

VIM - main help file

Move around: Use the cursor keys, or "h" to go left, "j" to go down, "k" to go up, "l" to go right.

Close this window: Use ":q<Enter>".

Get out of Vim: Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. [bars](#)) and hit **CTRL-]**.

With the mouse: ":set mouse=a" to enable the mouse (in xterm or GUI).
Double-click the left mouse button on a tag, e.g. [bars](#) .

Jump back: Type **CTRL-O**. Repeat to go further back.

Get specific help: It is possible to go directly to whatever you want help on, by giving an argument to the [:help](#) command.
Prepend something to specify the context: [help-context](#)

WHAT	PREPEND	EXAMPLE
Normal mode command		:help x
Visual mode command	v_	:help v_u
Insert mode command	i_	:help i_<Esc>
Command-line command	:	:help :quit
Command-line editing	c_	:help c_
Vim command argument	-	:help -r
Option	'	:help 'textwidth'
Regular expression	/	:help /[

See [help-summary](#) for more contexts and an explanation.
See [notation](#) for an explanation of the help syntax.

Search for help: Type ":help word", then hit **CTRL-D** to see matching help entries for "word".
Or use ":helpgrep word". [:helpgrep](#)

Getting started: Do the Vim tutor, a 30-minute interactive course for the basic commands, see [vimtutor](#) .
Read the user manual from start to end: [usr_01.txt](#)

Vim stands for Vi IMproved. Most of Vim was made by Bram Moolenaar, but only through the help of many others. See [credits](#) .

doc-file-list Q_ct

BASIC:

[quickref](#) Overview of the most common commands you will use

[tutor](#) 30-minute interactive course for beginners

[copying](#) About copyrights

[iccf](#) Helping poor children in Uganda

[sponsor](#) Sponsor Vim development, become a registered Vim user

[www](#) Vim on the World Wide Web

[bugs](#) Where to send bug reports

USER MANUAL: These files explain how to accomplish an editing task.

[usr_toc.txt](#) Table Of Contents

Getting Started

usr_01.txt	About the manuals
usr_02.txt	The first steps in Vim
usr_03.txt	Moving around
usr_04.txt	Making small changes
usr_05.txt	Set your settings
usr_06.txt	Using syntax highlighting
usr_07.txt	Editing more than one file
usr_08.txt	Splitting windows
usr_09.txt	Using the GUI
usr_10.txt	Making big changes
usr_11.txt	Recovering from a crash
usr_12.txt	Clever tricks

Editing Effectively

usr_20.txt	Typing command-line commands quickly
usr_21.txt	Go away and come back
usr_22.txt	Finding the file to edit
usr_23.txt	Editing other files
usr_24.txt	Inserting quickly
usr_25.txt	Editing formatted text
usr_26.txt	Repeating
usr_27.txt	Search commands and patterns
usr_28.txt	Folding
usr_29.txt	Moving through programs
usr_30.txt	Editing programs
usr_31.txt	Exploiting the GUI
usr_32.txt	The undo tree

Tuning Vim

usr_40.txt	Make new commands
usr_41.txt	Write a Vim script
usr_42.txt	Add new menus
usr_43.txt	Using filetypes
usr_44.txt	Your own syntax highlighted
usr_45.txt	Select your language

Writing Vim scripts

usr_50.txt	Advanced Vim script writing
usr_51.txt	Create a plugin
usr_52.txt	Write plugins using Vim9 script

Making Vim Run

usr_90.txt	Installing Vim
----------------------------	----------------

REFERENCE MANUAL: These files explain every detail of Vim.

[reference_toc](#)

General subjects

intro.txt	general introduction to Vim; notation used in help files
help.txt	overview and quick reference (this file)
helphelp.txt	about using the help files
index.txt	alphabetical index of all commands
help-tags	all the tags you can jump to (index of tags)

howto.txt	how to do the most common editing tasks
tips.txt	various tips on using Vim
message.txt	(error) messages and explanations
quotes.txt	remarks from users of Vim
todo.txt	known problems and desired extensions
develop.txt	development of Vim
debug.txt	debugging Vim itself
uganda.txt	Vim distribution conditions and what to do with your money

Basic editing

starting.txt	starting Vim, Vim command arguments, initialisation
editing.txt	editing and writing files
motion.txt	commands for moving around
scroll.txt	scrolling the text in the window
insert.txt	Insert and Replace mode
change.txt	deleting and replacing text
undo.txt	Undo and Redo
repeat.txt	repeating commands, Vim scripts and debugging
visual.txt	using the Visual mode (selecting a text area)
various.txt	various remaining commands
recover.txt	recovering from a crash

Advanced editing

cmdline.txt	Command-line editing
options.txt	description of all options
pattern.txt	regexp patterns and search commands
map.txt	key mapping and abbreviations
tagrch.txt	tags and special searches
windows.txt	commands for using multiple windows and buffers
tabpage.txt	commands for using multiple tab pages
spell.txt	spell checking
diff.txt	working with two to eight versions of the same file
autocmd.txt	automatically executing commands on an event
eval.txt	expression evaluation, conditional commands
builtin.txt	builtin functions
userfunc.txt	defining user functions
channel.txt	Jobs, Channels, inter-process communication
fold.txt	hide (fold) ranges of lines

Special issues

testing.txt	testing Vim and Vim scripts
print.txt	printing
remote.txt	using Vim as a server or client
term.txt	using different terminals and mice
terminal.txt	Terminal window support
popup.txt	popup window support
vim9.txt	using Vim9 script
vim9class.txt	using Vim9 script classes

Programming language support

indent.txt	automatic indenting for C and other languages
syntax.txt	syntax highlighting
textprop.txt	Attaching properties to text for highlighting or other
filetype.txt	settings done specifically for a type of file

quickfix.txt commands for a quick edit-compile-fix cycle
 ft_ada.txt Ada (the programming language) support
 ft_context.txt Filetype plugin for ConTeXt
 ft_mp.txt Filetype plugin for METAFONT and MetaPost
 ft_ps1.txt Filetype plugin for Windows PowerShell
 ft_raku.txt Filetype plugin for Raku
 ft_rust.txt Filetype plugin for Rust
 ft_sql.txt about the SQL filetype plugin

Language support

digraph.txt list of available digraphs
 mbyte.txt multibyte text support
 mlang.txt non-English language support
 rileft.txt right-to-left editing mode
 arabic.txt Arabic language support and editing
 farsi.txt Farsi (Persian) editing
 hebrew.txt Hebrew language support and editing
 russian.txt Russian language support and editing
 hangulin.txt Hangul (Korean) input mode

GUI

gui.txt Graphical User Interface (GUI)
 gui_w32.txt Win32 GUI
 gui_x11.txt X11 GUI

Interfaces

if_cscope.txt using Cscope with Vim
 if_lua.txt Lua interface
 if_mzsch.txt MzScheme interface
 if_perl.txt Perl interface
 if_pyth.txt Python interface
 if_tcl.txt Tcl interface
 if_ole.txt OLE automation interface for Win32
 if_ruby.txt Ruby interface
 debugger.txt Interface with a debugger
 netbeans.txt NetBeans External Editor interface
 sign.txt debugging signs

Versions

vi_diff.txt Main differences between Vim and Vi
 version4.txt Differences between Vim version 3.0 and 4.x
 version5.txt Differences between Vim version 4.6 and 5.x
 version6.txt Differences between Vim version 5.7 and 6.x
 version7.txt Differences between Vim version 6.4 and 7.x
 version8.txt Differences between Vim version 7.4 and 8.x
 version9.txt Differences between Vim version 8.2 and 9.0

[sys-file-list](#)

Remarks about specific systems

os_390.txt OS/390 Unix
 os_amiga.txt Amiga
 os_beos.txt BeOS and BeBox
 os_dos.txt MS-DOS and MS-Windows common items
 os_haiku.txt Haiku
 os_mac.txt Macintosh

os_mint.txt	Atari MiNT
os_msdos.txt	MS-DOS (plain DOS and DOS box under Windows)
os_os2.txt	OS/2
os_qnx.txt	QNX
os_risc.txt	RISC-OS
os_unix.txt	Unix
os_vms.txt	VMS
os_win32.txt	MS-Windows

standard-plugin-list

Standard plugins

pi_getscript.txt	Downloading latest version of Vim scripts
pi_gzip.txt	Reading and writing compressed files
pi_logipat.txt	Logical operators on patterns
pi_netrw.txt	Reading and writing files over a network
pi_paren.txt	Highlight matching parens
pi_spec.txt	Filetype plugin to work with rpm spec files
pi_tar.txt	Tar file explorer
pi_vimball.txt	Create a self-installing Vim script
pi_zip.txt	Zip archive explorer

LOCAL ADDITIONS:

local-additions

vim_faq.txt	Frequently asked questions
-------------	----------------------------

bars

Bars example

Now that you've jumped here with **CTRL-]** or a double mouse click, you can use **CTRL-T**, **CTRL-O**, `g<RightMouse>`, or `<C-RightMouse>` to go back to where you were.

Note that tags are within | characters, but when highlighting is enabled these characters are hidden. That makes it easier to read a command.

Anyway, you can use **CTRL-]** on any word, also when it is not within |, and Vim will try to find help for it. Especially for options in single quotes, e.g. **'compatible'**.

VIM REFERENCE MANUAL by Bram Moolenaar

Quick reference guide

tag	subject	tag	subject	quickref	Contents
Q_ct	list of help files	Q_re	Repeating commands		
Q_lr	motion: Left-right	Q_km	Key mapping		
Q_ud	motion: Up-down	Q_ab	Abbreviations		
Q_tm	motion: Text object	Q_op	Options		
Q_pa	motion: Pattern searches	Q_ur	Undo/Redo commands		
Q_ma	motion: Marks	Q_et	External commands		
Q_vm	motion: Various	Q_qf	Quickfix commands		
Q_ta	motion: Using tags	Q_vc	Various commands		
Q_sc	Scrolling	Q_ce	Ex: Command-line editing		
Q_in	insert: Inserting text	Q_ra	Ex: Ranges		
Q_ai	insert: Keys	Q_ex	Ex: Special characters		
Q_ss	insert: Special keys	Q_st	Starting Vim		
Q_di	insert: Digraphs	Q_ed	Editing a file		
Q_si	insert: Special inserts	Q_fl	Using the argument list		
Q_de	change: Deleting text	Q_wq	Writing and quitting		
Q_cm	change: Copying and moving	Q_ac	Automatic commands		
Q_ch	change: Changing text	Q_wi	Multi-window commands		
Q_co	change: Complex	Q_bu	Buffer list commands		
Q_vi	Visual mode	Q_sy	Syntax highlighting		
Q_to	Text objects	Q_gu	GUI commands		
		Q_fo	Folding		

N is used to indicate an optional count that can be given before the command.

Q_lr Left-right motions

h	N h	left (also: CTRL-H , <BS>, or <Left> key)
l	N l	right (also: <Space> or <Right> key)
0	0	to first character in the line (also: <Home> key)
^	^	to first non-blank character in the line
\$	N \$	to the last character in the line (N-1 lines lower) (also: <End> key)
g0	g0	to first character in screen line (differs from "0" when lines wrap)
g^	g^	to first non-blank character in screen line (differs from "^" when lines wrap)
g\$	N g\$	to last character in screen line (differs from "\$" when lines wrap)
gm	gm	to middle of the screen line
gM	gM	to middle of the line
bar	N	to column N (default: 1)
f	N f{char}	to the Nth occurrence of {char} to the right
F	N F{char}	to the Nth occurrence of {char} to the left
t	N t{char}	till before the Nth occurrence of {char} to the right

T	N	T{char}	till before the Nth occurrence of {char} to the left
;	N	;	repeat the last "f", "F", "t", or "T" N times
,	N	,	repeat the last "f", "F", "t", or "T" N times in opposite direction

Q_ud Up-down motions

k	N	k	up N lines (also: CTRL-P and <Up>)
j	N	j	down N lines (also: CTRL-J , CTRL-N , <NL>, and <Down>)
-	N	-	up N lines, on the first non-blank character
+	N	+	down N lines, on the first non-blank character (also: CTRL-M and <CR>)
_	N	_	down N-1 lines, on the first non-blank character
G	N	G	goto line N (default: last line), on the first non-blank character
gg	N	gg	goto line N (default: first line), on the first non-blank character
N%	N	%	goto line N percentage down in the file; N must be given, otherwise it is the % command
gk	N	gk	up N screen lines (differs from "k" when line wraps)
gj	N	gj	down N screen lines (differs from "j" when line wraps)

Q_tm Text object motions

w	N	w	N words forward
W	N	W	N blank-separated WORD s forward
e	N	e	forward to the end of the Nth word
E	N	E	forward to the end of the Nth blank-separated WORD
b	N	b	N words backward
B	N	B	N blank-separated WORD s backward
ge	N	ge	backward to the end of the Nth word
gE	N	gE	backward to the end of the Nth blank-separated WORD
)	N)	N sentences forward
(N	(N sentences backward
}	N	}	N paragraphs forward
{	N	{	N paragraphs backward
]]	N]]	N sections forward, at start of section
[[N	[[N sections backward, at start of section
][N][N sections forward, at end of section
[]	N	[]	N sections backward, at end of section
[(N	[(N times back to unclosed '('
[{	N	[{	N times back to unclosed '{'
[m	N	[m	N times back to start of method (for Java)
[M	N	[M	N times back to end of method (for Java)
])	N])	N times forward to unclosed ')'
]}	N]}	N times forward to unclosed '}'
]m	N]m	N times forward to start of method (for Java)
]M	N]M	N times forward to end of method (for Java)
[#	N	[#	N times back to unclosed "#if" or "#else"
]#	N]#	N times forward to unclosed "#else" or "#endif"
[star	N	[*	N times back to start of comment "/*"
]star	N]*	N times forward to end of comment "*/"

Q_pa Pattern searches

/	N	/{pattern}[/[offset]]<CR>	search forward for the Nth occurrence of {pattern}
?	N	?{pattern}[?[offset]]<CR>	search backward for the Nth occurrence of {pattern}
/<CR>	N	/<CR>	repeat last search, in the forward direction
?<CR>	N	?<CR>	repeat last search, in the backward direction
n	N	n	repeat last search
N	N	N	repeat last search, in opposite direction
star	N	*	search forward for the identifier under the cursor
#	N	#	search backward for the identifier under the cursor
gstar	N	g*	like "*", but also find partial matches
g#	N	g#	like "#", but also find partial matches
gd		gd	goto local declaration of identifier under the cursor
gD		gD	goto global declaration of identifier under the cursor

pattern Special characters in search patterns

	meaning	magic	nomagic
	matches any single character	.	\.
	matches start of line	^	^
	matches <EOL>	\$	\$
	matches start of word	\<	\<
	matches end of word	\>	\>
	matches a single char from the range	[a-z]	\[a-z]
	matches a single char not in the range	[^a-z]	\[^a-z]
	matches an identifier char	\i	\i
	idem but excluding digits	\I	\I
	matches a keyword character	\k	\k
	idem but excluding digits	\K	\K
	matches a file name character	\f	\f
	idem but excluding digits	\F	\F
	matches a printable character	\p	\p
	idem but excluding digits	\P	\P
	matches a white space character	\s	\s
	matches a non-white space character	\S	\S
	matches <Esc>	\e	\e
	matches <Tab>	\t	\t
	matches <CR>	\r	\r
	matches <BS>	\b	\b
	matches 0 or more of the preceding atom	*	*
	matches 1 or more of the preceding atom	\+	\+
	matches 0 or 1 of the preceding atom	\=	\=
	matches 2 to 5 of the preceding atom	\{2,5}	\{2,5}
	separates two alternatives	\	\
	group a pattern into an atom	\(\)	\(\)

search-offset Offsets allowed after search command

[num]	[num] lines downwards, in column 1
+ [num]	[num] lines downwards, in column 1

-[num] [num] lines upwards, in column 1
 e[+num] [num] characters to the right of the end of the match
 e[-num] [num] characters to the left of the end of the match
 s[+num] [num] characters to the right of the start of the match
 s[-num] [num] characters to the left of the start of the match
 b[+num] [num] identical to s[+num] above (mnemonic: begin)
 b[-num] [num] identical to s[-num] above (mnemonic: begin)
 ;{search-command} execute {search-command} next

Q_ma Marks and motions

m m{a-zA-Z} mark current position with mark {a-zA-Z}
 `a `{a-z} go to mark {a-z} within current file
 `A `{A-Z} go to mark {A-Z} in any file
 `0 `{0-9} go to the position where Vim was previously exited
 `` go to the position before the last jump
 `quote `` go to the position when last editing this file
 `[`[go to the start of the previously operated or put text
 `] `] go to the end of the previously operated or put text
 `< `< go to the start of the (previous) Visual area
 `> `> go to the end of the (previous) Visual area
 `.` `.` go to the position of the last change in this file
 `' `{a-zA-Z0-9[]'<>.`} same as ` , but on the first non-blank in the line
 :marks :marks print the active marks
 CTRL-O N CTRL-O go to Nth older position in jump list
 CTRL-I N CTRL-I go to Nth newer position in jump list
 :ju :ju[mps] print the jump list

Q_vm Various motions

% % find the next brace, bracket, comment, or "#if"/
 "#else"/"#endif" in this line and go to its match
 H N H go to the Nth line in the window, on the first
 non-blank
 M M go to the middle line in the window, on the first
 non-blank
 L N L go to the Nth line from the bottom, on the first
 non-blank
 go N go go to Nth byte in the buffer
 :go :[range]go[to] [off] go to [off] byte in the buffer

Q_ta Using tags

:ta :ta[g][!] {tag} jump to tag {tag}
 :ta :[count]ta[g][!] jump to [count]'th newer tag in tag list
 CTRL-] CTRL-] jump to the tag under cursor, unless changes
 have been made
 :ts :ts[elect][!] [tag] list matching tags and select one to jump to
 :tjump :tj[ump][!] [tag] jump to tag [tag] or select from list when
 there are multiple matches
 :ltag :lt[ag][!] [tag] jump to tag [tag] and add matching tags to the
 location list

<code>:tags</code>	<code>:tags</code>	print tag list
<code>CTRL-T</code>	<code>N CTRL-T</code>	jump back from Nth older tag in tag list
<code>:po</code>	<code>:[count]po[p][!]</code>	jump back from [count]'th older tag in tag list
<code>:tnext</code>	<code>:[count]tn[ext][!]</code>	jump to [count]'th next matching tag
<code>:tp</code>	<code>:[count]tp[revious][!]</code>	jump to [count]'th previous matching tag
<code>:tr</code>	<code>:[count]tr[ewind][!]</code>	jump to [count]'th matching tag
<code>:tl</code>	<code>:tl[ast][!]</code>	jump to last matching tag
<hr/>		
<code>:ptag</code>	<code>:pt[ag] {tag}</code>	open a preview window to show tag {tag}
<code>CTRL-W_}</code>	<code>CTRL-W }</code>	like <code>CTRL-]</code> but show tag in preview window
<code>:pts</code>	<code>:pts[elect]</code>	like <code>":tselect"</code> but show tag in preview window
<code>:ptjump</code>	<code>:ptj[ump]</code>	like <code>":tjump"</code> but show tag in preview window
<code>:pclose</code>	<code>:pc[lose]</code>	close tag preview window
<code>CTRL-W_z</code>	<code>CTRL-W z</code>	close tag preview window

Q_sc Scrolling

<code>CTRL-E</code>	<code>N CTRL-E</code>	window N lines downwards (default: 1)
<code>CTRL-D</code>	<code>N CTRL-D</code>	window N lines Downwards (default: 1/2 window)
<code>CTRL-F</code>	<code>N CTRL-F</code>	window N pages Forwards (downwards)
<code>CTRL-Y</code>	<code>N CTRL-Y</code>	window N lines upwards (default: 1)
<code>CTRL-U</code>	<code>N CTRL-U</code>	window N lines Upwards (default: 1/2 window)
<code>CTRL-B</code>	<code>N CTRL-B</code>	window N pages Backwards (upwards)
<code>z<CR></code>	<code>z<CR> or zt</code>	redraw, current line at top of window
<code>z.</code>	<code>z. or zz</code>	redraw, current line at center of window
<code>z-</code>	<code>z- or zb</code>	redraw, current line at bottom of window

These only work when `'wrap'` is off:

<code>zh</code>	<code>N zh</code>	scroll screen N characters to the right
<code>zl</code>	<code>N zl</code>	scroll screen N characters to the left
<code>zH</code>	<code>N zH</code>	scroll screen half a screenwidth to the right
<code>zL</code>	<code>N zL</code>	scroll screen half a screenwidth to the left

Q_in Inserting text

<code>a</code>	<code>N a</code>	append text after the cursor (N times)
<code>A</code>	<code>N A</code>	append text at the end of the line (N times)
<code>i</code>	<code>N i</code>	insert text before the cursor (N times) (also: <code><Insert></code>)
<code>I</code>	<code>N I</code>	insert text before the first non-blank in the line (N times)
<code>gI</code>	<code>N gI</code>	insert text in column 1 (N times)
<code>o</code>	<code>N o</code>	open a new line below the current line, append text (N times)
<code>O</code>	<code>N O</code>	open a new line above the current line, append text (N times)
<code>:startinsert</code>	<code>:star[tinsert][!]</code>	start Insert mode, append when [!] used
<code>:startreplace</code>	<code>:startr[eplace][!]</code>	start Replace mode, at EOL when [!] used

in Visual block mode:

<code>v_b_I</code>	<code>I</code>	insert the same text in front of all the selected lines
<code>v_b_A</code>	<code>A</code>	append the same text after all the selected lines

Q_ai Insert mode keys

`insert-index` alphabetical index of Insert mode commands

leaving Insert mode:

<code>i_<Esc></code>	<code><Esc></code>	end Insert mode, back to Normal mode
<code>i_CTRL-C</code>	CTRL-C	like <code><Esc></code> , but do not use an abbreviation
<code>i_CTRL-O</code>	CTRL-O <code>{command}</code>	execute <code>{command}</code> and return to Insert mode

moving around:

<code>i_<Up></code>	cursor keys	move cursor left/right/up/down
<code>i_<S-Left></code>	shift-left/right	one word left/right
<code>i_<S-Up></code>	shift-up/down	one screenful backward/forward
<code>i_<End></code>	<code><End></code>	cursor after last character in the line
<code>i_<Home></code>	<code><Home></code>	cursor to first character in the line

`Q_ss` Special keys in Insert mode

<code>i_CTRL-V</code>	CTRL-V <code>{char}..</code>	insert character literally, or enter decimal byte value
<code>i_<NL></code>	<code><NL></code> or <code><CR></code> or CTRL-M or CTRL-J	begin new line
<code>i_CTRL-E</code>	CTRL-E	insert the character from below the cursor
<code>i_CTRL-Y</code>	CTRL-Y	insert the character from above the cursor
<code>i_CTRL-A</code>	CTRL-A	insert previously inserted text
<code>i_CTRL-@</code>	CTRL-@	insert previously inserted text and stop Insert mode
<code>i_CTRL-R</code>	CTRL-R <code>{register}</code>	insert the contents of a register
<code>i_CTRL-N</code>	CTRL-N	insert next match of identifier before the cursor
<code>i_CTRL-P</code>	CTRL-P	insert previous match of identifier before the cursor
<code>i_CTRL-X</code>	CTRL-X ...	complete the word before the cursor in various ways
<code>i_<BS></code>	<code><BS></code> or CTRL-H	delete the character before the cursor
<code>i_</code>	<code></code>	delete the character under the cursor
<code>i_CTRL-W</code>	CTRL-W	delete word before the cursor
<code>i_CTRL-U</code>	CTRL-U	delete all entered characters in the current line
<code>i_CTRL-T</code>	CTRL-T	insert one shiftwidth of indent in front of the current line
<code>i_CTRL-D</code>	CTRL-D	delete one shiftwidth of indent in front of the current line
<code>i_0_CTRL-D</code>	<code>0</code> CTRL-D	delete all indent in the current line
<code>i_^_CTRL-D</code>	<code>^</code> CTRL-D	delete all indent in the current line, restore indent in next line

`Q_di` Digraphs

<code>:dig</code>	<code>:dig[raphs]</code>	show current list of digraphs
<code>:dig</code>	<code>:dig[raphs]</code> <code>{char1}{char2}</code> <code>{number}</code> ...	add digraph(s) to the list

In Insert or Command-line mode:

<code>i_CTRL-K</code>	CTRL-K <code>{char1}</code> <code>{char2}</code>
-----------------------	---

enter digraph
i_digraph {char1} <BS> {char2}
enter digraph if 'digraph' option set

Q_si Special inserts

:r :r [file] insert the contents of [file] below the cursor
:r! :r! {command} insert the standard output of {command} below the cursor

Q_de Deleting text

x N x delete N characters under and after the cursor
 N delete N characters under and after the cursor
X N X delete N characters before the cursor
d N d{motion} delete the text that is moved over with {motion}
v_d {visual}d delete the highlighted text
dd N dd delete N lines
D N D delete to the end of the line (and N-1 more lines)
J N J join N-1 lines (delete <EOL>s)
v_J {visual}J join the highlighted lines
gJ N gJ like "J", but without inserting spaces
v_gJ {visual}gJ like "{visual}J", but without inserting spaces
:d :[range]d [x] delete [range] lines [into register x]

Q_cm Copying and moving text

quote "{char} use register {char} for the next delete, yank, or put
:reg :reg show the contents of all registers
:reg :reg {arg} show the contents of registers mentioned in {arg}
y N y{motion} yank the text moved over with {motion} into a register
v_y {visual}y yank the highlighted text into a register
yy N yy yank N lines into a register
Y N Y yank N lines into a register
p N p put a register after the cursor position (N times)
P N P put a register before the cursor position (N times)
]p N]p like p, but adjust indent to current line
[p N [p like P, but adjust indent to current line
gp N gp like p, but leave cursor after the new text
gP N gP like P, but leave cursor after the new text

Q_ch Changing text

r N r{char} replace N characters with {char}
gr N gr{char} replace N characters without affecting layout
R N R enter Replace mode (repeat the entered text N times)
gR N gR enter virtual Replace mode: Like Replace mode but without affecting layout
v_b_r {visual}r{char}
in Visual block mode: Replace each char of the selected text with {char}

(change = delete text and enter Insert mode)

c N c{motion} change the text that is moved over with {motion}

v_c		{visual}c	change the highlighted text
cc	N	cc	change N lines
S	N	S	change N lines
C	N	C	change to the end of the line (and N-1 more lines)
s	N	s	change N characters
v_b_c		{visual}c	in Visual block mode: Change each of the selected lines with the entered text
v_b_C		{visual}C	in Visual block mode: Change each of the selected lines until end-of-line with the entered text
~	N	~	switch case for N characters and advance cursor
v_~		{visual}~	switch case for highlighted text
v_u		{visual}u	make highlighted text lowercase
v_U		{visual}U	make highlighted text uppercase
g~		g~{motion}	switch case for the text that is moved over with {motion}
gu		gu{motion}	make the text that is moved over with {motion} lowercase
gU		gU{motion}	make the text that is moved over with {motion} uppercase
v_g?		{visual}g?	perform rot13 encoding on highlighted text
g?		g?{motion}	perform rot13 encoding on the text that is moved over with {motion}
CTRL-A	N	CTRL-A	add N to the number at or after the cursor
CTRL-X	N	CTRL-X	subtract N from the number at or after the cursor
<	N	<{motion}	move the lines that are moved over with {motion} one shiftwidth left
<<	N	<<	move N lines one shiftwidth left
>	N	>{motion}	move the lines that are moved over with {motion} one shiftwidth right
>>	N	>>	move N lines one shiftwidth right
gq	N	gq{motion}	format the lines that are moved over with {motion} to 'textwidth' length
:ce		:[range]ce[nter] [width]	center the lines in [range]
:le		:[range]le[ft] [indent]	left-align the lines in [range] (with [indent])
:ri		:[range]ri[ght] [width]	right-align the lines in [range]

Q_co Complex changes

!	N	!{motion}{command}<CR>	filter the lines that are moved over through {command}
!!	N	!!{command}<CR>	filter N lines through {command}
v_!		{visual}!{command}<CR>	filter the highlighted lines through {command}
:range!		:[range]! {command}<CR>	filter [range] lines through {command}
=	N	= {motion}	filter the lines that are moved over through 'equalprg'

```

==      N ==      filter N lines through 'equalprg'
v_-=    {visual}=  filter the highlighted lines through 'equalprg'
:s      :[range]s[ubstitute]/{pattern}/{string}/[g][c]
        substitute {pattern} by {string} in [range] lines;
        with [g], replace all occurrences of {pattern};
        with [c], confirm each replacement
:s      :[range]s[ubstitute] [g][c]
        repeat previous ":s" with new range and options
&       &        Repeat previous ":s" on current line without options
:ret    :[range]ret[ab][!] [tabstop]
        set 'tabstop' to new value and adjust white space
        accordingly

```

Q_vi Visual mode

visual-index list of Visual mode commands.

```

v      v      start highlighting characters } move cursor and use
V      V      start highlighting linewise   } operator to affect
CTRL-V CTRL-V start highlighting blockwise } highlighted text
v_o    o      exchange cursor position with start of highlighting
gv     gv     start highlighting on previous visual area
v_v    v      highlight characters or stop highlighting
v_V    V      highlight linewise or stop highlighting
v_CTRL-V CTRL-V highlight blockwise or stop highlighting

```

Q_to Text objects (only in Visual mode or after an operator)

```

v_aw    N aw    Select "a word"
v_iw    N iw    Select "inner word"
v_aW    N aW    Select "a WORD "
v_iW    N iW    Select "inner WORD "
v_as    N as    Select "a sentence"
v_is    N is    Select "inner sentence"
v_ap    N ap    Select "a paragraph"
v_ip    N ip    Select "inner paragraph"
v_ab    N ab    Select "a block" (from "[" to ")")
v_ib    N ib    Select "inner block" (from "[" to ")")
v_aB    N aB    Select "a Block" (from "{" to "}")
v_iB    N iB    Select "inner Block" (from "{" to "}")
v_a>    N a>    Select "a <> block"
v_i>    N i>    Select "inner <> block"
v_at    N at    Select "a tag block" (from <aaa> to </aaa>)
v_it    N it    Select "inner tag block" (from <aaa> to </aaa>)
v_a'    N a'    Select "a single quoted string"
v_i'    N i'    Select "inner single quoted string"
v_aquote N a"    Select "a double quoted string"
v_iquote N i"    Select "inner double quoted string"
v_a`    N a`    Select "a backward quoted string"
v_i`    N i`    Select "inner backward quoted string"

```

Q_re Repeating commands

```

.      N .      repeat last change (with count replaced with N)
q      q{a-z}   record typed characters into register {a-z}
q      q{A-Z}   record typed characters, appended to register {a-z}
q      q        stop recording
@      N @{a-z}  execute the contents of register {a-z} (N times)
@@     N @@      repeat previous @{a-z} (N times)
:@     :@{a-z}   execute the contents of register {a-z} as an Ex
          command
:@@    :@@      repeat previous :@{a-z}
:g     :[range]g[lobal]/{pattern}/[cmd]
          execute Ex command [cmd] (default: ":p") on the lines
          within [range] where {pattern} matches
:g     :[range]g[lobal]!/{pattern}/[cmd]
          execute Ex command [cmd] (default: ":p") on the lines
          within [range] where {pattern} does NOT match
:so    :so[urce] {file}
          read Ex commands from {file}
:so    :so[urce]! {file}
          read Vim commands from {file}
:sl    :sl[EEP] [sec]
          don't do anything for [sec] seconds
gs     N gs     goto Sleep for N seconds

```

Q_km Key mapping

```

:map    :ma[p] {lhs} {rhs}  map {lhs} to {rhs} in Normal and Visual mode
:map!   :ma[p]! {lhs} {rhs} map {lhs} to {rhs} in Insert and Command-line
          mode
:noremap :no[re]map[!] {lhs} {rhs}
          same as ":map", no remapping for this {rhs}
:unmap  :unm[ap] {lhs}      remove the mapping of {lhs} for Normal and
          Visual mode
:unmap! :unm[ap]! {lhs}    remove the mapping of {lhs} for Insert and
          Command-line mode
:map_l  :ma[p] [lhs]        list mappings (starting with [lhs]) for
          Normal and Visual mode
:map_l! :ma[p]! [lhs]      list mappings (starting with [lhs]) for
          Insert and Command-line mode
:cmap   :cmap/:cunmap/:cnoremap
          like ":map!"/":unmap!"/":noremap!" but for
          Command-line mode only
:imap   :imap/:iunmap/:inoremap
          like ":map!"/":unmap!"/":noremap!" but for
          Insert mode only
:nmap   :nmap/:nunmap/:nnoremap
          like ":map"/":unmap"/":noremap" but for
          Normal mode only
:vmap   :vmap/:vunmap/:vnoremap
          like ":map"/":unmap"/":noremap" but for
          Visual mode only
:omap   :omap/:ounmap/:onoremap
          like ":map"/":unmap"/":noremap" but only for
          when an operator is pending

```

```

:mapc      :mapc[lear]          remove mappings for Normal and Visual mode
:mapc      :mapc[lear]!      remove mappings for Insert and Cmdline mode
:imapc     :imapc[lear]      remove mappings for Insert mode
:vmapc     :vmapc[lear]      remove mappings for Visual mode
:omapc     :omapc[lear]      remove mappings for Operator-pending mode
:nmapc     :nmapc[lear]      remove mappings for Normal mode
:cmmapc    :cmmapc[lear]     remove mappings for Cmdline mode
:mkexrc    :mk[exrc][!] [file] write current mappings, abbreviations, and
                                     settings to [file] (default: ".exrc";
                                     use ! to overwrite)

:mkvimrc   :mkv[imrc][!] [file]
                                     same as ":mkexrc", but with default ".vimrc"

:mksession :mks[ession][!] [file]
                                     like ":mkvimrc", but store current files,
                                     windows, etc. too, to be able to continue
                                     this session later

```

Q_ab Abbreviations

```

:abbreviate :ab[breviate] {lhs} {rhs} add abbreviation for {lhs} to {rhs}
:abbreviate :ab[breviate] {lhs}      show abbr's that start with {lhs}
:abbreviate :ab[breviate]            show all abbreviations
:unabbreviate :una[bbreviate] {lhs}  remove abbreviation for {lhs}
:noreabbrev  :norea[bbrev] [lhs] [rhs] like ":ab", but don't remap [rhs]
:iabbrev     :iab/:iunab/:inoreab    like ":ab", but only for Insert mode
:cabbrev     :cab/:cunab/:cnoreab    like ":ab", but only for
                                     Command-line mode

:abclear     :abc[lear]              remove all abbreviations
:cabclear    :cabc[lear]             remove all abbr's for Cmdline mode
:iabclear    :iabc[lear]             remove all abbr's for Insert mode

```

Q_op Options

```

:set         :se[t]                show all modified options
:set         :se[t] all              show all non-termcap options
:set         :se[t] termcap          show all termcap options
:set         :se[t] {option}         set boolean option (switch it on),
                                     show string or number option

:set         :se[t] no{option}       reset boolean option (switch it off)
:set         :se[t] inv{option}      invert boolean option
:set         :se[t] {option}={value} set string/number option to {value}
:set         :se[t] {option}+={value} append {value} to string option, add
                                     {value} to number option

:set         :se[t] {option}-={value} remove {value} to string option,
                                     subtract {value} from number option

:set         :se[t] {option}?        show value of {option}
:set         :se[t] {option}&        reset {option} to its default value

:setlocal   :setl[ocal]             like ":set" but set the local value
                                     for options that have one

:setglobal  :setg[lobal]           like ":set" but set the global value
                                     of a local option

:fix        :fix[del]              set value of 't_kD' according to

```

`:options` `:opt[ions]` value of `'t_kb'`
 open a new window to view and set options, grouped by functionality, a one line explanation and links to the help

Short explanation of each option:

`option-list`

<code>'aleph'</code>	<code>'al'</code>	ASCII code of the letter Aleph (Hebrew)
<code>'allowrevins'</code>	<code>'ari'</code>	allow <code>CTRL_</code> in Insert and Command-line mode
<code>'altkeymap'</code>	<code>'akm'</code>	obsolete option for Farsi
<code>'ambiwidth'</code>	<code>'ambw'</code>	what to do with Unicode chars of ambiguous width
<code>'antialias'</code>	<code>'anti'</code>	Mac OS X: use smooth, antialiased fonts
<code>'arabic'</code>	<code>'arab'</code>	for Arabic as a default second language
<code>'arabicshape'</code>	<code>'arshape'</code>	do shaping for Arabic characters
<code>'autochdir'</code>	<code>'acd'</code>	change directory to the file in the current window
<code>'autoindent'</code>	<code>'ai'</code>	take indent for new line from previous line
<code>'autoread'</code>	<code>'ar'</code>	autom. read file when changed outside of Vim
<code>'autoshelldir'</code>	<code>'asd'</code>	change directory to the shell's current directory
<code>'autowrite'</code>	<code>'aw'</code>	automatically write file if changed
<code>'autowriteall'</code>	<code>'awa'</code>	as <code>'autowrite'</code> , but works with more commands
<code>'background'</code>	<code>'bg'</code>	"dark" or "light", used for highlight colors
<code>'backspace'</code>	<code>'bs'</code>	how backspace works at start of line
<code>'backup'</code>	<code>'bk'</code>	keep backup file after overwriting a file
<code>'backupcopy'</code>	<code>'bkc'</code>	make backup as a copy, don't rename the file
<code>'backupdir'</code>	<code>'bdir'</code>	list of directories for the backup file
<code>'backupext'</code>	<code>'bex'</code>	extension used for the backup file
<code>'backupskip'</code>	<code>'bsk'</code>	no backup for files that match these patterns
<code>'balloondelay'</code>	<code>'bdlay'</code>	delay in mS before a balloon may pop up
<code>'ballooneval'</code>	<code>'beval'</code>	switch on balloon evaluation in the GUI
<code>'balloonevalterm'</code>	<code>'bevalterm'</code>	switch on balloon evaluation in the terminal
<code>'balloonexpr'</code>	<code>'bexpr'</code>	expression to show in balloon
<code>'belloff'</code>	<code>'bo'</code>	do not ring the bell for these reasons
<code>'binary'</code>	<code>'bin'</code>	read/write/edit file in binary mode
<code>'bioskey'</code>	<code>'biosk'</code>	MS-DOS: use bios calls for input characters
<code>'bomb'</code>		prepend a Byte Order Mark to the file
<code>'breakat'</code>	<code>'brk'</code>	characters that may cause a line break
<code>'breakindent'</code>	<code>'bri'</code>	wrapped line repeats indent
<code>'breakindentopt'</code>	<code>'briopt'</code>	settings for <code>'breakindent'</code>
<code>'browseidir'</code>	<code>'bsdir'</code>	which directory to start browsing in
<code>'bufhidden'</code>	<code>'bh'</code>	what to do when buffer is no longer in window
<code>'buflisted'</code>	<code>'bl'</code>	whether the buffer shows up in the buffer list
<code>'buftype'</code>	<code>'bt'</code>	special type of buffer
<code>'casemap'</code>	<code>'cmp'</code>	specifies how case of letters is changed
<code>'cdhome'</code>	<code>'cdh'</code>	change directory to the home directory by <code>":cd"</code>
<code>'cdpath'</code>	<code>'cd'</code>	list of directories searched with <code>":cd"</code>
<code>'cedit'</code>		key used to open the command-line window
<code>'charconvert'</code>	<code>'ccv'</code>	expression for character encoding conversion
<code>'cindent'</code>	<code>'cin'</code>	do C program indenting
<code>'cinkeys'</code>	<code>'cink'</code>	keys that trigger indent when <code>'cindent'</code> is set
<code>'cinoptions'</code>	<code>'cino'</code>	how to do indenting when <code>'cindent'</code> is set
<code>'cinscopedecls'</code>	<code>'cinsd'</code>	words that are recognized by <code>'cino-g'</code>
<code>'cinwords'</code>	<code>'cinw'</code>	words where <code>'si'</code> and <code>'cin'</code> add an indent
<code>'clipboard'</code>	<code>'cb'</code>	use the clipboard as the unnamed register
<code>'cmdheight'</code>	<code>'ch'</code>	number of lines to use for the command-line

'cmdwinheight'	'cwh'	height of the command-line window
'colorcolumn'	'cc'	columns to highlight
'columns'	'co'	number of columns in the display
'comments'	'com'	patterns that can start a comment line
'commentstring'	'cms'	template for comments; used for fold marker
'compatible'	'cp'	behave Vi-compatible as much as possible
'complete'	'cpt'	specify how Insert mode completion works
'completefunc'	'cfu'	function to be used for Insert mode completion
'completeopt'	'cot'	options for Insert mode completion
'completepopup'	'cpp'	options for the Insert mode completion info popup
'completeslash'	'csl'	like 'shellslash' for completion
'concealcursor'	'cocu'	whether concealable text is hidden in cursor line
'conceallevel'	'cole'	whether concealable text is shown or hidden
'confirm'	'cf'	ask what to do about unsaved/read-only files
'conskey'	'consk'	get keys directly from console (MS-DOS only)
'copyindent'	'ci'	make 'autoindent' use existing indent structure
'cptions'	'cpo'	flags for Vi-compatible behavior
'cryptmethod'	'cm'	type of encryption to use for file writing
'cscopepathcomp'	'cspc'	how many components of the path to show
'cscopeprg'	'csprg'	command to execute cscope
'cscopequickfix'	'csqf'	use quickfix window for cscope results
'cscoperelative'	'csre'	Use cscope.out path basename as prefix
'cscopetag'	'cst'	use cscope for tag commands
'cscopetagorder'	'csto'	determines ":cstag" search order
'cscopeverbose'	'csverb'	give messages when adding a cscope database
'cursorbind'	'crb'	move cursor in window as it moves in other windows
'cursorcolumn'	'cuc'	highlight the screen column of the cursor
'cursorline'	'cul'	highlight the screen line of the cursor
'cursorlineopt'	'culopt'	settings for 'cursorline'
'debug'		set to "msg" to see all error messages
'define'	'def'	pattern to be used to find a macro definition
'delcombine'	'deco'	delete combining characters on their own
'dictionary'	'dict'	list of file names used for keyword completion
'diff'		use diff mode for the current window
'diffexpr'	'dex'	expression used to obtain a diff file
'diffopt'	'dip'	options for using diff mode
'digraph'	'dg'	enable the entering of digraphs in Insert mode
'directory'	'dir'	list of directory names for the swap file
'display'	'dy'	list of flags for how to display text
'eadirection'	'ead'	in which direction 'equalalways' works
'edcompatible'	'ed'	toggle flags of ":substitute" command
'emoji'	'emo'	emoji characters are considered full width
'encoding'	'enc'	encoding used internally
'endoffile'	'eof'	write CTRL-Z at end of the file
'endofline'	'eol'	write <EOL> for last line in file
'equalalways'	'ea'	windows are automatically made the same size
'equalprg'	'ep'	external program to use for "=" command
'errorbells'	'eb'	ring the bell for error messages
'errorfile'	'ef'	name of the errorfile for the QuickFix mode
'errorformat'	'efm'	description of the lines in the error file
'esckey'	'ek'	recognize function keys in Insert mode
'eventignore'	'ei'	autocommand events that are ignored
'expandtab'	'et'	use spaces when <Tab> is inserted
'exec'	'ex'	read .vimrc and .exec in the current directory

'fileencoding'	'fenc'	file encoding for multibyte text
'fileencodings'	'fencs'	automatically detected character encodings
'fileformat'	'ff'	file format used for file I/O
'fileformats'	'ffs'	automatically detected values for 'fileformat'
'fileignorecase'	'fic'	ignore case when using file names
'filetype'	'ft'	type of file, used for autocommands
'fillchars'	'fcs'	characters to use for displaying special items
'fixendofline'	'fixeol'	make sure last line in file has <EOL>
'fkmap'	'fk'	obsolete option for Farsi
'foldclose'	'fcl'	close a fold when the cursor leaves it
'foldcolumn'	'fdc'	width of the column used to indicate folds
'foldenable'	'fen'	set to display all folds open
'foldexpr'	'fde'	expression used when 'foldmethod' is "expr"
'foldignore'	'fdi'	ignore lines when 'foldmethod' is "indent"
'foldlevel'	'fdl'	close folds with a level higher than this
'foldlevelstart'	'fdls'	'foldlevel' when starting to edit a file
'foldmarker'	'fmr'	markers used when 'foldmethod' is "marker"
'foldmethod'	'fdm'	folding type
'foldminlines'	'fml'	minimum number of lines for a fold to be closed
'foldnestmax'	'fdn'	maximum fold depth
'foldopen'	'fdo'	for which commands a fold will be opened
'foldtext'	'fdt'	expression used to display for a closed fold
'formatexpr'	'fex'	expression used with "gq" command
'formatlistpat'	'flp'	pattern used to recognize a list header
'formatoptions'	'fo'	how automatic formatting is to be done
'formatprg'	'fp'	name of external program used with "gq" command
'fsync'	'fs'	whether to invoke fsync() after file write
'gdefault'	'gd'	the ":substitute" flag 'g' is default on
'grepformat'	'gfm'	format of 'grepprg' output
'grepprg'	'gp'	program to use for ":grep"
'guicursor'	'gcr'	GUI: settings for cursor shape and blinking
'guifont'	'gfn'	GUI: Name(s) of font(s) to be used
'guifontset'	'gfs'	GUI: Names of multibyte fonts to be used
'guifontwide'	'gfw'	list of font names for double-wide characters
'guiheadroom'	'ghr'	GUI: pixels room for window decorations
'guiligatures'	'gli'	GTK GUI: ASCII characters that can form shapes
'guioptions'	'go'	GUI: Which components and options are used
'guipty'		GUI: try to use a pseudo-tty for ":@" commands
'guitablabel'	'gtl'	GUI: custom label for a tab page
'guitabtooltip'	'gtt'	GUI: custom tooltip for a tab page
'helpfile'	'hf'	full path name of the main help file
'helpheight'	'hh'	minimum height of a new help window
'helplang'	'hlg'	preferred help languages
'hidden'	'hid'	don't unload buffer when it is abandon ed
'highlight'	'hl'	sets highlighting mode for various occasions
'history'	'hi'	number of command-lines that are remembered
'hkmap'	'hk'	Hebrew keyboard mapping
'hkmapprg'	'hkp'	phonetic Hebrew keyboard mapping
'hlsearch'	'hls'	highlight matches with last search pattern
'icon'		let Vim set the text of the window icon
'iconstring'		string to use for the Vim icon text
'ignorecase'	'ic'	ignore case in search patterns
'imactivatefunc'	'imaf'	function to enable/disable the X input method
'imactivatekey'	'imak'	key that activates the X input method

'imcmdline'	'imc'	use IM when starting to edit a command line
'imdisable'	'imd'	do not use the IM in any mode
'iminsert'	'imi'	use :lmap or IM in Insert mode
'imsearch'	'ims'	use :lmap or IM when typing a search pattern
'imstatusfunc'	'imsf'	function to obtain X input method status
'imstyle'	'imst'	specifies the input style of the input method
'include'	'inc'	pattern to be used to find an include file
'includeexpr'	'inex'	expression used to process an include line
'incsearch'	'is'	highlight match while typing search pattern
'indentexpr'	'inde'	expression used to obtain the indent of a line
'indentkeys'	'indk'	keys that trigger indenting with 'indentexpr'
'infercase'	'inf'	adjust case of match for keyword completion
'insertmode'	'im'	start the edit of a file in Insert mode
'isfname'	'isf'	characters included in file names and pathnames
'isident'	'isi'	characters included in identifiers
'iskeyword'	'isk'	characters included in keywords
'isprint'	'isp'	printable characters
'joinspaces'	'js'	two spaces after a period with a join command
'jumpoptions'	'jop'	specifies how jumping is done
'key'		encryption key
'keymap'	'kmp'	name of a keyboard mapping
'keymodel'	'km'	enable starting/stopping selection with keys
'keyprotocol'	'kpc'	what keyboard protocol to use for what terminal
'keywordprg'	'kp'	program to use for the "K" command
'langmap'	'lmap'	alphabetic characters for other language mode
'langmenu'	'lm'	language to be used for the menus
'langnoremap'	'lnr'	do not apply 'langmap' to mapped characters
'langremap'	'lrm'	do apply 'langmap' to mapped characters
'laststatus'	'ls'	tells when last window has status lines
'lazyredraw'	'lz'	don't redraw while executing macros
'linebreak'	'lbr'	wrap long lines at a blank
'lines'		number of lines in the display
'linespace'	'lsp'	number of pixel lines to use between characters
'lisp'		automatic indenting for Lisp
'lispoptions'	'lop'	changes how Lisp indenting is done
'lispwords'	'lw'	words that change how lisp indenting works
'list'		show <Tab> and <EOL>
'listchars'	'lcs'	characters for displaying in list mode
'loadplugins'	'lpl'	load plugin scripts when starting up
'luadll'		name of the Lua dynamic library
'macatsui'		Mac GUI: use ATSUI text drawing
'magic'		changes special characters in search patterns
'makeef'	'mef'	name of the errorfile for ":make"
'makeencoding'	'menc'	encoding of external make/grep commands
'makeprg'	'mp'	program to use for the ":make" command
'matchpairs'	'mps'	pairs of characters that "%" can match
'matchtime'	'mat'	tenths of a second to show matching paren
'maxcombine'	'mco'	maximum nr of combining characters displayed
'maxfuncdepth'	'mfd'	maximum recursive depth for user functions
'maxmapdepth'	'mmd'	maximum recursive depth for mapping
'maxmem'	'mm'	maximum memory (in Kbyte) used for one buffer
'maxmempattern'	'mmp'	maximum memory (in Kbyte) used for pattern search
'maxmemtot'	'mmt'	maximum memory (in Kbyte) used for all buffers
'menuitems'	'mis'	maximum number of items in a menu

'mkspellmem'	'msm'	memory used before <code>:mkspell</code> compresses the tree
'modeline'	'ml'	recognize modelines at start or end of file
'modelineexpr'	'mle'	allow setting expression options from a modeline
'modelines'	'mls'	number of lines checked for modelines
'modifiable'	'ma'	changes to the text are not possible
'modified'	'mod'	buffer has been modified
'more'		pause listings when the whole screen is filled
'mouse'		enable the use of mouse clicks
'mousefocus'	'mousef'	keyboard focus follows the mouse
'mousehide'	'mh'	hide mouse pointer while typing
'mousemodel'	'mousem'	changes meaning of mouse buttons
'mousemoveevent'	'mousemev'	report mouse moves with <code><MouseMove></code>
'mouseshape'	'mouses'	shape of the mouse pointer in different modes
'mousetime'	'mouset'	max time between mouse double-click
'mzquantum'	'mzq'	the interval between polls for MzScheme threads
'mzschemedll'		name of the MzScheme dynamic library
'mzschemegcdll'		name of the MzScheme dynamic library for GC
'nrformats'	'nf'	number formats recognized for CTRL-A command
'number'	'nu'	print the line number in front of each line
'numberwidth'	'nuw'	number of columns used for the line number
'omnifunc'	'ofu'	function for filetype-specific completion
'opendevic'	'odev'	allow reading/writing devices on MS-Windows
'operatorfunc'	'opfunc'	function to be called for <code>g@</code> operator
'osfiletype'	'oft'	no longer supported
'packpath'	'pp'	list of directories used for packages
'paragraphs'	'para'	nröff macros that separate paragraphs
'paste'		allow pasting text
'pastetoggle'	'pt'	key code that causes <code>'paste'</code> to toggle
'patchexpr'	'pex'	expression used to patch a file
'patchmode'	'pm'	keep the oldest version of a file
'path'	'pa'	list of directories searched with "gf" et.al.
'perldll'		name of the Perl dynamic library
'preserveindent'	'pi'	preserve the indent structure when reindenting
'previewheight'	'pvh'	height of the preview window
'previewpopup'	'pvp'	use popup window for preview
'previewwindow'	'pvw'	identifies the preview window
'printdevice'	'pdev'	name of the printer to be used for <code>:hardcopy</code>
'printencoding'	'penc'	encoding to be used for printing
'printexpr'	'pexpr'	expression used to print PostScript for <code>:hardcopy</code>
'printfont'	'pfn'	name of the font to be used for <code>:hardcopy</code>
'printhead'	'pheader'	format of the header used for <code>:hardcopy</code>
'printmbcharset'	'pmbcs'	CJK character set to be used for <code>:hardcopy</code>
'printmbfont'	'pmbfn'	font names to be used for CJK output of <code>:hardcopy</code>
'printoptions'	'popt'	controls the format of <code>:hardcopy</code> output
'prompt'	'prompt'	enable prompt in Ex mode
'pumheight'	'ph'	maximum height of the popup menu
'pumwidth'	'pw'	minimum width of the popup menu
'pythondll'		name of the Python 2 dynamic library
'pythonhome'		name of the Python 2 home directory
'pythonthreedll'		name of the Python 3 dynamic library
'pythonthreehome'		name of the Python 3 home directory
'pyxversion'	'pyx'	Python version used for <code>pyx*</code> commands
'quickfixtextfunc'	'qftf'	function for the text in the quickfix window
'quotescape'	'qe'	escape characters used in a string

'readonly'	'ro'	disallow writing the buffer
'redrawtime'	'rdt'	timeout for 'hlsearch' and :match highlighting
'regengine'	're'	default regexp engine to use
'relativenumber'	'rnu'	show relative line number in front of each line
'remap'		allow mappings to work recursively
'renderoptions'	'rop'	options for text rendering on Windows
'report'		threshold for reporting nr. of lines changed
'restorescreen'	'rs'	Win32: restore screen when exiting
'revins'	'ri'	inserting characters will work backwards
'rightleft'	'rl'	window is right-to-left oriented
'rightleftcmd'	'rlc'	commands for which editing works right-to-left
'rubydll'		name of the Ruby dynamic library
'ruler'	'ru'	show cursor line and column in the status line
'rulerformat'	'ruf'	custom format for the ruler
'runtimepath'	'rtp'	list of directories used for runtime files
'scroll'	'scr'	lines to scroll with CTRL-U and CTRL-D
'scrollbind'	'scb'	scroll in window as other windows scroll
'scrollfocus'	'scf'	scroll wheel applies to window under pointer
'scrolljump'	'sj'	minimum number of lines to scroll
'scrolloff'	'so'	minimum nr. of lines above and below cursor
'scrollopt'	'sbo'	how 'scrollbind' should behave
'sections'	'sect'	nr of macros that separate sections
'secure'		secure mode for reading .vimrc in current dir
'selection'	'sel'	what type of selection to use
'selectmode'	'slm'	when to use Select mode instead of Visual mode
'sessionoptions'	'ssop'	options for :mksession
'shell'	'sh'	name of shell to use for external commands
'shellcmdflag'	'shcf'	flag to shell to execute one command
'shellpipe'	'sp'	string to put output of ":make" in error file
'shellquote'	'shq'	quote character(s) for around shell command
'shellredir'	'srr'	string to put output of filter in a temp file
'shellslash'	'ssl'	use forward slash for shell file names
'shelltemp'	'stmp'	whether to use a temp file for shell commands
'shelltype'	'st'	Amiga: influences how to use a shell
'shellxescape'	'sxe'	characters to escape when 'shellxquote' is (
'shellxquote'	'sxq'	like 'shellquote' , but include redirection
'shiftround'	'sr'	round indent to multiple of shiftwidth
'shiftwidth'	'sw'	number of spaces to use for (auto)indent step
'shortmess'	'shm'	list of flags, reduce length of messages
'shortname'	'sn'	Filenames assumed to be 8.3 chars
'showbreak'	'sbr'	string to use at the start of wrapped lines
'showcmd'	'sc'	show (partial) command somewhere
'showcmdloc'	'sloc'	where to show (partial) command
'showfulltag'	'sft'	show full tag pattern when completing tag
'showmatch'	'sm'	briefly jump to matching bracket if insert one
'showmode'	'smd'	message on status line to show current mode
'showtabline'	'stal'	tells when the tab pages line is displayed
'sidescroll'	'ss'	minimum number of columns to scroll horizontal
'sidescrolloff'	'siso'	min. nr. of columns to left and right of cursor
'signcolumn'	'scl'	when to display the sign column
'smartcase'	'scs'	no ignore case when pattern has uppercase
'smartindent'	'si'	smart autoindenting for C programs
'smarttab'	'sta'	use 'shiftwidth' when inserting <Tab>
'smoothscroll'	'sms'	scroll by screen lines when 'wrap' is set

'softtabstop'	'sts'	number of spaces that <Tab> uses while editing
'spell'		enable spell checking
'spellcapcheck'	'spc'	pattern to locate end of a sentence
'spellfile'	'spf'	files where <code>zg</code> and <code>zw</code> store words
'spelllang'	'spl'	language(s) to do spell checking for
'spelloptions'	'spo'	options for spell checking
'spellsuggest'	'sps'	method(s) used to suggest spelling corrections
'splitbelow'	'sb'	new window from split is below the current one
'splitkeep'	'spk'	determines scroll behavior for split windows
'splitright'	'spr'	new window is put right of the current one
'startofline'	'sol'	commands move cursor to first non-blank in line
'statusline'	'stl'	custom format for the status line
'suffixes'	'su'	suffixes that are ignored with multiple match
'suffixesadd'	'sua'	suffixes added when searching for a file
'swapfile'	'swf'	whether to use a swapfile for a buffer
'swapsync'	'sws'	how to sync the swap file
'switchbuf'	'swb'	sets behavior when switching to another buffer
'synmaxcol'	'smc'	maximum column to find syntax items
'syntax'	'syn'	syntax to be loaded for current buffer
'tabline'	'tal'	custom format for the console tab pages line
'tabpagemax'	'tpm'	maximum number of tab pages for <code>-p</code> and "tab all"
'tabstop'	'ts'	number of spaces that <Tab> in file uses
'tagbsearch'	'tbs'	use binary searching in tags files
'tagcase'	'tc'	how to handle case when searching in tags files
'tagfunc'	'tfu'	function to get list of tag matches
'taglength'	'tl'	number of significant characters for a tag
'tagrelative'	'tr'	file names in tag file are relative
'tags'	'tag'	list of file names used by the tag command
'tagstack'	'tgst'	push tags onto the tag stack
'tcldll'		name of the Tcl dynamic library
'term'		name of the terminal
'termbidi'	'tbidi'	terminal takes care of bi-directionality
'termencoding'	'tenc'	character encoding used by the terminal
'termguicolors'	'tgc'	use GUI colors for the terminal
'termwinkey'	'twk'	key that precedes a Vim command in a terminal
'termwincscroll'	'twsl'	max number of scrollbar lines in a terminal window
'termwinsize'	'twsize'	size of a terminal window
'termwintype'	'twtype'	MS-Windows: type of pty to use for terminal window
'terse'		shorten some messages
'textauto'	'ta'	obsolete, use 'fileformats'
'textmode'	'tx'	obsolete, use 'fileformat'
'textwidth'	'tw'	maximum width of text that is being inserted
'thesaurus'	'tsr'	list of thesaurus files for keyword completion
'thesaurusfunc'	'tsrfu'	function to be used for thesaurus completion
'tildeop'	'top'	tilde command "~" behaves like an operator
'timeout'	'to'	time out on mappings and key codes
'timeoutlen'	'tm'	time out time in milliseconds
'title'		let Vim set the title of the window
'titlelen'		percentage of 'columns' used for window title
'titleold'		old title, restored when exiting
'titlestring'		string to use for the Vim window title
'toolbar'	'tb'	GUI: which items to show in the toolbar
'toolbariconsizes'	'tbis'	size of the toolbar icons (for GTK 2 only)
'ttimeout'		time out on mappings

'ttimeoutlen'	'ttm'	time out time for key codes in milliseconds
'ttybuiltin'	'tbi'	use built-in termcap before external termcap
'ttyfast'	'tf'	indicates a fast terminal connection
'ttymouse'	'ttym'	type of mouse codes generated
'ttyscroll'	'tsl'	maximum number of lines for a scroll
'ttytype'	'tty'	alias for ' term '
'undodir'	'udir'	where to store undo files
'undofile'	'udf'	save undo information in a file
'undolevels'	'ul'	maximum number of changes that can be undone
'undoreload'	'ur'	max nr of lines to save for undo on a buffer reload
'updatecount'	'uc'	after this many characters flush swap file
'updatetime'	'ut'	after this many milliseconds flush swap file
'varsofttabstop'	'vsts'	a list of number of spaces when typing <Tab>
'vartabstop'	'vts'	a list of number of spaces for <Tab>s
'verbose'	'vbs'	give informative messages
'verbosefile'	'vfile'	file to write messages in
'viewdir'	'vdir'	directory where to store files with :mkview
'viewoptions'	'vop'	specifies what to save for :mkview
'viminfo'	'vi'	use .viminfo file upon startup and exiting
'viminfofile'	'vif'	file name used for the viminfo file
'virtualedit'	've'	when to use virtual editing
'visualbell'	'vb'	use visual bell instead of beeping
'warn'		warn for shell command when buffer was changed
'weirdinvert'	'wiv'	for terminals that have weird inversion method
'whichwrap'	'ww'	allow specified keys to cross line boundaries
'wildchar'	'wc'	command-line character for wildcard expansion
'wildcharm'	'wcm'	like ' wildchar ' but also works when mapped
'wildignore'	'wig'	files matching these patterns are not completed
'wildignorecase'	'wic'	ignore case when completing file names
'wildmenu'	'wmnu'	use menu for command line completion
'wildmode'	'wim'	mode for ' wildchar ' command-line expansion
'wildoptions'	'wop'	specifies how command line completion is done
'winaltkeys'	'wak'	when the windows system handles ALT keys
'wincolor'	'wcr'	window-local highlighting
'window'	'wi'	nr of lines to scroll for CTRL-F and CTRL-B
'winfixbuf'	'wfb'	keep window focused on a single buffer
'winfixheight'	'wfh'	keep window height when opening/closing windows
'winfixwidth'	'wfw'	keep window width when opening/closing windows
'winheight'	'wh'	minimum number of lines for the current window
'winminheight'	'wmh'	minimum number of lines for any window
'winminwidth'	'wmw'	minimal number of columns for any window
'winptydll'		name of the winpty dynamic library
'winwidth'	'wiw'	minimal number of columns for current window
'wrap'		long lines wrap and continue on the next line
'wrapmargin'	'wm'	chars from the right where wrapping starts
'wrapscan'	'ws'	searches wrap around the end of the file
'write'		writing to a file is allowed
'writeany'	'wa'	write to file with no need for "!" override
'writebackup'	'wb'	make a backup before overwriting a file
'writedelay'	'wd'	delay this many msec for each char (for debug)
'xtermcodes'		request terminal codes from an xterm

Q_ur

Undo/Redo commands

u	N	u	undo last N changes
CTRL-R	N	CTRL-R	redo last N undone changes
U		U	restore last changed line

Q_et External commands

:shell	:sh[ell]	start a shell
:!	!{command}	execute {command} with a shell
K	K	lookup keyword under the cursor with 'keywordprg' program (default: "man")

Q_qf Quickfix commands

:cc	:cc [nr]	display error [nr] (default is the same again)
:cnext	:cn	display the next error
:cprevious	:cp	display the previous error
:clist	:cl	list all errors
:cfile	:cf	read errors from the file 'errorfile'
:cgetbuffer	:cgetb	like :cbuffer but don't jump to the first error
:cgetfile	:cg	like :cfile but don't jump to the first error
:cgetexpr	:cgete	like :cexpr but don't jump to the first error
:caddfile	:caddf	add errors from the error file to the current quickfix list
:caddexpr	:cad	add errors from an expression to the current quickfix list
:cbuffer	:cb	read errors from text in a buffer
:cexpr	:cex	read errors from an expression
:cquit	:cq	quit without writing and return error code (to the compiler)
:make	:make [args]	start make, read errors, and jump to first error
:grep	:gr[ep] [args]	execute 'grepprg' to find matches and jump to the first one

Q_vc Various commands

CTRL-L	CTRL-L	clear and redraw the screen
CTRL-G	CTRL-G	show current file name (with path) and cursor position
ga	ga	show ascii value of character under cursor in decimal, hex, and octal
g8	g8	for utf-8 encoding: show byte sequence for character under cursor in hex
g_CTRL-G	g CTRL-G	show cursor column, line, and character position
CTRL-C	CTRL-C	during searches: Interrupt the search
dos-CTRL-Break	CTRL-Break	MS-Windows: during searches: Interrupt the search
		while entering a count: delete last character
:version	:ve[rsion]	show version information
:mode	:mode N	set screen mode to N (obsolete)
:normal	:norm[al][!] {commands}	execute Normal mode commands
Q	Q	switch to "Ex" mode

<code>:redir</code>	<code>:redir >{file}</code>	redirect messages to <code>{file}</code>
<code>:silent</code>	<code>:silent[!] {command}</code>	execute <code>{command}</code> silently
<code>:confirm</code>	<code>:confirm {command}</code>	quit, write, etc., asking about unsaved changes or read-only files
<code>:browse</code>	<code>:browse {command}</code>	open/read/write file, using a file selection dialog

Q_ce Command-line editing

<code>c_<Esc></code>	<code><Esc></code>	abandon command-line (if 'wildchar' is <code><Esc></code> , type it twice)
<code>c_CTRL-V</code>	CTRL-V <code>{char}</code>	insert <code>{char}</code> literally
<code>c_CTRL-V</code>	CTRL-V <code>{number}</code>	enter decimal value of character (up to three digits)
<code>c_CTRL-K</code>	CTRL-K <code>{char1}</code> <code>{char2}</code>	enter digraph (See <code>Q_di</code>)
<code>c_CTRL-R</code>	CTRL-R <code>{register}</code>	insert the contents of a register
<code>c_<Left></code>	<code><Left>/<Right></code>	cursor left/right
<code>c_<S-Left></code>	<code><S-Left>/<S-Right></code>	cursor one word left/right
<code>c_CTRL-B</code>	CTRL-B/CTRL-E	cursor to beginning/end of command-line
<code>c_<BS></code>	<code><BS></code>	delete the character in front of the cursor
<code>c_</code>	<code></code>	delete the character under the cursor
<code>c_CTRL-W</code>	CTRL-W	delete the word in front of the cursor
<code>c_CTRL-U</code>	CTRL-U	remove all characters
<code>c_<Up></code>	<code><Up>/<Down></code>	recall older/newer command-line that starts with current command
<code>c_<S-Up></code>	<code><S-Up>/<S-Down></code>	recall older/newer command-line from history
<code>c_CTRL-G</code>	CTRL-G	next match when 'incsearch' is active
<code>c_CTRL-T</code>	CTRL-T	previous match when 'incsearch' is active
<code>:history</code>	<code>:his[tory]</code>	show older command-lines

Context-sensitive completion on the command-line:

<code>c_wildchar</code>	'wildchar' (default: <code><Tab></code>)	do completion on the pattern in front of the cursor; if there are multiple matches, beep and show the first one; further 'wildchar' will show the next ones
<code>c_CTRL-D</code>	CTRL-D	list all names that match the pattern in front of the cursor
<code>c_CTRL-A</code>	CTRL-A	insert all names that match pattern in front of cursor
<code>c_CTRL-L</code>	CTRL-L	insert longest common part of names that match pattern
<code>c_CTRL-N</code>	CTRL-N	after 'wildchar' with multiple matches: go to next match
<code>c_CTRL-P</code>	CTRL-P	after 'wildchar' with multiple matches: go to previous match

Q_ra		Ex ranges
:range	,	separates two line numbers
:range	;	idem, set cursor to the first line number before interpreting the second one
:range	{number}	an absolute line number
:range	.	the current line
:range	\$	the last line in the file
:range	%	equal to 1,\$ (the entire file)
:range	*	equal to '<,>' (visual area)
:range	't	position of mark t
:range	/[pattern]/	the next line where {pattern} matches
:range	?[pattern]?	the previous line where {pattern} matches
:range	+ [num]	add [num] to the preceding line number (default: 1)
:range	- [num]	subtract [num] from the preceding line number (default: 1)

Q_ex Special Ex characters

:bar		separates two commands (not for ":global" and "!!")
:quote	"	begins comment
:_%	%	current file name (only where a file name is expected)
:_#	# [num]	alternate file name [num] (only where a file name is expected)
Note: The next seven are typed literally; these are not special keys!		
:<abuf>	<abuf>	buffer number, for use in an autocommand (only where a file name is expected)
:<afile>	<afile>	file name, for use in an autocommand (only where a file name is expected)
:<amatch>	<amatch>	what matched with the pattern, for use in an autocommand (only where a file name is expected)
:<cword>	<cword>	word under the cursor (only where a file name is expected)
:<cWORD>	<cWORD>	WORD under the cursor (only where a file name is expected) (see WORD)
:<cfile>	<cfile>	file name under the cursor (only where a file name is expected)
:<sfile>	<sfile>	file name of a ":source"d file, within that file (only where a file name is expected)

After "%", "#", "<cfile>", "<sfile>" or "<afile>"

::p	:p	full path
::h	:h	head (file name removed)
::t	:t	tail (file name only)
::r	:r	root (extension removed)
::e	:e	extension
::s	:s/{pat}/{repl}/	substitute {pat} with {repl}

Q_st Starting Vim

-vim	vim [options]	start editing with an empty buffer
-file	vim [options] {file} ..	start editing one or more files
--	vim [options] -	read file from stdin
-tag	vim [options] -t {tag}	edit the file associated with {tag}
-qf	vim [options] -q [fname]	start editing in QuickFix mode, display the first error

Most useful Vim arguments (for full list see [startup-options](#))

-gui	-g	start GUI (also allows other options)
+	+ [num]	put the cursor at line [num] (default: last line)
++c	++{command}	execute {command} after loading the file
++/	++/{pat} {file} ..	put the cursor at the first occurrence of {pat}
-v	-v	Vi mode, start ex in Normal mode
-e	-e	Ex mode, start vim in Ex mode
-R	-R	Read-only mode, implies -n
-m	-m	modifications not allowed (resets 'write' option)
-d	-d	diff mode diff
-b	-b	binary mode
-l	-l	lisp mode
-A	-A	Arabic mode ('arabic' is set)
-F	-F	Farsi mode ('fkmap' and 'rightleft' are set)
-H	-H	Hebrew mode ('hkmap' and 'rightleft' are set)
-V	-V	Verbose, give informative messages
-C	-C	Compatible, set the 'compatible' option
-N	-N	Nocompatible, reset the 'compatible' option
-r	-r	give list of swap files
-r	-r {file} ..	recover aborted edit session
-n	-n	do not create a swap file
-o	-o [num]	open [num] windows (default: one for each file)
-f	-f	GUI: foreground process, don't fork Amiga: do not restart Vim to open a window (for e.g., mail)
-s	-s {scriptin}	first read commands from the file {scriptin}
-w	-w {scriptout}	write typed chars to file {scriptout} (append)
-W	-W {scriptout}	write typed chars to file {scriptout} (overwrite)
-T	-T {terminal}	set terminal name
-d	-d {device}	Amiga: open {device} to be used as a console
-u	-u {vimrc}	read inits from {vimrc} instead of other inits
-U	-U {gvimrc}	idem, for when starting the GUI
-i	-i {viminfo}	read info from {viminfo} instead of other files
---	--	end of options, other arguments are file names
--help	--help	show list of arguments and exit
--version	--version	show version info and exit
--	-	read file from stdin

Q_ed Editing a file

Without !: Fail if changes have been made to the current buffer.

With !: Discard any changes to the current buffer.

:edit_f	:e[dit][!] {file}	edit {file}
:edit	:e[dit][!]	reload the current file
:enew	:ene[w][!]	edit a new, unnamed buffer

<code>:find</code>	<code>:fin[d][!] {file}</code>	find <code>{file}</code> in ' <code>path</code> ' and edit it
<code>CTRL-^</code>	<code>N CTRL-^</code>	edit alternate file N (equivalent to <code>":e #N"</code>)
<code>gf</code>	<code>gf</code> or <code>]f</code>	edit the file whose name is under the cursor
<code>:pwd</code>	<code>:pwd</code>	print the current directory name
<code>:cd</code>	<code>:cd [path]</code>	change the current directory to <code>[path]</code>
<code>:cd-</code>	<code>:cd -</code>	back to previous current directory
<code>:file</code>	<code>:f[file]</code>	print the current file name and the cursor position
<code>:file</code>	<code>:f[file] {name}</code>	set the current file name to <code>{name}</code>
<code>:files</code>	<code>:files</code>	show alternate file names

`Q_fl` Using the argument list `argument-list`

<code>:args</code>	<code>:ar[gs]</code>	print the argument list, with the current file in <code>"[]"</code>
<code>:all</code>	<code>:all</code> or <code>:sall</code>	open a window for every file in the arg list
<code>:wn</code>	<code>:wn[ext][!]</code>	write file and edit next file
<code>:wn</code>	<code>:wn[ext][!] {file}</code>	write to <code>{file}</code> and edit next file, unless <code>{file}</code> exists; With <code>!</code> , overwrite existing file
<code>:wN</code>	<code>:wN[ext][!] [file]</code>	write file and edit previous file

	in current window	in new window	
<code>:argument</code>	<code>:argu[ment] N</code>	<code>:sar[gument] N</code>	edit file N
<code>:next</code>	<code>:n[ext]</code>	<code>:sn[ext]</code>	edit next file
<code>:next_f</code>	<code>:n[ext] {arglist}</code>	<code>:sn[ext] {arglist}</code>	define new arg list and edit first file
<code>:Next</code>	<code>:N[ext]</code>	<code>:sN[ext]</code>	edit previous file
<code>:first</code>	<code>:fir[st]</code>	<code>:sfir[st]</code>	edit first file
<code>:last</code>	<code>:la[st]</code>	<code>:sla[st]</code>	edit last file

`Q_wq` Writing and quitting

<code>:w</code>	<code>:[range]w[rite][!]</code>	write to the current file
<code>:w_f</code>	<code>:[range]w[rite] {file}</code>	write to <code>{file}</code> , unless it already exists
<code>:w_f</code>	<code>:[range]w[rite]! {file}</code>	write to <code>{file}</code> . Overwrite an existing file
<code>:w_a</code>	<code>:[range]w[rite][!] >></code>	append to the current file
<code>:w_a</code>	<code>:[range]w[rite][!] >> {file}</code>	append to <code>{file}</code>
<code>:w_c</code>	<code>:[range]w[rite] !{cmd}</code>	execute <code>{cmd}</code> with <code>[range]</code> lines as standard input
<code>:up</code>	<code>:[range]up[date][!]</code>	write to current file if modified
<code>:wall</code>	<code>:wa[ll][!]</code>	write all changed buffers
<code>:q</code>	<code>:q[uit]</code>	quit current buffer, unless changes have been made; Exit Vim when there are no other non-help buffers
<code>:q</code>	<code>:q[uit]!</code>	quit current buffer always, discard any changes. Exit Vim when there are no other non-help buffers
<code>:qa</code>	<code>:qa[ll]</code>	exit Vim, unless changes have been made
<code>:qa</code>	<code>:qa[ll]!</code>	exit Vim always, discard any changes

<code>:cq</code>	<code>:cq</code>	quit without writing and return error code
<code>:wq</code>	<code>:wq[!]</code>	write the current file and exit
<code>:wq</code>	<code>:wq[!] {file}</code>	write to <code>{file}</code> and exit
<code>:xit</code>	<code>:x[it][!] [file]</code>	like <code>":wq"</code> but write only when changes have been made
<code>ZZ</code>	<code>ZZ</code>	same as <code>":x"</code>
<code>ZQ</code>	<code>ZQ</code>	same as <code>":q!"</code>
<code>:xall</code>	<code>:xa[ll][!] or :wqall[!]</code>	write all changed buffers and exit
<code>:stop</code>	<code>:st[op][!]</code>	suspend Vim or start new shell; if <code>'aw'</code> option is set and <code>[!]</code> not given write the buffer
CTRL-Z	CTRL-Z	same as <code>":stop"</code>

Q_ac Automatic Commands

`viminfo-file` read registers, marks, history at startup, save when exiting.

<code>:rviminfo</code>	<code>:rv[iminfo] [file]</code>	read info from viminfo file <code>[file]</code>
<code>:rviminfo</code>	<code>:rv[iminfo]! [file]</code>	idem, overwrite existing info
<code>:wviminfo</code>	<code>:wv[iminfo] [file]</code>	add info to viminfo file <code>[file]</code>
<code>:wviminfo</code>	<code>:wv[iminfo]! [file]</code>	write info to viminfo file <code>[file]</code>

`modeline` Automatic option setting when editing a file

`modeline` `vim:{set-arg}: ..` In the first and last lines of the file (see `'ml'` option), `{set-arg}` is given as an argument to `":set"`

`autocommand` Automatic execution of commands on certain events.

<code>:autocmd</code>	<code>:au</code>	list all autocommands
<code>:autocmd</code>	<code>:au {event}</code>	list all autocommands for <code>{event}</code>
<code>:autocmd</code>	<code>:au {event} {pat}</code>	list all autocommands for <code>{event}</code> with <code>{pat}</code>
<code>:autocmd</code>	<code>:au {event} {pat} {cmd}</code>	enter new autocommands for <code>{event}</code> with <code>{pat}</code>
<code>:autocmd</code>	<code>:au!</code>	remove all autocommands
<code>:autocmd</code>	<code>:au! {event}</code>	remove all autocommands for <code>{event}</code>
<code>:autocmd</code>	<code>:au! * {pat}</code>	remove all autocommands for <code>{pat}</code>
<code>:autocmd</code>	<code>:au! {event} {pat}</code>	remove all autocommands for <code>{event}</code> with <code>{pat}</code>
<code>:autocmd</code>	<code>:au! {event} {pat} {cmd}</code>	remove all autocommands for <code>{event}</code> with <code>{pat}</code> and enter new one

Q_wi Multi-window commands

CTRL-W_s	CTRL-W s or <code>:split</code>	split window into two parts
<code>:split_f</code>	<code>:split {file}</code>	split window and edit <code>{file}</code> in one of them

<code>:vsplit</code>	<code>:vsplit {file}</code>	same, but split vertically
<code>:vertical</code>	<code>:vertical {cmd}</code>	make <code>{cmd}</code> split vertically

<code>:sfind</code>	<code>:sf[ind] {file}</code>	split window, find <code>{file}</code> in <code>'path'</code> and edit it
<code>:terminal</code>	<code>:terminal {cmd}</code>	open a terminal window
<code>CTRL-W_]</code>	CTRL-W]	split window and jump to tag under cursor
<code>CTRL-W_f</code>	CTRL-W f	split window and edit file name under the cursor
<code>CTRL-W_^</code>	CTRL-W ^	split window and edit alternate file
<code>CTRL-W_n</code>	CTRL-W n or <code>:new</code>	create new empty window
<code>CTRL-W_q</code>	CTRL-W q or <code>:q[uit]</code>	quit editing and close window
<code>CTRL-W_c</code>	CTRL-W c or <code>:clo[se]</code>	make buffer hidden and close window
<code>CTRL-W_o</code>	CTRL-W o or <code>:on[ly]</code>	make current window only one on the screen
<code>CTRL-W_j</code>	CTRL-W j	move cursor to window below
<code>CTRL-W_k</code>	CTRL-W k	move cursor to window above
<code>CTRL-W_CTRL-W</code>	CTRL-W CTRL-W	move cursor to window below (wrap)
<code>CTRL-W_W</code>	CTRL-W W	move cursor to window above (wrap)
<code>CTRL-W_t</code>	CTRL-W t	move cursor to top window
<code>CTRL-W_b</code>	CTRL-W b	move cursor to bottom window
<code>CTRL-W_p</code>	CTRL-W p	move cursor to previous active window
<code>CTRL-W_r</code>	CTRL-W r	rotate windows downwards
<code>CTRL-W_R</code>	CTRL-W R	rotate windows upwards
<code>CTRL-W_x</code>	CTRL-W x	exchange current window with next one
<code>CTRL-W_= CTRL-W_- CTRL-W_+ CTRL-W_--</code>	CTRL-W = CTRL-W - CTRL-W + CTRL-W _	make all windows equal height & width decrease current window height increase current window height set current window height (default: very high)
<code>CTRL-W_< CTRL-W_> CTRL-W_bar</code>	CTRL-W < CTRL-W > CTRL-W 	decrease current window width increase current window width set current window width (default: widest possible)

Q_bu Buffer list commands

<code>:buffers</code>	<code>:buffers</code> or <code>:files</code>	list all known buffer and file names
<code>:ball</code>	<code>:ball</code> or <code>:sball</code>	edit all args/buffers
<code>:unhide</code>	<code>:unhide</code> or <code>:sunhide</code>	edit all loaded buffers
<code>:badd</code>	<code>:badd {fname}</code>	add file name <code>{fname}</code> to the list
<code>:bunload</code>	<code>:bunload[!] [N]</code>	unload buffer [N] from memory
<code>:bdelete</code>	<code>:bdelete[!] [N]</code>	unload buffer [N] and delete it from the buffer list
	in current window	in new window
<code>:buffer</code>	<code>:[N]buffer [N]</code>	<code>:[N]sbuffer [N]</code> to arg/buf N
<code>:bnext</code>	<code>:[N]bnext [N]</code>	<code>:[N]sbnext [N]</code> to Nth next arg/buf
<code>:bNext</code>	<code>:[N]bNext [N]</code>	<code>:[N]sbNext [N]</code> to Nth previous arg/buf
<code>:bprevious</code>	<code>:[N]bprevious [N]</code>	<code>:[N]sbprevious [N]</code> to Nth previous arg/buf

<code>:bfirst</code>	<code>:bfirst</code>	<code>:sbfirst</code>	to first arg/buf
<code>:blast</code>	<code>:blast</code>	<code>:sblast</code>	to last arg/buf
<code>:bmodified</code>	<code>:[N]bmod [N]</code>	<code>:[N]sbmod [N]</code>	to Nth modified buf

Q_sy Syntax Highlighting

<code>:syn-on</code>	<code>:syntax on</code>	start using syntax highlighting
<code>:syn-off</code>	<code>:syntax off</code>	stop using syntax highlighting
<code>:syn-keyword</code>	<code>:syntax keyword {group-name} {keyword} ..</code>	add a syntax keyword item
<code>:syn-match</code>	<code>:syntax match {group-name} {pattern} ...</code>	add syntax match item
<code>:syn-region</code>	<code>:syntax region {group-name} {pattern} ...</code>	add syntax region item
<code>:syn-sync</code>	<code>:syntax sync [ccomment lines {N} ...]</code>	tell syntax how to sync
<code>:syntax</code>	<code>:syntax [list]</code>	list current syntax items
<code>:syn-clear</code>	<code>:syntax clear</code>	clear all syntax info
<code>:highlight</code>	<code>:highlight clear</code>	clear all highlight info
<code>:highlight</code>	<code>:highlight {group-name} {key}={arg} ..</code>	set highlighting for {group-name}
<code>:filetype</code>	<code>:filetype on</code>	switch on file type detection, without syntax highlighting
<code>:filetype</code>	<code>:filetype plugin indent on</code>	switch on file type detection, with automatic indenting and settings

Q_gu GUI commands

<code>:gui</code>	<code>:gui</code>	UNIX: start the GUI
<code>:gui</code>	<code>:gui {fname} ..</code>	idem, and edit {fname} ..
<code>:menu</code>	<code>:menu</code>	list all menus
<code>:menu</code>	<code>:menu {mpath}</code>	list menus starting with {mpath}
<code>:menu</code>	<code>:menu {mpath} {rhs}</code>	add menu {mpath}, giving {rhs}
<code>:menu</code>	<code>:menu {pri} {mpath} {rhs}</code>	idem, with priorities {pri}
<code>:menu</code>	<code>:menu ToolBar.{name} {rhs}</code>	add toolbar item, giving {rhs}
<code>:tmenu</code>	<code>:tmenu {mpath} {text}</code>	add tooltip to menu {mpath}
<code>:unmenu</code>	<code>:unmenu {mpath}</code>	remove menu {mpath}

Q_fo Folding

<code>'foldmethod'</code>	<code>set foldmethod=manual</code>	manual folding
	<code>set foldmethod=indent</code>	folding by indent
	<code>set foldmethod=expr</code>	folding by 'foldexpr'
	<code>set foldmethod=syntax</code>	folding by syntax regions
	<code>set foldmethod=marker</code>	folding by 'foldmarker'
<code>zf</code>	<code>zf{motion}</code>	operator: Define a fold manually

<code>:fold</code>	<code>:{range}fold</code>	define a fold for <code>{range}</code> lines
<code>zd</code>	<code>zd</code>	delete one fold under the cursor
<code>zD</code>	<code>zD</code>	delete all folds under the cursor
<code>zo</code>	<code>zo</code>	open one fold under the cursor
<code>zO</code>	<code>zO</code>	open all folds under the cursor
<code>zc</code>	<code>zc</code>	close one fold under the cursor
<code>zC</code>	<code>zC</code>	close all folds under the cursor
<code>zm</code>	<code>zm</code>	fold more: decrease 'foldlevel'
<code>zM</code>	<code>zM</code>	close all folds: make 'foldlevel' zero
<code>zr</code>	<code>zr</code>	reduce folding: increase 'foldlevel'
<code>zR</code>	<code>zR</code>	open all folds: make 'foldlevel' max.
<code>zn</code>	<code>zn</code>	fold none: reset 'foldenable'
<code>zN</code>	<code>zN</code>	fold normal set 'foldenable'
<code>zi</code>	<code>zi</code>	invert 'foldenable'

Overview

Getting Started

usr_01.txt	About the manuals
usr_02.txt	The first steps in Vim
usr_03.txt	Moving around
usr_04.txt	Making small changes
usr_05.txt	Set your settings
usr_06.txt	Using syntax highlighting
usr_07.txt	Editing more than one file
usr_08.txt	Splitting windows
usr_09.txt	Using the GUI
usr_10.txt	Making big changes
usr_11.txt	Recovering from a crash
usr_12.txt	Clever tricks

Editing Effectively

usr_20.txt	Typing command-line commands quickly
usr_21.txt	Go away and come back
usr_22.txt	Finding the file to edit
usr_23.txt	Editing other files
usr_24.txt	Inserting quickly
usr_25.txt	Editing formatted text
usr_26.txt	Repeating
usr_27.txt	Search commands and patterns
usr_28.txt	Folding
usr_29.txt	Moving through programs
usr_30.txt	Editing programs
usr_31.txt	Exploiting the GUI
usr_32.txt	The undo tree

Tuning Vim

usr_40.txt	Make new commands
usr_41.txt	Write a Vim script
usr_42.txt	Add new menus
usr_43.txt	Using filetypes
usr_44.txt	Your own syntax highlighted
usr_45.txt	Select your language (locale)

Writing Vim script

usr_50.txt	Advanced Vim script writing
usr_51.txt	Write plugins
usr_52.txt	Write larger plugins

Making Vim Run

usr_90.txt	Installing Vim
------------	----------------

Reference manual

[reference_toc](#) More detailed information for all commands

The user manual (an older version) is available as a single, ready to print HTML and PDF file here:

<http://vimdoc.sf.net>

Getting Started

Read this from start to end to learn the essential commands.

usr_01.txt	About the manuals
	01.1 Two manuals
	01.2 Vim installed
	01.3 Using the Vim tutor
	01.4 Copyright
usr_02.txt	The first steps in Vim
	02.1 Running Vim for the First Time
	02.2 Inserting text
	02.3 Moving around
	02.4 Deleting characters
	02.5 Undo and Redo
	02.6 Other editing commands
	02.7 Getting out
	02.8 Finding help
usr_03.txt	Moving around
	03.1 Word movement
	03.2 Moving to the start or end of a line
	03.3 Moving to a character
	03.4 Matching a paren
	03.5 Moving to a specific line
	03.6 Telling where you are
	03.7 Scrolling around
	03.8 Simple searches
	03.9 Simple search patterns
	03.10 Using marks
usr_04.txt	Making small changes
	04.1 Operators and motions
	04.2 Changing text
	04.3 Repeating a change
	04.4 Visual mode
	04.5 Moving text
	04.6 Copying text
	04.7 Using the clipboard
	04.8 Text objects
	04.9 Replace mode
	04.10 Conclusion
usr_05.txt	Set your settings

	05.1	The vimrc file
	05.2	The example vimrc file explained
	05.3	The defaults.vim file explained
	05.4	Simple mappings
	05.5	Adding a package
	05.6	Adding a plugin
	05.7	Adding a help file
	05.8	The option window
	05.9	Often used options
usr_06.txt		Using syntax highlighting
	06.1	Switching it on
	06.2	No or wrong colors?
	06.3	Different colors
	06.4	With colors or without colors
	06.5	Printing with colors
	06.6	Further reading
usr_07.txt		Editing more than one file
	07.1	Edit another file
	07.2	A list of files
	07.3	Jumping from file to file
	07.4	Backup files
	07.5	Copy text between files
	07.6	Viewing a file
	07.7	Changing the file name
usr_08.txt		Splitting windows
	08.1	Split a window
	08.2	Split a window on another file
	08.3	Window size
	08.4	Vertical splits
	08.5	Moving windows
	08.6	Commands for all windows
	08.7	Viewing differences with vimdiff
	08.8	Various
usr_09.txt		Using the GUI
	09.1	Parts of the GUI
	09.2	Using the mouse
	09.3	The clipboard
	09.4	Select mode
usr_10.txt		Making big changes
	10.1	Record and playback commands
	10.2	Substitution
	10.3	Command ranges
	10.4	The global command
	10.5	Visual block mode
	10.6	Reading and writing part of a file
	10.7	Formatting text
	10.8	Changing case
	10.9	Using an external program

usr_11.txt	Recovering from a crash
	11.1 Basic recovery
	11.2 Where is the swap file?
	11.3 Crashed or not?
	11.4 Further reading
usr_12.txt	Clever tricks
	12.1 Replace a word
	12.2 Change "Last, First" to "First Last"
	12.3 Sort a list
	12.4 Reverse line order
	12.5 Count words
	12.6 Find a man page
	12.7 Trim blanks
	12.8 Find where a word is used

Editing Effectively

Subjects that can be read independently.

usr_20.txt	Typing command-line commands quickly
	20.1 Command line editing
	20.2 Command line abbreviations
	20.3 Command line completion
	20.4 Command line history
	20.5 Command line window
usr_21.txt	Go away and come back
	21.1 Suspend and resume
	21.2 Executing shell commands
	21.3 Remembering information; viminfo
	21.4 Sessions
	21.5 Views
	21.6 Modelines
usr_22.txt	Finding the file to edit
	22.1 The file explorer
	22.2 The current directory
	22.3 Finding a file
	22.4 The buffer list
usr_23.txt	Editing other files
	23.1 DOS, Mac and Unix files
	23.2 Files on the internet
	23.3 Encryption
	23.4 Binary files
	23.5 Compressed files
usr_24.txt	Inserting quickly
	24.1 Making corrections
	24.2 Showing matches
	24.3 Completion
	24.4 Repeating an insert

	24.5	Copying from another line
	24.6	Inserting a register
	24.7	Abbreviations
	24.8	Entering special characters
	24.9	Digraphs
	24.10	Normal mode commands
usr_25.txt		Editing formatted text
	25.1	Breaking lines
	25.2	Aligning text
	25.3	Indents and tabs
	25.4	Dealing with long lines
	25.5	Editing tables
usr_26.txt		Repeating
	26.1	Repeating with Visual mode
	26.2	Add and subtract
	26.3	Making a change in many files
	26.4	Using Vim from a shell script
usr_27.txt		Search commands and patterns
	27.1	Ignoring case
	27.2	Wrapping around the file end
	27.3	Offsets
	27.4	Matching multiple times
	27.5	Alternatives
	27.6	Character ranges
	27.7	Character classes
	27.8	Matching a line break
	27.9	Examples
usr_28.txt		Folding
	28.1	What is folding?
	28.2	Manual folding
	28.3	Working with folds
	28.4	Saving and restoring folds
	28.5	Folding by indent
	28.6	Folding with markers
	28.7	Folding by syntax
	28.8	Folding by expression
	28.9	Folding unchanged lines
	28.10	Which fold method to use?
usr_29.txt		Moving through programs
	29.1	Using tags
	29.2	The preview window
	29.3	Moving through a program
	29.4	Finding global identifiers
	29.5	Finding local identifiers
usr_30.txt		Editing programs
	30.1	Compiling
	30.2	Indenting C files
	30.3	Automatic indenting

	30.4	Other indenting
	30.5	Tabs and spaces
	30.6	Formatting comments
usr_31.txt		Exploiting the GUI
	31.1	The file browser
	31.2	Confirmation
	31.3	Menu shortcuts
	31.4	Vim window position and size
	31.5	Various
usr_32.txt		The undo tree
	32.1	Undo up to a file write
	32.2	Numbering changes
	32.3	Jumping around the tree
	32.4	Time travelling

Tuning Vim

Make Vim work as you like it.

usr_40.txt		Make new commands
	40.1	Key mapping
	40.2	Defining command-line commands
	40.3	Autocommands
usr_41.txt		Write a Vim script
	41.1	Introduction
	41.2	Variables
	41.3	Expressions
	41.4	Conditionals
	41.5	Executing an expression
	41.6	Using functions
	41.7	Defining a function
	41.8	Lists and Dictionaries
	41.9	White space
	41.10	Line continuation
	41.11	Comments
	41.12	Fileformat
usr_42.txt		Add new menus
	42.1	Introduction
	42.2	Menu commands
	42.3	Various
	42.4	Toolbar and popup menus
usr_43.txt		Using filetypes
	43.1	Plugins for a filetype
	43.2	Adding a filetype
usr_44.txt		Your own syntax highlighted
	44.1	Basic syntax commands
	44.2	Keywords

- 44.3 Matches
- 44.4 Regions
- 44.5 Nested items
- 44.6 Following groups
- 44.7 Other arguments
- 44.8 Clusters
- 44.9 Including another syntax file
- 44.10 Synchronizing
- 44.11 Installing a syntax file
- 44.12 Portable syntax file layout

- usr_45.txt Select your language (locale)
 - 45.1 Language for Messages
 - 45.2 Language for Menus
 - 45.3 Using another encoding
 - 45.4 Editing files with a different encoding
 - 45.5 Entering language text

Writing Vim script

- usr_50.txt Advanced Vim script writing
 - 50.1 Exceptions
 - 50.2 Function with variable number of arguments
 - 50.3 Restoring the view
- usr_51.txt Write plugins
 - 51.1 Writing a generic plugin
 - 51.2 Writing a filetype plugin
 - 51.3 Writing a compiler plugin
 - 51.4 Distributing Vim scripts
- usr_52.txt Write larger plugins
 - 52.1 Export and import
 - 52.2 Autoloading
 - 52.3 Autoloading without import/export
 - 52.4 Other mechanisms to use
 - 52.5 Using a Vim9 script from legacy script

Making Vim Run

Before you can use Vim.

- usr_90.txt Installing Vim
 - 90.1 Unix
 - 90.2 MS-Windows
 - 90.3 Upgrading
 - 90.4 Common installation issues
 - 90.5 Uninstalling Vim

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

`usr_01.txt` For `Vim version 9.1`. Last change: 2023 May 12

VIM USER MANUAL - by Bram Moolenaar

About the manuals

This chapter introduces the manuals available with Vim. Read this to know the conditions under which the commands are explained.

- `01.1` Two manuals
- `01.2` Vim installed
- `01.3` Using the Vim tutor
- `01.4` Copyright

Next chapter: `usr_02.txt` The first steps in Vim
Table of contents: `usr_toc.txt`

`01.1` Two manuals

The Vim documentation consists of two parts:

1. The User manual
Task oriented explanations, from simple to complex. Reads from start to end like a book.
2. The Reference manual
Precise description of how everything in Vim works.

The notation used in these manuals is explained here: `notation`

JUMPING AROUND

The text contains hyperlinks between the two parts, allowing you to quickly jump between the description of an editing task and a precise explanation of the commands and options used for it. Use these two commands:

Press **CTRL-]** to jump to a subject under the cursor.
Press **CTRL-O** to jump back (repeat to go further back).

Many links are in vertical bars, like this: `bars` . The bars themselves may be hidden or invisible; see below. An option name, like `'number'`, a command in double quotes like `":write"` and any other word can also be used as a link. Try it out: Move the cursor to **CTRL-]** and press **CTRL-]** on it.

Other subjects can be found with the `":help"` command; see `help.txt` .

The bars and stars are usually hidden with the `conceal` feature. They also use `hl-Ignore` , using the same color for the text as the background. You can make them visible with:

```
:set conceallevel=0  
:hi link HelpBar Normal
```

`:hi link HelpStar Normal`

01.2 Vim installed

Most of the manuals assume that Vim has been properly installed. If you didn't do that yet, or if Vim doesn't run properly (e.g., files can't be found or in the GUI the menus do not show up) first read the chapter on installation: [usr_90.txt](#) .

not-compatible

The manuals often assume you are using Vim with Vi-compatibility switched off. For most commands this doesn't matter, but sometimes it is important, e.g., for multi-level undo. An easy way to make sure you are using a nice setup is to copy the example vimrc file. By doing this inside Vim you don't have to check out where it is located. How to do this depends on the system you are using:

Unix:

```
:!cp -i $VIMRUNTIME/vimrc_example.vim ~/.vimrc
```

MS-Windows:

```
:!copy $VIMRUNTIME/vimrc_example.vim $VIM/_vimrc
```

Amiga:

```
:!copy $VIMRUNTIME/vimrc_example.vim $VIM/.vimrc
```

If the file already exists you probably want to keep it.

If you start Vim now, the **'compatible'** option should be off. You can check it with this command:

```
:set compatible?
```

If it responds with "nocompatible" you are doing well. If the response is "compatible" you are in trouble. You will have to find out why the option is still set. Perhaps the file you wrote above is not found. Use this command to find out:

```
:scriptnames
```

If your file is not in the list, check its location and name. If it is in the list, there must be some other place where the **'compatible'** option is switched back on.

For more info see [vimrc](#) and [compatible-default](#) .

Note:

This manual is about using Vim in the normal way. There is an alternative called "evim" (easy Vim). This is still Vim, but used in a way that resembles a click-and-type editor like Notepad. It always stays in Insert mode, thus it feels very different. It is not explained in the user manual, since it should be mostly self-explanatory. See [evim-keys](#) for details.

01.3 Using the Vim tutor

[tutor](#) [vimtutor](#)

Instead of reading the text (boring!) you can use the vimtutor to learn your first Vim commands. This is a 30-minute tutorial that teaches the most basic Vim functionality hands-on.

On Unix, if Vim has been properly installed, you can start it from the shell:

```
vimtutor
```

On MS-Windows you can find it in the Program/Vim menu. Or execute vimtutor.bat in the \$VIMRUNTIME directory.

This will make a copy of the tutor file, so that you can edit it without the risk of damaging the original.

There are a few translated versions of the tutor. To find out if yours is available, use the two-letter language code. For French:

```
vimtutor fr
```

On Unix, if you prefer using the GUI version of Vim, use "gvimtutor" or "vimtutor -g" instead of "vimtutor".

For OpenVMS, if Vim has been properly installed, you can start vimtutor from a VMS prompt with:

```
@VIM:vimtutor
```

Optionally add the two-letter language code as above.

On other systems, you have to do a little work:

1. Copy the tutor file. You can do this with Vim (it knows where to find it):

```
vim --clean -c 'e $VIMRUNTIME/tutor/tutor' -c 'w! TUTORCOPY' -c 'q'
```

This will write the file "TUTORCOPY" in the current directory. To use a translated version of the tutor, append the two-letter language code to the filename. For French:

```
vim --clean -c 'e $VIMRUNTIME/tutor/tutor.fr' -c 'w! TUTORCOPY' -c 'q'
```

2. Edit the copied file with Vim:

```
vim --clean TUTORCOPY
```

The --clean argument makes sure Vim is started with nice defaults.

3. Delete the copied file when you are finished with it:

```
del TUTORCOPY
```

The Vim user manual and reference manual are Copyright (c) 1988 by Bram Moolenaar. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later. The latest version is presently available at:

<http://www.opencontent.org/openpub/>

People who contribute to the manuals must agree with the above copyright notice.

[frombook](#)

Parts of the user manual come from the book "Vi IMproved - Vim" by Steve Oualline (published by New Riders Publishing, ISBN: 0735710015). The Open Publication License applies to this book. Only selected parts are included and these have been modified (e.g., by removing the pictures, updating the text for Vim 6.0 and later, fixing mistakes). The omission of the [frombook](#) tag does not mean that the text does not come from the book.

Many thanks to Steve Oualline and New Riders for creating this book and publishing it under the OPL! It has been a great help while writing the user manual. Not only by providing literal text, but also by setting the tone and style.

If you make money through selling the manuals, you are strongly encouraged to donate part of the profit to help AIDS victims in Uganda. See [iccf](#) .

=====

Next chapter: [usr_02.txt](#) The first steps in Vim

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_02.txt For Vim version 9.1. Last change: 2021 Apr 24

VIM USER MANUAL - by Bram Moolenaar

The first steps in Vim

This chapter provides just enough information to edit a file with Vim. Not well or fast, but you can edit. Take some time to practice with these commands, they form the base for what follows.

- 02.1 Running Vim for the First Time
- 02.2 Inserting text
- 02.3 Moving around
- 02.4 Deleting characters
- 02.5 Undo and Redo
- 02.6 Other editing commands
- 02.7 Getting out
- 02.8 Finding help

Next chapter: [usr_03.txt](#) Moving around
Previous chapter: [usr_01.txt](#) About the manuals
Table of contents: [usr_toc.txt](#)

02.1 Running Vim for the First Time

To start Vim, enter this command:

```
gvim file.txt
```

In UNIX you can type this at any command prompt. If you are running Microsoft Windows, open a Command Prompt and enter the command.

In either case, Vim starts editing a file called file.txt. Because this is a new file, you get a blank window. This is what your screen will look like:

```
+-----+
|#
|~
|~
|~
|~
|"file.txt" [New file]
+-----+
```

('#' is the cursor position.)

The tilde (~) lines indicate lines not in the file. In other words, when Vim runs out of file to display, it displays tilde lines. At the bottom of the screen, a message line indicates the file is named file.txt and shows that you are creating a new file. The message information is temporary and other information overwrites it.

THE VIM COMMAND

The `gvim` command causes the editor to create a new window for editing. If you use this command:

```
vim file.txt
```

the editing occurs inside your command window. In other words, if you are running inside an xterm, the editor uses your xterm window. If you are using an MS-Windows command prompt window, the editing occurs inside this window. The text in the window will look the same for both versions, but with `gvim` you have extra features, like a menu bar. More about that later.

02.2 Inserting text

The Vim editor is a modal editor. That means that the editor behaves differently, depending on which mode you are in. The two basic modes are called Normal mode and Insert mode. In Normal mode the characters you type are commands. In Insert mode the characters are inserted as text.

Since you have just started Vim it will be in Normal mode. To start Insert mode you type the "i" command (i for Insert). Then you can enter the text. It will be inserted into the file. Do not worry if you make mistakes; you can correct them later. To enter the following programmer's limerick, this is what you type:

```
iA very intelligent turtle  
Found programming UNIX a hurdle
```

After typing "turtle" you press the `<Enter>` key to start a new line. Finally you press the `<Esc>` key to stop Insert mode and go back to Normal mode. You now have two lines of text in your Vim window:

```
+-----+  
|A very intelligent turtle|  
|Found programming UNIX a hurdle|  
|~|  
|~|  
| |  
+-----+
```

WHAT IS THE MODE?

To be able to see what mode you are in, type this command:

```
:set showmode
```

You will notice that when typing the colon Vim moves the cursor to the last line of the window. That's where you type colon commands (commands that start with a colon). Finish this command by pressing the `<Enter>` key (all commands that start with a colon are finished this way).

Now, if you type the "i" command Vim will display `--INSERT--` at the bottom of the window. This indicates you are in Insert mode.

```
+-----+
|A very intelligent turtle      |
|Found programming UNIX a hurdle|
|~                              |
|~                              |
|-- INSERT --                  |
+-----+
```

If you press <Esc> to go back to Normal mode the last line will be made blank.

GETTING OUT OF TROUBLE

One of the problems for Vim novices is mode confusion, which is caused by forgetting which mode you are in or by accidentally typing a command that switches modes. To get back to Normal mode, no matter what mode you are in, press the <Esc> key. Sometimes you have to press it twice. If Vim beeps back at you, you already are in Normal mode.

02.3 Moving around

After you return to Normal mode, you can move around by using these keys:

```
h   left                               hjkl
j   down
k   up
l   right
```

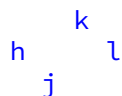
At first, it may appear that these commands were chosen at random. After all, who ever heard of using l for right? But actually, there is a very good reason for these choices: Moving the cursor is the most common thing you do in an editor, and these keys are on the home row of your right hand. In other words, these commands are placed where you can type them the fastest (especially when you type with ten fingers).

Note:

You can also move the cursor by using the arrow keys. If you do, however, you greatly slow down your editing because to press the arrow keys, you must move your hand from the text keys to the arrow keys. Considering that you might be doing it hundreds of times an hour, this can take a significant amount of time.

Also, there are keyboards which do not have arrow keys, or which locate them in unusual places; therefore, knowing the use of the hjkl keys helps in those situations.

One way to remember these commands is that h is on the left, l is on the right and j points down. In a picture:



The best way to learn these commands is by using them. Use the "i" command to insert some more lines of text. Then use the hjkl keys to move around and insert a word somewhere. Don't forget to press <Esc> to go back to Normal mode. The [vimtutor](#) is also a nice way to learn by doing.

For Japanese users, Hiroshi Iwatani suggested using this:

```

                Komsomolsk
                  ^
                  |
Huan Ho  <----->  Los Angeles
(Yellow river)
                  |
                  v
                Java (the island, not the programming language)

```

02.4 Deleting characters

To delete a character, move the cursor over it and type "x". (This is a throwback to the old days of the typewriter, when you deleted things by typing xxxx over them.) Move the cursor to the beginning of the first line, for example, and type xxxxxx (seven x's) to delete "A very ". The result should look like this:

```
+-----+
|intelligent turtle
|Found programming UNIX a hurdle
|~
|~
|
+-----+
```

Now you can insert new text, for example by typing:

```
iA young <Esc>
```

This begins an insert (the i), inserts the words "A young", and then exits insert mode (the final <Esc>). The result:

```
+-----+
|A young intelligent turtle
|Found programming UNIX a hurdle
|~
|~
|
+-----+
```

DELETING A LINE

To delete a whole line use the "dd" command. The following line will then move up to fill the gap:

```
+-----+
```

```
|Found programming UNIX a hurdle |
|~                               |
|~                               |
|~                               |
+-----+

```

DELETING A LINE BREAK

In Vim you can join two lines together, which means that the line break between them is deleted. The "J" command does this.

Take these two lines:

```
A young intelligent
turtle
```

Move the cursor to the first line and press "J":

```
A young intelligent turtle
```

02.5 Undo and Redo

Suppose you delete too much. Well, you can type it in again, but an easier way exists. The "u" command undoes the last edit. Take a look at this in action: After using "dd" to delete the first line, "u" brings it back.

Another one: Move the cursor to the A in the first line:

```
A young intelligent turtle
```

Now type xxxxxx to delete "A young". The result is as follows:

```
intelligent turtle
```

Type "u" to undo the last delete. That delete removed the g, so the undo restores the character.

```
g intelligent turtle
```

The next "u" command restores the next-to-last character deleted:

```
ng intelligent turtle
```

The next "u" command gives you the u, and so on:

```
ung intelligent turtle
oung intelligent turtle
young intelligent turtle
 young intelligent turtle
A young intelligent turtle
```

Note:

If you type "u" twice, and the result is that you get the same text

back, you have Vim configured to work Vi compatible. Look here to fix this: [not-compatible](#) .

This text assumes you work "The Vim Way". You might prefer to use the good old Vi way, but you will have to watch out for small differences in the text then.

REDO

If you undo too many times, you can press **CTRL-R** (redo) to reverse the preceding command. In other words, it undoes the undo. To see this in action, press **CTRL-R** twice. The character A and the space after it disappear:

```
young intelligent turtle
```

There's a special version of the undo command, the "U" (undo line) command. The undo line command undoes all the changes made on the last line that was edited. Typing this command twice cancels the preceding "U".

```
A very intelligent turtle      Delete very
  xxxx
A intelligent turtle          Delete turtle
          xxxxxx
A intelligent                  Restore line with "U"
A very intelligent turtle      Undo "U" with "u"
A intelligent
```

The "U" command is a change by itself, which the "u" command undoes and **CTRL-R** redoes. This might be a bit confusing. Don't worry, with "u" and **CTRL-R** you can go to any of the situations you had. More about that in section [32.2](#) .

02.6 Other editing commands

Vim has a large number of commands to change the text. See [Q_in](#) and below. Here are a few often used ones.

APPENDING

The "i" command inserts a character before the character under the cursor. That works fine; but what happens if you want to add stuff to the end of the line? For that you need to insert text after the cursor. This is done with the "a" (append) command.

For example, to change the line

```
and that's not saying much for the turtle.
to
and that's not saying much for the turtle!!!
```


move the cursor over to the dot at the end of the line. Then type "x" to delete the period. The cursor is now positioned at the end of the line on the e in turtle. Now type

```
a!!!<Esc>
```

to append three exclamation points after the e in turtle:

```
and that's not saying much for the turtle!!!
```

OPENING UP A NEW LINE

The "o" command creates a new, empty line below the cursor and puts Vim in Insert mode. Then you can type the text for the new line.

Suppose the cursor is somewhere in the first of these two lines:

```
A very intelligent turtle  
Found programming UNIX a hurdle
```

If you now use the "o" command and type new text:

```
oThat liked using Vim<Esc>
```

The result is:

```
A very intelligent turtle  
That liked using Vim  
Found programming UNIX a hurdle
```

The "O" command (uppercase) opens a line above the cursor.

USING A COUNT

Suppose you want to move up nine lines. You can type "kkkkkkkkk" or you can enter the command "9k". In fact, you can precede many commands with a number. Earlier in this chapter, for instance, you added three exclamation points to the end of a line by typing "a!!!<Esc>". Another way to do this is to use the command "3a!<Esc>". The count of 3 tells the command that follows to triple its effect. Similarly, to delete three characters, use the command "3x". The count always comes before the command it applies to.

=====

02.7 Getting out

To exit, use the "ZZ" command. This command writes the file and exits.

Note:

Unlike many other editors, Vim does not automatically make a backup file. If you type "ZZ", your changes are committed and there's no turning back. You can configure the Vim editor to produce backup files; see [07.4](#) .

DISCARDING CHANGES

Sometimes you will make a sequence of changes and suddenly realize you were better off before you started. Not to worry; Vim has a quit-and-throw-things-away command. It is:

```
:q!
```

Don't forget to press `<Enter>` to finish the command.

For those of you interested in the details, the three parts of this command are the colon (:), which enters Command-line mode; the q command, which tells the editor to quit; and the override command modifier (!).

The override command modifier is needed because Vim is reluctant to throw away changes. If you were to just type ":q", Vim would display an error message and refuse to exit:

```
E37: No write since last change (use ! to override)
```

By specifying the override, you are in effect telling Vim, "I know that what I'm doing looks stupid, but I really want to do this."

If you want to continue editing with Vim: The ":e!" command reloads the original version of the file.

02.8 Finding help

Everything you always wanted to know can be found in the Vim help files. Don't be afraid to ask!

If you know what you are looking for, it is usually easier to search for it using the help system, instead of using Google. Because the subjects follow a certain style guide.

Also the help has the advantage of belonging to your particular Vim version. You won't see help for commands added later. These would not work for you.

To get generic help use this command:

```
:help
```

You could also use the first function key `<F1>`. If your keyboard has a `<Help>` key it might work as well.

If you don't supply a subject, ":help" displays the general help window. The creators of Vim did something very clever (or very lazy) with the help system: They made the help window a normal editing window. You can use all the normal Vim commands to move through the help information. Therefore h, j, k, and l move left, down, up and right.

To get out of the help window, use the same command you use to get out of the editor: "ZZ". This will only close the help window, not exit Vim.

As you read the help text, you will notice some text enclosed in vertical bars

(for example, [help](#)). This indicates a hyperlink. If you position the cursor anywhere between the bars and press **CTRL-]** (jump to tag), the help system takes you to the indicated subject. (For reasons not discussed here, the Vim terminology for a hyperlink is tag. So **CTRL-]** jumps to the location of the tag given by the word under the cursor.)

After a few jumps, you might want to go back. **CTRL-T** (pop tag) takes you back to the preceding position. **CTRL-O** (jump to older position) also works nicely here.

At the top of the help screen, there is the notation `*help.txt*`. This name between "*" characters is used by the help system to define a tag (hyperlink destination).

See [29.1](#) for details about using tags.

To get help on a given subject, use the following command:

```
:help {subject}
```

To get help on the "x" command, for example, enter the following:

```
:help x
```

To find out how to delete text, use this command:

```
:help deleting
```

To get a complete index of all Vim commands, use the following command:

```
:help index
```

When you need to get help for a control character command (for example, **CTRL-A**), you need to spell it with the prefix "CTRL-".

```
:help CTRL-A
```

The Vim editor has many different modes. By default, the help system displays the normal-mode commands. For example, the following command displays help for the normal-mode **CTRL-H** command:

```
:help CTRL-H
```

To identify other modes, use a mode prefix. If you want the help for the insert-mode version of a command, use "i_". For **CTRL-H** this gives you the following command:

```
:help i_CTRL-H
```

When you start the Vim editor, you can use several command-line arguments. These all begin with a dash (-). To find what the `-t` argument does, for example, use the command:

```
:help -t
```

The Vim editor has a number of options that enable you to configure and customize the editor. If you want help for an option, you need to enclose it

in single quotation marks. To find out what the **'number'** option does, for example, use the following command:

```
:help 'number'
```

The table with all mode prefixes can be found below: [help-summary](#) .

Special keys are enclosed in angle brackets. To find help on the up-arrow key in Insert mode, for instance, use this command:

```
:help i_<Up>
```

If you see an error message that you don't understand, for example:

```
E37: No write since last change (use ! to override)
```

You can use the error ID at the start to find help about it:

```
:help E37
```

Summary: [help-summary](#)

- 1) Use Ctrl-D after typing a topic and let Vim show all available topics.
Or press Tab to complete:

```
:help some<Tab>
```

More information on how to use the help:

```
:help helphelp
```

- 2) Follow the links in bars to related help. You can go from the detailed help to the user documentation, which describes certain commands more from a user perspective and less detailed. E.g. after:

```
:help pattern.txt
```

You can see the user guide topics [03.9](#) and [usr_27.txt](#) in the introduction.

- 3) Options are enclosed in single apostrophes. To go to the help topic for the list option:

```
:help 'list'
```

If you only know you are looking for a certain option, you can also do:

```
:help options.txt
```

to open the help page which describes all option handling and then search using regular expressions, e.g. `textwidth`.

Certain options have their own namespace, e.g.:

```
:help cpo-<letter>
```

for the corresponding flag of the **'cptions'** settings, substitute `<letter>` by a specific flag, e.g.:

```
:help cpo-;
```

And for the **'guioptions'** flags:

```
:help go-<letter>
```

- 4) Normal mode commands do not have a prefix. To go to the help page for the "gt" command:

```
:help gt
```

- 5) Insert mode commands start with `i_`. Help for deleting a word:
`:help i_CTRL-W`
- 6) Visual mode commands start with `v_`. Help for jumping to the other side of the Visual area:
`:help v_o`
- 7) Command line editing and arguments start with `c_`. Help for using the command argument %:
`:help c_%`
- 8) Ex-commands always start with `:"`, so to go to the `:"s` command help:
`:help :s`
- 9) Commands specifically for debugging start with `>`. To go to the help for the `"cont"` debug command:
`:help >cont`
- 10) Key combinations. They usually start with a single letter indicating the mode for which they can be used. E.g.:
`:help i_CTRL-X`
takes you to the family of **CTRL-X** commands for insert mode which can be used to auto-complete different things. Note, that certain keys will always be written the same, e.g. Control will always be CTRL.
For normal mode commands there is no prefix and the topic is available at `:h CTRL-<Letter>`. E.g.
`:help CTRL-W`
In contrast
`:help c_CTRL-R`
will describe what the **CTRL-R** does when entering commands in the Command line and
`:help v_CTRL-A`
talks about incrementing numbers in visual mode and
`:help g_CTRL-A`
talks about the `"g<C-A>"` command (e.g. you have to press `"g"` then `<CTRL-A>`). Here the `"g"` stands for the normal command `"g"` which always expects a second key before doing something similar to the commands starting with `"z"`.
- 11) Regexp items always start with `/`. So to get help for the `"\+"` quantifier in Vim regexes:
`:help /\+`
If you need to know everything about regular expressions, start reading at:
`:help pattern.txt`
- 12) Registers always start with `"quote"`. To find out about the special `:"` register:
`:help quote:`
- 13) Vim script is available at
`:help eval.txt`
Certain aspects of the language are available at `:h expr-X` where `"X"` is a

single letter. E.g.

`:help expr-!`

will take you to the topic describing the "!" (Not) operator for Vim script.

Also important is

`:help function-list`

to find a short description of all functions available. Help topics for Vim script functions always include the "()", so:

`:help append()`

talks about the append Vim script function rather than how to append text in the current buffer.

- 14) Mappings are talked about in the help page `:h map.txt` . Use

`:help mapmode-i`

to find out about the `:imap` command. Also use `:map-topic`

to find out about certain subtopics particular for mappings. e.g:

`:help :map-local`

for buffer-local mappings or

`:help map-bar`

for how the '|' is handled in mappings.

- 15) Command definitions are talked about `:h command-topic`, so use

`:help command-bar`

to find out about the '!' argument for custom commands.

- 16) Window management commands always start with **CTRL-W**, so you find the corresponding help at `:h CTRL-W_letter`. E.g.

`:help CTRL-W_p`

for moving the previous accessed window. You can also access

`:help windows.txt`

and read your way through if you are looking for window handling commands.

- 17) Use `:helpgrep` to search in all help pages (and also of any installed plugins). See `:helpgrep` for how to use it.

To search for a topic:

`:helpgrep topic`

This takes you to the first match. To go to the next one:

`:cnext`

All matches are available in the quickfix window which can be opened with:

`:copen`

Move around to the match you like and press Enter to jump to that help.

- 18) The user manual. This describes help topics for beginners in a rather friendly way. Start at `usr_toc.txt` to find the table of content (as you might have guessed):

`:help usr_toc.txt`

Skim over the contents to find interesting topics. The "Digraphs" and "Entering special characters" items are in chapter 24, so to go to that particular help page:

`:help usr_24.txt`

Also if you want to access a certain chapter in the help, the chapter number can be accessed directly like this:

`:help 10.1`

which goes to chapter 10.1 in `usr_10.txt` and talks about recording macros.

- 19) Highlighting groups. Always start with `hl-groupname`. E.g.
`:help hl-WarningMsg`
talks about the `WarningMsg` highlighting group.
- 20) Syntax highlighting is namespaced to `:syn-topic`. E.g.
`:help :syn-conceal`
talks about the `conceal` argument for the `":syn"` command.
- 21) Quickfix commands usually start with `:c` while location list commands usually start with `:l`
- 22) Autocommand events can be found by their name:
`:help BufWinLeave`
To see all possible events:
`:help autocommand-events`
- 23) Command-line switches always start with `"-"`. So for the help of the `-f` command switch of Vim use:
`:help -f`
- 24) Optional features always start with `"+"`. To find out about the `conceal` feature use:
`:help +conceal`
- 25) Documentation for included filetype specific functionality is usually available in the form `ft-<filetype>-<functionality>`. So
`:help ft-c-syntax`
talks about the C syntax file and the option it provides. Sometimes, additional sections for omni completion
`:help ft-php-omni`
or filetype plugins
`:help ft-tex-plugin`
are available.
- 26) Error and Warning codes can be looked up directly in the help. So
`:help E297`
takes you exactly to the description of the swap error message and
`:help W10`
talks about the warning "Changing a readonly file".
Sometimes, however, those error codes are not described, but rather are listed at the Vim command that usually causes this. So:
`:help E128`
takes you to the `:function` command

=====

Next chapter: `usr_03.txt` Moving around

Copyright: see `manual-copyright` vim:tw=78:ts=8:noet:ft=help:norl:

Moving around

Before you can insert or delete text the cursor has to be moved to the right place. Vim has a large number of commands to position the cursor. This chapter shows you how to use the most important ones. You can find a list of these commands below [Q_lr](#) .

- 03.1 Word movement
- 03.2 Moving to the start or end of a line
- 03.3 Moving to a character
- 03.4 Matching a parenthesis
- 03.5 Moving to a specific line
- 03.6 Telling where you are
- 03.7 Scrolling around
- 03.8 Simple searches
- 03.9 Simple search patterns
- 03.10 Using marks

Next chapter: [usr_04.txt](#) Making small changes
 Previous chapter: [usr_02.txt](#) The first steps in Vim
 Table of contents: [usr_toc.txt](#)



03.1 Word movement

To move the cursor forward one word, use the "w" command. Like most Vim commands, you can use a numeric prefix to move past multiple words. For example, "3w" moves three words. This figure shows how it works (starting at the position marked with "x"):

```

This is a line with example text
x-->-->-->----->
  w  w  w    3w

```

Notice that "w" moves to the start of the next word if it already is at the start of a word.

The "b" command moves backward to the start of the previous word:

```

This is a line with example text
<-----<--<--<-----<--x
  b  b b    2b    b

```

There is also the "e" command that moves to the next end of a word and "ge", which moves to the previous end of a word:

```

This is a line with example text
<-----<-----x----->----->
  2ge  ge    e    2e

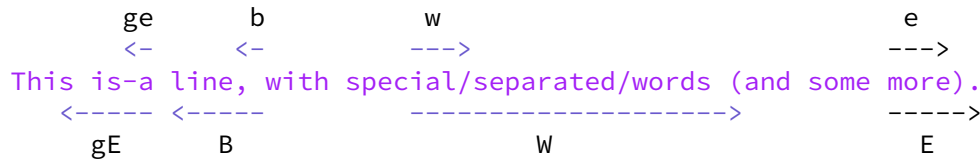
```

If you are at the last word of a line, the "w" command will take you to the first word in the next line. Thus you can use this to move through a paragraph, much faster than using "l". "b" does the same in the other direction.

A word ends at a non-word character, such as a ".", "-" or ")". To change what Vim considers to be a word, see the 'iskeyword' option. If you try this out in the help directly, 'iskeyword' needs to be reset for the examples to work:

```
:set iskeyword&
```

It is also possible to move by white-space separated WORDS. This is not a word in the normal sense, that's why the uppercase is used. The commands for moving by WORDS are also uppercase, as this figure shows:

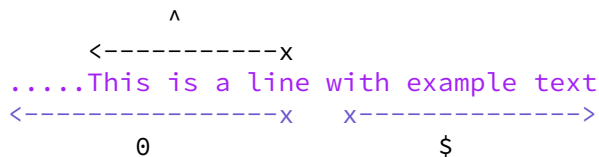


With this mix of lowercase and uppercase commands, you can quickly move forward and backward through a paragraph.

03.2 Moving to the start or end of a line

The "\$" command moves the cursor to the end of a line. If your keyboard has an <End> key it will do the same thing.

The "^" command moves to the first non-blank character of the line. The "0" command (zero) moves to the very first character of the line, and the <Home> key does the same thing. In a picture ("." indicates a space):



(the "....." indicates blanks here)

The "\$" command takes a count, like most movement commands. But moving to the end of the line several times doesn't make sense. Therefore it causes the editor to move to the end of another line. For example, "1\$" moves you to the end of the first line (the one you're on), "2\$" to the end of the next line, and so on.

The "0" command doesn't take a count argument, because the "0" would be part of the count. Unexpectedly, using a count with "^" doesn't have any effect.

03.3 Moving to a character

One of the most useful movement commands is the single-character search

command. The command "fx" searches forward in the line for the single character x. Hint: "f" stands for "Find".

For example, you are at the beginning of the following line. Suppose you want to go to the h of human. Just execute the command "fh" and the cursor will be positioned over the h:

```
To err is human. To really foul up you need a computer.
----->----->
      fh          fy
```

This also shows that the command "fy" moves to the end of the word really.

You can specify a count; therefore, you can go to the "l" of "foul" with "3fl":

```
To err is human. To really foul up you need a computer.
----->
                3fl
```

The "F" command searches to the left:

```
To err is human. To really foul up you need a computer.
<-----
                Fh
```

The "tx" command works like the "fx" command, except it stops one character before the searched character. Hint: "t" stands for "To". The backward version of this command is "Tx".

```
To err is human. To really foul up you need a computer.
<----->----->
      Th          tn
```

These four commands can be repeated with ";". ";," repeats in the other direction. The cursor is never moved to another line. Not even when the sentence continues.

Sometimes you will start a search, only to realize that you have typed the wrong command. You type "f" to search backward, for example, only to realize that you really meant "F". To abort a search, press <Esc>. So "f<Esc>" is an aborted forward search and doesn't do anything. **Note:** <Esc> cancels most operations, not just searches.

03.4 Matching a parenthesis

When writing a program you often end up with nested () constructs. Then the "%" command is very handy: It moves to the matching paren. If the cursor is on a "(" it will move to the matching ")". If it's on a ")" it will move to the matching "(".

```
                %
                <----->
if (a == (b * c) / d)
<----->
```

%

This also works for [] and {} pairs. (This can be defined with the 'matchpairs' option.)

When the cursor is not on a useful character, "%" will search forward to find one. Thus if the cursor is at the start of the line of the previous example, "%" will search forward and find the first "(".

```

if (a == (b * c) / d)
-----+----->
                %

```

Other ways to move around code can be found in [usr_29.txt](#).

=====

03.5 Moving to a specific line

If you are a C or C++ programmer, you are familiar with error messages such as the following:

```
prog.c:33: j undeclared (first use in this function)
```

This tells you that you might want to fix something on line 33. So how do you find line 33? One way is to do "9999k" to go to the top of the file and "32j" to go down thirty-two lines. It is not a good way, but it works. A much better way of doing things is to use the "G" command. With a count, this command positions you at the given line number. For example, "33G" puts you on line 33. (For a better way of going through a compiler's error list, see [usr_30.txt](#), for information on the :make command.)

With no argument, "G" positions you at the end of the file. A quick way to go to the start of a file use "gg". "1G" will do the same, but is a tiny bit more typing.

```

      | first line of a file      ^
      | text text text text      |
      | text text text text      | gg
7G   | text text text text      |
      | text text text text      |
      | text text text text      |
      | text text text text      | G
      | text text text text      |
      | last line of a file      V

```

Another way to move to a line is using the "%" command with a count. For example "50%" moves you to halfway the file. "90%" goes to near the end.

The previous assumes that you want to move to a line in the file, no matter if it's currently visible or not. What if you want to move to one of the lines you can see? This figure shows the three commands you can use:

```

      +-----+
H --> | text sample text      |

```

```

      | sample text          |
      | text sample text   |
M --> | sample text        |
      | text sample text   |
      | sample text        |
      | text sample text   |
L --> | sample text        |
      | text sample text   |
      +-----+

```

Hints: "H" stands for Home, "M" for Middle and "L" for Last. Alternatively, "H" for High, "M" for Middle and "L" for Low.

03.6 Telling where you are

To see where you are in a file, there are three ways:

1. Use the **CTRL-G** command. You get a message like this (assuming the **'ruler'** option is off):

```
"usr_03.txt" line 233 of 650 --35%-- col 45-52
```

This shows the name of the file you are editing, the line number where the cursor is, the total number of lines, the percentage of the way through the file and the column of the cursor.

Sometimes you will see a split column number. For example, "col 2-9". This indicates that the cursor is positioned on the second character, but because character one is a tab, occupying eight spaces worth of columns, the screen column is 9.

2. Set the **'number'** option. This will display a line number in front of every line:

```
:set number
```

To switch this off again:

```
:set nonumber
```

Since **'number'** is a boolean option, prepending "no" to its name has the effect of switching it off. A boolean option has only these two values, it is either on or off.

Vim has many options. Besides the boolean ones there are options with a numerical value and string options. You will see examples of this where they are used.

3. Set the **'ruler'** option. This will display the cursor position in the lower right corner of the Vim window:

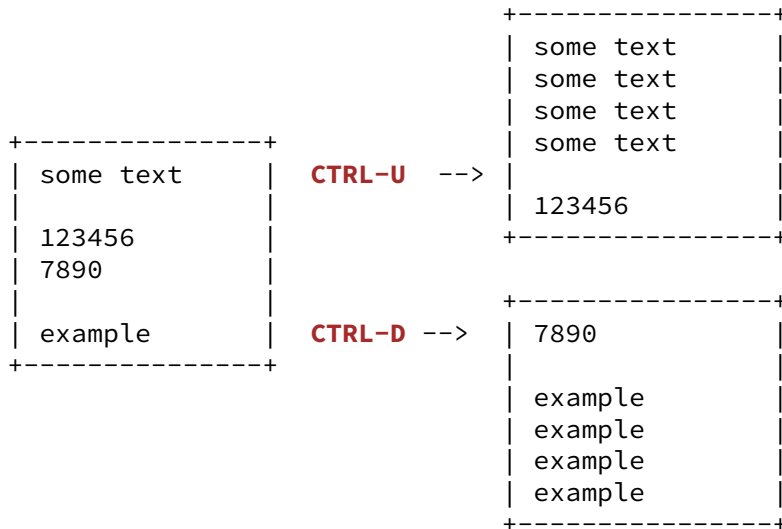
```
:set ruler
```

Using the **'ruler'** option has the advantage that it doesn't take much room, thus there is more space for your text.

03.7 Scrolling around

The **CTRL-U** command scrolls down half a screen of text. Think of looking through a viewing window at the text and moving this window up by half the height of the window. Thus the window moves up over the text, which is backward in the file. Don't worry if you have a little trouble remembering which end is up. Most users have the same problem.

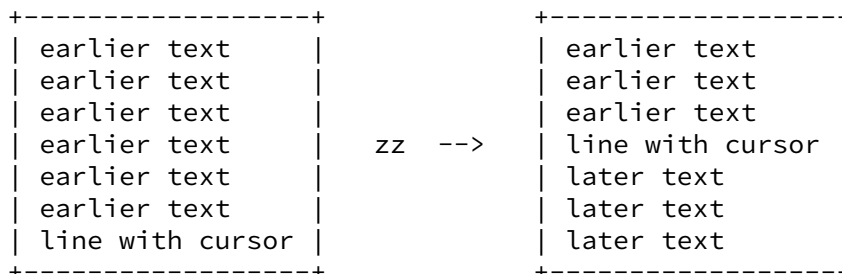
The **CTRL-D** command moves the viewing window down half a screen in the file, thus scrolls the text up half a screen.



To scroll one line at a time use **CTRL-E** (scroll up) and **CTRL-Y** (scroll down). Think of **CTRL-E** to give you one line Extra. (If you use MS-Windows compatible key mappings **CTRL-Y** will redo a change instead of scroll.)

To scroll forward by a whole screen (except for two lines) use **CTRL-F**. To scroll backwards, use **CTRL-B**. These should be easy to remember: F for Forwards and B for Backwards.

A common issue is that after moving down many lines with "j" your cursor is at the bottom of the screen. You would like to see the context of the line with the cursor. That's done with the "zz" command.



The "zt" command puts the cursor line at the top, "zb" at the bottom. There are a few more scrolling commands, see [Q_sc](#) . To always keep a few lines of

context around the cursor, use the **'scrolloff'** option.

03.8 Simple searches

To search for a string, use the `"/string"` command. To find the word `include`, for example, use the command:

```
/include
```

You will notice that when you type the `/"` the cursor jumps to the last line of the Vim window, like with colon commands. That is where you type the word. You can press the backspace key (backarrow or `<BS>`) to make corrections. Use the `<Left>` and `<Right>` cursor keys when necessary.

Pressing `<Enter>` executes the command.

Note:

The characters `. * [] ^ % / \ ? ~ $` have special meanings. If you want to use them in a search you must put a `\` in front of them. See below.

To find the next occurrence of the same string use the `"n"` command. Use this to find the first `#include` after the cursor:

```
/#include
```

And then type `"n"` several times. You will move to each `#include` in the text. You can also use a count if you know which match you want. Thus `"3n"` finds the third match. You can also use a count with `/"`: `"4/the"` goes to the fourth match of `"the"`.

The `"?"` command works like `/"` but searches backwards:

```
?word
```

The `"N"` command repeats the last search the opposite direction. Thus using `"N"` after a `/"` command searches backwards, using `"N"` after `"?"` searches forwards.

IGNORING CASE

Normally you have to type exactly what you want to find. If you don't care about upper or lowercase in a word, set the **'ignorecase'** option:

```
:set ignorecase
```

If you now search for `"word"`, it will also match `"Word"` and `"WORD"`. To match case again:

```
:set noignorecase
```

HISTORY

Suppose you do three searches:

```
/one  
/two  
/three
```

Now let's start searching by typing a simple "/" without pressing `<Enter>`. If you press `<Up>` (the cursor key), Vim puts `/three` on the command line. Pressing `<Enter>` at this point searches for three. If you do not press `<Enter>`, but press `<Up>` instead, Vim changes the prompt to `/two`. Another press of `<Up>` moves you to `/one`.

You can also use the `<Down>` cursor key to move through the history of search commands in the other direction.

If you know what a previously used pattern starts with, and you want to use it again, type that character before pressing `<Up>`. With the previous example, you can type `/o<Up>` and Vim will put `/one` on the command line.

The commands starting with ":" also have a history. That allows you to recall a previous command and execute it again. These two histories are separate.

SEARCHING FOR A WORD IN THE TEXT

Suppose you see the word `"TheLongFunctionName"` in the text and you want to find the next occurrence of it. You could type `"/TheLongFunctionName"`, but that's a lot of typing. And when you make a mistake Vim won't find it.

There is an easier way: Position the cursor on the word and use the `"*` command. Vim will grab the word under the cursor and use it as the search string.

The `"#` command does the same in the other direction. You can prepend a count: `"3*` searches for the third occurrence of the word under the cursor.

SEARCHING FOR WHOLE WORDS

If you type `"/the` it will also match `"there"`. To only find words that end in `"the"` use:

```
/the\>
```

The `"\>` item is a special marker that only matches at the end of a word. Similarly `"\<` only matches at the beginning of a word. Thus to search for the word `"the"` only:

```
/\<the\>
```

This does not match `"there"` or `"soothe"`. Notice that the `"*` and `"#` commands use these start-of-word and end-of-word markers to only find whole words (you can use `"g*` and `"g#` to match partial words).

HIGHLIGHTING MATCHES

While editing a program you see a variable called "nr". You want to check where it's used. You could move the cursor to "nr" and use the "*" command and press "n" to go along all the matches.

There is another way. Type this command:

```
:set hlsearch
```

If you now search for "nr", Vim will highlight all matches. That is a very good way to see where the variable is used, without the need to type commands.

To switch this off:

```
:set nohlsearch
```

Then you need to switch it on again if you want to use it for the next search command. If you only want to remove the highlighting, use this command:

```
:nohlsearch
```

This doesn't reset the option. Instead, it disables the highlighting. As soon as you execute a search command, the highlighting will be used again. Also for the "n" and "N" commands.

TUNING SEARCHES

There are a few options that change how searching works. These are the essential ones:

```
:set incsearch
```

This makes Vim display the match for the string while you are still typing it. Use this to check if the right match will be found. Then press <Enter> to really jump to that location. Or type more to change the search string.

```
:set nowrapscan
```

This stops the search at the end of the file. Or, when you are searching backwards, it stops the search at the start of the file. The **'wrapscan'** option is on by default, thus searching wraps around the end of the file.

INTERMEZZO

If you like one of the options mentioned before, and set it each time you use Vim, you can put the command in your Vim startup file.

Edit the file, as mentioned at [not-compatible](#) . Or use this command to find out where it is:

```
:scriptnames
```

Edit the file, for example with:

```
:edit ~/.vimrc
```

Then add a line with the command to set the option, just like you typed it in Vim. Example:

```
Go:set hlsearch<Esc>
```

"G" moves to the end of the file. "o" starts a new line, where you type the ":set" command. You end insert mode with <Esc>. Then write and close the file:

```
ZZ
```

If you now start Vim again, the 'hlsearch' option will already be set.

03.9 Simple search patterns

The Vim editor uses regular expressions to specify what to search for. Regular expressions are an extremely powerful and compact way to specify a search pattern. Unfortunately, this power comes at a price, because regular expressions are a bit tricky to specify.

In this section we mention only a few essential ones. More about search patterns and commands can be found in chapter 27 [usr_27.txt](#). You can find the full explanation here: [pattern](#).

BEGINNING AND END OF A LINE

The ^ character matches the beginning of a line. On an English-US keyboard you find it above the 6. The pattern "include" matches the word include anywhere on the line. But the pattern "^include" matches the word include only if it is at the beginning of a line.

The \$ character matches the end of a line. Therefore, "was\$" matches the word was only if it is at the end of a line.

Let's mark the places where "/the" matches in this example line with "x"s:

```
the solder holding one of the chips melted and the
xxx                               xxx                               xxx
```

Using "/the\$" we find this match:

```
the solder holding one of the chips melted and the
                                                                xxx
```

And with "/^the" we find this one:

```
the solder holding one of the chips melted and the
xxx
```

You can try searching with "/^the\$"; it will only match a single line consisting entirely of "the". White space does matter here, thus if a line contains a space after the word, like "the ", the pattern will not match.

MATCHING ANY SINGLE CHARACTER

The . (dot) character matches any existing character. For example, the pattern "c.m" matches a string whose first character is a c, whose second character is anything, and whose third character is m. Example:

```
We use a computer that became the cummin winter.
      xxx          xxx          xxx
```

MATCHING SPECIAL CHARACTERS

If you really want to match a dot, you must avoid its special meaning by putting a backslash before it.

If you search for "ter.", you will find these matches:

```
We use a computer that became the cummin winter.
      xxxx          xxxxx
```

Searching for "ter\." only finds the second match.

03.10 Using marks

When you make a jump to a position with the "G" command, Vim remembers the position from before this jump. This position is called a mark. To go back where you came from, use this command:

```
``
```

This ` is a backtick or open single-quote character.

If you use the same command a second time you will jump back again. That's because the `` command is a jump itself, and the position from before this jump is remembered.

Generally, every time you do a command that can move the cursor further than within the same line, this is called a jump. This includes the search commands "/" and "n" (it doesn't matter how far away the match is). But not the character searches with "fx" and "tx" or the word movements "w" and "e".

Also, "j" and "k" are not considered to be a jump, even when you use a count to make them move the cursor quite a long way away.

The `` command jumps back and forth, between two points. The **CTRL-O** command jumps to older positions (Hint: O for older). **CTRL-I** then jumps back to newer positions (Hint: for many common keyboard layouts, I is just next to O). Consider this sequence of commands:

```
33G
/^The
CTRL-O
```

You first jump to line 33, then search for a line that starts with "The". Then with **CTRL-O** you jump back to line 33. Another **CTRL-O** takes you back to where you started. If you now use **CTRL-I** you jump to line 33 again. And to the match for "The" with another **CTRL-I**.

		example text	^			
33G		example text		CTRL-O		CTRL-I
		example text				
	V	line 33 text	^		V	
		example text				
/^The		example text		CTRL-O		CTRL-I
	V	There you are			V	
		example text				

Note:

CTRL-I is the same as <Tab>.

The ":jumps" command gives a list of positions you jumped to. The entry which you used last is marked with a ">".

NAMED MARKS

bookmark

Vim enables you to place your own marks in the text. The command "ma" marks the place under the cursor as mark a. You can place 26 marks (a through z) in your text. You can't see them, it's just a position that Vim remembers.

To go to a mark, use the command `{mark}`, where {mark} is the mark letter. Thus to move to the a mark:

``a`

The command "'mark" (single quotation mark, or apostrophe) moves you to the beginning of the line containing the mark. This differs from the "`mark" command, which also moves you to the marked column.

The marks can be very useful when working on two related parts in a file. Suppose you have some text near the start of the file you need to look at, while working on some text near the end of the file.

Move to the text at the start and place the s (start) mark there:

`ms`

Then move to the text you want to work on and put the e (end) mark there:

`me`

Now you can move around, and when you want to look at the start of the file, you use this to jump there:

`'s`

Then you can use '' to jump back to where you were, or 'e to jump to the text you were working on at the end.

There is nothing special about using s for start and e for end, they are just easy to remember.

You can use this command to get a list of marks:

[:marks](#)

You will notice a few special marks. These include:

'	The cursor position before doing a jump
"	The cursor position when last editing the file
[Start of the last change
]	End of the last change

=====

Next chapter: [usr_04.txt](#) Making small changes

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Making small changes

This chapter shows you several ways of making corrections and moving text around. It teaches you the three basic ways to change text: operator-motion, Visual mode and text objects.

- 04.1 Operators and motions
- 04.2 Changing text
- 04.3 Repeating a change
- 04.4 Visual mode
- 04.5 Moving text
- 04.6 Copying text
- 04.7 Using the clipboard
- 04.8 Text objects
- 04.9 Replace mode
- 04.10 Conclusion

Next chapter: [usr_05.txt](#) Set your settings
Previous chapter: [usr_03.txt](#) Moving around
Table of contents: [usr_toc.txt](#)

04.1 Operators and motions

In chapter 2 you learned the "x" command to delete a single character. And using a count: "4x" deletes four characters.

The "dw" command deletes a word. You may recognize the "w" command as the move word command. In fact, the "d" command may be followed by any motion command, and it deletes from the current location to the place where the cursor winds up.

The "4w" command, for example, moves the cursor over four words. The "d4w" command deletes four words.

```
To err is human. To really foul up you need a computer.
```

```
----->  
          d4w
```

```
To err is human. you need a computer.
```

Vim only deletes up to the position where the motion takes the cursor. That's because Vim knows that you probably don't want to delete the first character of a word. If you use the "e" command to move to the end of a word, Vim guesses that you do want to include that last character:

```
To err is human. you need a computer.
```

```
----->  
          d2e
```

```
To err is human. a computer.
```

Whether the character under the cursor is included depends on the command you used to move to that character. The reference manual calls this "exclusive" when the character isn't included and "inclusive" when it is.

The "\$" command moves to the end of a line. The "d\$" command deletes from the cursor to the end of the line. This is an inclusive motion, thus the last character of the line is included in the delete operation:

```
To err is human. a computer.
                ----->
                d$
```

```
To err is human
```

There is a pattern here: operator-motion. You first type an operator command. For example, "d" is the delete operator. Then you type a motion command like "4l" or "w". This way you can operate on any text you can move over.

04.2 Changing text

Another operator is "c", change. It acts just like the "d" operator, except it leaves you in Insert mode. For example, "cw" changes a word. Or more specifically, it deletes a word and then puts you in Insert mode.

```
To err is human
----->
c2wbe<Esc>
```

```
To be human
```

This "c2wbe<Esc>" contains these bits:

c	the change operator
2w	move two words (they are deleted and Insert mode started)
be	insert this text
<Esc>	back to Normal mode

You will have noticed something strange: The space before "human" isn't deleted. There is a saying that for every problem there is an answer that is simple, clear, and wrong. That is the case with the example used here for the "cw" command. The c operator works just like the d operator, with one exception: "cw". It actually works like "ce", change to end of word. Thus the space after the word isn't included. This is an exception that dates back to the old Vi. Since many people are used to it now, the inconsistency has remained in Vim.

MORE CHANGES

Like "dd" deletes a whole line, "cc" changes a whole line. It keeps the existing indent (leading white space) though.

Just like "d\$" deletes until the end of the line, "c\$" changes until the end of the line. It's like doing "d\$" to delete the text and then "a" to start Insert mode and append new text.

SHORTCUTS

Some operator-motion commands are used so often that they have been given a single-letter command:

x	stands for	dl	(delete character under the cursor)
X	stands for	dh	(delete character left of the cursor)
D	stands for	d\$	(delete to end of the line)
C	stands for	c\$	(change to end of the line)
s	stands for	cl	(change one character)
S	stands for	cc	(change a whole line)

WHERE TO PUT THE COUNT

The commands "3dw" and "d3w" delete three words. If you want to get really picky about things, the first command, "3dw", deletes one word three times; the command "d3w" deletes three words once. This is a difference without a distinction. You can actually put in two counts, however. For example, "3d2w" deletes two words, repeated three times, for a total of six words.

REPLACING WITH ONE CHARACTER

The "r" command is not an operator. It waits for you to type a character, and will replace the character under the cursor with it. You could do the same with "cl" or with the "s" command, but with "r" you don't have to press <Esc> to get back out of insert mode.

```
there is somerhing grong here
rT          rt    rw
```

```
There is something wrong here
```

Using a count with "r" causes that many characters to be replaced with the same character. Example:

```
There is something wrong here
5rx
```

```
There is something xxxxx here
```

To replace a character with a line break use "r<Enter>". This deletes one character and inserts a line break. Using a count here only applies to the number of characters deleted: "4r<Enter>" replaces four characters with one line break.

=====

04.3 Repeating a change

The "." command is one of the simplest yet powerful commands in Vim. It repeats the last change. For instance, suppose you are editing an HTML file and want to delete all the tags. You position the cursor on the first < and delete the with the command "df>". You then go to the < of the next and delete it using the "." command. The "." command executes the last change command (in this case, "df>"). To delete another tag, position the cursor on the < and use the "." command.

```

                                To <B>generate</B> a table of <B>contents
f<  find first <  ---->
df> delete to >   -->
f<  find next <   ----->
.   repeat df>    ---->
f<  find next <   ----->
.   repeat df>    -->

```

The "." command works for all changes you make, except for "u" (undo), **CTRL-R** (redo) and commands that start with a colon (:).

Another example: You want to change the word "four" to "five". It appears several times in your text. You can do this quickly with this sequence of commands:

```

/four<Enter>  find the first string "four"
cwfive<Esc>   change the word to "five"
n             find the next "four"
.            repeat the change to "five"
n             find the next "four"
.            repeat the change
etc.

```

04.4 Visual mode

To delete simple items the operator-motion changes work quite well. But often it's not so easy to decide which command will move over the text you want to change. Then you can use Visual mode.

You start Visual mode by pressing "v". You move the cursor over the text you want to work on. While you do this, the text is highlighted. Finally type the operator command.

For example, to delete from the middle of one word to the middle of another word:

```

This is an examination sample of visual mode
----->
      vellld

```

```

This is an example of visual mode

```

When doing this you don't really have to count how many times you have to press "l" to end up in the right position. You can immediately see what text will be deleted when you press "d".

If at any time you decide you don't want to do anything with the highlighted text, just press `<Esc>` and Visual mode will stop without doing anything.

SELECTING LINES

If you want to work on whole lines, use "V" to start Visual mode. You will see right away that the whole line is highlighted, without moving around. When you move left or right nothing changes. When you move up or down the selection is extended whole lines at a time.

For example, select three lines with "Vjj":

```
+-----+
| text more text |
>> | more text more text | | Vjj
selected lines >> | text text text | | V
>> | text more |
| more text more |
+-----+
```

SELECTING BLOCKS

If you want to work on a rectangular block of characters, use **CTRL-V** to start Visual mode. This is very useful when working on tables.

name	Q1	Q2	Q3
pierre	123	455	234
john	0	90	39
steve	392	63	334

To delete the middle "Q2" column, move the cursor to the "Q" of "Q2". Press **CTRL-V** to start blockwise Visual mode. Now move the cursor three lines down with "3j" and to the next word with "w". You can see the first character of the last column is included. To exclude it, use "h". Now press "d" and the middle column is gone.

GOING TO THE OTHER SIDE

If you have selected some text in Visual mode, and discover that you need to change the other end of the selection, use the "o" command (Hint: o for other end). The cursor will go to the other end, and you can move the cursor to change where the selection starts. Pressing "o" again brings you back to the other end.

When using blockwise selection, you have four corners. "o" only takes you to one of the other corners, diagonally. Use "O" to move to the other corner in the same line.

Note that "o" and "O" in Visual mode work very differently from Normal mode, where they open a new line below or above the cursor.

04.5 Moving text

When you delete something with "d", "x", or another command, the text is saved. You can paste it back by using the "p" command. (The Vim name for this is put).

Take a look at how this works. First you will delete an entire line, by putting the cursor on the line you want to delete and typing "dd". Now you move the cursor to where you want to put the line and use the "p" (put) command. The line is inserted on the line below the cursor.

```
a line      a line      a line
line 2  dd  line 3  p  line 3
line 3                                line 2
```

Because you deleted an entire line, the "p" command placed the text line below the cursor. If you delete part of a line (a word, for instance), the "p" command puts it just after the cursor.

```
Some more boring try text to out commands.
```

```
---->
dw
```

```
Some more boring text to out commands.
```

```
----->
welp
```

```
Some more boring text to try out commands.
```

MORE ON PUTTING

The "P" command puts text like "p", but before the cursor. When you deleted a whole line with "dd", "P" will put it back above the cursor. When you deleted a word with "dw", "P" will put it back just before the cursor.

You can repeat putting as many times as you like. The same text will be used.

You can use a count with "p" and "P". The text will be repeated as many times as specified with the count. Thus "dd" and then "3p" puts three copies of the same deleted line.

SWAPPING TWO CHARACTERS

Frequently when you are typing, your fingers get ahead of your brain (or the other way around?). The result is a typo such as "teh" for "the". Vim makes it easy to correct such problems. Just put the cursor on the e of "teh" and execute the command "xp". This works as follows: "x" deletes the character e and places it in a register. "p" puts the text after the cursor, which is after the h.

```
teh      th      the
 x       p
```

04.6 Copying text

To copy text from one place to another, you could delete it, use "u" to undo the deletion and then "p" to put it somewhere else. There is an easier way: yanking. The "y" operator copies text into a register. Then a "p" command can be used to put it.

Yanking is just a Vim name for copying. The "c" letter was already used for the change operator, and "y" was still available. Calling this operator "yank" made it easier to remember to use the "y" key.

Since "y" is an operator, you use "yw" to yank a word. A count is possible as usual. To yank two words use "y2w". Example:

```
let sqr = LongVariable *
          ----->
          y2w

let sqr = LongVariable *
          p

let sqr = LongVariable * LongVariable
```

Notice that "yw" includes the white space after a word. If you don't want this, use "ye".

The "yy" command yanks a whole line, just like "dd" deletes a whole line. Unexpectedly, while "D" deletes from the cursor to the end of the line, "Y" works like "yy", it yanks the whole line. Watch out for this inconsistency! Use "y\$" to yank to the end of the line.

```
  a text line  yy      a text line      p      a text line
  line 2      line 2      line 2
  last line   last line   a text line
                                   last line
```

04.7 Using the clipboard

If you are using the GUI version of Vim (gvim), you can find the "Copy" item in the "Edit" menu. First select some text with Visual mode, then use the Edit/Copy menu item. The selected text is now copied to the clipboard. You can paste the text in other programs. In Vim itself too.

If you have copied text to the clipboard in another application, you can paste it in Vim with the Edit/Paste menu item. This works in Normal mode and Insert mode. In Visual mode the selected text is replaced with the pasted text.

The "Cut" menu item deletes the text before it's put on the clipboard. The "Copy", "Cut" and "Paste" items are also available in the popup menu (only when there is a popup menu, of course). If your Vim has a toolbar, you can also find these items there.

If you are not using the GUI, or if you don't like using a menu, you have to use another way. You use the normal "y" (yank) and "p" (put) commands, but prepend "*" (double-quote star) before it. To copy a line to the clipboard:

```
"*yy
```

To put text from the clipboard back into the text:

```
"*p
```

This only works on versions of Vim that include clipboard support. More about the clipboard can be found in section [09.3](#) and here: [clipboard](#).

04.8 Text objects

If the cursor is in the middle of a word and you want to delete that word, you need to move back to its start before you can do "dw". There is a simpler way to do this: "daw".

```
this is some example text.  
      daw
```

```
this is some text.
```

The "d" of "daw" is the delete operator. "aw" is a text object. Hint: "aw" stands for "A Word". Thus "daw" is "Delete A Word". To be precise, the white space after the word is also deleted (or the white space before the word if at the end of the line).

Using text objects is the third way to make changes in Vim. We already had operator-motion and Visual mode. Now we add operator-text object.

It is very similar to operator-motion, but instead of operating on the text between the cursor position before and after a movement command, the text object is used as a whole. It doesn't matter where in the object the cursor was.

To change a whole sentence use "cis". Take this text:

```
Hello there. This  
is an example. Just  
some text.
```

Move to the start of the second line, on "is an". Now use "cis":

```
Hello there.    Just  
some text.
```

The cursor is in between the blanks in the first line. Now you type the new sentence "Another line.":

```
Hello there. Another line. Just  
some text.
```

"cis" consists of the "c" (change) operator and the "is" text object. This stands for "Inner Sentence". There is also the "as" ("A Sentence") object. The difference is that "as" includes the white space after the sentence and "is" doesn't. If you would delete a sentence, you want to delete the white space at the same time, thus use "das". If you want to type new text the white space can remain, thus you use "cis".

You can also use text objects in Visual mode. It will include the text object in the Visual selection. Visual mode continues, thus you can do this several times. For example, start Visual mode with "v" and select a sentence with "as". Now you can repeat "as" to include more sentences. Finally you use an operator to do something with the selected sentences.

You can find a long list of text objects here: [text-objects](#) .

04.9 Replace mode

The "R" command causes Vim to enter replace mode. In this mode, each character you type replaces the one under the cursor. This continues until you type `<Esc>`.

In this example you start Replace mode on the first "t" of "text":

```
This is text.  
Rinteresting.<Esc>
```

```
This is interesting.
```

You may have noticed that this command replaced 5 characters in the line with twelve others. The "R" command automatically extends the line if it runs out of characters to replace. It will not continue on the next line.

You can switch between Insert mode and Replace mode with the `<Insert>` key.

When you use `<BS>` (backspace) to make a correction, you will notice that the old text is put back. Thus it works like an undo command for the previously typed character.

04.10 Conclusion

The operators, movement commands and text objects give you the possibility to make lots of combinations. Now that you know how they work, you can use N operators with M movement commands to make N * M commands!

You can find a list of operators here: [operator](#) .

For example, there are many other ways to delete pieces of text. Here are a few common ones:

x	delete character under the cursor (short for "dl")
X	delete character before the cursor (short for "dh")
D	delete from cursor to end of line (short for "d\$")
dw	delete from cursor to next start of word

db delete from cursor to previous start of word
diw delete word under the cursor (excluding white space)
daw delete word under the cursor (including white space)
dG delete until the end of the file
dgg delete until the start of the file

If you use "c" instead of "d" they become change commands. And with "y" you yank the text. And so forth.

There are a few common commands to make changes that didn't fit somewhere else:

- ~ Change case of the character under the cursor, and move the cursor to the next character. This is not an operator (unless **'tildeop'** is set), thus you can't use it with a motion command. It does work in Visual mode, where it changes case for all the selected text.

- I Start Insert mode after moving the cursor to the first non-blank in the line.

- A Start Insert mode after moving the cursor to the end of the line.



Next chapter: [usr_05.txt](#) Set your settings

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_05.txt For Vim version 9.1. Last change: 2023 Sep 12

VIM USER MANUAL - by Bram Moolenaar

Set your settings

Vim can be tuned to work like you want it to. This chapter shows you how to make Vim start with options set to different values. Add plugins to extend Vim's capabilities. Or define your own macros.

- 05.1 The vimrc file
- 05.2 The example vimrc file explained
- 05.3 The defaults.vim file explained
- 05.4 Simple mappings
- 05.5 Adding a package
- 05.6 Adding a plugin
- 05.7 Adding a help file
- 05.8 The option window
- 05.9 Often used options

Next chapter: [usr_06.txt](#) Using syntax highlighting
Previous chapter: [usr_04.txt](#) Making small changes
Table of contents: [usr_toc.txt](#)

05.1 The vimrc file

[vimrc-intro](#)

You probably got tired of typing commands that you use very often. To start Vim with all your favorite option settings and mappings, you write them in what is called the vimrc file. Vim executes the commands in this file when it starts up.

If you already have a vimrc file (e.g., when your sysadmin has one setup for you), you can edit it this way:

```
:edit $MYVIMRC
```

If you don't have a vimrc file yet, see [vimrc](#) to find out where you can create a vimrc file. Also, the `":version"` command mentions the name of the "user vimrc file" Vim looks for.

For Unix and Macintosh this file is always used and is recommended:

```
~/.vimrc
```

For MS-Windows you can use one of these:

```
$HOME/_vimrc  
$VIM/_vimrc
```

If you are creating the vimrc file for the first time, it is recommended to put this line at the top:

```
source $VIMRUNTIME/defaults.vim
```

This initializes Vim for new users (as opposed to traditional Vi users). See [defaults.vim](#) for the details.

The vimrc file can contain all the commands that you type after a colon. The simplest ones are for setting options. For example, if you want Vim to always start with the **'incsearch'** option on, add this line your vimrc file:

```
set incsearch
```

For this new line to take effect you need to exit Vim and start it again. Later you will learn how to do this without exiting Vim.

This chapter only explains the most basic items. For more information on how to write a Vim script file: [usr_41.txt](#) .

05.2 The example vimrc file explained vimrc_example.vim

In the first chapter was explained how the example vimrc (included in the Vim distribution) file can be used to make Vim startup in not-compatible mode (see [not-compatible](#)). The file can be found here:

```
$VIMRUNTIME/vimrc_example.vim
```

In this section we will explain the various commands used in this file. This will give you hints about how to set up your own preferences. Not everything will be explained though. Use the `":help"` command to find out more.

```
" Get the defaults that most users want.
source $VIMRUNTIME/defaults.vim
```

This loads the "defaults.vim" file in the \$VIMRUNTIME directory. This sets up Vim for how most users like it. If you are one of the few that don't, then comment out this line. The commands are explained below:

[defaults.vim-explained](#)

```
if has("vms")
  set nobackup
else
  set backup
  if has('persistent_undo')
    set undofile
  endif
endif
```

This tells Vim to keep a backup copy of a file when overwriting it. But not on the VMS system, since it keeps old versions of files already. The backup file will have the same name as the original file with "~" added. See [07.4](#)

This also sets the **'undofile'** option, if available. This will store the

multi-level undo information in a file. The result is that when you change a file, exit Vim, and then edit the file again, you can undo the changes made previously. It's a very powerful and useful feature, at the cost of storing a file. For more information see [undo-persistence](#) .

The "if" command is very useful to set options only when some condition is met. More about that in [usr_41.txt](#) .

```
if &t_Co > 2 || has("gui_running")
  set hlsearch
endif
```

This switches on the '**hlsearch**' option, telling Vim to highlight matches with the last used search pattern.

```
augroup vimrcEx
  au!
  autocmd FileType text setlocal textwidth=78
augroup END
```

This makes Vim break text to avoid lines getting longer than 78 characters. But only for files that have been detected to be plain text. There are actually two parts here. "autocmd FileType text" is an autocommand. This defines that when the file type is set to "text" the following command is automatically executed. "setlocal textwidth=78" sets the '**textwidth**' option to 78, but only locally in one file.

The wrapper with "augroup vimrcEx" and "augroup END" makes it possible to delete the autocommand with the "au!" command. See [:augroup](#) .

```
if has('syntax') && has('eval')
  packadd! matchit
endif
```

This loads the "matchit" plugin if the required features are available. It makes the % command more powerful. This is explained at [matchit-install](#) .

05.3 The defaults.vim file explained [defaults.vim-explained](#)

The [defaults.vim](#) file is loaded when the user has no vimrc file. When you create a new vimrc file, add this line near the top to keep using it:

```
source $VIMRUNTIME/defaults.vim
```

Or use the vimrc_example.vim file, as explained above.

The following explains what defaults.vim is doing.

```
if exists('skip_defaults_vim')
  finish
endif
```

Loading defaults.vim can be disabled with this command:

```
let skip_defaults_vim = 1
```

This has to be done in the system vimrc file. See [system-vimrc](#). If you have a user vimrc this is not needed, since defaults.vim will not be loaded automatically.

```
set nocompatible
```

As mentioned in the first chapter, these manuals explain Vim working in an improved way, thus not completely Vi compatible. Setting the **'compatible'** option off, thus **'nocompatible'** takes care of this.

```
set backspace=indent,eol,start
```

This specifies where in Insert mode the **<BS>** is allowed to delete the character in front of the cursor. The three items, separated by commas, tell Vim to delete the white space at the start of the line, a line break and the character before where Insert mode started. See **'backspace'**.

```
set history=200
```

Keep 200 commands and 200 search patterns in the history. Use another number if you want to remember fewer or more lines. See **'history'**.

```
set ruler
```

Always display the current cursor position in the lower right corner of the Vim window. See **'ruler'**.

```
set showcmd
```

Display an incomplete command in the lower right corner of the Vim window, left of the ruler. For example, when you type "2f", Vim is waiting for you to type the character to find and "2f" is displayed. When you press "w" next, the "2fw" command is executed and the displayed "2f" is removed.

```
+-----+
|text in the Vim window|
|~|
|~|
|-- VISUAL --          2f    43,8    17% |
+-----+
^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^   ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^   ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^
  'showmode'              'showcmd'  'ruler'
```

```
set wildmenu
```

Display completion matches in a status line. That is when you type <Tab> and there is more than one match. See '[wildmenu](#)'.

```
set ttimeout
set ttimeoutlen=100
```

This makes typing Esc take effect more quickly. Normally Vim waits a second to see if the Esc is the start of an escape sequence. If you have a very slow remote connection, increase the number. See '[ttimeout](#)'.

```
set display=truncate
```

Show @@@ in the last line if it is truncated, instead of hiding the whole line. See '[display](#)'.

```
set incsearch
```

Display the match for a search pattern when halfway typing it. See '[incsearch](#)'.

```
set nrformats=octal
```

Do not recognize numbers starting with a zero as octal. See '[nrformats](#)'.

```
map Q gq
```

This defines a key mapping. More about that in the next section. This defines the "Q" command to do formatting with the "gq" operator. This is how it worked before Vim 5.0. Otherwise the "Q" command starts Ex mode, but you will not need it.

```
inoremap <C-U> <C-G>u<C-U>
```

CTRL-U in insert mode deletes all entered text in the current line. Use **CTRL-G u** to first break undo, so that you can undo **CTRL-U** after inserting a line break. Revert with `":iunmap <C-U>"`.

```
if has('mouse')
  set mouse=a
endif
```

Enable using the mouse if available. See '[mouse](#)'.

```
vnoremap _g y:exe "grep /" .. escape(@, '\\/') .. "/" *.c *.h"<CR>
```

This mapping yanks the visually selected text and searches for it in C files. You can see that a mapping can be used to do quite complicated things. Still, it is just a sequence of commands that are executed like you typed them.

[syntax on](#)

Enable highlighting files in color. See [syntax](#) .

[filetype plugin indent on](#)

[vimrc-filetype](#)

This switches on three very clever mechanisms:

1. Filetype detection.

Whenever you start editing a file, Vim will try to figure out what kind of file this is. When you edit "main.c", Vim will see the ".c" extension and recognize this as a "c" filetype. When you edit a file that starts with "#!/bin/sh", Vim will recognize it as a "sh" filetype.

The filetype detection is used for syntax highlighting and the other two items below.

See [filetypes](#) .

2. Using filetype plugin files

Many different filetypes are edited with different options. For example, when you edit a "c" file, it's very useful to set the '**cindent**' option to automatically indent the lines. These commonly useful option settings are included with Vim in filetype plugins. You can also add your own, see [write-filetype-plugin](#) .

3. Using indent files

When editing programs, the indent of a line can often be computed automatically. Vim comes with these indent rules for a number of filetypes. See [:filetype-indent-on](#) and '**indentexpr**'.

[restore-cursor](#) [last-position-jump](#)

```
augroup RestoreCursor
  autocmd!
  autocmd BufReadPost *
    \ let line = line("\")
    \ | if line >= 1 && line <= line("$") && &filetype !~# 'commit'
    \   && index(['xd', 'gitrebase'], &filetype) == -1
    \ |   execute "normal! g\"
    \ | endif
augroup END
```

Another autocommand. This time it is used after reading any file. The complicated stuff after it checks if the '"' mark is defined, and jumps to it if so. It doesn't do that for a commit or rebase message, which are likely a different one than last time, and when using xxd(1) to filter and edit

binary files, which transforms input files back and forth, causing them to have dual nature, so to speak. See also [using-xxd](#) .

The backslash at the start of a line is used to continue the command from the previous line. That avoids a line getting very long. See [line-continuation](#) . This only works in a Vim script file, not when typing commands at the command line.

```
command DiffOrig vert new | set bt=nofile | r ++edit # | 0d_ | diffthis
\ | wincmd p | diffthis
```

This adds the ":DiffOrig" command. Use this in a modified buffer to see the differences with the file it was loaded from. See [diff](#) and [:DiffOrig](#) .

```
set nolangremap
```

Prevent that the langmap option applies to characters that result from a mapping. If set (default), this may break plugins (but it's backward compatible). See '[langremap](#)'.

05.4 Simple mappings

A mapping enables you to bind a set of Vim commands to a single key. Suppose, for example, that you need to surround certain words with curly braces. In other words, you need to change a word such as "amount" into "{amount}". With the :map command, you can tell Vim that the F5 key does this job. The command is as follows:

```
:map <F5> i{<Esc>ea}<Esc>
```

Note:

When entering this command, you must enter <F5> by typing four characters. Similarly, <Esc> is not entered by pressing the <Esc> key, but by typing five characters. Watch out for this difference when reading the manual!

Let's break this down:

- <F5> The F5 function key. This is the trigger key that causes the command to be executed as the key is pressed.

- i{<Esc> Insert the { character. The <Esc> key ends Insert mode.

- e Move to the end of the word.

- a}<Esc> Append the } to the word.

After you execute the ":map" command, all you have to do to put {} around a word is to put the cursor on the first character and press F5.

In this example, the trigger is a single key; it can be any string. But when you use an existing Vim command, that command will no longer be available.

You better avoid that.

One key that can be used with mappings is the backslash. Since you probably want to define more than one mapping, add another character. You could map "\p" to add parentheses around a word, and "\c" to add curly braces, for example:

```
:map \p i(<Esc>ea)<Esc>
:map \c i{<Esc>ea}<Esc>
```

You need to type the \ and the p quickly after another, so that Vim knows they belong together.

The ":map" command (with no arguments) lists your current mappings. At least the ones for Normal mode. More about mappings in section [40.1](#).

05.5 Adding a package add-package matchit-install

A package is a set of files that you can add to Vim. There are two kinds of packages: optional and automatically loaded on startup.

The Vim distribution comes with a few packages that you can optionally use. For example, the matchit plugin. This plugin makes the "%" command jump to matching HTML tags, if/else/endif in Vim scripts, etc. Very useful, although it's not backwards compatible (that's why it is not enabled by default).

To start using the matchit plugin, add one line to your vimrc file:

```
packadd! matchit
```

That's all! After restarting Vim you can find help about this plugin:

```
:help matchit
```

This works, because when `:packadd` loaded the plugin it also added the package directory in '[runtimepath](#)', so that the help file can be found.

You can find packages on the Internet in various places. It usually comes as an archive or as a repository. For an archive you can follow these steps:

1. create the package directory:

```
mkdir -p ~/.vim/pack/fancy
```

"fancy" can be any name of your liking. Use one that describes the package.
2. unpack the archive in that directory. This assumes the top directory in the archive is "start":

```
cd ~/.vim/pack/fancy
unzip /tmp/fancy.zip
```

If the archive layout is different make sure that you end up with a path like this:

```
~/.vim/pack/fancy/start/fancytext/plugin/fancy.vim
```

Here "fancytext" is the name of the package, it can be anything else.

Adding the editorconfig package editorconfig-install

Similar to the matchit package, to load the distributed editorconfig plugin

when Vim starts, add the following line to your vimrc file:
`packadd! editorconfig`

After restarting your Vim, the plugin is active and you can read about it at:
`:h editorconfig.txt`

More information about packages can be found here: [packages](#) .

05.6 Adding a plugin [add-plugin](#) [plugin](#)

Vim's functionality can be extended by adding plugins. A plugin is nothing more than a Vim script file that is loaded automatically when Vim starts. You can add a plugin very easily by dropping it in your plugin directory.
{not available when Vim was compiled without the |+eval| feature}

There are two types of plugins:

- global plugin: Used for all kinds of files
- filetype plugin: Only used for a specific type of file

The global plugins will be discussed first, then the filetype ones
[add-filetype-plugin](#) .

GLOBAL PLUGINS [standard-plugin](#) [distributed-plugins](#)

When you start Vim, it will automatically load a number of global plugins. You don't have to do anything for this. They add functionality that most people will want to use, but which was implemented as a Vim script instead of being compiled into Vim. You can find them listed in the help index
[standard-plugin-list](#) .

For locally installed plugins and packages (which come with a separated help file) a similar list can be found in the help section [local-additions](#) .

Also see [load-plugins](#) .

[add-global-plugin](#)

You can add a global plugin to add functionality that will always be present when you use Vim. There are only two steps for adding a global plugin:

1. Get a copy of the plugin.
2. Drop it in the right directory.

GETTING A GLOBAL PLUGIN

Where can you find plugins?

- Some are always loaded, you can see them in the directory `$VIMRUNTIME/plugin`.
- Some come with Vim. You can find them in the directory `$VIMRUNTIME/macros` and its sub-directories and under `$VIM/vimfiles/pack/dist/opt/`.
- Download from the net. There is a large collection on <http://www.vim.org>.
- They are sometimes posted in a Vim [maillist](#) .
- You could write one yourself, see [write-plugin](#) .

Some plugins come as a vimball archive, see [vimball](#) .
Some plugins can be updated automatically, see [getscript](#) .

USING A GLOBAL PLUGIN

First read the text in the plugin itself to check for any special conditions.
Then copy the file to your plugin directory:

system	plugin directory
Unix	~/.vim/plugin/
PC	\$HOME/vimfiles/plugin or \$VIM/vimfiles/plugin
Amiga	s:vimfiles/plugin
Macintosh	\$VIM:vimfiles:plugin
Mac OS X	~/.vim/plugin/

Example for Unix (assuming you didn't have a plugin directory yet):

```
mkdir ~/.vim
mkdir ~/.vim/plugin
cp /tmp/yourplugin.vim ~/.vim/plugin
```

That's all! Now you can use the commands defined in this plugin.

Instead of putting plugins directly into the plugin/ directory, you may better organize them by putting them into subdirectories under plugin/.
As an example, consider using "~/.vim/plugin/perl/*.vim" for all your Perl plugins.

FILETYPE PLUGINS

[add-filetype-plugin](#) [ftplugins](#)

The Vim distribution comes with a set of plugins for different filetypes that you can start using with this command:

```
:filetype plugin on
```

That's all! See [vimrc-filetype](#) .

If you are missing a plugin for a filetype you are using, or you found a better one, you can add it. There are two steps for adding a filetype plugin:

1. Get a copy of the plugin.
2. Drop it in the right directory.

GETTING A FILETYPE PLUGIN

You can find them in the same places as the global plugins. Watch out if the type of file is mentioned, then you know if the plugin is a global or a filetype one. The scripts in \$VIMRUNTIME/macros are global ones, the filetype plugins are in \$VIMRUNTIME/ftplugin.

USING A FILETYPE PLUGIN

ftplugin-name

You can add a filetype plugin by dropping it in the right directory. The name of this directory is in the same directory mentioned above for global plugins, but the last part is "ftplugin". Suppose you have found a plugin for the "stuff" filetype, and you are on Unix. Then you can move this file to the ftplugin directory:

```
mv thefile ~/.vim/ftplugin/stuff.vim
```

If that file already exists you already have a plugin for "stuff". You might want to check if the existing plugin doesn't conflict with the one you are adding. If it's OK, you can give the new one another name:

```
mv thefile ~/.vim/ftplugin/stuff_too.vim
```

The underscore is used to separate the name of the filetype from the rest, which can be anything. If you use "otherstuff.vim" it wouldn't work, it would be loaded for the "otherstuff" filetype.

On MS-DOS like filesystems you cannot use long filenames. You would run into trouble if you add a second plugin and the filetype has more than six characters. You can use an extra directory to get around this:

```
mkdir $VIM/vimfiles/ftplugin/fortran
copy thefile $VIM/vimfiles/ftplugin/fortran/too.vim
```

The generic names for the filetype plugins are:

```
ftplugin/<filetype>.vim
ftplugin/<filetype>_<name>.vim
ftplugin/<filetype>/<name>.vim
```

Here "<name>" can be any name that you prefer. Examples for the "stuff" filetype on Unix:

```
~/.vim/ftplugin/stuff.vim
~/.vim/ftplugin/stuff_def.vim
~/.vim/ftplugin/stuff/header.vim
```

The <filetype> part is the name of the filetype the plugin is to be used for. Only files of this filetype will use the settings from the plugin. The <name> part of the plugin file doesn't matter, you can use it to have several plugins for the same filetype. **Note** that it must end in ".vim".

Further reading:

[filetype-plugins](#)

Documentation for the filetype plugins and information about how to avoid that mappings cause problems.

[load-plugins](#)

When the global plugins are loaded during startup.

[ftplugin-override](#)

Overruling the settings from a global plugin.

[write-plugin](#)

How to write a plugin script.

[plugin-details](#)

For more information about using plugins or when your plugin doesn't work.

05.7 Adding a help file**add-local-help**

If you are lucky, the plugin you installed also comes with a help file. We will explain how to install the help file, so that you can easily find help for your new plugin.

Let us use the "doit.vim" plugin as an example. This plugin comes with documentation: "doit.txt". Let's first copy the plugin to the right directory. This time we will do it from inside Vim. (You may skip some of the "mkdir" commands if you already have the directory.)

```
:!mkdir ~/.vim
:!mkdir ~/.vim/plugin
:!cp /tmp/doit.vim ~/.vim/plugin
```

The "cp" command is for Unix, on MS-Windows you can use "copy".

Now create a "doc" directory in one of the directories in 'runtimepath'.

```
:!mkdir ~/.vim/doc
```

Copy the help file to the "doc" directory.

```
:!cp /tmp/doit.txt ~/.vim/doc
```

Now comes the trick, which allows you to jump to the subjects in the new help file: Generate the local tags file with the `:helptags` command.

```
:helptags ~/.vim/doc
```

Now you can use the

```
:help doit
```

command to find help for "doit" in the help file you just added. You can see an entry for the local help file when you do:

```
:help local-additions
```

The title lines from the local help files are automatically added to this section. There you can see which local help files have been added and jump to them through the tag.

For writing a local help file, see [write-local-help](#) .

05.8 The option window

If you are looking for an option that does what you want, you can search in the help files here: `options` . Another way is by using this command:

```
:options
```

This opens a new window, with a list of options with a one-line explanation. The options are grouped by subject. Move the cursor to a subject and press `<Enter>` to jump there. Press `<Enter>` again to jump back. Or use **CTRL-O**.

You can change the value of an option. For example, move to the "displaying text" subject. Then move the cursor down to this line:

```
set wrap      nowrap
```

When you hit `<Enter>`, the line will change to:

```
set nowrap    wrap
```

The option has now been switched off.

Just above this line is a short description of the **'wrap'** option. Move the cursor one line up to place it in this line. Now hit `<Enter>` and you jump to the full help on the **'wrap'** option.

For options that take a number or string argument you can edit the value. Then press `<Enter>` to apply the new value. For example, move the cursor a few lines up to this line:

```
set so=0
```

Position the cursor on the zero with `"$"`. Change it into a five with `"r5"`. Then press `<Enter>` to apply the new value. When you now move the cursor around you will notice that the text starts scrolling before you reach the border. This is what the **'scrolloff'** option does, it specifies an offset from the window border where scrolling starts.

05.9 Often used options

There are an awful lot of options. Most of them you will hardly ever use. Some of the more useful ones will be mentioned here. Don't forget you can find more help on these options with the `:"help"` command, with single quotes before and after the option name. For example:

```
:"help 'wrap'
```

In case you have messed up an option value, you can set it back to the default by putting an ampersand (&) after the option name. Example:

```
:"set iskeyword&
```

NOT WRAPPING LINES

Vim normally wraps long lines, so that you can see all of the text. Sometimes it's better to let the text continue right of the window. Then you need to scroll the text left-right to see all of a long line. Switch wrapping off with this command:

```
:set nowrap
```

Vim will automatically scroll the text when you move to text that is not displayed. To see a context of ten characters, do this:

```
:set sidescroll=10
```

This doesn't change the text in the file, only the way it is displayed.

WRAPPING MOVEMENT COMMANDS

Most commands for moving around will stop moving at the start and end of a line. You can change that with the **'whichwrap'** option. This sets it to the default value:

```
:set whichwrap=b,s
```

This allows the `<BS>` key, when used in the first position of a line, to move the cursor to the end of the previous line. And the `<Space>` key moves from the end of a line to the start of the next one.

To allow the cursor keys `<Left>` and `<Right>` to also wrap, use this command:

```
:set whichwrap=b,s,<,>
```

This is still only for Normal mode. To let `<Left>` and `<Right>` do this in Insert mode as well:

```
:set whichwrap=b,s,<,>,[,]
```

There are a few other flags that can be added, see **'whichwrap'**.

VIEWING TABS

When there are tabs in a file, you cannot see where they are. To make them visible:

```
:set list
```

Now every tab is displayed as `^I`. And a `$` is displayed at the end of each line, so that you can spot trailing spaces that would otherwise go unnoticed.

A disadvantage is that this looks ugly when there are many Tabs in a file. If you have a color terminal, or are using the GUI, Vim can show the spaces and tabs as highlighted characters. Use the **'listchars'** option:

```
:set listchars=tab:>-,trail:-
```

Now every tab will be displayed as `">---"` (with more or less `"-"`) and trailing white space as `"-"`. Looks a lot better, doesn't it?

KEYWORDS

The **'iskeyword'** option specifies which characters can appear in a word:

```
:set iskeyword
iskeyword=@,48-57,_,192-255
```

The "@" stands for all alphabetic letters. "48-57" stands for ASCII characters 48 to 57, which are the numbers 0 to 9. "192-255" are the printable latin characters.

Sometimes you will want to include a dash in keywords, so that commands like "w" consider "upper-case" to be one word. You can do it like this:

```
:set iskeyword+==
:set iskeyword
iskeyword=@,48-57,_,192-255,-
```

If you look at the new value, you will see that Vim has added a comma for you. To remove a character use "-=". For example, to remove the underscore:

```
:set iskeyword=-_
:set iskeyword
iskeyword=@,48-57,192-255,-
```

This time a comma is automatically deleted.

ROOM FOR MESSAGES

When Vim starts there is one line at the bottom that is used for messages. When a message is long, it is either truncated, thus you can only see part of it, or the text scrolls and you have to press **<Enter>** to continue.

You can set the **'cmdheight'** option to the number of lines used for messages. Example:

```
:set cmdheight=3
```

This does mean there is less room to edit text, thus it's a compromise.

=====

Next chapter: [usr_06.txt](#) Using syntax highlighting

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_06.txt For Vim version 9.1. Last change: 2021 Nov 07

VIM USER MANUAL - by Bram Moolenaar

Using syntax highlighting

Black and white text is boring. With colors your file comes to life. This not only looks nice, it also speeds up your work. Change the colors used for the different sorts of text. Print your text, with the colors you see on the screen.

- 06.1 Switching it on
- 06.2 No or wrong colors?
- 06.3 Different colors
- 06.4 With colors or without colors
- 06.5 Printing with colors
- 06.6 Further reading

Next chapter: [usr_07.txt](#) Editing more than one file
Previous chapter: [usr_05.txt](#) Set your settings
Table of contents: [usr_toc.txt](#)

06.1 Switching it on

It all starts with one simple command:

```
:syntax enable
```

That should work in most situations to get color in your files. Vim will automagically detect the type of file and load the right syntax highlighting. Suddenly comments are blue, keywords brown and strings red. This makes it easy to overview the file. After a while you will find that black&white text slows you down!

If you always want to use syntax highlighting, put the `":syntax enable"` command in your `vimrc` file.

If you want syntax highlighting only when the terminal supports colors, you can put this in your `vimrc` file:

```
if &t_Co > 1
    syntax enable
endif
```

If you want syntax highlighting only in the GUI version, put the `":syntax enable"` command in your `gvimrc` file.

06.2 No or wrong colors?

There can be a number of reasons why you don't see colors:

- Your terminal does not support colors.
Vim will use bold, italic and underlined text, but this doesn't look very nice. You probably will want to try to get a terminal with colors. For Unix, I recommend the `xterm` from the XFree86 project:
`xfree-xterm` .

- Your terminal does support colors, but Vim doesn't know this.
Make sure your `$TERM` setting is correct. For example, when using an `xterm` that supports colors:

```
setenv TERM xterm-color
```

or (depending on your shell):

```
TERM=xterm-color; export TERM
```

The terminal name must match the terminal you are using. If it still doesn't work, have a look at `xterm-color` , which shows a few ways to make Vim display colors (not only for an `xterm`).

- The file type is not recognized.
Vim doesn't know all file types, and sometimes it's near to impossible to tell what language a file uses. Try this command:

```
:set filetype
```

If the result is `"filetype="` then the problem is indeed that Vim doesn't know what type of file this is. You can set the type manually:

```
:set filetype=fortran
```

To see which types are available, look in the directory `$VIMRUNTIME/syntax`. For the GUI you can use the Syntax menu. Setting the filetype can also be done with a `modeline` , so that the file will be highlighted each time you edit it. For example, this line can be used in a Makefile (put it near the start or end of the file):

```
# vim: syntax=make
```

You might know how to detect the file type yourself. Often the file name extension (after the dot) can be used.
See `new-filetype` for how to tell Vim to detect that file type.

- There is no highlighting for your file type.
You could try using a similar file type by manually setting it as mentioned above. If that isn't good enough, you can write your own syntax file, see `mysyntaxfile` .

Or the colors could be wrong:

- The colored text is very hard to read.

Vim guesses the background color that you are using. If it is black (or another dark color) it will use light colors for text. If it is white (or another light color) it will use dark colors for text. If Vim guessed wrong the text will be hard to read. To solve this, set the **'background'** option. For a dark background:

```
:set background=dark
```

And for a light background:

```
:set background=light
```

Make sure you put this `_before_` the `":syntax enable"` command, otherwise the colors will already have been set. You could do `":syntax reset"` after setting **'background'** to make Vim set the default colors again.

- The colors are wrong when scrolling bottom to top. Vim doesn't read the whole file to parse the text. It starts parsing wherever you are viewing the file. That saves a lot of time, but sometimes the colors are wrong. A simple fix is hitting **CTRL-L**. Or scroll back a bit and then forward again. For a real fix, see `:syn-sync`. Some syntax files have a way to make it look further back, see the help for the specific syntax file. For example, `tex.vim` for the TeX syntax.

06.3 Different colors

`:syn-default-override`

If you don't like the default colors, you can select another color scheme. In the GUI use the Edit/Color Scheme menu. You can also type the command:

```
:colorscheme evening
```

"evening" is the name of the color scheme. There are several others you might want to try out. Look in the directory `$VIMRUNTIME/colors`.

When you found the color scheme that you like, add the `":colorscheme"` command to your `vimrc` file.

You could also write your own color scheme. This is how you do it:

1. Select a color scheme that comes close. Copy this file to your own Vim directory. For Unix, this should work:

```
!mkdir ~/.vim/colors
!cp $VIMRUNTIME/colors/morning.vim ~/.vim/colors/mine.vim
```

This is done from Vim, because it knows the value of `$VIMRUNTIME`.

2. Edit the color scheme file. These entries are useful:

<code>term</code>	attributes in a B&W terminal
<code>cterm</code>	attributes in a color terminal

<code>ctermbg</code>	background color in a color terminal
<code>ctermbg</code>	background color in a color terminal
<code>gui</code>	attributes in the GUI
<code>guifg</code>	foreground color in the GUI
<code>guibg</code>	background color in the GUI

For example, to make comments green:

```
:highlight Comment ctermfg=green guifg=green
```

Attributes you can use for "cterm" and "gui" are "bold" and "underline". If you want both, use "bold,underline". For details see the `:highlight` command.

3. Tell Vim to always use your color scheme. Put this line in your `vimrc` :

```
colorscheme mine
```

If you want to see what the most often used color combinations look like, use this command:

```
:runtime syntax/colortest.vim
```

You will see text in various color combinations. You can check which ones are readable and look nice. These aren't the only colors available to you though. You can specify `#rrggbb` hex colors and you can define new names for hex colors in `v:colornames` like so:

```
let v:colornames['mine_red'] = '#aa0000'
```

If you are authoring a color scheme for others to use, it is important to define these colors only when they do not exist:

```
call extend(v:colornames, {'mine_red': '#aa0000'}, 'keep')
```

This allows users of the color scheme to override the precise definition of that color prior to loading your color scheme. For example, in a `.vimrc` file:

```
runtime colors/lists/css_colors.vim
let v:colornames['your_red'] = v:colornames['css_red']
colorscheme yourscheme
```

As a color scheme author, you should be able to rely on some color names for GUI colors. These are defined in `colors/lists/default.vim`. All such files found on the `'runtimepath'` are loaded each time the `colorscheme` command is run. A canonical list is provided by the vim distribution, which should include all X11 colors (previously defined in `rgb.txt`).

06.4 With colors or without colors

Displaying text in color takes a lot of effort. If you find the displaying too slow, you might want to disable syntax highlighting for a moment:

```
:syntax clear
```

When editing another file (or the same one) the colors will come back.

If you want to stop highlighting completely use:

```
:syntax off
```

This will completely disable syntax highlighting and remove it immediately for all buffers. See `:syntax-off` for more details.

`:syn-manual`

If you want syntax highlighting only for specific files, use this:

```
:syntax manual
```

This will enable the syntax highlighting, but not switch it on automatically when starting to edit a buffer. To switch highlighting on for the current buffer, set the `'syntax'` option:

```
:set syntax=ON
```

06.5 Printing with colors

`syntax-printing`

In the MS-Windows version you can print the current file with this command:

```
:hardcopy
```

You will get the usual printer dialog, where you can select the printer and a few settings. If you have a color printer, the paper output should look the same as what you see inside Vim. But when you use a dark background the colors will be adjusted to look good on white paper.

There are several options that change the way Vim prints:

```
'printdevice'  
'printhead'  
'printfont'  
'printoptions'
```

To print only a range of lines, use Visual mode to select the lines and then type the command:

```
v100j:hardcopy
```

"v" starts Visual mode. "100j" moves a hundred lines down, they will be highlighted. Then `:"hardcopy`" will print those lines. You can use other commands to move in Visual mode, of course.

This also works on Unix, if you have a PostScript printer. Otherwise, you will have to do a bit more work. You need to convert the text to HTML first, and then print it from a web browser.

Convert the current file to HTML with this command:

```
:TOhtml
```

In case that doesn't work:

```
:source $VIMRUNTIME/syntax/2html.vim
```

You will see it crunching away, this can take quite a while for a large file. Some time later another window shows the HTML code. Now write this somewhere (doesn't matter where, you throw it away later):

```
:write main.c.html
```

Open this file in your favorite browser and print it from there. If all goes well, the output should look exactly as it does in Vim. See [2html.vim](#) for details. Don't forget to delete the HTML file when you are done with it.

Instead of printing, you could also put the HTML file on a web server, and let others look at the colored text.

06.6 Further reading

[usr_44.txt](#) Your own syntax highlighted.
[syntax](#) All the details.

Next chapter: [usr_07.txt](#) Editing more than one file

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_07.txt For Vim version 9.1. Last change: 2020 Mar 23

VIM USER MANUAL - by Bram Moolenaar

Editing more than one file

No matter how many files you have, you can edit them without leaving Vim. Define a list of files to work on and jump from one to the other. Copy text from one file and put it in another one.

- 07.1 Edit another file
- 07.2 A list of files
- 07.3 Jumping from file to file
- 07.4 Backup files
- 07.5 Copy text between files
- 07.6 Viewing a file
- 07.7 Changing the file name

Next chapter: [usr_08.txt](#) Splitting windows
Previous chapter: [usr_06.txt](#) Using syntax highlighting
Table of contents: [usr_toc.txt](#)

07.1 Edit another file

So far you had to start Vim for every file you wanted to edit. There is a simpler way. To start editing another file, use this command:

```
:edit foo.txt
```

You can use any file name instead of "foo.txt". Vim will close the current file and open the new one. If the current file has unsaved changes, however, Vim displays an error message and does not open the new file:

```
E37: No write since last change (use ! to override)
```

Note:

Vim puts an error ID at the start of each error message. If you do not understand the message or what caused it, look in the help system for this ID. In this case:

```
:help E37
```

At this point, you have a number of alternatives. You can write the file using this command:

```
:write
```

Or you can force Vim to discard your changes and edit the new file, using the force (!) character:

```
:edit! foo.txt
```

If you want to edit another file, but not write the changes in the current file yet, you can make it hidden:

```
:hide edit foo.txt
```

The text with changes is still there, but you can't see it. This is further explained in section [22.4](#) : The buffer list.

07.2 A list of files

You can start Vim to edit a sequence of files. For example:

```
vim one.c two.c three.c
```

This command starts Vim and tells it that you will be editing three files. Vim displays just the first file. After you have done your thing in this file, to edit the next file you use this command:

```
:next
```

If you have unsaved changes in the current file, you will get an error message and the ":next" will not work. This is the same problem as with ":edit" mentioned in the previous section. To abandon the changes:

```
:next!
```

But mostly you want to save the changes and move on to the next file. There is a special command for this:

```
:wnext
```

This does the same as using two separate commands:

```
:write  
:next
```

WHERE AM I?

To see which file in the argument list you are editing, look in the window title. It should show something like "(2 of 3)". This means you are editing the second file out of three files.

If you want to see the list of files, use this command:

```
:args
```

This is short for "arguments". The output might look like this:

```
one.c [two.c] three.c
```

These are the files you started Vim with. The one you are currently editing, "two.c", is in square brackets.

MOVING TO OTHER ARGUMENTS

To go back one file:

```
:previous
```

This is just like the ":next" command, except that it moves in the other direction. Again, there is a shortcut command for when you want to write the file first:

```
:wprevious
```

To move to the very last file in the list:

```
:last
```

And to move back to the first one again:

```
:first
```

There is no ":wlast" or ":wfirst" command though!

You can use a count for ":next" and ":previous". To skip two files forward:

```
:2next
```

AUTOMATIC WRITING

When moving around the files and making changes, you have to remember to use ":write". Otherwise you will get an error message. If you are sure you always want to write modified files, you can tell Vim to automatically write them:

```
:set autowrite
```

When you are editing a file which you may not want to write, switch it off again:

```
:set noautowrite
```

EDITING ANOTHER LIST OF FILES

You can redefine the list of files without the need to exit Vim and start it again. Use this command to edit three other files:

```
:args five.c six.c seven.h
```

Or use a wildcard, like it's used in the shell:

```
:args *.txt
```

Vim will take you to the first file in the list. Again, if the current file has changes, you can either write the file first, or use `":args!"` (with ! added) to abandon the changes.

DID YOU EDIT THE LAST FILE?

`arglist-quit`

When you use a list of files, Vim assumes you want to edit them all. To protect you from exiting too early, you will get this error when you didn't edit the last file in the list yet:

`E173: 46 more files to edit`

If you really want to exit, just do it again. Then it will work (but not when you did other commands in between).

07.3 Jumping from file to file

To quickly jump between two files, press `CTRL-^` (on English-US keyboards the ^ is above the 6 key). Example:

```
:args one.c two.c three.c
```

You are now in one.c.

```
:next
```

Now you are in two.c. Now use `CTRL-^` to go back to one.c. Another `CTRL-^` and you are back in two.c. Another `CTRL-^` and you are in one.c again. If you now do:

```
:next
```

You are in three.c. Notice that the `CTRL-^` command does not change the idea of where you are in the list of files. Only commands like `":next"` and `":previous"` do that.

The file you were previously editing is called the "alternate" file. When you just started Vim `CTRL-^` will not work, since there isn't a previous file.

PREDEFINED MARKS

After jumping to another file, you can use two predefined marks which are very useful:

```
`"
```

This takes you to the position where the cursor was when you left the file. Another mark that is remembered is the position where you made the last change:

```
`.
```

Suppose you are editing the file "one.txt". Somewhere halfway through the file you use "x" to delete a character. Then you go to the last line with "G" and write the file with ":w". You edit several other files, and then use ":edit one.txt" to come back to "one.txt". If you now use `` Vim jumps to the last line of the file. Using `.` takes you to the position where you deleted the character. Even when you move around in the file `` and `.` will take you to the remembered position. At least until you make another change or leave the file.

FILE MARKS

In section 03.10 was explained how you can place a mark in a file with "mx" and jump to that position with "`x". That works within one file. If you edit another file and place marks there, these are specific for that file. Thus each file has its own set of marks, they are local to the file.

So far we were using marks with a lowercase letter. There are also marks with an uppercase letter. These are global, they can be used from any file. For example suppose that we are editing the file "foo.txt". Go to halfway down the file ("50%") and place the F mark there (F for foo):

```
50%mF
```

Now edit the file "bar.txt" and place the B mark (B for bar) at its last line:

```
GmB
```

Now you can use the "'F" command to jump back to halfway foo.txt. Or edit yet another file, type "'B" and you are at the end of bar.txt again.

The file marks are remembered until they are placed somewhere else. Thus you can place the mark, do hours of editing and still be able to jump back to that mark.

It's often useful to think of a simple connection between the mark letter and where it is placed. For example, use the H mark in a header file, M in a Makefile and C in a C code file.

To see where a specific mark is, give an argument to the ":marks" command:

```
:marks M
```

You can also give several arguments:

```
:marks MCP
```

Don't forget that you can use **CTRL-O** and **CTRL-I** to jump to older and newer positions without placing marks there.

07.4 Backup files

Usually Vim does not produce a backup file. If you want to have one, all you need to do is execute the following command:

```
:set backup
```

The name of the backup file is the original file with a ~ added to the end. If your file is named data.txt, for example, the backup file name is data.txt~.

If you do not like the fact that the backup files end with ~, you can change the extension:

```
:set backupext=.bak
```

This will use data.txt.bak instead of data.txt~.

Another option that matters here is **'backupdir'**. It specifies where the backup file is written. The default, to write the backup in the same directory as the original file, will mostly be the right thing.

Note:

When the **'backup'** option isn't set but the **'writebackup'** is, Vim will still create a backup file. However, it is deleted as soon as writing the file was completed successfully. This functions as a safety against losing your original file when writing fails in some way (disk full is the most common cause; being hit by lightning might be another, although less common).

KEEPING THE ORIGINAL FILE

If you are editing source files, you might want to keep the file before you make any changes. But the backup file will be overwritten each time you write the file. Thus it only contains the previous version, not the first one.

To make Vim keep the original file, set the **'patchmode'** option. This specifies the extension used for the first backup of a changed file. Usually you would do this:

```
:set patchmode=.orig
```

When you now edit the file data.txt for the first time, make changes and write the file, Vim will keep a copy of the unchanged file under the name "data.txt.orig".

If you make further changes to the file, Vim will notice that "data.txt.orig" already exists and leave it alone. Further backup files will then be called "data.txt~" (or whatever you specified with **'backupext'**).

If you leave **'patchmode'** empty (that is the default), the original file will not be kept.

07.5 Copy text between files

This explains how to copy text from one file to another. Let's start with a simple example. Edit the file that contains the text you want to copy. Move the cursor to the start of the text and press "v". This starts Visual mode. Now move the cursor to the end of the text and press "y". This yanks (copies) the selected text.

To copy the above paragraph, you would do:

```
:edit thisfile
/This
vjjjj$y
```

Now edit the file you want to put the text in. Move the cursor to the character where you want the text to appear after. Use "p" to put the text there.

```
:edit otherfile
/There
p
```

Of course you can use many other commands to yank the text. For example, to select whole lines start Visual mode with "V". Or use **CTRL-V** to select a rectangular block. Or use "Y" to yank a single line, "yaw" to yank-a-word, etc.

The "p" command puts the text after the cursor. Use "P" to put the text before the cursor. Notice that Vim remembers if you yanked a whole line or a block, and puts it back that way.

USING REGISTERS

When you want to copy several pieces of text from one file to another, having to switch between the files and writing the target file takes a lot of time. To avoid this, copy each piece of text to its own register.

A register is a place where Vim stores text. Here we will use the registers named a to z (later you will find out there are others). Let's copy a sentence to the f register (f for First):

```
"fyas
```

The "yas" command yanks a sentence like before. It's the "f" that tells Vim the text should be placed in the f register. This must come just before the yank command.

Now yank three whole lines to the l register (l for line):

```
"l3Y
```

The count could be before the "l" just as well. To yank a block of text to the b (for block) register:

```
CTRL-Vjjw"by
```

Notice that the register specification "b" is just before the "y" command. This is required. If you would have put it before the "w" command, it would not have worked.

Now you have three pieces of text in the f, l and b registers. Edit another file, move around and place the text where you want it:

```
"fp
```

Again, the register specification "f" comes before the "p" command.

You can put the registers in any order. And the text stays in the register

until you yank something else into it. Thus you can put it as many times as you like.

When you delete text, you can also specify a register. Use this to move several pieces of text around. For example, to delete-a-word and write it in the w register:

```
"wdaw
```

Again, the register specification comes before the delete command "d".

APPENDING TO A FILE

When collecting lines of text into one file, you can use this command:

```
:write >> logfile
```

This will write the text of the current file to the end of "logfile". Thus it is appended. This avoids that you have to copy the lines, edit the log file and put them there. Thus you save two steps. But you can only append to the end of a file.

To append only a few lines, select them in Visual mode before typing ":write". In chapter 10 you will learn other ways to select a range of lines.

07.6 Viewing a file

Sometimes you only want to see what a file contains, without the intention to ever write it back. There is the risk that you type ":w" without thinking and overwrite the original file anyway. To avoid this, edit the file read-only.

To start Vim in readonly mode, use this command:

```
vim -R file
```

On Unix this command should do the same thing:

```
view file
```

You are now editing "file" in read-only mode. When you try using ":w" you will get an error message and the file won't be written.

When you try to make a change to the file Vim will give you a warning:

```
W10: Warning: Changing a readonly file
```

The change will be done though. This allows for formatting the file, for example, to be able to read it easily.

If you make changes to a file and forgot that it was read-only, you can still write it. Add the ! to the write command to force writing.

If you really want to forbid making changes in a file, do this:

```
vim -M file
```

Now every attempt to change the text will fail. The help files are like this, for example. If you try to make a change you get this error message:

```
E21: Cannot make changes, 'modifiable' is off
```

You could use the `-M` argument to setup Vim to work in a viewer mode. This is only voluntary though, since these commands will remove the protection:

```
:set modifiable
:set write
```

07.7 Changing the file name

A clever way to start editing a new file is by using an existing file that contains most of what you need. For example, you start writing a new program to move a file. You know that you already have a program that copies a file, thus you start with:

```
:edit copy.c
```

You can delete the stuff you don't need. Now you need to save the file under a new name. The `:"saveas` command can be used for this:

```
:saveas move.c
```

Vim will write the file under the given name, and edit that file. Thus the next time you do `:"write`", it will write `move.c`. `copy.c` remains unmodified.

When you want to change the name of the file you are editing, but don't want to write the file, you can use this command:

```
:file move.c
```

Vim will mark the file as "not edited". This means that Vim knows this is not the file you started editing. When you try to write the file, you might get this message:

```
E13: File exists (use ! to override)
```

This protects you from accidentally overwriting another file.

Next chapter: [usr_08.txt](#) Splitting windows

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Splitting windows

Display two different files above each other. Or view two locations in the file at the same time. See the difference between two files by putting them side by side. All this is possible with split windows.

- 08.1 Split a window
- 08.2 Split a window on another file
- 08.3 Window size
- 08.4 Vertical splits
- 08.5 Moving windows
- 08.6 Commands for all windows
- 08.7 Viewing differences with vimdiff
- 08.8 Various
- 08.9 Tab pages

Next chapter: [usr_09.txt](#) Using the GUI
Previous chapter: [usr_07.txt](#) Editing more than one file
Table of contents: [usr_toc.txt](#)

08.1 Split a window

The easiest way to open a new window is to use the following command:

```
:split
```

This command splits the screen into two windows and leaves the cursor in the top one:

```
+-----+
|/* file one.c */|
|~|
|~|
|one.c=====|
|/* file one.c */|
|~|
|one.c=====|
| |
+-----+
```

What you see here is two windows on the same file. The line with "=====" is the status line. It displays information about the window above it. (In practice the status line will be in reverse video.)

The two windows allow you to view two parts of the same file. For example, you could make the top window show the variable declarations of a program, and the bottom one the code that uses these variables.

The **CTRL-W** w command can be used to jump between the windows. If you are in

the top window, **CTRL-W w** jumps to the window below it. If you are in the bottom window it will jump to the first window. (CTRL-W **CTRL-W** does the same thing, in case you let go of the CTRL key a bit later.)

CLOSE THE WINDOW

To close a window, use the command:

```
:close
```

Actually, any command that quits editing a file works, like ":quit" and "ZZ". But ":close" prevents you from accidentally exiting Vim when you close the last window.

CLOSING ALL OTHER WINDOWS

If you have opened a whole bunch of windows, but now want to concentrate on one of them, this command will be useful:

```
:only
```

This closes all windows, except for the current one. If any of the other windows has changes, you will get an error message and that window won't be closed.



08.2 Split a window on another file

The following command opens a second window and starts editing the given file:

```
:split two.c
```

If you were editing one.c, then the result looks like this:

```
+-----+
|/* file two.c */|
|~|
|~|
|two.c=====|
|/* file one.c */|
|~|
|one.c=====|
|_|
+-----+
```

To open a window on a new, empty file, use this:

```
:new
```

You can repeat the ":split" and ":new" commands to create as many windows as you like.

08.3 Window size

The `:split` command can take a number argument. If specified, this will be the height of the new window. For example, the following opens a new window three lines high and starts editing the file `alpha.c`:

```
:3split alpha.c
```

For existing windows you can change the size in several ways. When you have a working mouse, it is easy: Move the mouse pointer to the status line that separates two windows, and drag it up or down.

To increase the size of a window:

```
CTRL-W +
```

To decrease it:

```
CTRL-W -
```

Both of these commands take a count and increase or decrease the window size by that many lines. Thus `"4 CTRL-W +"` make the window four lines higher.

To set the window height to a specified number of lines:

```
{height}CTRL-W _
```

That's: a number `{height}`, `CTRL-W` and then an underscore (the `-` key with Shift on English-US keyboards).

To make a window as high as it can be, use the `CTRL-W _` command without a count.

USING THE MOUSE

In Vim you can do many things very quickly from the keyboard. Unfortunately, the window resizing commands require quite a bit of typing. In this case, using the mouse is faster. Position the mouse pointer on a status line. Now press the left mouse button and drag. The status line will move, thus making the window on one side higher and the other smaller.

OPTIONS

The `'winheight'` option can be set to a minimal desired height of a window and `'winminheight'` to a hard minimum height.

Likewise, there is `'winwidth'` for the minimal desired width and `'winminwidth'` for the hard minimum width.

The `'equalalways'` option, when set, makes Vim equalize the windows sizes when a window is closed or opened.

08.4 Vertical splits

The ":split" command creates the new window above the current one. To make the window appear at the left side, use:

```
:vsplit
```

or:

```
:vsplit two.c
```

The result looks something like this:

```
+-----+
|/* file two.c */  |/* file one.c */ |
|~                 |~                 |
|~                 |~                 |
|~                 |~                 |
|two.c=====one.c=====|
|                 |                 |
+-----+
```

Actually, the | lines in the middle will be in reverse video. This is called the vertical separator. It separates the two windows left and right of it.

There is also the ":vnew" command, to open a vertically split window on a new, empty file. Another way to do this:

```
:vertical new
```

The ":vertical" command can be inserted before another command that splits a window. This will cause that command to split the window vertically instead of horizontally. (If the command doesn't split a window, it works unmodified.)

MOVING BETWEEN WINDOWS

Since you can split windows horizontally and vertically as much as you like, you can create almost any layout of windows. Then you can use these commands to move between them:

CTRL-W h	move to the window on the left
CTRL-W j	move to the window below
CTRL-W k	move to the window above
CTRL-W l	move to the window on the right
CTRL-W t	move to the TOP window
CTRL-W b	move to the BOTTOM window

You will notice the same letters as used for moving the cursor. And the cursor keys can also be used, if you like.

More commands to move to other windows: [Q_wi](#) .

08.5 Moving windows

You have split a few windows, but now they are in the wrong place. Then you need a command to move the window somewhere else. For example, you have three windows like this:

```
+-----+
|/* file two.c */|
|~|
|~|
|two.c=====|
|/* file three.c */|
|~|
|~|
|three.c=====|
|/* file one.c */|
|~|
|one.c=====|
|_|
+-----+
```

Clearly the last one should be at the top. Go to that window (using **CTRL-W w**) and then type this command:

CTRL-W K

This uses the uppercase letter K. What happens is that the window is moved to the very top. You will notice that K is again used for moving upwards.

When you have vertical splits, **CTRL-W K** will move the current window to the top and make it occupy the full width of the Vim window. If this is your layout:

```
+-----+
|/* two.c */ |/* three.c */ |/* one.c */ | | |
|~| |~| |~|
|~| |~| |~|
|~| |~| |~|
|~| |~| |~|
|~| |~| |~|
|two.c=====three.c=====one.c=====|
|_|
+-----+
```

Then using **CTRL-W K** in the middle window (three.c) will result in:

```
+-----+
|/* three.c */|
|~|
|~|
|three.c=====|
|/* two.c */ |/* one.c */|
|~| |~|
|two.c=====one.c=====|
|_|
+-----+
```

The other three similar commands (you can probably guess these now):

```
CTRL-W H      move window to the far left
CTRL-W J      move window to the bottom
CTRL-W L      move window to the far right
```

08.6 Commands for all windows

When you have several windows open and you want to quit Vim, you can close each window separately. A quicker way is using this command:

```
:qall
```

This stands for "quit all". If any of the windows contain changes, Vim will not exit. The cursor will automatically be positioned in a window with changes. You can then either use ":write" to save the changes, or ":quit!" to throw them away.

If you know there are windows with changes, and you want to save all these changes, use this command:

```
:wall
```

This stands for "write all". But actually, it only writes files with changes. Vim knows it doesn't make sense to write files that were not changed.

And then there is the combination of ":qall" and ":wall": the "write and quit all" command:

```
:wqall
```

This writes all modified files and quits Vim.

Finally, there is a command that quits Vim and throws away all changes:

```
:qall!
```

Be careful, there is no way to undo this command!

OPENING A WINDOW FOR ALL ARGUMENTS

To make Vim open a window for each file, start it with the "-o" argument:

```
vim -o one.txt two.txt three.txt
```

This results in:

```
+-----+
|file one.txt|
|~          |
|one.txt=====|
|file two.txt|
```

```

|~
|two.txt=====|
|file three.txt|
|~
|three.txt=====|
|
+-----+

```

The "-0" argument is used to get vertically split windows.

When Vim is already running, the ":all" command opens a window for each file in the argument list. ":vertical all" does it with vertical splits.

08.7 Viewing differences with vimdiff

There is a special way to start Vim, which shows the differences between two files. Let's take a file "main.c" and insert a few characters in one line. Write this file with the '**backup**' option set, so that the backup file "main.c~" will contain the previous version of the file.

Type this command in a shell (not in Vim):

```
vimdiff main.c~ main.c
```

Vim will start, with two windows side by side. You will only see the line in which you added characters, and a few lines above and below it.

```

VV                VV
+-----+
|+ +--123 lines: /* a|+ +--123 lines: /* a| <- fold
| text              | text
| text              | text
| text              | text
| text              | changed text    <- changed line
| text              | text
| text              | -----      <- deleted line
| text              | text
| text              | text
| text              | text
|+ +--432 lines: text|+ +--432 lines: text| <- fold
| ~                 | ~
| ~                 | ~
|main.c~=====main.c=====
|
+-----+

```

(This picture doesn't show the highlighting, use the vimdiff command for a better look.)

The lines that were not modified have been collapsed into one line. This is called a closed fold. They are indicated in the picture with "<- fold". Thus the single fold line at the top stands for 123 text lines. These lines are equal in both files.

The line marked with "<- changed line" is highlighted, and the inserted text is displayed with another color. This clearly shows what the difference

is between the two files.

The line that was deleted is displayed with "---" in the main.c window. See the "<- deleted line" marker in the picture. These characters are not really there. They just fill up main.c, so that it displays the same number of lines as the other window.

THE FOLD COLUMN

Each window has a column on the left with a slightly different background. In the picture above these are indicated with "VV". You notice there is a plus character there, in front of each closed fold. Move the mouse pointer to that plus and click the left button. The fold will open, and you can see the text that it contains.

The fold column contains a minus sign for an open fold. If you click on this -, the fold will close.

Obviously, this only works when you have a working mouse. You can also use "zo" to open a fold and "zc" to close it.

DIFFING IN VIM

Another way to start in diff mode can be done from inside Vim. Edit the "main.c" file, then make a split and show the differences:

```
:edit main.c
:vertical difffsplit main.c~
```

The ":vertical" command is used to make the window split vertically. If you omit this, you will get a horizontal split.

If you have a patch or diff file, you can use the third way to start diff mode. First edit the file to which the patch applies. Then tell Vim the name of the patch file:

```
:edit main.c
:vertical diffpatch main.c.diff
```

WARNING: The patch file must contain only one patch, for the file you are editing. Otherwise you will get a lot of error messages, and some files might be patched unexpectedly.

The patching will only be done to the copy of the file in Vim. The file on your harddisk will remain unmodified (until you decide to write the file).

SCROLL BINDING

When the files have more changes, you can scroll in the usual way. Vim will try to keep both the windows start at the same position, so you can easily see the differences side by side.

When you don't want this for a moment, use this command:

```
:set noscrollbind
```


JUMPING TO CHANGES

When you have disabled folding in some way, it may be difficult to find the changes. Use this command to jump forward to the next change:

```
]c
```

To go the other way use:

```
[c
```

Prepended a count to jump further away.

REMOVING CHANGES

You can move text from one window to the other. This either removes differences or adds new ones. Vim doesn't keep the highlighting updated in all situations. To update it use this command:

```
:diffupdate
```

To remove a difference, you can move the text in a highlighted block from one window to another. Take the "main.c" and "main.c~" example above. Move the cursor to the left window, on the line that was deleted in the other window. Now type this command:

```
dp
```

The change will be removed by putting the text of the current window in the other window. "dp" stands for "diff put".

You can also do it the other way around. Move the cursor to the right window, to the line where "changed" was inserted. Now type this command:

```
do
```

The change will now be removed by getting the text from the other window. Since there are no changes left now, Vim puts all text in a closed fold. "do" stands for "diff obtain". "dg" would have been better, but that already has a different meaning ("dgg" deletes from the cursor until the first line).

For details about diff mode, see [vimdiff](#) .

08.8 Various

The **'laststatus'** option can be used to specify when the last window has a statusline:

0	never
1	only when there are split windows (the default)
2	always

Many commands that edit another file have a variant that splits the window. For Command-line commands this is done by prepending an "s". For example: ":tag" jumps to a tag, ":stag" splits the window and jumps to a tag.

For Normal mode commands a **CTRL-W** is prepended. **CTRL-^** jumps to the alternate file, **CTRL-W CTRL-^** splits the window and edits the alternate file.

The '**splitbelow**' option can be set to make a new window appear below the current window. The '**splitright**' option can be set to make a vertically split window appear right of the current window.

When splitting a window you can prepend a modifier command to tell where the window is to appear:

:leftabove {cmd}	left or above the current window
:aboveleft {cmd}	idem
:rightbelow {cmd}	right or below the current window
:belowright {cmd}	idem
:topleft {cmd}	at the top or left of the Vim window
:botright {cmd}	at the bottom or right of the Vim window

08.9 Tab pages

You will have noticed that windows never overlap. That means you quickly run out of screen space. The solution for this is called Tab pages.

Assume you are editing "thisfile". To create a new tab page use this command:

```
:tabedit thatfile
```

This will edit the file "thatfile" in a window that occupies the whole Vim window. And you will notice a bar at the top with the two file names:

```
+-----+
| thisfile | /thatfile/ _____X| (thatfile is bold)
|/* thatfile */
|that
|that
|~
|~
|~
|
+-----+
```

You now have two tab pages. The first one has a window for "thisfile" and the second one a window for "thatfile". It's like two pages that are on top of each other, with a tab sticking out of each page showing the file name.

Now use the mouse to click on "thisfile" in the top line. The result is

```
+-----+
| /thisfile/ | thatfile _____X| (thisfile is bold)
```

```

|/* thisfile */
|this
|this
|~
|~
|~
+-----+

```

Thus you can switch between tab pages by clicking on the label in the top line. If you don't have a mouse or don't want to use it, you can use the "gt" command. Mnemonic: Goto Tab.

Now let's create another tab page with the command:

```
:tab split
```

This makes a new tab page with one window that is editing the same buffer as the window we were in:

```

+-----+
| thisfile | /thisfile/ | thatfile __X| (thisfile is bold)
|/* thisfile */
|this
|this
|~
|~
|~
+-----+

```

You can put ":tab" before any Ex command that opens a window. The window will be opened in a new tab page. Another example:

```
:tab help gt
```

Will show the help text for "gt" in a new tab page.

A few more things you can do with tab pages:

- click with the mouse in the space after the last label
The next tab page will be selected, like with "gt".
- click with the mouse on the "X" in the top right corner
The current tab page will be closed. Unless there are unsaved changes in the current tab page.
- double click with the mouse in the top line
A new tab page will be created.
- the "tabonly" command
Closes all tab pages except the current one. Unless there are unsaved changes in other tab pages.

For more information about tab pages see [tab-page](#) .



Next chapter: [usr_09.txt](#) Using the GUI

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Using the GUI

Vim works in an ordinary terminal, while gVim has a Graphical User Interface (GUI). It can do the same things and a few more. The GUI offers menus, a toolbar, scrollbars and other items. This chapter is about these extra things that the GUI offers.

- 09.1 Parts of the GUI
- 09.2 Using the mouse
- 09.3 The clipboard
- 09.4 Select mode

Next chapter: [usr_10.txt](#) Making big changes
 Previous chapter: [usr_08.txt](#) Splitting windows
 Table of contents: [usr_toc.txt](#)

09.1 Parts of the GUI

You might have an icon on your desktop that starts gvim. Otherwise, one of these commands should do it:

```
gvim file.txt
vim -g file.txt
```

If this doesn't work you don't have a version of Vim with GUI support. You will have to install one first.

Vim will open a window and display "file.txt" in it. What the window looks like depends on the version of Vim. It should resemble the following picture (for as far as this can be shown in ASCII!).

```
+-----+
| file.txt + (~/dir) - VIM                               X | <- window title
+-----+
| File Edit Tools Syntax Buffers Window Help           | <- menubar
+-----+
| aaa bbb ccc ddd eee fff ggg hhh iii jjj             | <- toolbar
| aaa bbb ccc ddd eee fff ggg hhh iii jjj             |
+-----+
| file text                                             | ^ |
| ~                                                     | # |
| ~                                                     | # | <- scrollbar
| ~                                                     | # |
| ~                                                     | # |
| ~                                                     | # |
| ~                                                     | V |
+-----+
```

The largest space is occupied by the file text. This shows the file in the

same way as in a terminal. With some different colors and another font perhaps.

THE WINDOW TITLE

At the very top is the window title. This is drawn by your window system. Vim will set the title to show the name of the current file. First comes the name of the file. Then some special characters and the directory of the file in parens. These special characters can be present:

- The file cannot be modified (e.g., a help file)
- + The file contains changes
- = The file is read-only
- =+ The file is read-only, contains changes anyway

If nothing is shown you have an ordinary, unchanged file.

THE MENUBAR

You know how menus work, right? Vim has the usual items, plus a few more. Browse them to get an idea of what you can use them for. A relevant submenu is Edit/Global Settings. You will find these entries:

- Toggle Toolbar make the toolbar appear/disappear
- Toggle Bottom Scrollbar make a scrollbar appear/disappear at the bottom
- Toggle Left Scrollbar make a scrollbar appear/disappear at the left
- Toggle Right Scrollbar make a scrollbar appear/disappear at the right

On most systems you can tear-off the menus. Select the top item of the menu, the one that looks like a dashed line. You will get a separate window with the items of the menu. It will hang around until you close the window.

THE TOOLBAR

This contains icons for the most often used actions. Hopefully the icons are self-explanatory. There are tooltips to get an extra hint (move the mouse pointer to the icon without clicking and don't move it for a second).

The "Edit/Global Settings/Toggle Toolbar" menu item can be used to make the toolbar disappear. If you never want a toolbar, use this command in your vimrc file:

```
:set guioptions-=T
```

This removes the 'T' flag from the **'guioptions'** option. Other parts of the GUI can also be enabled or disabled with this option. See the help for it.

THE SCROLLBARS

By default there is one scrollbar on the right. It does the obvious thing.

When you split the window, each window will get its own scrollbar.

You can make a horizontal scrollbar appear with the menu item Edit/Global Settings/Toggle Bottom Scrollbar. This is useful in diff mode, or when the `'wrap'` option has been reset (more about that later).

When there are vertically split windows, only the windows on the right side will have a scrollbar. However, when you move the cursor to a window on the left, it will be this one that the scrollbar controls. This takes a bit of time to get used to.

When you work with vertically split windows, consider adding a scrollbar on the left. This can be done with a menu item, or with the `'guioptions'` option:

```
:set guioptions+=l
```

This adds the `'l'` flag to `'guioptions'`.

09.2 Using the mouse

Standards are wonderful. In Microsoft Windows, you can use the mouse to select text in a standard manner. The X Window system also has a standard system for using the mouse. Unfortunately, these two standards are not the same.

Fortunately, you can customize Vim. You can make the behavior of the mouse work like an X Window system mouse or a Microsoft Windows mouse. The following command makes the mouse behave like an X Window mouse:

```
:behave xterm
```

The following command makes the mouse work like a Microsoft Windows mouse:

```
:behave mswin
```

The default behavior of the mouse on UNIX systems is `xterm`. The default behavior on a Microsoft Windows system is selected during the installation process. For details about what the two behaviors are, see `:behave`. Here follows a summary.

XTERM MOUSE BEHAVIOR

Left mouse click	position the cursor
Left mouse drag	select text in Visual mode
Middle mouse click	paste text from the clipboard
Right mouse click	extend the selected text until the mouse pointer

MSWIN MOUSE BEHAVIOR

Left mouse click	position the cursor
Left mouse drag	select text in Select mode (see 09.4)
Left mouse click, with Shift	extend the selected text until the mouse pointer

Middle mouse click	paste text from the clipboard
Right mouse click	display a pop-up menu

The mouse can be further tuned. Check out these options if you want to change the way how the mouse works:

<code>'mouse'</code>	in which mode the mouse is used by Vim
<code>'mousemodel'</code>	what effect a mouse click has
<code>'mousetime'</code>	time between clicks for a double-click
<code>'mousehide'</code>	hide the mouse while typing
<code>'selectmode'</code>	whether the mouse starts Visual or Select mode

09.3 The clipboard

In section 04.7 the basic use of the clipboard was explained. There is one essential thing to explain about X-windows: There are actually two places to exchange text between programs. MS-Windows doesn't have this.

In X-Windows there is the "current selection". This is the text that is currently highlighted. In Vim this is the Visual area (this assumes you are using the default option settings). You can paste this selection in another application without any further action.

For example, in this text select a few words with the mouse. Vim will switch to Visual mode and highlight the text. Now start another `gvim`, without a file name argument, so that it displays an empty window. Click the middle mouse button. The selected text will be inserted.

The "current selection" will only remain valid until some other text is selected. After doing the paste in the other `gvim`, now select some characters in that window. You will notice that the words that were previously selected in the other `gvim` window are displayed differently. This means that it no longer is the current selection.

You don't need to select text with the mouse, using the keyboard commands for Visual mode works just as well.

THE REAL CLIPBOARD

Now for the other place with which text can be exchanged. We call this the "real clipboard", to avoid confusion. Often both the "current selection" and the "real clipboard" are called clipboard, you'll have to get used to that.

To put text on the real clipboard, select a few different words in one of the `gvims` you have running. Then use the Edit/Copy menu entry. Now the text has been copied to the real clipboard. You can't see this, unless you have some application that shows the clipboard contents (e.g., KDE's Klipper).

Now select the other `gvim`, position the cursor somewhere and use the Edit/Paste menu. You will see the text from the real clipboard is inserted.

USING BOTH

This use of both the "current selection" and the "real clipboard" might sound a bit confusing. But it is very useful. Let's show this with an example. Use one `gvim` with a text file and perform these actions:

- Select two words in Visual mode.
- Use the Edit/Copy menu to get these words onto the clipboard.
- Select one other word in Visual mode.
- Use the Edit/Paste menu item. What will happen is that the single selected word is replaced with the two words from the clipboard.
- Move the mouse pointer somewhere else and click the middle button. You will see that the word you just overwrote with the clipboard is inserted here.

If you use the "current selection" and the "real clipboard" with care, you can do a lot of useful editing with them.

USING THE KEYBOARD

If you don't like using the mouse, you can access the current selection and the real clipboard with two registers. The "*" register is for the current selection.

To make text become the current selection, use Visual mode. For example, to select a whole line just press "V".

To insert the current selection before the cursor:

```
"*p
```

Notice the uppercase "P". The lowercase "p" puts the text after the cursor.

The "+" register is used for the real clipboard. For example, to copy the text from the cursor position until the end of the line to the clipboard:

```
"y$
```

Remember, "y" is yank, which is Vim's copy command.

To insert the contents of the real clipboard before the cursor:

```
"+p
```

It's the same as for the current selection, but uses the plus (+) register instead of the star (*) register.

09.4 Select mode

And now something that is used more often on MS-Windows than on X-Windows. But both can do it. You already know about Visual mode. Select mode is like Visual mode, because it is also used to select text. But there is an obvious difference: When typing text, the selected text is deleted and the typed text replaces it.

To start working with Select mode, you must first enable it (for MS-Windows it is probably already enabled, but you can do this anyway):

```
:set selectmode+=mouse
```

Now use the mouse to select some text. It is highlighted like in Visual mode. Now press a letter. The selected text is deleted, and the single letter replaces it. You are in Insert mode now, thus you can continue typing.

Since typing normal text causes the selected text to be deleted, you can not use the normal movement commands "hjkl", "w", etc. Instead, use the shifted function keys. `<S-Left>` (shifted cursor left key) moves the cursor left. The selected text is changed like in Visual mode. The other shifted cursor keys do what you expect. `<S-End>` and `<S-Home>` also work.

You can tune the way Select mode works with the `'selectmode'` option.

=====

Next chapter: [usr_10.txt](#) Making big changes

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Making big changes

In chapter 4 several ways to make small changes were explained. This chapter goes into making changes that are repeated or can affect a large amount of text. The Visual mode allows doing various things with blocks of text. Use an external program to do really complicated things.

- 10.1 Record and playback commands
- 10.2 Substitution
- 10.3 Command ranges
- 10.4 The global command
- 10.5 Visual block mode
- 10.6 Reading and writing part of a file
- 10.7 Formatting text
- 10.8 Changing case
- 10.9 Using an external program

Next chapter: [usr_11.txt](#) Recovering from a crash
Previous chapter: [usr_09.txt](#) Using the GUI
Table of contents: [usr_toc.txt](#)

10.1 Record and playback commands

The "." command repeats the preceding change. But what if you want to do something more complex than a single change? That's where command recording comes in. There are three steps:

1. The "q{register}" command starts recording keystrokes into the register named {register}. The register name must be between a and z.
2. Type your commands.
3. To finish recording, press q (without any extra character).

You can now execute the macro by typing the command "@{register}".

Take a look at how to use these commands in practice. You have a list of filenames that look like this:

```
stdio.h
fcntl.h
unistd.h
stdlib.h
```

And what you want is the following:

```
#include "stdio.h"
#include "fcntl.h"
#include "unistd.h"
#include "stdlib.h"
```

You start by moving to the first character of the first line. Next you execute the following commands:

qa	Start recording a macro in register a.
^	Move to the beginning of the line.
i#include "<Esc>	Insert the string #include " at the beginning of the line.
\$	Move to the end of the line.
a"<Esc>	Append the character double quotation mark (") to the end of the line.
j	Go to the next line.
q	Stop recording the macro.

Now that you have done the work once, you can repeat the change by typing the command "@a" three times.

The "@a" command can be preceded by a count, which will cause the macro to be executed that number of times. In this case you would type:

```
3@a
```

MOVE AND EXECUTE

You might have the lines you want to change in various places. Just move the cursor to each location and use the "@a" command. If you have done that once, you can do it again with "@@". That's a bit easier to type. If you now execute register b with "@b", the next "@@" will use register b.

If you compare the playback method with using ".", there are several differences. First of all, "." can only repeat one change. As seen in the example above, "@a" can do several changes, and move around as well. Secondly, "." can only remember the last change. Executing a register allows you to make any changes and then still use "@a" to replay the recorded commands. Finally, you can use 26 different registers. Thus you can remember 26 different command sequences to execute.

USING REGISTERS

The registers used for recording are the same ones you used for yank and delete commands. This allows you to mix recording with other commands to manipulate the registers.

Suppose you have recorded a few commands in register n. When you execute this with "@n" you notice you did something wrong. You could try recording again, but perhaps you will make another mistake. Instead, use this trick:

G	Go to the end of the file.
o<Esc>	Create an empty line.
"np	Put the text from the n register. You now see the commands you typed as text in the file.
{edits}	Change the commands that were wrong. This is just like editing text.
0	Go to the start of the line.
"ny\$	Yank the corrected commands into the n

```
register.  
dd Delete the scratch line.
```

Now you can execute the corrected commands with "@n". (If your recorded commands include line breaks, adjust the last two items in the example to include all the lines.)

APPENDING TO A REGISTER

So far we have used a lowercase letter for the register name. To append to a register, use an uppercase letter.

Suppose you have recorded a command to change a word to register c. It works properly, but you would like to add a search for the next word to change. This can be done with:

```
qC/word<Enter>q
```

You start with "qC", which records to the c register and appends. Thus writing to an uppercase register name means to append to the register with the same letter, but lowercase.

This works both with recording and with yank and delete commands. For example, you want to collect a sequence of lines into the a register. Yank the first line with:

```
"aY
```

Now move to the second line, and type:

```
"AY
```

Repeat this command for all lines. The a register now contains all those lines, in the order you yanked them.

10.2 Substitution

find-replace

The ":substitute" command enables you to perform string replacements on a whole range of lines. The general form of this command is as follows:

```
:[range]substitute/from/to/[flags]
```

This command changes the "from" string to the "to" string in the lines specified with [range]. For example, you can change "Professor" to "Teacher" in all lines with the following command:

```
:%substitute/Professor/Teacher/
```

Note:

The ":substitute" command is almost never spelled out completely. Most of the time, people use the abbreviated version ":s". From here on the abbreviation will be used.

The "%" before the command specifies the command works on all lines. Without a range, ":s" only works on the current line. More about ranges in the next section [10.3](#) .

By default, the ":substitute" command changes only the first occurrence on each line. For example, the preceding command changes the line:

```
Professor Smith criticized Professor Johnson today.
```

to:

```
Teacher Smith criticized Professor Johnson today.
```

To change every occurrence on the line, you need to add the g (global) flag. The command:

```
:%s/Professor/Teacher/g
```

results in (starting with the original line):

```
Teacher Smith criticized Teacher Johnson today.
```

Other flags include p (print), which causes the ":substitute" command to print out the last line it changes. The c (confirm) flag tells ":substitute" to ask you for confirmation before it performs each substitution. Enter the following:

```
:%s/Professor/Teacher/c
```

Vim finds the first occurrence of "Professor" and displays the text it is about to change. You get the following prompt:

```
replace with Teacher (y/n/a/q/l/^E/^Y)?
```

At this point, you must enter one of the following answers:

y	Yes; make this change.
n	No; skip this match.
a	All; make this change and all remaining ones without further confirmation.
q	Quit; don't make any more changes.
l	Last; make this change and then quit.
CTRL-E	Scroll the text one line up.
CTRL-Y	Scroll the text one line down.

The "from" part of the substitute command is actually a pattern. The same kind as used for the search command. For example, this command only substitutes "the" when it appears at the start of a line:

```
:s/^the/these/
```

If you are substituting with a "from" or "to" part that includes a slash, you need to put a backslash before it. A simpler way is to use another character

instead of the slash. A plus, for example:

```
:s+one/two+one or two+
```

10.3 Command ranges

The ":substitute" command, and many other : commands, can be applied to a selection of lines. This is called a range.

The simple form of a range is {number},{number}. For example:

```
:1,5s/this/that/g
```

Executes the substitute command on the lines 1 to 5. Line 5 is included. The range is always placed before the command.

A single number can be used to address one specific line:

```
:54s/President/Fool/
```

Some commands work on the whole file when you do not specify a range. To make them work on the current line the "." address is used. The ":write" command works like that. Without a range, it writes the whole file. To make it write only the current line into a file:

```
:.write otherfile
```

The first line always has number one. How about the last line? The "\$" character is used for this. For example, to substitute in the lines from the cursor to the end:

```
::,$s/yes/no/
```

The "%" range that we used before, is actually a short way to say "1,\$", from the first to the last line.

USING A PATTERN IN A RANGE

Suppose you are editing a chapter in a book, and want to replace all occurrences of "grey" with "gray". But only in this chapter, not in the next one. You know that only chapter boundaries have the word "Chapter" in the first column. This command will work then:

```
?:^Chapter?,/^Chapter/s=grey=gray=g
```

You can see a search pattern is used twice. The first "?^Chapter?" finds the line above the current position that matches this pattern. Thus the ?pattern? range is used to search backwards. Similarly, "/^Chapter/" is used to search forward for the start of the next chapter.

To avoid confusion with the slashes, the "=" character was used in the substitute command here. A slash or another character would have worked as well.

ADD AND SUBTRACT

There is a slight error in the above command: If the title of the next chapter had included "grey" it would be replaced as well. Maybe that's what you wanted, but what if you didn't? Then you can specify an offset.

To search for a pattern and then use the line above it:

```
/Chapter/-1
```

You can use any number instead of the 1. To address the second line below the match:

```
/Chapter/+2
```

The offsets can also be used with the other items in a range. Look at this one:

```
:.+3,$-5
```

This specifies the range that starts three lines below the cursor and ends five lines before the last line in the file.

USING MARKS

Instead of figuring out the line numbers of certain positions, remembering them and typing them in a range, you can use marks.

Place the marks as mentioned in chapter 3. For example, use "mt" to mark the top of an area and "mb" to mark the bottom. Then you can use this range to specify the lines between the marks (including the lines with the marks):

```
:'t,'b
```

VISUAL MODE AND RANGES

You can select text with Visual mode. If you then press ":" to start a colon command, you will see this:

```
:'<,'>
```

Now you can type the command and it will be applied to the range of lines that was visually selected.

Note:

When using Visual mode to select part of a line, or using **CTRL-V** to select a block of text, the colon commands will still apply to whole lines. This might change in a future version of Vim.

The '<' and '>' are actually marks, placed at the start and end of the Visual selection. The marks remain at their position until another Visual selection is made. Thus you can use the "'<" command to jump to position where the Visual area started. And you can mix the marks with other items:


```
:',,$
```

This addresses the lines from the end of the Visual area to the end of the file.

A NUMBER OF LINES

When you know how many lines you want to change, you can type the number and then ":". For example, when you type "5:", you will get:

```
:.+.4
```

Now you can type the command you want to use. It will use the range "." (current line) until "+4" (four lines down). Thus it spans five lines.

10.4 The global command

The ":global" command is one of the more powerful features of Vim. It allows you to find a match for a pattern and execute a command there. The general form is:

```
:[range]global/{pattern}/{command}
```

This is similar to the ":substitute" command. But, instead of replacing the matched text with other text, the command `{command}` is executed.

Note:

The command executed for ":global" must be one that starts with a colon. Normal mode commands can not be used directly. The `:normal` command can do this for you.

Suppose you want to change "foobar" to "barfoo", but only in C++ style comments. These comments start with "//". Use this command:

```
:g//+s/foobar/barfoo/g
```

This starts with ":g". That is short for ":global", just like ":s" is short for ":substitute". Then the pattern, enclosed in plus characters. Since the pattern we are looking for contains a slash, this uses the plus character to separate the pattern. Next comes the substitute command that changes "foobar" into "barfoo".

The default range for the global command is the whole file. Thus no range was specified in this example. This is different from ":substitute", which works on one line without a range.

The command isn't perfect, since it also matches lines where "//" appears halfway through a line, and the substitution will also take place before the "//".

Just like with ":substitute", any pattern can be used. When you learn more complicated patterns later, you can use them here.

10.5 Visual block mode

With **CTRL-V** you can start selection of a rectangular area of text. There are a few commands that do something special with the text block.

There is something special about using the "\$" command in Visual block mode. When the last motion command used was "\$", all lines in the Visual selection will extend until the end of the line, also when the line with the cursor is shorter. This remains effective until you use a motion command that moves the cursor horizontally. Thus using "j" keeps it, "h" stops it.

INSERTING TEXT

The command "I{string}<Esc>" inserts the text {string} in each line, just left of the visual block. You start by pressing **CTRL-V** to enter visual block mode. Now you move the cursor to define your block. Next you type I to enter Insert mode, followed by the text to insert. As you type, the text appears on the first line only.

After you press <Esc> to end the insert, the text will magically be inserted in the rest of the lines contained in the visual selection. Example:

```
include one
include two
include three
include four
```

Move the cursor to the "o" of "one" and press **CTRL-V**. Move it down with "3j" to "four". You now have a block selection that spans four lines. Now type:

```
Imain.<Esc>
```

The result:

```
include main.one
include main.two
include main.three
include main.four
```

If the block spans short lines that do not extend into the block, the text is not inserted in that line. For example, make a Visual block selection that includes the word "long" in the first and last line of this text, and thus has no text selected in the second line:

```
This is a long line
short
Any other long line
```

```
^^^^ selected block
```

Now use the command "Ivery <Esc>". The result is:

```
This is a very long line
```

```
short
Any other very long line
```

In the short line no text was inserted.

If the string you insert contains a newline, the "I" acts just like a Normal insert command and affects only the first line of the block.

The "A" command works the same way, except that it appends after the right side of the block. And it does insert text in a short line. Thus you can make a choice whether you do or don't want to append text to a short line.

There is one special case for "A": Select a Visual block and then use "\$" to make the block extend to the end of each line. Using "A" now will append the text to the end of each line.

Using the same example from above, and then typing "\$A XXX<Esc>", you get this result:

```
This is a long line XXX
short XXX
Any other long line XXX
```

This really requires using the "\$" command. Vim remembers that it was used. Making the same selection by moving the cursor to the end of the longest line with other movement commands will not have the same result.

CHANGING TEXT

The Visual block "c" command deletes the block and then throws you into Insert mode to enable you to type in a string. The string will be inserted in each line in the block.

Starting with the same selection of the "long" words as above, then typing "c_LONG_<Esc>", you get this:

```
This is a _LONG_ line
short
Any other _LONG_ line
```

Just like with "I" the short line is not changed. Also, you can't enter a newline in the new text.

The "C" command deletes text from the left edge of the block to the end of line. It then puts you in Insert mode so that you can type in a string, which is added to the end of each line.

Starting with the same text again, and typing "Cnew text<Esc>" you get:

```
This is a new text
short
Any other new text
```

Notice that, even though only the "long" word was selected, the text after it is deleted as well. Thus only the location of the left edge of the visual block really matters.

Again, short lines that do not reach into the block are excluded.

Other commands that change the characters in the block:

~	swap case	(a -> A and A -> a)
U	make uppercase	(a -> A and A -> A)
u	make lowercase	(a -> a and A -> a)

FILLING WITH A CHARACTER

To fill the whole block with one character, use the "r" command. Again, starting with the same example text from above, and then typing "rx":

```
This is a xxxx line
short
Any other xxxx line
```

Note:

If you want to include characters beyond the end of the line in the block, check out the '**virtualedit**' feature in chapter 25.

SHIFTING

The command ">" shifts the selected text to the right one shift amount, inserting whitespace. The starting point for this shift is the left edge of the visual block.

With the same example again, ">" gives this result:

```
This is a          long line
short
Any other          long line
```

The shift amount is specified with the '**shiftwidth**' option. To change it to use 4 spaces:

```
:set shiftwidth=4
```

The "<" command removes one shift amount of whitespace at the left edge of the block. This command is limited by the amount of text that is there; so if there is less than a shift amount of whitespace available, it removes what it can.

JOINING LINES

The "J" command joins all selected lines together into one line. Thus it removes the line breaks. Actually, the line break, leading white space and trailing white space is replaced by one space. Two spaces are used after a line ending (that can be changed with the '**joinspaces**' option).

Let's use the example that we got so familiar with now. The result of using the "J" command:

This is a long line short Any other long line

The "J" command doesn't require a blockwise selection. It works with "v" and "V" selection in exactly the same way.

If you don't want the white space to be changed, use the "gJ" command.

10.6 Reading and writing part of a file

When you are writing an e-mail message, you may want to include another file. This can be done with the ":read {filename}" command. The text of the file is put below the cursor line.

Starting with this text:

```
Hi John,  
Here is the diff that fixes the bug:  
Bye, Pierre.
```

Move the cursor to the second line and type:

```
:read patch
```

The file named "patch" will be inserted, with this result:

```
Hi John,  
Here is the diff that fixes the bug:  
2c2  
<      for (i = 0; i <= length; ++i)  
---  
>      for (i = 0; i < length; ++i)  
Bye, Pierre.
```

The ":read" command accepts a range. The file will be put below the last line number of this range. Thus ":\$r patch" appends the file "patch" at the end of the file.

What if you want to read the file above the first line? This can be done with the line number zero. This line doesn't really exist, you will get an error message when using it with most commands. But this command is allowed:

```
:0read patch
```

The file "patch" will be put above the first line of the file.

WRITING A RANGE OF LINES

To write a range of lines to a file, the ":write" command can be used. Without a range it writes the whole file. With a range only the specified lines are written:

```
:$write tempo
```

This writes the lines from the cursor until the end of the file into the file

"tempo". If this file already exists you will get an error message. Vim protects you from accidentally overwriting an existing file. If you know what you are doing and want to overwrite the file, append !:

```
!.,$write! tempo
```

CAREFUL: The ! must follow the ":write" command immediately, without white space. Otherwise it becomes a filter command, which is explained later in this chapter.

APPENDING TO A FILE

In the first section of this chapter was explained how to collect a number of lines into a register. The same can be done to collect lines in a file. Write the first line with this command:

```
!.write collection
```

Now move the cursor to the second line you want to collect, and type this:

```
!.write >>collection
```

The ">>" tells Vim the "collection" file is not to be written as a new file, but the line must be appended at the end. You can repeat this as many times as you like.

10.7 Formatting text

When you are typing plain text, it's nice if the length of each line is automatically trimmed to fit in the window. To make this happen while inserting text, set the '**textwidth**' option:

```
:set textwidth=72
```

You might remember that in the example vimrc file this command was used for every text file. Thus if you are using that vimrc file, you were already using it. To check the current value of '**textwidth**':

```
:set textwidth
```

Now lines will be broken to take only up to 72 characters. But when you insert text halfway through a line, or when you delete a few words, the lines will get too long or too short. Vim doesn't automatically reformat the text.

To tell Vim to format the current paragraph:

```
gqap
```

This starts with the "gq" command, which is an operator. Following is "ap", the text object that stands for "a paragraph". A paragraph is separated from the next paragraph by an empty line.

Note:

A blank line, which contains white space, does NOT separate paragraphs. This is hard to notice!

Instead of "ap" you could use any motion or text object. If your paragraphs are properly separated, you can use this command to format the whole file:

`ggqG`

"gg" takes you to the first line, "gq" is the format operator and "G" the motion that jumps to the last line.

In case your paragraphs aren't clearly defined, you can format just the lines you manually select. Move the cursor to the first line you want to format. Start with the command "gqj". This formats the current line and the one below it. If the first line was short, words from the next line will be appended. If it was too long, words will be moved to the next line. The cursor moves to the second line. Now you can use "." to repeat the command. Keep doing this until you are at the end of the text you want to format.

10.8 Changing case

You have text with section headers in lowercase. You want to make the word "section" all uppercase. Do this with the "gU" operator. Start with the cursor in the first column:

```
section header      gUw      SECTION header
----->
```

The "gu" operator does exactly the opposite:

```
SECTION header      guw      section header
----->
```

You can also use "g~" to swap case. All these are operators, thus they work with any motion command, with text objects and in Visual mode.

To make an operator work on lines you double it. The delete operator is "d", thus to delete a line you use "dd". Similarly, "gugu" makes a whole line lowercase. This can be shortened to "guu". "gUgU" is shortened to "gUU" and "g~g~" to "g~~". Example:

```
Some GIRLS have Fun  g~~  SOME girls HAVE FUN
----->
```

10.9 Using an external program

Vim has a very powerful set of commands, it can do anything. But there may still be something that an external command can do better or faster.

The command "`!{motion}{program}`" takes a block of text and filters it through an external program. In other words, it runs the system command represented by `{program}`, giving it the block of text represented by `{motion}` as input. The output of this command then replaces the selected block.

Because this summarizes badly if you are unfamiliar with UNIX filters, take

a look at an example. The sort command sorts a file. If you execute the following command, the unsorted file input.txt will be sorted and written to output.txt. (This works on both UNIX and Microsoft Windows.)

```
sort <input.txt >output.txt
```

Now do the same thing in Vim. You want to sort lines 1 through 5 of a file. You start by putting the cursor on line 1. Next you execute the following command:

```
!5G
```

The "!" tells Vim that you are performing a filter operation. The Vim editor expects a motion command to follow, indicating which part of the file to filter. The "5G" command tells Vim to go to line 5, so it now knows that it is to filter lines 1 (the current line) through 5.

In anticipation of the filtering, the cursor drops to the bottom of the screen and a ! prompt displays. You can now type in the name of the filter program, in this case "sort". Therefore, your full command is as follows:

```
!5Gsort<Enter>
```

The result is that the sort program is run on the first 5 lines. The output of the program replaces these lines.

```
line 55          line 11
line 33          line 22
line 11          -->  line 33
line 22          line 44
line 44          line 55
last line       last line
```

The "!!" command filters the current line through a filter. In Unix the "date" command prints the current time and date. "!!date<Enter>" replaces the current line with the output of "date". This is useful to add a timestamp to a file.

WHEN IT DOESN'T WORK

Starting a shell, sending it text and capturing the output requires that Vim knows how the shell works exactly. When you have problems with filtering, check the values of these options:

'shell'	specifies the program that Vim uses to execute external programs.
'shellcmdflag'	argument to pass a command to the shell
'shellquote'	quote to be used around the command
'shellxquote'	quote to be used around the command and redirection
'shelltype'	kind of shell (only for the Amiga)
'shellslash'	use forward slashes in the command (only for MS-Windows and alike)
'shellredir'	string used to write the command output into a file

On Unix this is hardly ever a problem, because there are two kinds of shells:

"sh" like and "csh" like. Vim checks the `'shell'` option and sets related options automatically, depending on whether it sees "csh" somewhere in `'shell'`.

On MS-Windows, however, there are many different shells and you might have to tune the options to make filtering work. Check the help for the options for more information.

READING COMMAND OUTPUT

To read the contents of the current directory into the file, use this:

on Unix:

```
:read !ls
```

on MS-Windows:

```
:read !dir
```

The output of the "ls" or "dir" command is captured and inserted in the text, below the cursor. This is similar to reading a file, except that the "!" is used to tell Vim that a command follows.

The command may have arguments. And a range can be used to tell where Vim should put the lines:

```
:0read !date -u
```

This inserts the current time and date in UTC format at the top of the file. (Well, if you have a date command that accepts the "-u" argument.) [Note](#) the difference with using "!!date": that replaced a line, while ":read !date" will insert a line.

WRITING TEXT TO A COMMAND

The Unix command "wc" counts words. To count the words in the current file:

```
:write !wc
```

This is the same write command as before, but instead of a file name the "!" character is used and the name of an external command. The written text will be passed to the specified command as its standard input. The output could look like this:

```
4      47    249
```

The "wc" command isn't verbose. This means you have 4 lines, 47 words and 249 characters.

Watch out for this mistake:

```
:write! wc
```

This will write the file "wc" in the current directory, with force. White space is important here!

REDRAWING THE SCREEN

If the external command produced an error message, the display may have been messed up. Vim is very efficient and only redraws those parts of the screen that it knows need redrawing. But it can't know about what another program has written. To tell Vim to redraw the screen:

`CTRL-L`

=====

Next chapter: [usr_11.txt](#) Recovering from a crash

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_11.txt For Vim version 9.1. Last change: 2020 Oct 25

VIM USER MANUAL - by Bram Moolenaar

Recovering from a crash

Did your computer crash? And you just spent hours editing? Don't panic! Vim stores enough information to be able to restore most of your work. This chapter shows you how to get your work back and explains how the swap file is used.

- 11.1 Basic recovery
- 11.2 Where is the swap file?
- 11.3 Crashed or not?
- 11.4 Further reading

Next chapter: [usr_12.txt](#) Clever tricks
Previous chapter: [usr_10.txt](#) Making big changes
Table of contents: [usr_toc.txt](#)

11.1 Basic recovery

In most cases recovering a file is quite simple, assuming you know which file you were editing (and the harddisk is still working). Start Vim on the file, with the "-r" argument added:

```
vim -r help.txt
```

Vim will read the swap file (used to store text you were editing) and may read bits and pieces of the original file. If Vim recovered your changes you will see these messages (with different file names, of course):

```
Using swap file ".help.txt.swp"  
Original file "~/vim/runtime/doc/help.txt"  
Recovery completed. You should check if everything is OK.  
(You might want to write out this file under another name  
and run diff with the original file to check for changes)  
You may want to delete the .swp file now.
```

To be on the safe side, write this file under another name:

```
:write help.txt.recovered
```

Compare the file with the original file to check if you ended up with what you expected. Vimdiff is very useful for this [08.7](#) . For example:

```
:write help.txt.recovered  
:edit #  
:diffsp help.txt
```

Watch out for the original file to contain a more recent version (you saved the file just before the computer crashed). And check that no lines are

missing (something went wrong that Vim could not recover).

If Vim produces warning messages when recovering, read them carefully. This is rare though.

If the recovery resulted in text that is exactly the same as the file contents, you will get this message:

```
Using swap file ".help.txt.swp"
Original file "~/vim/runtime/doc/help.txt"
Recovery completed. Buffer contents equals file contents.
You may want to delete the .swp file now.
```

This usually happens if you already recovered your changes, or you wrote the file after making changes. It is safe to delete the swap file now.

It is normal that the last few changes can not be recovered. Vim flushes the changes to disk when you don't type for about four seconds, or after typing about two hundred characters. This is set with the **'updatetime'** and **'updatecount'** options. Thus when Vim didn't get a chance to save itself when the system went down, the changes after the last flush will be lost.

If you were editing without a file name, give an empty string as argument:

```
vim -r ""
```

You must be in the right directory, otherwise Vim can't find the swap file.

11.2 Where is the swap file?

Vim can store the swap file in several places. Normally it is in the same directory as the original file. To find it, change to the directory of the file, and use:

```
vim -r
```

Vim will list the swap files that it can find. It will also look in other directories where the swap file for files in the current directory may be located. It will not find swap files in any other directories though, it doesn't search the directory tree.

The output could look like this:

```
Swap files found:
  In current directory:
1.   .main.c.swp
      owned by: mool   dated: Tue May 29 21:00:25 2001
      file name: ~/mool/vim/vim6/src/main.c
      modified: YES
      user name: mool   host name: masaka.moolenaar.net
      process ID: 12525
  In directory ~/tmp:
      -- none --
  In directory /var/tmp:
      -- none --
```

```
In directory /tmp:
-- none --
```

If there are several swap files that look like they may be the one you want to use, a list is given of these swap files and you are requested to enter the number of the one you want to use. Carefully look at the dates to decide which one you want to use.

In case you don't know which one to use, just try them one by one and check the resulting files if they are what you expected.

USING A SPECIFIC SWAP FILE

If you know which swap file needs to be used, you can recover by giving the swap file name. Vim will then find out the name of the original file from the swap file.

Example:

```
vim -r .help.txt.swo
```

This is also handy when the swap file is in another directory than expected. Vim recognizes files with the pattern `*.s[uvw][a-z]` as swap files.

If this still does not work, see what file names Vim reports and rename the files accordingly. Check the `'directory'` option to see where Vim may have put the swap file.

Note:

Vim tries to find the swap file by searching the directories in the `'dir'` option, looking for files that match `"filename.sw?"`. If wildcard expansion doesn't work (e.g., when the `'shell'` option is invalid), Vim does a desperate try to find the file `"filename.swp"`. If that fails too, you will have to give the name of the swapfile itself to be able to recover the file.

11.3 Crashed or not?

ATTENTION E325

Vim tries to protect you from doing stupid things. Suppose you innocently start editing a file, expecting the contents of the file to show up. Instead, Vim produces a very long message:

```
E325: ATTENTION
Found a swap file by the name ".main.c.swp"
  owned by: mool   dated: Tue May 29 21:09:28 2001
  file name: ~mool/vim/vim6/src/main.c
  modified: no
  user name: mool   host name: masaka.moolenaar.net
  process ID: 12559 (still running)
While opening file "main.c"
  dated: Tue May 29 19:46:12 2001
```

- (1) Another program may be editing the same file.
If this is the case, be careful not to end up with two

different instances of the same file when making changes.
Quit, or continue with caution.

- (2) An edit session for this file crashed.
If this is the case, use `":recover"` or `"vim -r main.c"`
to recover the changes (see `":help recovery"`).
If you did this already, delete the swap file `".main.c.swp"`
to avoid this message.

You get this message, because, when starting to edit a file, Vim checks if a swap file already exists for that file. If there is one, there must be something wrong. It may be one of these two situations.

1. Another edit session is active on this file. Look in the message for the line with "process ID". It might look like this:

```
process ID: 12559 (still running)
```

The text "(still running)" indicates that the process editing this file runs on the same computer. When working on a non-Unix system you will not get this extra hint. When editing a file over a network, you may not see the hint, because the process might be running on another computer. In those two cases you must find out what the situation is yourself.

If there is another Vim editing the same file, continuing to edit will result in two versions of the same file. The one that is written last will overwrite the other one, resulting in loss of changes. You better quit this Vim.

2. The swap file might be the result from a previous crash of Vim or the computer. Check the dates mentioned in the message. If the date of the swap file is newer than the file you were editing, and this line appears:

```
modified: YES
```

Then you very likely have a crashed edit session that is worth recovering.

If the date of the file is newer than the date of the swap file, then either it was changed after the crash (perhaps you recovered it earlier, but didn't delete the swap file?), or else the file was saved before the crash but after the last write of the swap file (then you're lucky: you don't even need that old swap file). Vim will warn you for this with this extra line:

```
NEWER than swap file!
```

NOTE that in the following situation Vim knows the swap file is not useful and will automatically delete it:

- The file is a valid swap file (Magic number is correct).
- The flag that the file was modified is not set.
- The process is not running.

You can programmatically deal with this situation with the `FileChangedShell` autocommand event.

UNREADABLE SWAP FILE

Sometimes the line

[cannot be read]

will appear under the name of the swap file. This can be good or bad, depending on circumstances.

It is good if a previous editing session crashed without having made any changes to the file. Then a directory listing of the swap file will show that it has zero bytes. You may delete it and proceed.

It is slightly bad if you don't have read permission for the swap file. You may want to view the file read-only, or quit. On multi-user systems, if you yourself did the last changes under a different login name, a logout followed by a login under that other name might cure the "read error". Or else you might want to find out who last edited (or is editing) the file and have a talk with them.

It is very bad if it means there is a physical read error on the disk containing the swap file. Fortunately, this almost never happens. You may want to view the file read-only at first (if you can), to see the extent of the changes that were "forgotten". If you are the one in charge of that file, be prepared to redo your last changes.

WHAT TO DO?

swap-exists-choices

If dialogs are supported you will be asked to select one of six choices:

Swap file ".main.c.swp" already exists!

[O]pen Read-Only, (E)dit anyway, (R)ecover, (Q)uit, (A)bort, (D)elete it:

- O Open the file readonly. Use this when you just want to view the file and don't need to recover it. You might want to use this when you know someone else is editing the file, but you just want to look in it and not make changes.
- E Edit the file anyway. Use this with caution! If the file is being edited in another Vim, you might end up with two versions of the file. Vim will try to warn you when this happens, but better be safe than sorry.
- R Recover the file from the swap file. Use this if you know that the swap file contains changes that you want to recover.
- Q Quit. This avoids starting to edit the file. Use this if there is another Vim editing the same file.

When you just started Vim, this will exit Vim. When starting Vim with files in several windows, Vim quits only if there is a swap file for the first one. When using an edit command, the file will not be loaded and you are taken back to the previously edited file.

A Abort. Like Quit, but also abort further commands. This is useful when loading a script that edits several files, such as a session with multiple windows.

D Delete the swap file. Use this when you are sure you no longer need it. For example, when it doesn't contain changes, or when the file itself is newer than the swap file.

On Unix this choice is only offered when the process that created the swap file does not appear to be running.

If you do not get the dialog (you are running a version of Vim that does not support it), you will have to do it manually. To recover the file, use this command:

```
:recover
```

Vim cannot always detect that a swap file already exists for a file. This is the case when the other edit session puts the swap files in another directory or when the path name for the file is different when editing it on different machines. Therefore, don't rely on Vim always warning you.

If you really don't want to see this message, you can add the 'A' flag to the '**shortmess**' option. But it's very unusual that you need this.

For remarks about encryption and the swap file, see `:recover-crypt` .

For programmatic access to the swap file, see `swapinfo()` .

11.4 Further reading

<code>swap-file</code>	An explanation about where the swap file will be created and what its name is.
<code>:preserve</code>	Manually flushing the swap file to disk.
<code>:swapname</code>	See the name of the swap file for the current file.
<code>'updatecount'</code>	Number of key strokes after which the swap file is flushed to disk.
<code>'updatetime'</code>	Timeout after which the swap file is flushed to disk.
<code>'swapsync'</code>	Whether the disk is synced when the swap file is flushed.
<code>'directory'</code>	List of directory names where to store the swap file.
<code>'maxmem'</code>	Limit for memory usage before writing text to the swap file.
<code>'maxmemtot'</code>	Same, but for all files in total.

Next chapter: `usr_12.txt` Clever tricks

Copyright: see `manual-copyright` vim:tw=78:ts=8:noet:ft=help:norl:

Clever tricks

By combining several commands you can make Vim do nearly everything. In this chapter a number of useful combinations will be presented. This uses the commands introduced in the previous chapters and a few more.

- 12.1 Replace a word
- 12.2 Change "Last, First" to "First Last"
- 12.3 Sort a list
- 12.4 Reverse line order
- 12.5 Count words
- 12.6 Find a man page
- 12.7 Trim blanks
- 12.8 Find where a word is used

Next chapter: [usr_20.txt](#) Typing command-line commands quickly
Previous chapter: [usr_11.txt](#) Recovering from a crash
Table of contents: [usr_toc.txt](#)

12.1 Replace a word

The substitute command can be used to replace all occurrences of a word with another word:

```
:%s/four/4/g
```

The "%" range means to replace in all lines. The "g" flag at the end causes all words in a line to be replaced.

This will not do the right thing if your file also contains "thirtyfour". It would be replaced with "thirty4". To avoid this, use the "\<" item to match the start of a word:

```
:%s/\<four/4/g
```

Obviously, this still goes wrong on "fourteen". Use "\>" to match the end of a word:

```
:%s/\<four\>/4/g
```

If you are programming, you might want to replace "four" in comments, but not in the code. Since this is difficult to specify, add the "c" flag to have the substitute command prompt you for each replacement:

```
:%s/\<four\>/4/gc
```

REPLACING IN SEVERAL FILES

Suppose you want to replace a word in more than one file. You could edit each file and type the command manually. It's a lot faster to use record and playback.

Let's assume you have a directory with C++ files, all ending in ".cpp". There is a function called "GetResp" that you want to rename to "GetAnswer".

```
vim *.cpp          Start Vim, defining the argument list to
                   contain all the C++ files. You are now in the
                   first file.
qq                Start recording into the q register
:%s/\<GetResp\>/GetAnswer/g
                   Do the replacements in the first file.
:wnext            Write this file and move to the next one.
q                Stop recording.
@q               Execute the q register. This will replay the
                   substitution and ":wnext". You can verify
                   that this doesn't produce an error message.
999@q            Execute the q register on the remaining files.
```

At the last file you will get an error message, because ":wnext" cannot move to the next file. This stops the execution, and everything is done.

Note:

When playing back a recorded sequence, an error stops the execution. Therefore, make sure you don't get an error message when recording.

There is one catch: If one of the .cpp files does not contain the word "GetResp", you will get an error and replacing will stop. To avoid this, add the "e" flag to the substitute command:

```
:%s/\<GetResp\>/GetAnswer/ge
```

The "e" flag tells ":substitute" that not finding a match is not an error.

=====

12.2 Change "Last, First" to "First Last"

You have a list of names in this form:

```
Doe, John
Smith, Peter
```

You want to change that to:

```
John Doe
Peter Smith
```

This can be done with just one command:

```
:%s/\([^\,]*\), \(.*\)/\2 \1/
```

Let's break this down in parts. Obviously it starts with a substitute command. The "%" is the line range, which stands for the whole file. Thus

the substitution is done in every line in the file.

The arguments for the substitute command are `"/from/to/"`. The slashes separate the "from" pattern and the "to" string. This is what the "from" pattern contains:

```

                                                    \([^\,]*\), \(.*\)
The first part between \( \) matches "Last"  \([^\,]*\)
    match anything but a comma                [^\,]*
    any number of times                       *
matches ", " literally                        , \( \)
The second part between \( \) matches "First" \(.*\)
    any character                             .
    any number of times                       *
```

In the "to" part we have `"\2"` and `"\1"`. These are called backreferences. They refer to the text matched by the `"\(\)"` parts in the pattern. `"\2"` refers to the text matched by the second `"\(\)"`, which is the "First" name. `"\1"` refers to the first `"\(\)"`, which is the "Last" name.

You can use up to nine backreferences in the "to" part of a substitute command. `"\0"` stands for the whole matched pattern. There are a few more special items in a substitute command, see [sub-replace-special](#).

12.3 Sort a list

In a Makefile you often have a list of files. For example:

```
OBJS = \
    version.o \
    pch.o \
    getopt.o \
    util.o \
    getopt1.o \
    inp.o \
    patch.o \
    backup.o
```

To sort this list, filter the text through the external sort command:

```
/^OBJS
j
:./^$/-1!sort
```

This goes to the first line, where "OBJS" is the first thing in the line. Then it goes one line down and filters the lines until the next empty line. You could also select the lines in Visual mode and then use `"!sort"`. That's easier to type, but more work when there are many lines.

The result is this:

```
OBJS = \
    backup.o
    getopt.o \
    getopt1.o \
    inp.o \
```

```
patch.o \  
pch.o \  
util.o \  
version.o \  

```

Notice that a backslash at the end of each line is used to indicate the line continues. After sorting, this is wrong! The "backup.o" line that was at the end didn't have a backslash. Now that it sorts to another place, it must have a backslash.

The simplest solution is to add the backslash with "A \<Esc>". You can keep the backslash in the last line, if you make sure an empty line comes after it. That way you don't have this problem again.

12.4 Reverse line order

The `:global` command can be combined with the `:move` command to move all the lines before the first line, resulting in a reversed file. The command is:

```
:global/^/move 0
```

Abbreviated:

```
:g/^/m 0
```

The "^" regular expression matches the beginning of the line (even if the line is blank). The `:move` command moves the matching line to after the imaginary zeroth line, so the current matching line becomes the first line of the file. As the `:global` command is not confused by the changing line numbering, `:global` proceeds to match all remaining lines of the file and puts each as the first.

This also works on a range of lines. First move to above the first line and mark it with "mt". Then move the cursor to the last line in the range and type:

```
:'t+1,.g/^/m 't
```

12.5 Count words

Sometimes you have to write a text with a maximum number of words. Vim can count the words for you.

When the whole file is what you want to count the words in, use this command:

```
g CTRL-G
```

Do not type a space after the g, this is just used here to make the command easy to read.

The output looks like this:

```
Col 1 of 0; Line 141 of 157; Word 748 of 774; Byte 4489 of 4976
```

You can see on which word you are (748), and the total number of words in the file (774).

When the text is only part of a file, you could move to the start of the text, type "g **CTRL-G**", move to the end of the text, type "g **CTRL-G**" again, and then use your brain to compute the difference in the word position. That's a good exercise, but there is an easier way. With Visual mode, select the text you want to count words in. Then type g **CTRL-G**. The result:

```
Selected 5 of 293 Lines; 70 of 1884 Words; 359 of 10928 Bytes
```

For other ways to count words, lines and other items, see [count-items](#) .

12.6 Find a man page

[find-manpage](#)

While editing a shell script or C program, you are using a command or function that you want to find the man page for (this is on Unix). Let's first use a simple way: Move the cursor to the word you want to find help on and press

K

Vim will run the external "man" program on the word. If the man page is found, it is displayed. This uses the normal pager to scroll through the text (mostly the "more" program). When you get to the end pressing **<Enter>** will get you back into Vim.

A disadvantage is that you can't see the man page and the text you are working on at the same time. There is a trick to make the man page appear in a Vim window. First, load the man filetype plugin:

```
:runtime! ftplugin/man.vim
```

Put this command in your vimrc file if you intend to do this often. Now you can use the **":Man**" command to open a window on a man page:

```
:Man csh
```

You can scroll around and the text is highlighted. This allows you to find the help you were looking for. Use **CTRL-W w** to jump to the window with the text you were working on.

To find a man page in a specific section, put the section number first. For example, to look in section 3 for "echo":

```
:Man 3 echo
```

To jump to another man page, which is in the text with the typical form "word(1)", press **CTRL-]** on it. Further **":Man**" commands will use the same window.

To display a man page for the word under the cursor, use this:

\K

(If you redefined the <Leader>, use it instead of the backslash). For example, you want to know the return value of "strstr()" while editing this line:

```
if ( strstr (input, "aap") == )
```

Move the cursor to somewhere on "strstr" and type "\K". A window will open to display the man page for strstr().

12.7 Trim blanks

Some people find spaces and tabs at the end of a line useless, wasteful, and ugly. To remove whitespace at the end of every line, execute the following command:

```
:%s/\s\+$//
```

The line range "%" is used, thus this works on the whole file. The pattern that the ":substitute" command matches with is "\s\+\$". This finds white space characters (\s), 1 or more of them (\+), before the end-of-line (\$). Later will be explained how you write patterns like this, see [usr_27.txt](#).

The "to" part of the substitute command is empty: "//". Thus it replaces with nothing, effectively deleting the matched white space.

Another wasteful use of spaces is placing them before a tab. Often these can be deleted without changing the amount of white space. But not always! Therefore, you can best do this manually. Use this search command:

```
/
```

You cannot see it, but there is a space before a tab in this command. Thus it's "<Space><Tab>". Now use "x" to delete the space and check that the amount of white space doesn't change. You might have to insert a tab if it does change. Type "n" to find the next match. Repeat this until no more matches can be found.

12.8 Find where a word is used

If you are a UNIX user, you can use a combination of Vim and the grep command to edit all the files that contain a given word. This is extremely useful if you are working on a program and want to view or edit all the files that contain a specific variable.

For example, suppose you want to edit all the C program files that contain the word "frame_counter". To do this you use the command:

```
vim `grep -l frame_counter *.c`
```

Let's look at this command in detail. The grep command searches through a set of files for a given word. Because the -l argument is specified, the command will only list the files containing the word and not print the matching lines. The word it is searching for is "frame_counter". Actually, this can be any

regular expression. (Note: What grep uses for regular expressions is not exactly the same as what Vim uses.)

The entire command is enclosed in backticks (`). This tells the UNIX shell to run this command and pretend that the results were typed on the command line. So what happens is that the grep command is run and produces a list of files, these files are put on the Vim command line. This results in Vim editing the file list that is the output of grep. You can then use commands like `":next"` and `":first"` to browse through the files.

FINDING EACH LINE

The above command only finds the files in which the word is found. You still have to find the word within the files.

Vim has a built-in command that you can use to search a set of files for a given string. If you want to find all occurrences of "error_string" in all C program files, for example, enter the following command:

```
:grep error_string *.c
```

This causes Vim to search for the string "error_string" in all the specified files (*.c). The editor will now open the first file where a match is found and position the cursor on the first matching line. To go to the next matching line (no matter in what file it is), use the `":cnext"` command. To go to the previous match, use the `":cprev"` command. Use `":clist"` to see all the matches and where they are.

The `":grep"` command uses the external commands `grep` (on Unix) or `findstr` (on Windows). You can change this by setting the option `'grepprog'`.

=====

Next chapter: [usr_20.txt](#) Typing command-line commands quickly

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_20.txt For Vim version 9.1. Last change: 2021 Nov 18

VIM USER MANUAL - by Bram Moolenaar

Typing command-line commands quickly

Vim has a few generic features that makes it easier to enter commands. Colon commands can be abbreviated, edited and repeated. Completion is available for nearly everything.

- 20.1 Command line editing
- 20.2 Command line abbreviations
- 20.3 Command line completion
- 20.4 Command line history
- 20.5 Command line window

Next chapter: [usr_21.txt](#) Go away and come back
Previous chapter: [usr_12.txt](#) Clever tricks
Table of contents: [usr_toc.txt](#)

20.1 Command line editing

When you use a colon (:) command or search for a string with / or ?, Vim puts the cursor on the bottom of the screen. There you type the command or search pattern. This is called the Command line. Also when it's used for entering a search command.

The most obvious way to edit the command you type is by pressing the <BS> key. This erases the character before the cursor. To erase another character, typed earlier, first move the cursor with the cursor keys.

For example, you have typed this:

```
:s/col/pig/
```

Before you hit <Enter>, you notice that "col" should be "cow". To correct this, you type <Left> five times. The cursor is now just after "col". Type <BS> and "w" to correct:

```
:s/cow/pig/
```

Now you can press <Enter> directly. You don't have to move the cursor to the end of the line before executing the command.

The most often used keys to move around in the command line:

<Left>	one character left
<Right>	one character right
<S-Left> or <C-Left>	one word left
<S-Right> or <C-Right>	one word right
CTRL-B or <Home>	to begin of command line
CTRL-E or <End>	to end of command line

Note:

<S-Left> (cursor left key with Shift key pressed) and <C-Left> (cursor left key with Control pressed) will not work on all keyboards. Same for the other Shift and Control combinations.

You can also use the mouse to move the cursor.

DELETING

As mentioned, <BS> deletes the character before the cursor. To delete a whole word use **CTRL-W**.

```
/the fine pig
```

CTRL-W

```
/the fine
```

CTRL-U removes all text, thus allows you to start all over again.

OVERSTRIKE

The <Insert> key toggles between inserting characters and replacing the existing ones. Start with this text:

```
/the fine pig
```

Move the cursor to the start of "fine" with <S-Left> twice (or <Left> eight times, if <S-Left> doesn't work). Now press <Insert> to switch to overstrike and type "great":

```
/the greatpig
```

Oops, we lost the space. Now, don't use <BS>, because it would delete the "t" (this is different from Replace mode). Instead, press <Insert> to switch from overstrike to inserting, and type the space:

```
/the great pig
```

CANCELLING

You thought of executing a : or / command, but changed your mind. To get rid of what you already typed, without executing it, press **CTRL-C** or <Esc>.

Note:

<Esc> is the universal "get out" key. Unfortunately, in the good old Vi pressing <Esc> in a command line executed the command! Since that might be considered to be a bug, Vim uses <Esc> to cancel the command. But with the '**cptions**' option it can be made Vi compatible. And when using a mapping (which might be written for Vi) <Esc> also works Vi compatible. Therefore, using **CTRL-C** is a method that always works.

If you are at the start of the command line, pressing <BS> will cancel the command. It's like deleting the ":" or "/" that the line starts with.

20.2 Command line abbreviations

Some of the ":" commands are really long. We already mentioned that ":substitute" can be abbreviated to ":s". This is a generic mechanism, all ":" commands can be abbreviated.

How short can a command get? There are 26 letters, and many more commands. For example, ":set" also starts with ":s", but ":s" doesn't start a ":set" command. Instead ":set" can be abbreviated to ":se".

When the shorter form of a command could be used for two commands, it stands for only one of them. There is no logic behind which one, you have to learn them. In the help files the shortest form that works is mentioned. For example:

```
:s[ubstitute]
```

This means that the shortest form of ":substitute" is ":s". The following characters are optional. Thus ":su" and ":sub" also work.

In the user manual we will either use the full name of command, or a short version that is still readable. For example, ":function" can be abbreviated to ":fu". But since most people don't understand what that stands for, we will use ":fun". (Vim doesn't have a ":funny" command, otherwise ":fun" would be confusing too.)

It is recommended that in Vim scripts you write the full command name. That makes it easier to read back when you make later changes. Except for some often used commands like ":w" (":write") and ":r" (":read").

A particularly confusing one is ":end", which could stand for ":endif", ":endwhile" or ":endfunction". Therefore, always use the full name.

SHORT OPTION NAMES

In the user manual the long version of the option names is used. Many options also have a short name. Unlike ":" commands, there is only one short name that works. For example, the short name of 'autoindent' is 'ai'. Thus these two commands do the same thing:

```
:set autoindent  
:set ai
```

You can find the full list of long and short names here: [option-list](#) .

20.3 Command line completion

This is one of those Vim features that, by itself, is a reason to switch from Vi to Vim. Once you have used this, you can't do without.

Suppose you have a directory that contains these files:

```
info.txt
intro.txt
bodyofthepaper.txt
```

To edit the last one, you use the command:

```
:edit bodyofthepaper.txt
```

It's easy to type this wrong. A much quicker way is:

```
:edit b<Tab>
```

Which will result in the same command. What happened? The `<Tab>` key does completion of the word before the cursor. In this case "b". Vim looks in the directory and finds only one file that starts with a "b". That must be the one you are looking for, thus Vim completes the file name for you.

Now type:

```
:edit i<Tab>
```

Vim will beep, and give you:

```
:edit info.txt
```

The beep means that Vim has found more than one match. It then uses the first match it found (alphabetically). If you press `<Tab>` again, you get:

```
:edit intro.txt
```

Thus, if the first `<Tab>` doesn't give you the file you were looking for, press it again. If there are more matches, you will see them all, one at a time.

If you press `<Tab>` on the last matching entry, you will go back to what you first typed:

```
:edit i
```

Then it starts all over again. Thus Vim cycles through the list of matches. Use **CTRL-P** to go through the list in the other direction:

```

<-----<Tab>-----+
|
:edit i      <Tab> -->      :edit info.txt      <Tab> -->      :edit intro.txt
|      <-- CTRL-P      <-- CTRL-P
+-----CTRL-P----->
```

CONTEXT

When you type `":set i"` instead of `":edit i"` and press `<Tab>` you get:

```
:set icon
```

Hey, why didn't you get `":set info.txt"`? That's because Vim has context sensitive completion. The kind of words Vim will look for depends on the command before it. Vim knows that you cannot use a file name just after a `":set"` command, but you can use an option name.

Again, if you repeat typing the `<Tab>`, Vim will cycle through all matches. There are quite a few, it's better to type more characters first:

```
:set isk<Tab>
```

Gives:

```
:set iskeyword
```

Now type `"="` and press `<Tab>`:

```
:set iskeyword=@,48-57,_,192-255
```

What happens here is that Vim inserts the old value of the option. Now you can edit it.

What is completed with `<Tab>` is what Vim expects in that place. Just try it out to see how it works. In some situations you will not get what you want. That's either because Vim doesn't know what you want, or because completion was not implemented for that situation. In that case you will get a `<Tab>` inserted (displayed as `^I`).

LIST MATCHES

When there are many matches, you would like to see an overview. Do this by pressing **CTRL-D**. For example, pressing **CTRL-D** after:

```
:set is
```

results in:

```
:set is
incsearch  isfname  isident  iskeyword  isprint
:set is
```

Vim lists the matches and then comes back with the text you typed. You can now check the list for the item you wanted. If it isn't there, you can use `<BS>` to correct the word. If there are many matches, type a few more characters before pressing `<Tab>` to complete the rest.

If you have watched carefully, you will have noticed that `"incsearch"` doesn't start with `"is"`. In this case `"is"` stands for the short name of `"incsearch"`. (Many options have a short and a long name.) Vim is clever enough to know that you might have wanted to expand the short name of the option into the long name.

THERE IS MORE

The **CTRL-L** command completes the word to the longest unambiguous string. If you type `:"edit i"` and there are files `"info.txt"` and `"info_backup.txt"` you will get `:"edit info"`.

The **'wildmode'** option can be used to change the way completion works. The **'wildmenu'** option can be used to get a menu-like list of matches. Use the **'suffixes'** option to specify files that are less important and appear at the end of the list of files. The **'wildignore'** option specifies files that are not listed at all.

More about all of this here: [cmdline-completion](#)

20.4 Command line history

In chapter 3 we briefly mentioned the history. The basics are that you can use the `<Up>` key to recall an older command line. `<Down>` then takes you back to newer commands.

There are actually five histories. The ones we will mention here are for `:"` commands and for `/"` and `?"` search commands. The `/"` and `?"` commands share the same history, because they are both search commands. The three other histories are for expressions, debug mode commands and input lines for the `input()` function. [cmdline-history](#)

Suppose you have done a `:"set"` command, typed ten more colon commands and then want to repeat that `:"set"` command again. You could press `:"` and then ten times `<Up>`. There is a quicker way:

```
:se<Up>
```

Vim will now go back to the previous command that started with `"se"`. You have a good chance that this is the `:"set"` command you were looking for. At least you should not have to press `<Up>` very often (unless `:"set"` commands is all you have done).

The `<Up>` key will use the text typed so far and compare it with the lines in the history. Only matching lines will be used.

If you do not find the line you were looking for, use `<Down>` to go back to what you typed and correct that. Or use **CTRL-U** to start all over again.

To see all the lines in the history:

```
:history
```

That's the history of `:"` commands. The search history is displayed with this command:

```
:history /
```

CTRL-P will work like `<Up>`, except that it doesn't matter what you already typed. Similarly for **CTRL-N** and `<Down>`. **CTRL-P** stands for previous, **CTRL-N**

for next.

20.5 Command line window

Typing the text in the command line works differently from typing text in Insert mode. It doesn't allow many commands to change the text. For most commands that's OK, but sometimes you have to type a complicated command. That's where the command line window is useful.

Open the command line window with this command:

q:

Vim now opens a (small) window at the bottom. It contains the command line history, and an empty line at the end:

```
+-----+
|other window|
|~           |
|file.txt=====|
|:e c       |
|:e config.h.in|
|:set path=.,/usr/include,,|
|:set iskeyword=@,48-57,_,192-255|
|:set is    |
|:q         |
|:         |
|command-line=====|
|          |
+-----+
```

You are now in Normal mode. You can use the "hjkl" keys to move around. For example, move up with "5k" to the ":e config.h.in" line. Type "\$h" to go to the "i" of "in" and type "cwout". Now you have changed the line to:

:e config.h.out

Now press **<Enter>** and this command will be executed. The command line window will close.

The **<Enter>** command will execute the line under the cursor. It doesn't matter whether Vim is in Insert mode or in Normal mode.

Changes in the command line window are lost. They do not result in the history to be changed. Except that the command you execute will be added to the end of the history, like with all executed commands.

The command line window is very useful when you want to have overview of the history, lookup a similar command, change it a bit and execute it. A search command can be used to find something.

In the previous example the "?config" search command could have been used to find the previous command that contains "config". It's a bit strange, because you are using a command line to search in the command line window. While typing that search command you can't open another command line window, there can be only one.

Next chapter: [usr_21.txt](#) Go away and come back

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_21.txt For Vim version 9.1. Last change: 2019 Apr 25

VIM USER MANUAL - by Bram Moolenaar

Go away and come back

This chapter goes into mixing the use of other programs with Vim. Either by executing program from inside Vim or by leaving Vim and coming back later. Furthermore, this is about the ways to remember the state of Vim and restore it later.

- 21.1 Suspend and resume
- 21.2 Executing shell commands
- 21.3 Remembering information; viminfo
- 21.4 Sessions
- 21.5 Views
- 21.6 Modelines

Next chapter: [usr_22.txt](#) Finding the file to edit
Previous chapter: [usr_20.txt](#) Typing command-line commands quickly
Table of contents: [usr_toc.txt](#)

21.1 Suspend and resume

Like most Unix programs Vim can be suspended by pressing **CTRL-Z**. This stops Vim and takes you back to the shell it was started in. You can then do any other commands until you are bored with them. Then bring back Vim with the "fg" command.

```
CTRL-Z
{any sequence of shell commands}
fg
```

You are right back where you left Vim, nothing has changed.

In case pressing **CTRL-Z** doesn't work, you can also use ":suspend". Don't forget to bring Vim back to the foreground, you would lose any changes that you made!

Only Unix has support for this. On other systems Vim will start a shell for you. This also has the functionality of being able to execute shell commands. But it's a new shell, not the one that you started Vim from.

When you are running the GUI you can't go back to the shell where Vim was started. **CTRL-Z** will minimize the Vim window instead.

21.2 Executing shell commands

To execute a single shell command from Vim use ":{command}". For example, to see a directory listing:

```
:!ls
:!dir
```

The first one is for Unix, the second one for MS-Windows.

Vim will execute the program. When it ends you will get a prompt to hit `<Enter>`. This allows you to have a look at the output from the command before returning to the text you were editing.

The `!` is also used in other places where a program is run. Let's take a look at an overview:

```
:{program}          execute {program}
:r !{program}       execute {program} and read its output
:w !{program}       execute {program} and send text to its input
:[range]!{program} filter text through {program}
```

Notice that the presence of a range before `!{program}` makes a big difference. Without it executes the program normally, with the range a number of text lines is filtered through the program.

Executing a whole row of programs this way is possible. But a shell is much better at it. You can start a new shell this way:

```
:shell
```

This is similar to using `CTRL-Z` to suspend Vim. The difference is that a new shell is started.

When using the GUI the shell will be using the Vim window for its input and output. Since Vim is not a terminal emulator, this will not work perfectly. If you have trouble, try toggling the `'guipty'` option. If this still doesn't work well enough, start a new terminal to run the shell in. For example with:

```
!:xterm&
```

21.3 Remembering information; viminfo

After editing for a while you will have text in registers, marks in various files, a command line history filled with carefully crafted commands. When you exit Vim all of this is lost. But you can get it back!

The viminfo file is designed to store status information:

- Command-line and Search pattern history
- Text in registers
- Marks for various files
- The buffer list
- Global variables

Each time you exit Vim it will store this information in a file, the viminfo file. When Vim starts again, the viminfo file is read and the information restored.

The `'viminfo'` option is set by default to restore a limited number of items. You might want to set it to remember more information. This is done through the following command:

```
:set viminfo=string
```

The string specifies what to save. The syntax of this string is an option character followed by an argument. The option/argument pairs are separated by commas.

Take a look at how you can build up your own viminfo string. First, the ' option is used to specify how many files for which you save marks (a-z). Pick a nice even number for this option (1000, for instance). Your command now looks like this:

```
:set viminfo='1000
```

The f option controls whether global marks (A-Z and 0-9) are stored. If this option is 0, none are stored. If it is 1 or you do not specify an f option, the marks are stored. You want this feature, so now you have this:

```
:set viminfo='1000,f1
```

The < option controls how many lines are saved for each of the registers. By default, all the lines are saved. If 0, nothing is saved. To avoid adding thousands of lines to your viminfo file (which might never get used and makes starting Vim slower) you use a maximum of 500 lines:

```
:set viminfo='1000,f1,<500
```

Other options you might want to use:

:	number of lines to save from the command line history
@	number of lines to save from the input line history
/	number of lines to save from the search history
r	removable media, for which no marks will be stored (can be used several times)
!	global variables that start with an uppercase letter and don't contain lowercase letters
h	disable 'hlsearch' highlighting when starting
%	the buffer list (only restored when starting Vim without file arguments)
c	convert the text using 'encoding'
n	name used for the viminfo file (must be the last option)

See the **'viminfo'** option and [viminfo-file](#) for more information.

When you run Vim multiple times, the last one exiting will store its information. This may cause information that previously exiting Vims stored to be lost. Each item can be remembered only once.

GETTING BACK TO WHERE YOU STOPPED VIM

You are halfway editing a file and it's time to leave for holidays. You exit Vim and go enjoy yourselves, forgetting all about your work. After a couple of weeks you start Vim, and type:

```
'0
```

And you are right back where you left Vim. So you can get on with your work.

Vim creates a mark each time you exit Vim. The last one is '0. The position that '0 pointed to is made '1. And '1 is made to '2, and so forth. Mark '9 is lost.

The `:marks` command is useful to find out where '0 to '9 will take you.

GETTING BACK TO SOME FILE

If you want to go back to a file that you edited recently, but not when exiting Vim, there is a slightly more complicated way. You can see a list of files by typing the command:

```
:oldfiles
1: ~/.viminfo
2: ~/text/resume.txt
3: /tmp/draft
```

Now you would like to edit the second file, which is in the list preceded by "2:". You type:

```
:e #<2
```

Instead of ":e" you can use any command that has a file name argument, the "#<2" item works in the same place as "%" (current file name) and "#" (alternate file name). So you can also split the window to edit the third file:

```
:split #<3
```

That #<123 thing is a bit complicated when you just want to edit a file. Fortunately there is a simpler way:

```
:browse oldfiles
1: ~/.viminfo
2: ~/text/resume.txt
3: /tmp/draft
-- More --
```

You get the same list of files as with `:oldfiles`. If you want to edit "resume.txt" first press "q" to stop the listing. You will get a prompt:

```
Type number and <Enter> (empty cancels):
```

Type "2" and press <Enter> to edit the second file.

More info at `:oldfiles`, `v:oldfiles` and `c_#<`.

MOVE INFO FROM ONE VIM TO ANOTHER

You can use the `":wviminfo"` and `":rviminfo"` commands to save and restore the information while still running Vim. This is useful for exchanging register

contents between two instances of Vim, for example. In the first Vim do:

```
:wviminfo! ~/tmp/viminfo
```

And in the second Vim do:

```
:rviminfo! ~/tmp/viminfo
```

Obviously, the "w" stands for "write" and the "r" for "read".

The ! character is used by ":wviminfo" to forcefully overwrite an existing file. When it is omitted, and the file exists, the information is merged into the file.

The ! character used for ":rviminfo" means that all the information is used, this may overwrite existing information. Without the ! only information that wasn't set is used.

These commands can also be used to store info and use it again later. You could make a directory full of viminfo files, each containing info for a different purpose.

21.4 Sessions

Suppose you are editing along, and it is the end of the day. You want to quit work and pick up where you left off the next day. You can do this by saving your editing session and restoring it the next day.

A Vim session contains all the information about what you are editing. This includes things such as the file list, window layout, global variables, options and other information. (Exactly what is remembered is controlled by the '**sessionoptions**' option, described below.)

The following command creates a session file:

```
:mksession vimbook.vim
```

Later if you want to restore this session, you can use this command:

```
:source vimbook.vim
```

If you want to start Vim and restore a specific session, you can use the following command:

```
vim -S vimbook.vim
```

This tells Vim to read a specific file on startup. The 'S' stands for session (actually, you can source any Vim script with -S, thus it might as well stand for "source").

The windows that were open are restored, with the same position and size as before. Mappings and option values are like before.

What exactly is restored depends on the '**sessionoptions**' option. The default value is:

```
"blank,buffers,curdir,folds,help,options,tabpages,winsize,terminal".
```

blank	keep empty windows
buffers	all buffers, not only the ones in a window

curdir	the current directory
fold	fold, also manually created ones
help	the help window
options	all options and mappings
tabpages	all tab pages
winsize	window sizes
terminal	include terminal windows

Change this to your liking. To also restore the size of the Vim window, for example, use:

```
:set sessionoptions+=resize
```

SESSION HERE, SESSION THERE

The obvious way to use sessions is when working on different projects. Suppose you store your session files in the directory "~/.vim". You are currently working on the "secret" project and have to switch to the "boring" project:

```
:wall
:mksession! ~/.vim/secret.vim
:source ~/.vim/boring.vim
```

This first uses ":wall" to write all modified files. Then the current session is saved, using ":mksession!". This overwrites the previous session. The next time you load the secret session you can continue where you were at this point. And finally you load the new "boring" session.

If you open help windows, split and close various windows, and generally mess up the window layout, you can go back to the last saved session:

```
:source ~/.vim/boring.vim
```

Thus you have complete control over whether you want to continue next time where you are now, by saving the current setup in a session, or keep the session file as a starting point.

Another way of using sessions is to create a window layout that you like to use, and save this in a session. Then you can go back to this layout whenever you want.

For example, this is a nice layout to use:

```

+-----+
|                VIM - main help file                |
|Move around:  Use the cursor keys, or "h|
|help.txt=====|
|explorer    |
|dir         |~
|dir         |~
|file        |~
|file        |~
|file        |~
+-----+

```

```

|file      |~
~/===== [No File]=====|
|
+-----+

```

This has a help window at the top, so that you can read this text. The narrow vertical window on the left contains a file explorer. This is a Vim plugin that lists the contents of a directory. You can select files to edit there. More about this in the next chapter.

Create this from a just started Vim with:

```

:help
CTRL-W w
:vertical split ~/

```

You can resize the windows a bit to your liking. Then save the session with:

```

:mksession ~/.vim/mine.vim

```

Now you can start Vim with this layout:

```

vim -S ~/.vim/mine.vim

```

Hint: To open a file you see listed in the explorer window in the empty window, move the cursor to the filename and press "O". Double clicking with the mouse will also do this.

UNIX AND MS-WINDOWS

Some people have to do work on MS-Windows systems one day and on Unix another day. If you are one of them, consider adding "slash" and "unix" to **'sessionoptions'**. The session files will then be written in a format that can be used on both systems. This is the command to put in your vimrc file:

```

:set sessionoptions+=unix,slash

```

Vim will use the Unix format then, because the MS-Windows Vim can read and write Unix files, but Unix Vim can't read MS-Windows format session files. Similarly, MS-Windows Vim understands file names with / to separate names, but Unix Vim doesn't understand \.

SESSIONS AND VIMINFO

Sessions store many things, but not the position of marks, contents of registers and the command line history. You need to use the viminfo feature for these things.

In most situations you will want to use sessions separately from viminfo. This can be used to switch to another session, but keep the command line history. And yank text into registers in one session, and paste it back in another session.

You might prefer to keep the info with the session. You will have to do this yourself then. Example:

```
:mksession! ~/.vim/secret.vim
:wviminfo! ~/.vim/secret.viminfo
```

And to restore this again:

```
:source ~/.vim/secret.vim
:rviminfo! ~/.vim/secret.viminfo
```

21.5 Views

A session stores the looks of the whole of Vim. When you want to store the properties for one window only, use a view.

The use of a view is for when you want to edit a file in a specific way. For example, you have line numbers enabled with the **'number'** option and defined a few folds. Just like with sessions, you can remember this view on the file and restore it later. Actually, when you store a session, it stores the view of each window.

There are two basic ways to use views. The first is to let Vim pick a name for the view file. You can restore the view when you later edit the same file. To store the view for the current window:

```
:mkview
```

Vim will decide where to store the view. When you later edit the same file you get the view back with this command:

```
:loadview
```

That's easy, isn't it?

Now you want to view the file without the **'number'** option on, or with all folds open, you can set the options to make the window look that way. Then store this view with:

```
:mkview 1
```

Obviously, you can get this back with:

```
:loadview 1
```

Now you can switch between the two views on the file by using `":loadview"` with and without the `"1"` argument.

You can store up to ten views for the same file this way, one unnumbered and nine numbered 1 to 9.

A VIEW WITH A NAME

The second basic way to use views is by storing the view in a file with a name you choose. This view can be loaded while editing another file. Vim will then switch to editing the file specified in the view. Thus you can use this to quickly switch to editing another file, with all its options set as you saved them.

For example, to save the view of the current file:

```
:mkview ~/.vim/main.vim
```

You can restore it with:

```
:source ~/.vim/main.vim
```

21.6 Modelines

When editing a specific file, you might set options specifically for that file. Typing these commands each time is boring. Using a session or view for editing a file doesn't work when sharing the file between several people.

The solution for this situation is adding a modeline to the file. This is a line of text that tells Vim the values of options, to be used in this file only.

A typical example is a C program where you make indents by a multiple of 4 spaces. This requires setting the `'shiftwidth'` option to 4. This modeline will do that:

```
/* vim:set shiftwidth=4: */
```

Put this line as one of the first or last five lines in the file. When editing the file, you will notice that `'shiftwidth'` will have been set to four. When editing another file, it's set back to the default value of eight.

For some files the modeline fits well in the header, thus it can be put at the top of the file. For text files and other files where the modeline gets in the way of the normal contents, put it at the end of the file.

The `'modelines'` option specifies how many lines at the start and end of the file are inspected for containing a modeline. To inspect ten lines:

```
:set modelines=10
```

The `'modeline'` option can be used to switch this off. Do this when you are working as root on Unix or Administrator on MS-Windows, or when you don't trust the files you are editing:

```
:set nomodeline
```

Use this format for the modeline:

```
any-text vim:set {option}={value} ... : any-text
```

The "any-text" indicates that you can put any text before and after the part that Vim will use. This allows making it look like a comment, like what was done above with `/*` and `*/`.

The "vim:" part is what makes Vim recognize this line. There must be white space before "vim", or "vim" must be at the start of the line. Thus using something like "gvim:" will not work.

The part between the colons is a `":set"` command. It works the same way as typing the `":set"` command, except that you need to insert a backslash before a colon (otherwise it would be seen as the end of the modeline).

Another example:

```
// vim:set textwidth=72 dir=c\:\tmp: use c:\tmp here
```

There is an extra backslash before the first colon, so that it's included in the ":set" command. The text after the second colon is ignored, thus a remark can be placed there.

For more details see [modeline](#) .

=====

Next chapter: [usr_22.txt](#) Finding the file to edit

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_22.txt For Vim version 9.1. Last change: 2020 Mar 28

VIM USER MANUAL - by Bram Moolenaar

Finding the file to edit

Files can be found everywhere. So how do you find them? Vim offers various ways to browse the directory tree. There are commands to jump to a file that is mentioned in another. And Vim remembers which files have been edited before.

- 22.1 The file browser
- 22.2 The current directory
- 22.3 Finding a file
- 22.4 The buffer list

Next chapter: [usr_23.txt](#) Editing other files
Previous chapter: [usr_21.txt](#) Go away and come back
Table of contents: [usr_toc.txt](#)

22.1 The file browser

Vim has a plugin that makes it possible to edit a directory. Try this:

```
:edit .
```

Through the magic of autocommands and Vim scripts, the window will be filled with the contents of the directory. It looks like this:

```
" =====  
" Netrw Directory Listing (netrw v109)  
" Sorted by name  
" Sort sequence: [\/]$,\.h$,\.c$,\.cpp$,*,\.info$,\.swp$,\.o$\\.obj$,\.bak$  
" Quick Help: <F1>:help -:go up dir D:delete R:rename s:sort-by x:exec  
" =====  
../  
./  
check/  
Makefile  
autocmd.txt  
change.txt  
eval.txt~  
filetype.txt~  
help.txt.info
```

You can see these items:

1. The name of the browsing tool and its version number
2. The name of the browsing directory
3. The method of sorting (may be by name, time, or size)
4. How names are to be sorted (directories first, then *.h files, *.c files, etc)

5. How to get help (use the <F1> key), and an abbreviated listing of available commands
6. A listing of files, including "../", which allows one to list the parent directory.

If you have syntax highlighting enabled, the different parts are highlighted so as to make it easier to spot them.

You can use Normal mode Vim commands to move around in the text. For example, move the cursor atop a file and press <Enter>; you will then be editing that file. To go back to the browser use ":edit ." again, or use ":Explore".

CTRL-O also works.

Try using <Enter> while the cursor is atop a directory name. The result is that the file browser moves into that directory and displays the items found there. Pressing <Enter> on the first directory "../" moves you one level higher. Pressing "-" does the same thing, without the need to move to the "../" item first.

You can press <F1> to get help on the things you can do in the netrw file browser. This is what you get:

```

9. Directory Browsing      netrw-browse  netrw-dir    netrw-list  netrw-help
MAPS                        netrw-maps
<F1>.....Help.....|netrw-help|
<cr>.....Browsing.....|netrw-cr|
<del>.....Deleting Files or Directories.....|netrw-delete|
-.....Going Up.....|netrw--|
a.....Hiding Files or Directories.....|netrw-a|
mb.....Bookmarking a Directory.....|netrw-mb|
gb.....Changing to a Bookmarked Directory.....|netrw-gb|
cd.....Make Browsing Directory The Current Dir....|netrw-c|
d.....Make A New Directory.....|netrw-d|
D.....Deleting Files or Directories.....|netrw-D|
<c-h>.....Edit File/Directory Hiding List.....|netrw-ctrl-h|
i.....Change Listing Style.....|netrw-i|
<c-l>.....Refreshing the Listing.....|netrw-ctrl-l|
o.....Browsing with a Horizontal Split.....|netrw-o|
p.....Use Preview Window.....|netrw-p|
P.....Edit in Previous Window.....|netrw-p|
q.....Listing Bookmarks and History.....|netrw-qb|
r.....Reversing Sorting Order.....|netrw-r|
(etc)

```

The <F1> key thus brings you to a netrw directory browsing contents help page. It's a regular help page; use the usual CTRL-] to jump to tagged help items and CTRL-O to jump back.

To select files for display and editing: (with the cursor is atop a filename)

```

<enter>      Open the file in the current window.      netrw-cr
o            Horizontally split window and display file netrw-o
v            Vertically split window and display file  netrw-v

```

p	Use the preview-window	netrw-p
P	Edit in the previous window	netrw-P
t	Open file in a new tab	netrw-t

The following normal-mode commands may be used to control the browser display:

i	Controls listing style (thin, long, wide, and tree). The long listing includes size and date information.
s	Repeatedly pressing s will change the way the files are sorted; one may sort on name, modification time, or size.
r	Reverse the sorting order.

As a sampling of extra normal-mode commands:

cd	Change Vim's notion of the current directory to be the same as the browser directory. (see g:netrw_keepdir to control this, too)
R	Rename the file or directory under the cursor; a prompt will be issued for the new name.
D	Delete the file or directory under the cursor; a confirmation request will be issued.
mb gb	Make bookmark/goto bookmark

One may also use command mode; again, just a sampling:

:Explore [directory]	Browse specified/current directory
:NetrwSettings	A comprehensive list of your current netrw settings with help linkage.

The netrw browser is not limited to just your local machine; one may use urls such as: (that trailing / is important)

```
:Explore ftp://somehost/path/to/dir/
:e scp://somehost/path/to/dir/
```

See [netrw-browse](#) for more.

22.2 The current directory

Just like the shell, Vim has the concept of a current directory. Suppose you are in your home directory and want to edit several files in a directory "VeryLongFileName". You could do:

```
:edit VeryLongFileName/file1.txt
:edit VeryLongFileName/file2.txt
:edit VeryLongFileName/file3.txt
```

To avoid much of the typing, do this:

```
:cd VeryLongFileName
:edit file1.txt
```

```
:edit file2.txt
:edit file3.txt
```

The ":cd" command changes the current directory. You can see what the current directory is with the ":pwd" command:

```
:pwd
/home/Bram/VeryLongFileName
```

Vim remembers the last directory that you used. Use "cd -" to go back to it. Example:

```
:pwd
/home/Bram/VeryLongFileName
:cd /etc
:pwd
/etc
:cd -
:pwd
/home/Bram/VeryLongFileName
:cd -
:pwd
/etc
```

WINDOW LOCAL DIRECTORY

When you split a window, both windows use the same current directory. When you want to edit a number of files somewhere else in the new window, you can make it use a different directory, without changing the current directory in the other window. This is called a local directory.

```
:pwd
/home/Bram/VeryLongFileName
:split
:lcd /etc
:pwd
/etc
CTRL-W w
:pwd
/home/Bram/VeryLongFileName
```

So long as no `:lcd` command has been used, all windows share the same current directory. Doing a `:cd` command in one window will also change the current directory of the other window.

For a window where `:lcd` has been used a different current directory is remembered. Using `:cd` or `:lcd` in other windows will not change it.

When using a `:cd` command in a window that uses a different current directory, it will go back to using the shared directory.

TAB LOCAL DIRECTORY

When you open a new tab page, it uses the directory of the window in the

previous tab page from which the new tab page was opened. You can change the directory of the current tab page using the `:tcd` command. All the windows in a tab page share this directory except for windows with a window-local directory. Any new windows opened in this tab page will use this directory as the current working directory. Using a `:cd` command in a tab page will not change the working directory of tab pages which have a tab local directory. When the global working directory is changed using the `:cd` command in a tab page, it will also change the current tab page working directory.

22.3 Finding a file

You are editing a C program that contains this line:

```
#include "inits.h"
```

You want to see what is in that "inits.h" file. Move the cursor on the name of the file and type:

```
gf
```

Vim will find the file and edit it.

What if the file is not in the current directory? Vim will use the `'path'` option to find the file. This option is a list of directory names where to look for your file.

Suppose you have your include files located in "c:/prog/include". This command will add it to the `'path'` option:

```
:set path+=c:/prog/include
```

This directory is an absolute path. No matter where you are, it will be the same place. What if you have located files in a subdirectory, below where the file is? Then you can specify a relative path name. This starts with a dot:

```
:set path+=./proto
```

This tells Vim to look in the directory "proto", below the directory where the file in which you use "gf" is. Thus using "gf" on "inits.h" will make Vim look for "proto/inits.h", starting in the directory of the file.

Without the "./", thus "proto", Vim would look in the "proto" directory below the current directory. And the current directory might not be where the file that you are editing is located.

The `'path'` option allows specifying the directories where to search for files in many more ways. See the help on the `'path'` option.

The `'isfname'` option is used to decide which characters are included in the file name, and which ones are not (e.g., the " character in the example above).

When you know the file name, but it's not to be found in the file, you can type it:

```
:find inits.h
```

Vim will then use the **'path'** option to try and locate the file. This is the same as the `":edit"` command, except for the use of **'path'**.

To open the found file in a new window use **CTRL-W f** instead of `"gf"`, or use `":sfind"` instead of `":find"`.

A nice way to directly start Vim to edit a file somewhere in the **'path'**:

```
vim "+find stdio.h"
```

This finds the file `"stdio.h"` in your value of **'path'**. The quotes are necessary to have one argument `+c` .

22.4 The buffer list

The Vim editor uses the term buffer to describe a file being edited. Actually, a buffer is a copy of the file that you edit. When you finish changing the buffer, you write the contents of the buffer to the file. Buffers not only contain file contents, but also all the marks, settings, and other stuff that goes with it.

HIDDEN BUFFERS

Suppose you are editing the file `one.txt` and need to edit the file `two.txt`. You could simply use `":edit two.txt"`, but since you made changes to `one.txt` that won't work. You also don't want to write `one.txt` yet. Vim has a solution for you:

```
:hide edit two.txt
```

The buffer `"one.txt"` disappears from the screen, but Vim still knows that you are editing this buffer, so it keeps the modified text. This is called a hidden buffer: The buffer contains text, but you can't see it.

The argument of `":hide"` is another command. `":hide"` makes that command behave as if the **'hidden'** option was set. You could also set this option yourself. The effect is that when any buffer is abandoned, it becomes hidden.

Be careful! When you have hidden buffers with changes, don't exit Vim without making sure you have saved all the buffers.

INACTIVE BUFFERS

When a buffer has been used once, Vim remembers some information about it. When it is not displayed in a window and it is not hidden, it is still in the buffer list. This is called an inactive buffer. Overview:

Active	Appears in a window, text loaded.
Hidden	Not in a window, text loaded.
Inactive	Not in a window, no text loaded.

The inactive buffers are remembered, because Vim keeps information about them, like marks. And remembering the file name is useful too, so that you can see which files you have edited. And edit them again.

LISTING BUFFERS

View the buffer list with this command:

```
:buffers
```

A command which does the same, is not so obvious to list buffers, but is much shorter to type:

```
:ls
```

The output could look like this:

```
1 #h "help.txt" line 62
2 %a + "usr_21.txt" line 1
3 "usr_toc.txt" line 1
```

The first column contains the buffer number. You can use this to edit the buffer without having to type the name, see below.

After the buffer number come the flags. Then the name of the file and the line number where the cursor was the last time.

The flags that can appear are these (from left to right):

```
u      Buffer is unlisted  unlisted-buffer .
%      Current buffer.
#      Alternate buffer.
a      Buffer is loaded and displayed.
h      Buffer is loaded but hidden.
=      Buffer is read-only.
-      Buffer is not modifiable, the 'modifiable' option is off.
+      Buffer has been modified.
```

EDITING A BUFFER

You can edit a buffer by its number. That avoids having to type the file name:

```
:buffer 2
```

But the only way to know the number is by looking in the buffer list. You can use the name, or part of it, instead:

```
:buffer help
```

Vim will find the best match for the name you type. If there is only one buffer that matches the name, it will be used. In this case "help.txt".

To open a buffer in a new window:

```
:sbuffer 3
```

This works with a name as well.

USING THE BUFFER LIST

You can move around in the buffer list with these commands:

```
:bnext      go to next buffer
:bprevious  go to previous buffer
:bfirst     go to the first buffer
:blast     go to the last buffer
```

To remove a buffer from the list, use this command:

```
:bdelete 3
```

Again, this also works with a name.

If you delete a buffer that was active (visible in a window), that window will be closed. If you delete the current buffer, the current window will be closed. If it was the last window, Vim will find another buffer to edit. You can't be editing nothing!

Note:

Even after removing the buffer with `":bdelete"` Vim still remembers it. It's actually made "unlisted", it no longer appears in the list from `":buffers"`. The `":buffers!"` command will list unlisted buffers (yes, Vim can do the impossible). To really make Vim forget about a buffer, use `":bwipe"`. Also see the **'buflisted'** option.

=====

Next chapter: [usr_23.txt](#) Editing other files

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_23.txt For Vim version 9.1. Last change: 2020 Dec 19

VIM USER MANUAL - by Bram Moolenaar

Editing other files

This chapter is about editing files that are not ordinary files. With Vim you can edit files that are compressed or encrypted. Some files need to be accessed over the internet. With some restrictions, binary files can be edited as well.

- 23.1 DOS, Mac and Unix files
- 23.2 Files on the internet
- 23.3 Encryption
- 23.4 Binary files
- 23.5 Compressed files

Next chapter: [usr_24.txt](#) Inserting quickly
Previous chapter: [usr_22.txt](#) Finding the file to edit
Table of contents: [usr_toc.txt](#)

23.1 DOS, Mac and Unix files

Back in the early days, the old Teletype machines used two characters to start a new line. One to move the carriage back to the first position (carriage return, <CR>), another to move the paper up (line feed, <LF>).

When computers came out, storage was expensive. Some people decided that they did not need two characters for end-of-line. The UNIX people decided they could use <New Line> or <NL> only for end-of-line. The Apple people standardized on <CR>. The Microsoft Windows folks decided to keep the old <CR><NL> (we use <NL> for line feed in the help text).

This means that if you try to move a file from one system to another, you have line-break problems. The Vim editor automatically recognizes the different file formats and handles things properly behind your back.

The option **'fileformats'** contains the various formats that will be tried when a new file is edited. The following command, for example, tells Vim to try UNIX format first and MS-DOS format second:

```
:set fileformats=unix,dos
```

You will notice the format in the message you get when editing a file. You don't see anything if you edit a native file format. Thus editing a Unix file on Unix won't result in a remark. But when you edit a dos file, Vim will notify you of this:

```
"/tmp/test" [dos] 3L, 71C
```

For a Mac file you would see "[mac]".

The detected file format is stored in the **'fileformat'** option. To see which format you have, execute the following command:

```
:set fileformat?
```

The three names that Vim uses are:

unix	<NL>
dos	<CR><NL>
mac	<CR>

USING THE MAC FORMAT

On Unix, <NL> is used to break a line. It's not unusual to have a <CR> character halfway a line. Incidentally, this happens quite often in Vi (and Vim) scripts.

On the Macintosh, where <CR> is the line break character, it's possible to have a <NL> character halfway a line.

The result is that it's not possible to be 100% sure whether a file containing both <CR> and <NL> characters is a Mac or a Unix file. Therefore, Vim assumes that on Unix you probably won't edit a Mac file, and doesn't check for this type of file. To check for this format anyway, add "mac" to **'fileformats'**:

```
:set fileformats+=mac
```

Then Vim will take a guess at the file format. Watch out for situations where Vim guesses wrong.

OVERRULING THE FORMAT

If you use the good old Vi and try to edit an MS-DOS format file, you will find that each line ends with a ^M character. (^M is <CR>). The automatic detection avoids this. Suppose you do want to edit the file that way? Then you need to overrule the format:

```
:edit ++ff=unix file.txt
```

The "++" string is an item that tells Vim that an option name follows, which overrules the default for this single command. "++ff" is used for **'fileformat'**. You could also use "++ff=mac" or "++ff=dos".

This doesn't work for any option, only "++ff" and "++enc" are currently implemented. The full names "++fileformat" and "++encoding" also work.

CONVERSION

You can use the **'fileformat'** option to convert from one file format to another. Suppose, for example, that you have an MS-DOS file named README.TXT that you want to convert to UNIX format. Start by editing the MS-DOS format file:

```
vim README.TXT
```

Vim will recognize this as a dos format file. Now change the file format to UNIX:

```
:set fileformat=unix
:write
```

The file is written in Unix format.

23.2 Files on the internet

Someone sends you an e-mail message, which refers to a file by its URL. For example:

You can find the information here:
<ftp://ftp.vim.org/pub/vim/README>

You could start a program to download the file, save it on your local disk and then start Vim to edit it.

There is a much simpler way. Move the cursor to any character of the URL. Then use this command:

```
gf
```

With a bit of luck, Vim will figure out which program to use for downloading the file, download it and edit the copy. To open the file in a new window use **CTRL-W f**.

If something goes wrong you will get an error message. It's possible that the URL is wrong, you don't have permission to read it, the network connection is down, etc. Unfortunately, it's hard to tell the cause of the error. You might want to try the manual way of downloading the file.

Accessing files over the internet works with the netrw plugin. Currently URLs with these formats are recognized:

ftp://	uses ftp
rcp://	uses rcp
scp://	uses scp
http://	uses wget (reading only)

Vim doesn't do the communication itself, it relies on the mentioned programs to be available on your computer. On most Unix systems "ftp" and "rcp" will be present. "scp" and "wget" might need to be installed.

Vim detects these URLs for each command that starts editing a new file, also with ":edit" and ":split", for example. Write commands also work, except for http://.

For more information, also about passwords, see [netrw](#) .

23.3 Encryption

Some information you prefer to keep to yourself. For example, when writing a test on a computer that students also use. You don't want clever students to figure out a way to read the questions before the exam starts. Vim can encrypt the file for you, which gives you some protection.

To start editing a new file with encryption, use the "-x" argument to start Vim. Example:

```
vim -x exam.txt
```

Vim prompts you for a key used for encrypting and decrypting the file:

```
Enter encryption key:
```

Carefully type the secret key now. You cannot see the characters you type, they will be replaced by stars. To avoid the situation that a typing mistake will cause trouble, Vim asks you to enter the key again:

```
Enter same key again:
```

You can now edit this file normally and put in all your secrets. When you finish editing the file and tell Vim to exit, the file is encrypted and written.

When you edit the file with Vim, it will ask you to enter the same key again. You don't need to use the "-x" argument. You can also use the normal ":edit" command. Vim adds a magic string to the file by which it recognizes that the file was encrypted.

If you try to view this file using another program, all you get is garbage. Also, if you edit the file with Vim and enter the wrong key, you get garbage. Vim does not have a mechanism to check if the key is the right one (this makes it much harder to break the key).

SWITCHING ENCRYPTION ON AND OFF

To disable the encryption of a file, set the 'key' option to an empty string:

```
:set key=
```

The next time you write the file this will be done without encryption.

Setting the 'key' option to enable encryption is not a good idea, because the password appears in the clear. Anyone shoulder-surfing can read your password.

To avoid this problem, the ":X" command was created. It asks you for an encryption key, just like the "-x" argument did:

```
:X
Enter encryption key: *****
Enter same key again: *****
```

LIMITS ON ENCRYPTION

The encryption algorithm used by Vim is not very strong. It is good enough to keep out the casual prowler, but not good enough to keep out a cryptology expert with lots of time on his hands. The text in the swap file and the undo file is also encrypted. However, this is done block-by-block and may reduce the time needed to crack a password. You can disable the swap file, but then a crash will cause you to lose your work, since Vim keeps all the text in

memory only. The undo file can be disabled with the only disadvantage that you can't undo after unloading the buffer.

To avoid using a swap file, supply the `-n` argument on the command line. For example, to edit the encrypted file "file.txt" without a swap file use the following command:

```
vim -x -n file.txt
```

When already editing a file, the swapfile can be disabled with:

```
:setlocal noswapfile
```

Since there is no swapfile, recovery will be impossible. Save the file a bit more often to avoid the risk of losing your changes.

While the file is in memory, it is in plain text. Anyone with privilege can look in the editor's memory and discover the contents of the file.

If you use a viminfo file, be aware that the contents of text registers are written out in the clear as well.

If you really want to secure the contents of a file, edit it only on a portable computer not connected to a network, use good encryption tools, and keep the computer locked up in a big safe when not in use.

23.4 Binary files

You can edit binary files with Vim. Vim wasn't really made for this, thus there are a few restrictions. But you can read a file, change a character and write it back, with the result that only that one character was changed and the file is identical otherwise.

To make sure that Vim does not use its clever tricks in the wrong way, add the `"-b"` argument when starting Vim:

```
vim -b datafile
```

This sets the **'binary'** option. The effect of this is that unexpected side effects are turned off. For example, **'textwidth'** is set to zero, to avoid automatic formatting of lines. And files are always read in Unix file format.

Binary mode can be used to change a message in a program. Be careful not to insert or delete any characters, it would stop the program from working. Use "R" to enter replace mode.

Many characters in the file will be unprintable. To see them in Hex format:

```
:set display=uhex
```

Otherwise, the "ga" command can be used to see the value of the character under the cursor. The output, when the cursor is on an `<Esc>`, looks like this:

```
<^[] 27, Hex 1b, Octal 033
```

There might not be many line breaks in the file. To get some overview switch

the **'wrap'** option off:

```
:set nowrap
```

BYTE POSITION

To see on which byte you are in the file use this command:

```
g CTRL-G
```

The output is verbose:

```
Col 9-16 of 9-16; Line 277 of 330; Word 1806 of 2058; Byte 10580 of 12206
```

The last two numbers are the byte position in the file and the total number of bytes. This takes into account how **'fileformat'** changes the number of bytes that a line break uses.

To move to a specific byte in the file, use the "go" command. For example, to move to byte 2345:

```
2345go
```

USING XXD

A real binary editor shows the text in two ways: as it is and in hex format. You can do this in Vim by first converting the file with the "xxd" program. This comes with Vim.

First edit the file in binary mode:

```
vim -b datafile
```

Now convert the file to a hex dump with xxd:

```
:%!xxd
```

The text will look like this:

```
00000000: 1f8b 0808 39d7 173b 0203 7474 002b 4e49  ....9..;..tt.+NI
00000010: 4b2c 8660 eb9c ecac c462 eb94 345e 2e30  K,.`.....b..4^.0
00000020: 373b 2731 0b22 0ca6 c1a2 d669 1035 39d9  7;'1.".....i.59.
```

You can now view and edit the text as you like. Vim treats the information as ordinary text. Changing the hex does not cause the printable character to be changed, or the other way around.

Finally convert it back with:

```
:%!xxd -r
```

Only changes in the hex part are used. Changes in the printable text part on the right are ignored.

See the manual page of xxd for more information.

23.5 Compressed files

This is easy: You can edit a compressed file just like any other file. The "gzip" plugin takes care of decompressing the file when you edit it. And compressing it again when you write it.

These compression methods are currently supported:

.Z	compress
.gz	gzip
.bz2	bzip2

Vim uses the mentioned programs to do the actual compression and decompression. You might need to install the programs first.

Next chapter: [usr_24.txt](#) Inserting quickly

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Inserting quickly

When entering text, Vim offers various ways to reduce the number of keystrokes and avoid typing mistakes. Use Insert mode completion to repeat previously typed words. Abbreviate long words to short ones. Type characters that aren't on your keyboard.

- 24.1 Making corrections
- 24.2 Showing matches
- 24.3 Completion
- 24.4 Repeating an insert
- 24.5 Copying from another line
- 24.6 Inserting a register
- 24.7 Abbreviations
- 24.8 Entering special characters
- 24.9 Digraphs
- 24.10 Normal mode commands

Next chapter: [usr_25.txt](#) Editing formatted text
Previous chapter: [usr_23.txt](#) Editing other files
Table of contents: [usr_toc.txt](#)

24.1 Making corrections

The `<BS>` key was already mentioned. It deletes the character just before the cursor. The `` key does the same for the character under (after) the cursor.

When you typed a whole word wrong, use **CTRL-W**:

```
The horse had fallen to the sky
                                CTRL-W
The horse had fallen to the
```

If you really messed up a line and want to start over, use **CTRL-U** to delete it. This keeps the text after the cursor and the indent. Only the text from the first non-blank to the cursor is deleted. With the cursor on the "f" of "fallen" in the next line pressing **CTRL-U** does this:

```
The horse had fallen to the
                                CTRL-U
fallen to the
```

When you spot a mistake a few words back, you need to move the cursor there to correct it. For example, you typed this:

```
The horse had follen to the ground
```

You need to change "follen" to "fallen". With the cursor at the end, you

would type this to correct it:

	<code><Esc>4blraA</code>
get out of Insert mode	<code><Esc></code>
four words back	<code>4b</code>
move on top of the "o"	<code>l</code>
replace with "a"	<code>ra</code>
restart Insert mode	<code>A</code>

Another way to do this:

	<code><C-Left><C-Left><C-Left><C-Left><Right>a<End></code>
four words back	<code><C-Left><C-Left><C-Left><C-Left></code>
move on top of the "o"	<code><Right></code>
delete the "o"	<code></code>
insert an "a"	<code>a</code>
go to end of the line	<code><End></code>

This uses special keys to move around, while remaining in Insert mode. This resembles what you would do in a modeless editor. It's easier to remember, but takes more time (you have to move your hand from the letters to the cursor keys, and the `<End>` key is hard to press without looking at the keyboard).

These special keys are most useful when writing a mapping that doesn't leave Insert mode. The extra typing doesn't matter then.

An overview of the keys you can use in Insert mode:

<code><C-Home></code>	to start of the file
<code><PageUp></code>	a whole screenful up
<code><Home></code>	to start of line
<code><S-Left></code>	one word left
<code><C-Left></code>	one word left
<code><S-Right></code>	one word right
<code><C-Right></code>	one word right
<code><End></code>	to end of the line
<code><PageDown></code>	a whole screenful down
<code><C-End></code>	to end of the file

There are a few more, see [ins-special-special](#) .

24.2 Showing matches

When you type a `)` it would be nice to see with which `(` it matches. To make Vim do that use this command:

```
:set showmatch
```

When you now type a text like `"(example)"`, as soon as you type the `)` Vim will briefly move the cursor to the matching `(`, keep it there for half a second, and move back to where you were typing.

In case there is no matching `(`, Vim will beep. Then you know that you might have forgotten the `(` somewhere, or typed a `)` too many.

The match will also be shown for [] and {} pairs. You don't have to wait with typing the next character, as soon as Vim sees it the cursor will move back and inserting continues as before.

You can change the time Vim waits with the `'matchtime'` option. For example, to make Vim wait one and a half second:

```
:set matchtime=15
```

The time is specified in tenths of a second.

24.3 Completion

Vim can automatically complete words on insertion. You type the first part of a word, press **CTRL-P**, and Vim guesses the rest.

Suppose, for example, that you are creating a C program and want to type in the following:

```
total = ch_array[0] + ch_array[1] + ch_array[2];
```

You start by entering the following:

```
total = ch_array[0] + ch_
```

At this point, you tell Vim to complete the word using the command **CTRL-P**. Vim searches for a word that starts with what's in front of the cursor. In this case, it is "ch_", which matches with the word `ch_array`. So typing **CTRL-P** gives you the following:

```
total = ch_array[0] + ch_array
```

After a little more typing, you get this (ending in a space):

```
total = ch_array[0] + ch_array[1] +
```

If you now type **CTRL-P** Vim will search again for a word that completes the word before the cursor. Since there is nothing in front of the cursor, it finds the first word backwards, which is "ch_array". Typing **CTRL-P** again gives you the next word that matches, in this case "total". A third **CTRL-P** searches further back. If there is nothing else, it causes the editor to run out of words, so it returns to the original text, which is nothing. A fourth **CTRL-P** causes the editor to start over again with "ch_array".

To search forward, use **CTRL-N**. Since the search wraps around the end of the file, **CTRL-N** and **CTRL-P** will find the same matches, but in a different sequence. Hint: **CTRL-N** is Next-match and **CTRL-P** is Previous-match.

The Vim editor goes through a lot of effort to find words to complete. By default, it searches the following places:

1. Current file
2. Files in other windows
3. Other loaded files (hidden buffers)
4. Files which are not loaded (inactive buffers)

5. Tag files
6. All files #included by the current file

OPTIONS

You can customize the search order with the **'complete'** option.

The **'ignorecase'** option is used. When it is set, case differences are ignored when searching for matches.

A special option for completion is **'infercase'**. This is useful to find matches while ignoring case ('ignorecase' must be set) but still using the case of the word typed so far. Thus if you type "For" and Vim finds a match "fortunately", it will result in "Fortunately".

COMPLETING SPECIFIC ITEMS

If you know what you are looking for, you can use these commands to complete with a certain type of item:

CTRL-X CTRL-F	file names
CTRL-X CTRL-L	whole lines
CTRL-X CTRL-D	macro definitions (also in included files)
CTRL-X CTRL-I	current and included files
CTRL-X CTRL-K	words from a dictionary
CTRL-X CTRL-T	words from a thesaurus
CTRL-X CTRL-]	tags
CTRL-X CTRL-V	Vim command line

After each of them **CTRL-N** can be used to find the next match, **CTRL-P** to find the previous match.

More information for each of these commands here: [ins-completion](#) .

COMPLETING FILE NAMES

Let's take **CTRL-X CTRL-F** as an example. This will find file names. It scans the current directory for files and displays each one that matches the word in front of the cursor.

Suppose, for example, that you have the following files in the current directory:

```
main.c  sub_count.c  sub_done.c  sub_exit.c
```

Now enter Insert mode and start typing:

The exit code is in the file sub

At this point, you enter the command **CTRL-X CTRL-F**. Vim now completes the current word "sub" by looking at the files in the current directory. The first match is sub_count.c. This is not the one you want, so you match the next file by typing **CTRL-N**. This match is sub_done.c. Typing **CTRL-N** again

takes you to sub_exit.c. The results:

```
The exit code is in the file sub_exit.c
```

If the file name starts with / (Unix) or C:\ (MS-Windows) you can find all files in the file system. For example, type "/u" and **CTRL-X CTRL-F**. This will match "/usr" (this is on Unix):

```
the file is found in /usr/
```

If you now press **CTRL-N** you go back to "/u". Instead, to accept the "/usr/" and go one directory level deeper, use **CTRL-X CTRL-F** again:

```
the file is found in /usr/X11R6/
```

The results depend on what is found in your file system, of course. The matches are sorted alphabetically.

COMPLETING IN SOURCE CODE

Source code files are well structured. That makes it possible to do completion in an intelligent way. In Vim this is called Omni completion. In some other editors it's called intellisense, but that is a trademark.

The key to Omni completion is **CTRL-X CTRL-O**. Obviously the O stands for Omni here, so that you can remember it easier. Let's use an example for editing C source:

```
{
    struct foo *p;
    p->
```

The cursor is after "p->". Now type **CTRL-X CTRL-O**. Vim will offer you a list of alternatives, which are the items that "struct foo" contains. That is quite different from using **CTRL-P**, which would complete any word, while only members of "struct foo" are valid here.

For Omni completion to work you may need to do some setup. At least make sure filetype plugins are enabled. Your vimrc file should contain a line like this:

```
filetype plugin on
```

Or:

```
filetype plugin indent on
```

For C code you need to create a tags file and set the **'tags'** option. That is explained [ft-c-omni](#). For other filetypes you may need to do something similar, look below [compl-omni-filetypes](#). It only works for specific filetypes. Check the value of the **'omnifunc'** option to find out if it would work.

```
=====
24.4 Repeating an insert
```

If you press **CTRL-A**, the editor inserts the text you typed the last time you were in Insert mode.

Assume, for example, that you have a file that begins with the following:

```
"file.h"  
/* Main program begins */
```

You edit this file by inserting "#include " at the beginning of the first line:

```
#include "file.h"  
/* Main program begins */
```

You go down to the beginning of the next line using the commands "j^". You now start to insert a new "#include" line. So you type:

```
i CTRL-A
```

The result is as follows:

```
#include "file.h"  
#include /* Main program begins */
```

The "#include " was inserted because **CTRL-A** inserts the text of the previous insert. Now you type "main.h"<Enter> to finish the line:

```
#include "file.h"  
#include "main.h"  
/* Main program begins */
```

The **CTRL-@** command does a **CTRL-A** and then exits Insert mode. That's a quick way of doing exactly the same insertion again.

24.5 Copying from another line

The **CTRL-Y** command inserts the character above the cursor. This is useful when you are duplicating a previous line. For example, you have this line of C code:

```
b_array[i]->s_next = a_array[i]->s_next;
```

Now you need to type the same line, but with "s_prev" instead of "s_next". Start the new line, and press **CTRL-Y** 14 times, until you are at the "n" of "next":

```
b_array[i]->s_next = a_array[i]->s_next;  
b_array[i]->s_
```

Now you type "prev":

```
b_array[i]->s_next = a_array[i]->s_next;  
b_array[i]->s_prev
```


Continue pressing **CTRL-Y** until the following "next":

```
b_array[i]->s_next = a_array[i]->s_next;
b_array[i]->s_prev = a_array[i]->s_
```

Now type "prev;" to finish it off.

The **CTRL-E** command acts like **CTRL-Y** except it inserts the character below the cursor.

24.6 Inserting a register

The command **CTRL-R {register}** inserts the contents of the register. This is useful to avoid having to type a long word. For example, you need to type this:

```
r = VeryLongFunction(a) + VeryLongFunction(b) + VeryLongFunction(c)
```

The function name is defined in a different file. Edit that file and move the cursor on top of the function name there, and yank it into register v:

```
"vyiw
```

"v is the register specification, "yiw" is yank-inner-word. Now edit the file where the new line is to be inserted, and type the first letters:

```
r =
```

Now use **CTRL-R v** to insert the function name:

```
r = VeryLongFunction
```

You continue to type the characters in between the function name, and use **CTRL-R v** two times more.

You could have done the same with completion. Using a register is useful when there are many words that start with the same characters.

If the register contains characters such as **<BS>** or other special characters, they are interpreted as if they had been typed from the keyboard. If you do not want this to happen (you really want the **<BS>** to be inserted in the text), use the command **CTRL-R CTRL-R {register}**.

24.7 Abbreviations

An abbreviation is a short word that takes the place of a long one. For example, "ad" stands for "advertisement". Vim enables you to type an abbreviation and then will automatically expand it for you.

To tell Vim to expand "ad" into "advertisement" every time you insert it, use the following command:

```
:iabbrev ad advertisement
```

Now, when you type "ad", the whole word "advertisement" will be inserted into the text. This is triggered by typing a character that can't be part of a word, for example a space:

What Is Entered	What You See
I saw the a	I saw the a
I saw the ad	I saw the ad
I saw the ad<Space>	I saw the advertisement<Space>

The expansion doesn't happen when typing just "ad". That allows you to type a word like "add", which will not get expanded. Only whole words are checked for abbreviations.

ABBREVIATING SEVERAL WORDS

It is possible to define an abbreviation that results in multiple words. For example, to define "JB" as "Jack Benny", use the following command:

```
:iabbrev JB Jack Benny
```

As a programmer, I use two rather unusual abbreviations:

```
:iabbrev #b /*****  
:iabbrev #e <Space>*****/
```

These are used for creating boxed comments. The comment starts with #b, which draws the top line. I then type the comment text and use #e to draw the bottom line.

Notice that the #e abbreviation begins with a space. In other words, the first two characters are space-star. Usually Vim ignores spaces between the abbreviation and the expansion. To avoid that problem, I spell space as seven characters: <, S, p, a, c, e, >.

Note:

":iabbrev" is a long word to type. ":iab" works just as well. That's abbreviating the abbreviate command!

FIXING TYPING MISTAKES

It's very common to make the same typing mistake every time. For example, typing "teh" instead of "the". You can fix this with an abbreviation:

```
:abbreviate teh the
```

You can add a whole list of these. Add one each time you discover a common mistake.

LISTING ABBREVIATIONS

The ":abbreviate" command lists the abbreviations:

```

:abbreviate
i #e          *****/
i #b          /*****
i JB          Jack Benny
i ad          advertisement
! teh        the

```

The "i" in the first column indicates Insert mode. These abbreviations are only active in Insert mode. Other possible characters are:

```

c          Command-line mode          :cabbrev
!          both Insert and Command-line mode  :abbreviate

```

Since abbreviations are not often useful in Command-line mode, you will mostly use the ":iabbrev" command. That avoids, for example, that "ad" gets expanded when typing a command like:

```
:edit ad
```

DELETING ABBREVIATIONS

To get rid of an abbreviation, use the ":unabbreviate" command. Suppose you have the following abbreviation:

```
:abbreviate @f fresh
```

You can remove it with this command:

```
:unabbreviate @f
```

While you type this, you will notice that @f is expanded to "fresh". Don't worry about this, Vim understands it anyway (except when you have an abbreviation for "fresh", but that's very unlikely).

To remove all the abbreviations:

```
:abclear
```

":unabbreviate" and ":abclear" also come in the variants for Insert mode (":iunabbreviate" and ":iabclear") and Command-line mode (":cunabbreviate" and ":cabclear").

REMAPPING ABBREVIATIONS

There is one thing to watch out for when defining an abbreviation: The resulting string should not be mapped. For example:

```
:abbreviate @a adder
:imap dd disk-door
```

When you now type @a, you will get "adisk-doorer". That's not what you want. To avoid this, use the ":noreabbrev" command. It does the same as

":abbreviate", but avoids that the resulting string is used for mappings:

```
:noreabbrev @a adder
```

Fortunately, it's unlikely that the result of an abbreviation is mapped.

24.8 Entering special characters

The **CTRL-V** command is used to insert the next character literally. In other words, any special meaning the character has, it will be ignored. For example:

```
CTRL-V <Esc>
```

Inserts an escape character. Thus you don't leave Insert mode. (Don't type the space after **CTRL-V**, it's only to make this easier to read).

Note:

On MS-Windows **CTRL-V** is used to paste text. Use **CTRL-Q** instead of **CTRL-V**. On Unix, on the other hand, **CTRL-Q** does not work on some terminals, because it has a special meaning.

You can also use the command **CTRL-V {digits}** to insert a character with the decimal number **{digits}**. For example, the character number 127 is the **** character (but not necessarily the **** key!). To insert **** type:

```
CTRL-V 127
```

You can enter characters up to 255 this way. When you type fewer than two digits, a non-digit will terminate the command. To avoid the need of typing a non-digit, prepend one or two zeros to make three digits.

All the next commands insert a **<Tab>** and then a dot:

```
CTRL-V 9.  
CTRL-V 09.  
CTRL-V 009.
```

To enter a character in hexadecimal, use an "x" after the **CTRL-V**:

```
CTRL-V x7f
```

This also goes up to character 255 (CTRL-V xff). You can use "o" to type a character as an octal number and two more methods allow you to type up to a 16 bit and a 32 bit number (e.g., for a Unicode character):

```
CTRL-V o123  
CTRL-V u1234  
CTRL-V U12345678
```

24.9 Digraphs

Some characters are not on the keyboard. For example, the copyright character

(©). To type these characters in Vim, you use digraphs, where two characters represent one. To enter a ©, for example, you press three keys:

`CTRL-K Co`

To find out what digraphs are available, use the following command:

`:digraphs`

Vim will display the digraph table. Here are three lines of it:

```
AC ~_ 159 NS | 160 !I ; 161 Ct ¢ 162 Pd £ 163 Cu ¤ 164 Ye ¥ 165
BB | 166 SE § 167 ': " 168 Co © 169 -a ª 170 << « 171 NO ¬ 172
-- 173 Rg ® 174 'm ¯ 175 DG ° 176 +- ± 177 2S ² 178 3S ³ 179
```

This shows, for example, that the digraph you get by typing `CTRL-K Pd` is the character (£). This is character number 163 (decimal).

`Pd` is short for Pound. Most digraphs are selected to give you a hint about the character they will produce. If you look through the list you will understand the logic.

You can exchange the first and second character, if there is no digraph for that combination. Thus `CTRL-K dP` also works. Since there is no digraph for "dP" Vim will also search for a "Pd" digraph.

Note:

The digraphs depend on the character set that Vim assumes you are using. Always use `:digraphs` to find out which digraphs are currently available.

You can define your own digraphs. Example:

`:digraph a" ä`

This defines that `CTRL-K a"` inserts an ä character. You can also specify the character with a decimal number. This defines the same digraph:

`:digraph a" 228`

More information about digraphs here: [digraphs](#)

Another way to insert special characters is with a keymap. More about that here: [45.5](#)

24.10 Normal mode commands

Insert mode offers a limited number of commands. In Normal mode you have many more. When you want to use one, you usually leave Insert mode with `<Esc>`, execute the Normal mode command, and re-enter Insert mode with "i" or "a".

There is a quicker way. With `CTRL-O {command}` you can execute any Normal mode command from Insert mode. For example, to delete from the cursor to the end of the line:

`CTRL-O D`

You can execute only one Normal mode command this way. But you can specify a register or a count. A more complicated example:

`CTRL-O "g3dw`

This deletes up to the third word into register g.

=====

Next chapter: [usr_25.txt](#) Editing formatted text

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Editing formatted text

Text hardly ever comes in one sentence per line. This chapter is about breaking sentences to make them fit on a page and other formatting. Vim also has useful features for editing single-line paragraphs and tables.

- 25.1 Breaking lines
- 25.2 Aligning text
- 25.3 Indents and tabs
- 25.4 Dealing with long lines
- 25.5 Editing tables

Next chapter: [usr_26.txt](#) Repeating
Previous chapter: [usr_24.txt](#) Inserting quickly
Table of contents: [usr_toc.txt](#)

25.1 Breaking lines

Vim has a number of functions that make dealing with text easier. By default, the editor does not perform automatic line breaks. In other words, you have to press `<Enter>` yourself. This is useful when you are writing programs where you want to decide where the line ends. It is not so good when you are creating documentation and want the text to be at most 70 character wide.

If you set the `'textwidth'` option, Vim automatically inserts line breaks. Suppose, for example, that you want a very narrow column of only 30 characters. You need to execute the following command:

```
:set textwidth=30
```

Now you start typing (ruler added):

```
      1      2      3
12345678901234567890123456789012345
I taught programming for a whi
```

If you type `"l"` next, this makes the line longer than the 30-character limit. When Vim sees this, it inserts a line break and you get the following:

```
      1      2      3
12345678901234567890123456789012345
I taught programming for a
whil
```

Continuing on, you can type in the rest of the paragraph:

```
      1      2      3
12345678901234567890123456789012345
I taught programming for a
```

```
while. One time, I was stopped
by the Fort Worth police,
because my homework was too
hard. True story.
```

You do not have to type newlines; Vim puts them in automatically.

Note:

The **'wrap'** option makes Vim display lines with a line break, but this doesn't insert a line break in the file.

REFORMATTING

The Vim editor is not a word processor. In a word processor, if you delete something at the beginning of the paragraph, the line breaks are reworked. In Vim they are not; so if you delete the word "programming" from the first line, all you get is a short line:

```
          1          2          3
12345678901234567890123456789012345
I taught for a
while. One time, I was stopped
by the Fort Worth police,
because my homework was too
hard. True story.
```

This does not look good. To get the paragraph into shape you use the "gq" operator.

Let's first use this with a Visual selection. Starting from the first line, type:

```
v4jgq
```

"v" to start Visual mode, "4j" to move to the end of the paragraph and then the "gq" operator. The result is:

```
          1          2          3
12345678901234567890123456789012345
I taught for a while. One
time, I was stopped by the
Fort Worth police, because my
homework was too hard. True
story.
```

Note: there is a way to do automatic formatting for specific types of text layouts, see [auto-format](#) .

Since "gq" is an operator, you can use one of the three ways to select the text it works on: With Visual mode, with a movement and with a text object.

The example above could also be done with "gq4j". That's less typing, but you have to know the line count. A more useful motion command is "}". This moves to the end of a paragraph. Thus "gq}" formats from the cursor to the end of the current paragraph.

A very useful text object to use with "gq" is the paragraph. Try this:

```
gqap
```

"ap" stands for "a-paragraph". This formats the text of one paragraph (separated by empty lines). Also the part before the cursor.

If you have your paragraphs separated by empty lines, you can format the whole file by typing this:

```
ggqG
```

"gg" to move to the first line, "ggG" to format until the last line.

Warning: If your paragraphs are not properly separated, they will be joined together. A common mistake is to have a line with a space or tab. That's a blank line, but not an empty line.

Vim is able to format more than just plain text. See [fo-table](#) for how to change this. See the '[joinspaces](#)' option to change the number of spaces used after a full stop.

It is possible to use an external program for formatting. This is useful if your text can't be properly formatted with Vim's builtin command. See the '[formatprg](#)' option.

25.2 Aligning text

To center a range of lines, use the following command:

```
:{range}center [width]
```

{range} is the usual command-line range. [width] is an optional line width to use for centering. If [width] is not specified, it defaults to the value of '[textwidth](#)'. (If '[textwidth](#)' is 0, the default is 80.)

For example:

```
:1,5center 40
```

results in the following:

```
    I taught for a while. One
      time, I was stopped by the
Fort Worth police, because my
      homework was too hard. True
        story.
```

RIGHT ALIGNMENT

Similarly, the ":right" command right-justifies the text:

```
:1,5right 37
```

gives this result:

I taught for a while. One
time, I was stopped by the
Fort Worth police, because my
homework was too hard. True
story.

LEFT ALIGNMENT

Finally there is this command:

```
{range}left [margin]
```

Unlike `:center` and `:right`, however, the argument to `:left` is not the length of the line. Instead it is the left margin. If it is omitted, the text will be put against the left side of the screen (using a zero margin would do the same). If it is 5, the text will be indented 5 spaces. For example, use these commands:

```
:1left 5  
:2,5left
```

This results in the following:

I taught for a while. One
time, I was stopped by the
Fort Worth police, because my
homework was too hard. True
story.

JUSTIFYING TEXT

Vim has no built-in way of justifying text. However, there is a neat macro package that does the job. To use this package, execute the following command:

```
:packadd justify
```

Or put this line in your `vimrc` :

```
packadd! justify
```

This Vim script file defines a new visual command `"_j`". To justify a block of text, highlight the text in Visual mode and then execute `"_j`".

Look in the file for more explanations. To go there, do `"gf`" on this name: `$VIMRUNTIME/pack/dist/opt/justify/plugin/justify.vim`.

An alternative is to filter the text through an external program. Example:

```
:%!fmt
```

25.3 Indents and tabs

Indents can be used to make text stand out from the rest. The example texts in this manual, for example, are indented by eight spaces or a tab. You would normally enter this by typing a tab at the start of each line. Take this text:

```
the first line
the second line
```

This is entered by typing a tab, some text, <Enter>, tab and more text.

The **'autoindent'** option inserts indents automatically:

```
:set autoindent
```

When a new line is started it gets the same indent as the previous line. In the above example, the tab after the <Enter> is not needed anymore.

INCREASING INDENT

To increase the amount of indent in a line, use the ">" operator. Often this is used as ">>", which adds indent to the current line.

The amount of indent added is specified with the **'shiftwidth'** option. The default value is 8. To make ">>" insert four spaces worth of indent, for example, type this:

```
:set shiftwidth=4
```

When used on the second line of the example text, this is what you get:

```
the first line
    the second line
```

"4>>" will increase the indent of four lines.

TABSTOP

If you want to make indents a multiple of 4, you set **'shiftwidth'** to 4. But when pressing a <Tab> you still get 8 spaces worth of indent. To change this, set the **'softtabstop'** option:

```
:set softtabstop=4
```

This will make the <Tab> key insert 4 spaces worth of indent. If there are already four spaces, a <Tab> character is used (saving seven characters in the file). (If you always want spaces and no tab characters, set the **'expandtab'** option.)

Note:

You could set the **'tabstop'** option to 4. However, if you edit the file another time, with **'tabstop'** set to the default value of 8, it will look wrong. In other programs and when printing the indent will also be wrong. Therefore it is recommended to keep **'tabstop'** at eight all the time. That's the standard value everywhere.

CHANGING TABS

You edit a file which was written with a tabstop of 3. In Vim it looks ugly, because it uses the normal tabstop value of 8. You can fix this by setting **'tabstop'** to 3. But you have to do this every time you edit this file.

Vim can change the use of tabstops in your file. First, set **'tabstop'** to make the indents look good, then use the `":retab"` command:

```
:set tabstop=3
:retab 8
```

The `":retab"` command will change **'tabstop'** to 8, while changing the text such that it looks the same. It changes spans of white space into tabs and spaces for this. You can now write the file. Next time you edit it the indents will be right without setting an option.

Warning: When using `":retab"` on a program, it may change white space inside a string constant. Therefore it's a good habit to use `"\t"` instead of a real tab.

25.4 Dealing with long lines

Sometimes you will be editing a file that is wider than the number of columns in the window. When that occurs, Vim wraps the lines so that everything fits on the screen.

If you switch the **'wrap'** option off, each line in the file shows up as one line on the screen. Then the ends of the long lines disappear off the screen to the right.

When you move the cursor to a character that can't be seen, Vim will scroll the text to show it. This is like moving a viewport over the text in the horizontal direction.

By default, Vim does not display a horizontal scrollbar in the GUI. If you want to enable one, use the following command:

```
:set guioptions+=b
```

One horizontal scrollbar will appear at the bottom of the Vim window.

If you don't have a scrollbar or don't want to use it, use these commands to scroll the text. The cursor will stay in the same place, but it's moved back into the visible text if necessary.

zh	scroll right
4zh	scroll four characters right
zH	scroll half a window width right
ze	scroll right to put the cursor at the end
zl	scroll left
4zl	scroll four characters left
zL	scroll half a window width left
zs	scroll left to put the cursor at the start

Let's attempt to show this with one line of text. The cursor is on the "w" of "which". The "current window" above the line indicates the text that is

currently visible. The "window"s below the text indicate the text that is visible after the command left of it.

```

                                |<-- current window -->|
                                some long text, part of which is visible in the window
ze      |<--      window      -->|
zH      |<--      window      -->|
4zh     |<--      window      -->|
zh      |<--      window      -->|
zL      |<--      window      -->|
4zL     |<--      window      -->|
zL      |<--      window      -->|
zs      |<--      window      -->|

```

MOVING WITH WRAP OFF

When **'wrap'** is off and the text has scrolled horizontally, you can use the following commands to move the cursor to a character you can see. Thus text left and right of the window is ignored. These never cause the text to scroll:

```

g0      to first visible character in this line
g^      to first non-blank visible character in this line
gm      to middle of screen line
gM      to middle of the text in this line
g$      to last visible character in this line

```

```

                                |<--      window      -->|
                                some long text, part of which is visible in one line
                                g0  g^   gm   gM  g$

```

BREAKING AT WORDS

edit-no-break

When preparing text for use by another program, you might have to make paragraphs without a line break. A disadvantage of using **'nowrap'** is that you can't see the whole sentence you are working on. When **'wrap'** is on, words are broken halfway, which makes them hard to read.

A good solution for editing this kind of paragraph is setting the **'linebreak'** option. Vim then breaks lines at an appropriate place when displaying the line. The text in the file remains unchanged.

Without **'linebreak'** text might look like this:

```

+-----+
|letter generation program for a b|
|ank. They wanted to send out a s|
|pecial, personalized letter to th|
|eir richest 1000 customers. Unfo|
|rtunately for the programmer, he |
+-----+

```

After:

```
:set linebreak
```

it looks like this:

```
+-----+
|letter generation program for a |
|bank. They wanted to send out a |
|special, personalized letter to |
|their richest 1000 customers.   |
|Unfortunately for the programmer,|
+-----+
```

Related options:

'breakat' specifies the characters where a break can be inserted.

'showbreak' specifies a string to show at the start of broken line.

Set **'textwidth'** to zero to avoid a paragraph to be split.

MOVING BY VISIBLE LINES

The "j" and "k" commands move to the next and previous lines. When used on a long line, this means moving a lot of screen lines at once.

To move only one screen line, use the "gj" and "gk" commands. When a line doesn't wrap they do the same as "j" and "k". When the line does wrap, they move to a character displayed one line below or above.

You might like to use these mappings, which bind these movement commands to the cursor keys:

```
:map <Up> gk
:map <Down> gj
```

TURNING A PARAGRAPH INTO ONE LINE

[edit-paragraph-join](#)

If you want to import text into a program like MS-Word, each paragraph should be a single line. If your paragraphs are currently separated with empty lines, this is how you turn each paragraph into a single line:

```
:g/./,/^$/join
```

That looks complicated. Let's break it up in pieces:

<pre>:g/./</pre>	A ":global" command that finds all lines that contain at least one character.
<pre>,/^\$/</pre>	A range, starting from the current line (the non-empty line) until an empty line.
<pre>join</pre>	The ":join" command joins the range of lines together into one line.

Starting with this text, containing eight lines broken at column 30:

```
+-----+
|A letter generation program      |
|for a bank. They wanted to      |
|send out a special,             |
+-----+
```

```
|personalized letter.          |
|                              |
|To their richest 1000        |
|customers. Unfortunately for  |
|the programmer,              |
+-----+

```

You end up with two lines:

```
+-----+
|A letter generation program for a |
|bank. They wanted to send out a s |
|pecial, personalized letter.      |
|To their richest 1000 customers.   |
|Unfortunately for the programmer,  |
+-----+

```

Note that this doesn't work when the separating line is blank but not empty; when it contains spaces and/or tabs. This command does work with blank lines:

```
:g/\S/,/^\s*$/join
```

This still requires a blank or empty line at the end of the file for the last paragraph to be joined.

25.5 Editing tables

Suppose you are editing a table with four columns:

```
nice table      test 1      test 2      test 3
input A         0.534
input B         0.913
```

You need to enter numbers in the third column. You could move to the second line, use "A", enter a lot of spaces and type the text.

For this kind of editing there is a special option:

```
set virtualedit=all
```

Now you can move the cursor to positions where there isn't any text. This is called "virtual space". Editing a table is a lot easier this way.

Move the cursor by searching for the header of the last column:

```
/test 3
```

Now press "j" and you are right where you can enter the value for "input A". Typing "0.693" results in:

```
nice table      test 1      test 2      test 3
input A         0.534         0.693
input B         0.913
```

Vim has automatically filled the gap in front of the new text for you. Now,

to enter the next field in this column use "Bj". "B" moves back to the start of a white space separated word. Then "j" moves to the place where the next field can be entered.

Note:

You can move the cursor anywhere in the display, also beyond the end of a line. But Vim will not insert spaces there, until you insert a character in that position.

COPYING A COLUMN

You want to add a column, which should be a copy of the third column and placed before the "test 1" column. Do this in seven steps:

1. Move the cursor to the left upper corner of this column, e.g., with "/test 3".
2. Press **CTRL-V** to start blockwise Visual mode.
3. Move the cursor down two lines with "2j". You are now in "virtual space": the "input B" line of the "test 3" column.
4. Move the cursor right, to include the whole column in the selection, plus the space that you want between the columns. "9l" should do it.
5. Yank the selected rectangle with "y".
6. Move the cursor to "test 1", where the new column must be placed.
7. Press "P".

The result should be:

```
nice table      test 3      test 1      test 2      test 3
input A         0.693      0.534
input B
```

Notice that the whole "test 1" column was shifted right, also the line where the "test 3" column didn't have text.

Go back to non-virtual cursor movements with:

```
:set virtualedit=
```

VIRTUAL REPLACE MODE

The disadvantage of using '**virtualedit**' is that it "feels" different. You can't recognize tabs or spaces beyond the end of line when moving the cursor around. Another method can be used: Virtual Replace mode.

Suppose you have a line in a table that contains both tabs and other characters. Use "rx" on the first tab:

```
inp      0.693  0.534  0.693
rx      |
        |
        v
inp0.693  0.534  0.693
```


The layout is messed up. To avoid that, use the "gr" command:

```
inp      0.693  0.534  0.693
grx      |
          |
          v
inpx     0.693  0.534  0.693
```

What happens is that the "gr" command makes sure the new character takes the right amount of screen space. Extra spaces or tabs are inserted to fill the gap. Thus what actually happens is that a tab is replaced by "x" and then blanks added to make the text after it keep its place. In this case a tab is inserted.

When you need to replace more than one character, you use the "R" command to go to Replace mode (see [04.9](#)). This messes up the layout and replaces the wrong characters:

```
inp      0      0.534  0.693
R0.786   |
          |
          v
inp      0.78634 0.693
```

The "gR" command uses Virtual Replace mode. This preserves the layout:

```
inp      0      0.534  0.693
gR0.786  |
          |
          v
inp      0.786  0.534  0.693
```

=====

Next chapter: [usr_26.txt](#) Repeating

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Repeating

An editing task is hardly ever unstructured. A change often needs to be made several times. In this chapter a number of useful ways to repeat a change will be explained.

- 26.1 Repeating with Visual mode
- 26.2 Add and subtract
- 26.3 Making a change in many files
- 26.4 Using Vim from a shell script

Next chapter: [usr_27.txt](#) Search commands and patterns
Previous chapter: [usr_25.txt](#) Editing formatted text
Table of contents: [usr_toc.txt](#)

26.1 Repeating with Visual mode

Visual mode is very handy for making a change in any sequence of lines. You can see the highlighted text, thus you can check if the correct lines are changed. But making the selection takes some typing. The "gv" command selects the same area again. This allows you to do another operation on the same text.

Suppose you have some lines where you want to change "2001" to "2002" and "2000" to "2001":

```
The financial results for 2001 are better
than for 2000. The income increased by 50%,
even though 2001 had more rain than 2000.

income          2000          2001
                45,403        66,234
```

First change "2001" to "2002". Select the lines in Visual mode, and use:

```
:s/2001/2002/g
```

Now use "gv" to reselect the same text. It doesn't matter where the cursor is. Then use ":s/2000/2001/g" to make the second change.

Obviously, you can repeat these changes several times.

26.2 Add and subtract

When repeating the change of one number into another, you often have a fixed offset. In the example above, one was added to each year. Instead of typing a substitute command for each year that appears, the **CTRL-A** command can be used.

Using the same text as above, search for a year:

```
/19[0-9][0-9]\|20[0-9][0-9]
```

Now press **CTRL-A**. The year will be increased by one:

```
The financial results for 2002 are better
than for 2000. The income increased by 50%,
even though 2001 had more rain than 2000.

income          2000          2001
                45,403         66,234
```

Use "n" to find the next year, and press "." to repeat the **CTRL-A** ("." is a bit quicker to type). Repeat "n" and "." for all years that appear.

Hint: set the **'hlsearch'** option to see the matches you are going to change, then you can look ahead and do it faster.

Adding more than one can be done by prepending the number to **CTRL-A**. Suppose you have this list:

```
1. item four
2. item five
3. item six
```

Move the cursor to "1." and type:

```
3 CTRL-A
```

The "1." will change to "4.". Again, you can use "." to repeat this on the other numbers.

Another example:

```
006    foo bar
007    foo bar
```

Using **CTRL-A** on these numbers results in:

```
007    foo bar
010    foo bar
```

7 plus one is 10? What happened here is that Vim recognized "007" as an octal number, because there is a leading zero. This notation is often used in C programs. If you do not want a number with leading zeros to be handled as octal, use this:

```
:set nrformats-=octal
```

The **CTRL-X** command does subtraction in a similar way.

26.3 Making a change in many files

Suppose you have a variable called "x_cnt" and you want to change it to "x_counter". This variable is used in several of your C files. You need to change it in all files. This is how you do it.

Put all the relevant files in the argument list:

```
:args *.c
```

This finds all C files and edits the first one. Now you can perform a substitution command on all these files:

```
:argdo %s/\<x_cnt\>/x_counter/ge | update
```

The ":argdo" command takes an argument that is another command. That command will be executed on all files in the argument list.

The "%s" substitute command that follows works on all lines. It finds the word "x_cnt" with "\<x_cnt\>". The "\<" and "\>" are used to match the whole word only, and not "px_cnt" or "x_cnt2".

The flags for the substitute command include "g" to replace all occurrences of "x_cnt" in the same line. The "e" flag is used to avoid an error message when "x_cnt" does not appear in the file. Otherwise ":argdo" would abort on the first file where "x_cnt" was not found.

The "|" separates two commands. The following "update" command writes the file only if it was changed. If no "x_cnt" was changed to "x_counter" nothing happens.

There is also the ":windo" command, which executes its argument in all windows. And ":bufdo" executes its argument on all buffers. Be careful with this, because you might have more files in the buffer list than you think. Check this with the ":buffers" command (or ":ls").

26.4 Using Vim from a shell script

Suppose you have a lot of files in which you need to change the string "-person-" to "Jones" and then print it. How do you do that? One way is to do a lot of typing. The other is to write a shell script to do the work.

The Vim editor does a superb job as a screen-oriented editor when using Normal mode commands. For batch processing, however, Normal mode commands do not result in clear, commented command files; so here you will use Ex mode instead. This mode gives you a nice command-line interface that makes it easy to put into a batch file. ("Ex command" is just another name for a command-line (:) command.)

The Ex mode commands you need are as follows:

```
%s/-person-/Jones/g
write tempfile
quit
```

You put these commands in the file "change.vim". Now to run the editor in batch mode, use this shell script:

```
for file in *.txt; do
    vim -e -s $file < change.vim
    lpr -r tempfile
done
```

The for-done loop is a shell construct to repeat the two lines in between,

while the `$file` variable is set to a different file name each time.

The second line runs the Vim editor in Ex mode (`-e` argument) on the file `$file` and reads commands from the file `"change.vim"`. The `-s` argument tells Vim to operate in silent mode. In other words, do not keep outputting the `:prompt`, or any other prompt for that matter.

The `"lpr -r tempfile"` command prints the resulting `"tempfile"` and deletes it (that's what the `-r` argument does).

READING FROM STDIN

Vim can read text on standard input. Since the normal way is to read commands there, you must tell Vim to read text instead. This is done by passing the `"-"` argument in place of a file. Example:

```
ls | vim -
```

This allows you to edit the output of the `"ls"` command, without first saving the text in a file.

If you use the standard input to read text from, you can use the `"-S"` argument to read a script:

```
producer | vim -S change.vim -
```

NORMAL MODE SCRIPTS

If you really want to use Normal mode commands in a script, you can use it like this:

```
vim -s script file.txt ...
```

Note:

`"-s"` has a different meaning when it is used without `"-e"`. Here it means to source the `"script"` as Normal mode commands. When used with `"-e"` it means to be silent, and doesn't use the next argument as a file name.

The commands in `"script"` are executed like you typed them. Don't forget that a line break is interpreted as pressing `<Enter>`. In Normal mode that moves the cursor to the next line.

To create the script you can edit the script file and type the commands. You need to imagine what the result would be, which can be a bit difficult. Another way is to record the commands while you perform them manually. This is how you do that:

```
vim -w script file.txt ...
```

All typed keys will be written to `"script"`. If you make a small mistake you can just continue and remember to edit the script later.

The `"-w"` argument appends to an existing script. That is good when you want to record the script bit by bit. If you want to start from scratch and start all over, use the `"-W"` argument. It overwrites any existing file.

Next chapter: [usr_27.txt](#) Search commands and patterns

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

In chapter 3 a few simple search patterns were mentioned [03.9](#) . Vim can do much more complex searches. This chapter explains the most often used ones. A detailed specification can be found here: [pattern](#)

- 27.1 Ignoring case
- 27.2 Wrapping around the file end
- 27.3 Offsets
- 27.4 Matching multiple times
- 27.5 Alternatives
- 27.6 Character ranges
- 27.7 Character classes
- 27.8 Matching a line break
- 27.9 Examples

Next chapter: [usr_28.txt](#) Folding
Previous chapter: [usr_26.txt](#) Repeating
Table of contents: [usr_toc.txt](#)

27.1 Ignoring case

By default, Vim's searches are case sensitive. Therefore, "include", "INCLUDE", and "Include" are three different words and a search will match only one of them.

Now switch on the **'ignorecase'** option:

```
:set ignorecase
```

Search for "include" again, and now it will match "Include", "INCLUDE" and "InClUDe". (Set the **'hlsearch'** option to quickly see where a pattern matches.)

You can switch this off again with:

```
:set noignorecase
```

But let's keep it set, and search for "INCLUDE". It will match exactly the same text as "include" did. Now set the **'smartcase'** option:

```
:set ignorecase smartcase
```

If you have a pattern with at least one uppercase character, the search becomes case sensitive. The idea is that you didn't have to type that uppercase character, so you must have done it because you wanted case to match. That's smart!

With these two options set you find the following matches:

pattern	matches
---------	---------

word	word, Word, WORD, WoRd, etc.
Word	Word
WORD	WORD
WoRd	WoRd

CASE IN ONE PATTERN

If you want to ignore case for one specific pattern, you can do this by prepending the "\c" string. Using "\C" will make the pattern to match case. This overrules the 'ignorecase' and 'smartcase' options, when "\c" or "\C" is used their value doesn't matter.

pattern	matches
\Cword	word
\CWord	Word
\cword	word, Word, WORD, WoRd, etc.
\cWord	word, Word, WORD, WoRd, etc.

A big advantage of using "\c" and "\C" is that it sticks with the pattern. Thus if you repeat a pattern from the search history, the same will happen, no matter if 'ignorecase' or 'smartcase' was changed.

Note:

The use of "\c" items in search patterns depends on the 'magic' option. In this chapter we will assume 'magic' is on, because that is the standard and recommended setting. If you would change 'magic', many search patterns would suddenly become invalid.

Note:

If your search takes much longer than you expected, you can interrupt it with **CTRL-C** on Unix and **CTRL-Break** on MS-Windows.

27.2 Wrapping around the file end

By default, a forward search starts searching for the given string at the current cursor location. It then proceeds to the end of the file. If it has not found the string by that time, it starts from the beginning and searches from the start of the file to the cursor location.

Keep in mind that when repeating the "n" command to search for the next match, you eventually get back to the first match. If you don't notice this you keep searching forever! To give you a hint, Vim displays this message:

```
search hit BOTTOM, continuing at TOP
```

If you use the "?" command, to search in the other direction, you get this message:

```
search hit TOP, continuing at BOTTOM
```

Still, you don't know when you are back at the first match. One way to see this is by switching on the 'ruler' option:

```
:set ruler
```

Vim will display the cursor position in the lower righthand corner of the window (in the status line if there is one). It looks like this:

```
101,29      84%
```

The first number is the line number of the cursor. Remember the line number where you started, so that you can check if you passed this position again.

NOT WRAPPING

To turn off search wrapping, use the following command:

```
:set nowrapscan
```

Now when the search hits the end of the file, an error message displays:

```
E385: search hit BOTTOM without match for: forever
```

Thus you can find all matches by going to the start of the file with "gg" and keep searching until you see this message.

If you search in the other direction, using "?", you get:

```
E384: search hit TOP without match for: forever
```

27.3 Offsets

By default, the search command leaves the cursor positioned on the beginning of the pattern. You can tell Vim to leave it some other place by specifying an offset. For the forward search command "/", the offset is specified by appending a slash (/) and the offset:

```
/default/2
```

This command searches for the pattern "default" and then moves to the beginning of the second line past the pattern. Using this command on the paragraph above, Vim finds the word "default" in the first line. Then the cursor is moved two lines down and lands on "an offset".

If the offset is a simple number, the cursor will be placed at the beginning of the line that many lines from the match. The offset number can be positive or negative. If it is positive, the cursor moves down that many lines; if negative, it moves up.

CHARACTER OFFSETS

The "e" offset indicates an offset from the end of the match. It moves the cursor onto the last character of the match. The command:

```
/const/e
```

puts the cursor on the "t" of "const".

From that position, adding a number moves forward that many characters. This command moves to the character just after the match:

```
/const/e+1
```

A positive number moves the cursor to the right, a negative number moves it to the left. For example:

```
/const/e-1
```

moves the cursor to the "s" of "const".

If the offset begins with "b", the cursor moves to the beginning of the pattern. That's not very useful, since leaving out the "b" does the same thing. It does get useful when a number is added or subtracted. The cursor then goes forward or backward that many characters. For example:

```
/const/b+2
```

Moves the cursor to the beginning of the match and then two characters to the right. Thus it lands on the "n".

REPEATING

To repeat searching for the previously used search pattern, but with a different offset, leave out the pattern:

```
/that  
//e
```

Is equal to:

```
/that/e
```

To repeat with the same offset:

```
/
```

"n" does the same thing. To repeat while removing a previously used offset:

```
//
```

SEARCHING BACKWARDS

The "?" command uses offsets in the same way, but you must use "?" to separate the offset from the pattern, instead of "/":

```
?const?e-2
```

The "b" and "e" keep their meaning, they don't change direction with the use

of "?".

START POSITION

When starting a search, it normally starts at the cursor position. When you specify a line offset, this can cause trouble. For example:

```
/const/-2
```

This finds the next word "const" and then moves two lines up. If you use "n" to search again, Vim could start at the current position and find the same "const" match. Then using the offset again, you would be back where you started. You would be stuck!

It could be worse: Suppose there is another match with "const" in the next line. Then repeating the forward search would find this match and move two lines up. Thus you would actually move the cursor back!

When you specify a character offset, Vim will compensate for this. Thus the search starts a few characters forward or backward, so that the same match isn't found again.

27.4 Matching multiple times

The "*" item specifies that the item before it can match any number of times. Thus:

```
/a*
```

matches "a", "aa", "aaa", etc. But also "" (the empty string), because zero times is included.

The "*" only applies to the item directly before it. Thus "ab*" matches "a", "ab", "abb", "abbb", etc. To match a whole string multiple times, it must be grouped into one item. This is done by putting "(" before it and ")" after it. Thus this command:

```
/\(ab\)*
```

Matches: "ab", "abab", "ababab", etc. And also "".

To avoid matching the empty string, use "+". This makes the previous item match one or more times.

```
/ab\+
```

Matches "ab", "abb", "abbb", etc. It does not match "a" when no "b" follows.

To match an optional item, use "\=". Example:

```
/folders\=
```

Matches "folder" and "folders".

SPECIFIC COUNTS

To match a specific number of items use the form "`\{n,m\}`". "n" and "m" are numbers. The item before it will be matched "n" to "m" times *inclusive*. Example:

```
/ab\{3,5}
```

matches "abbb", "abbbb" and "abbbbbb".

When "n" is omitted, it defaults to zero. When "m" is omitted it defaults to infinity. When ",m" is omitted, it matches exactly "n" times.

Examples:

pattern	match count
<code>\{,4\}</code>	0, 1, 2, 3 or 4
<code>\{3,\}</code>	3, 4, 5, etc.
<code>\{0,1\}</code>	0 or 1, same as <code>\=</code>
<code>\{0,\}</code>	0 or more, same as <code>*</code>
<code>\{1,\}</code>	1 or more, same as <code>\+</code>
<code>\{3\}</code>	3

MATCHING AS LITTLE AS POSSIBLE

The items so far match as many characters as they can find. To match as few as possible, use "`\{-n,m\}`". It works the same as "`\{n,m\}`", except that the minimal amount possible is used.

For example, use:

```
/ab\{-1,3}
```

Will match "ab" in "abbb". Actually, it will never match more than one b, because there is no reason to match more. It requires something else to force it to match more than the lower limit.

The same rules apply to removing "n" and "m". It's even possible to remove both of the numbers, resulting in "`\{-\}`". This matches the item before it zero or more times, as few as possible. The item by itself always matches zero times. It is useful when combined with something else. Example:

```
/a.\{-}b
```

This matches "axb" in "axbxb". If this pattern would be used:

```
/a.*b
```

It would try to match as many characters as possible with "`.*`", thus it matches "axbxb" as a whole.

27.5 Alternatives

The "or" operator in a pattern is "`\|`". Example:

```
/foo\|bar
```

This matches "foo" or "bar". More alternatives can be concatenated:

```
/one\|two\|three
```

Matches "one", "two" and "three".

To match multiple times, the whole thing must be placed in "\"(" and "\)":

```
/\(foo\|bar\)\/+
```

This matches "foo", "foobar", "foofoo", "barfoobar", etc.

Another example:

```
/end\(if\|while\|for\)
```

This matches "endif", "endwhile" and "endfor".

A related item is "\&". This requires that both alternatives match in the same place. The resulting match uses the last alternative. Example:

```
/forever\&...
```

This matches "for" in "forever". It will not match "fortuin", for example.

27.6 Character ranges

To match "a", "b" or "c" you could use "/a\|b\|c". When you want to match all letters from "a" to "z" this gets very long. There is a shorter method:

```
/[a-z]
```

The [] construct matches a single character. Inside you specify which characters to match. You can include a list of characters, like this:

```
/[0123456789abcdef]
```

This will match any of the characters included. For consecutive characters you can specify the range. "0-3" stands for "0123". "w-z" stands for "wxyz". Thus the same command as above can be shortened to:

```
/[0-9a-f]
```

To match the "-" character itself make it the first or last one in the range. These special characters are accepted to make it easier to use them inside a [] range (they can actually be used anywhere in the search pattern):

```
\e    <Esc>  
\t    <Tab>  
\r    <CR>  
\b    <BS>
```

There are a few more special cases for [] ranges, see /[] for the whole

story.

COMPLEMENTED RANGE

To avoid matching a specific character, use "^" at the start of the range. The [] item then matches everything but the characters included. Example:

```
/"[^"]*"
```

"	a double quote
[^"]	any character that is not a double quote
*	as many as possible
"	a double quote again

This matches "foo" and "3!x", including the double quotes.

PREDEFINED RANGES

A number of ranges are used very often. Vim provides a shortcut for these. For example:

```
/\a
```

Finds alphabetic characters. This is equal to using "[a-zA-Z]". Here are a few more of these:

item	matches	equivalent
\d	digit	[0-9]
\D	non-digit	[^0-9]
\x	hex digit	[0-9a-fA-F]
\X	non-hex digit	[^0-9a-fA-F]
\s	white space	[] (<Tab> and <Space>)
\S	non-white characters	[^] (not <Tab> and <Space>)
\l	lowercase alpha	[a-z]
\L	non-lowercase alpha	[^a-z]
\u	uppercase alpha	[A-Z]
\U	non-uppercase alpha	[^A-Z]

Note:

Using these predefined ranges works a lot faster than the character range it stands for.

These items can not be used inside []. Thus "[\d\l]" does NOT work to match a digit or lowercase alpha. Use "\(\d\|\l\)" instead.

See [/\s](#) for the whole list of these ranges.

27.7 Character classes

The character range matches a fixed set of characters. A character class is similar, but with an essential difference: The set of characters can be redefined without changing the search pattern.

For example, search for this pattern:

```
/\f\+
```

The "\f" item stands for file name characters. Thus this matches a sequence of characters that can be a file name.

Which characters can be part of a file name depends on the system you are using. On MS-Windows, the backslash is included, on Unix it is not. This is specified with the 'isfname' option. The default value for Unix is:

```
:set isfname
isfname=@,48-57,/,.,-,_,+,,#,$,%~,=
```

For other systems the default value is different. Thus you can make a search pattern with "\f" to match a file name, and it will automatically adjust to the system you are using it on.

Note:

Actually, Unix allows using just about any character in a file name, including white space. Including these characters in 'isfname' would be theoretically correct. But it would make it impossible to find the end of a file name in text. Thus the default value of 'isfname' is a compromise.

The character classes are:

item	matches	option
\i	identifier characters	'isident'
\I	like \i, excluding digits	
\k	keyword characters	'iskeyword'
\K	like \k, excluding digits	
\p	printable characters	'isprint'
\P	like \p, excluding digits	
\f	file name characters	'isfname'
\F	like \f, excluding digits	

27.8 Matching a line break

Vim can find a pattern that includes a line break. You need to specify where the line break happens, because all items mentioned so far don't match a line break.

To check for a line break in a specific place, use the "\n" item:

```
/one\ntwo
```

This will match at a line that ends in "one" and the next line starts with "two". To match "one two" as well, you need to match a space or a line break. The item to use for it is "_s":

```
/one\_stwo
```

To allow any amount of white space:

```
/one\s\+two
```

This also matches when "one " is at the end of a line and " two" at the start of the next one.

"\s" matches white space, "_s" matches white space or a line break. Similarly, "\a" matches an alphabetic character, and "_a" matches an alphabetic character or a line break. The other character classes and ranges can be modified in the same way by inserting a "_".

Many other items can be made to match a line break by prepending "_". For example: "_." matches any character or a line break.

Note:

"_.*" matches everything until the end of the file. Be careful with this, it can make a search command very slow.

Another example is "_[]", a character range that includes a line break:

```
/"\_[^"]*"
```

This finds a text in double quotes that may be split up in several lines.

27.9 Examples

Here are a few search patterns you might find useful. This shows how the items mentioned above can be combined.

FINDING A CALIFORNIA LICENSE PLATE

A sample license plate number is "1MGU103". It has one digit, three uppercase letters and three digits. Directly putting this into a search pattern:

```
/\d\u\u\u\d\d\d
```

Another way is to specify that there are three digits and letters with a count:

```
/\d\u\{3}\d\{3}
```

Using [] ranges instead:

```
/[0-9][A-Z]\{3}[0-9]\{3}
```

Which one of these you should use? Whichever one you can remember. The simple way you can remember is much faster than the fancy way that you can't. If you can remember them all, then avoid the last one, because it's both more typing and slower to execute.

FINDING AN IDENTIFIER

In C programs (and many other computer languages) an identifier starts with a letter and further consists of letters and digits. Underscores can be used too. This can be found with:

```
/\<\h\w*\>
```

"\<" and "\>" are used to find only whole words. "\h" stands for "[A-Za-z_]" and "\w" for "[0-9A-Za-z_]".

Note:

"\<" and "\>" depend on the **'iskeyword'** option. If it includes "-", for example, then "ident-" is not matched. In this situation use:

```
/\w\@<!\h\w*\w\@!
```

This checks if "\w" does not match before or after the identifier. See [/\@<!](#) and [/\@!](#) .

Next chapter: [usr_28.txt](#) Folding

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Folding

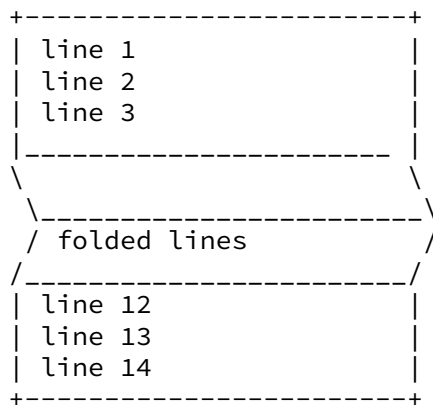
Structured text can be separated in sections. And sections in sub-sections. Folding allows you to display a section as one line, providing an overview. This chapter explains the different ways this can be done.

- 28.1 What is folding?
- 28.2 Manual folding
- 28.3 Working with folds
- 28.4 Saving and restoring folds
- 28.5 Folding by indent
- 28.6 Folding with markers
- 28.7 Folding by syntax
- 28.8 Folding by expression
- 28.9 Folding unchanged lines
- 28.10 Which fold method to use?

Next chapter: [usr_29.txt](#) Moving through programs
Previous chapter: [usr_27.txt](#) Search commands and patterns
Table of contents: [usr_toc.txt](#)

28.1 What is folding?

Folding is used to show a range of lines in the buffer as a single line on the screen. Like a piece of paper which is folded to make it shorter:



The text is still in the buffer, unchanged. Only the way lines are displayed is affected by folding.

The advantage of folding is that you can get a better overview of the structure of text, by folding lines of a section and replacing it with a line that indicates that there is a section.

28.2 Manual folding

Try it out: Position the cursor in a paragraph and type:

`zfap`

You will see that the paragraph is replaced by a highlighted line. You have created a fold. `zf` is an operator and `ap` a text object selection. You can use the `zf` operator with any movement command to create a fold for the text that it moved over. `zf` also works in Visual mode.

To view the text again, open the fold by typing:

`zo`

And you can close the fold again with:

`zc`

All the folding commands start with "z". With some fantasy, this looks like a folded piece of paper, seen from the side. The letter after the "z" has a mnemonic meaning to make it easier to remember the commands:

<code>zf</code>	F-old creation
<code>zo</code>	O-pen a fold
<code>zc</code>	C-lose a fold

Folds can be nested: A region of text that contains folds can be folded again. For example, you can fold each paragraph in this section, and then fold all the sections in this chapter. Try it out. You will notice that opening the fold for the whole chapter will restore the nested folds as they were, some may be open and some may be closed.

Suppose you have created several folds, and now want to view all the text. You could go to each fold and type "zo". To do this faster, use this command:

`zr`

This will R-educer the folding. The opposite is:

`zm`

This folds M-ore. You can repeat "zr" and "zm" to open and close nested folds of several levels.

If you have nested several levels deep, you can open all of them with:

`zR`

This R-educer folds until there are none left. And you can close all folds with:

`zM`

This folds M-ore and M-ore.

You can quickly disable the folding with the `zn` command. Then `zN` brings back the folding as it was. `zi` toggles between the two. This is a useful way of working:

- create folds to get overview on your file
- move around to where you want to do your work
- do `zi` to look at the text and edit it
- do `zi` again to go back to moving around

More about manual folding in the reference manual: [fold-manual](#)

28.3 Working with folds

When some folds are closed, movement commands like "j" and "k" move over a fold like it was a single, empty line. This allows you to quickly move around over folded text.

You can yank, delete and put folds as if it was a single line. This is very useful if you want to reorder functions in a program. First make sure that each fold contains a whole function (or a bit less) by selecting the right **'foldmethod'**. Then delete the function with "dd", move the cursor and put it with "p". If some lines of the function are above or below the fold, you can use Visual selection:

- put the cursor on the first line to be moved
- hit "V" to start Visual mode
- put the cursor on the last line to be moved
- hit "d" to delete the selected lines.
- move the cursor to the new position and "p"ut the lines there.

It is sometimes difficult to see or remember where a fold is located, thus where a `zo` command would actually work. To see the defined folds:

```
:set foldcolumn=4
```

This will show a small column on the left of the window to indicate folds. A "+" is shown for a closed fold. A "-" is shown at the start of each open fold and "|" at following lines of the fold.

You can use the mouse to open a fold by clicking on the "+" in the foldcolumn. Clicking on the "-" or a "|" below it will close an open fold.

- To open all folds at the cursor line use `zO` .
- To close all folds at the cursor line use `zC` .
- To delete a fold at the cursor line use `zd` .
- To delete all folds at the cursor line use `zD` .

When in Insert mode, the fold at the cursor line is never closed. That allows you to see what you type!

Folds are opened automatically when jumping around or moving the cursor left or right. For example, the "0" command opens the fold under the cursor (if **'foldopen'** contains "hor", which is the default). The **'foldopen'** option

can be changed to open folds for specific commands. If you want the line under the cursor always to be open, do this:

```
:set foldopen=all
```

Warning: You won't be able to move onto a closed fold then. You might want to use this only temporarily and then set it back to the default:

```
:set foldopen&
```

You can make folds close automatically when you move out of it:

```
:set foldclose=all
```

This will re-apply **'foldlevel'** to all folds that don't contain the cursor. You have to try it out if you like how this feels. Use **zm** to fold more and **zr** to fold less (reduce folds).

The folding is local to the window. This allows you to open two windows on the same buffer, one with folds and one without folds. Or one with all folds closed and one with all folds open.

28.4 Saving and restoring folds

When you abandon a file (starting to edit another one), the state of the folds is lost. If you come back to the same file later, all manually opened and closed folds are back to their default. When folds have been created manually, all folds are gone! To save the folds use the **:mkview** command:

```
:mkview
```

This will store the settings and other things that influence the view on the file. You can change what is stored with the **'viewoptions'** option. When you come back to the same file later, you can load the view again:

```
:loadview
```

You can store up to ten views on one file. For example, to save the current setup as the third view and load the second view:

```
:mkview 3  
:loadview 2
```

Note that when you insert or delete lines the views might become invalid. Also check out the **'viewdir'** option, which specifies where the views are stored. You might want to delete old views now and then.

28.5 Folding by indent

Defining folds with **zf** is a lot of work. If your text is structured by giving lower level items a larger indent, you can use the indent folding method. This will create folds for every sequence of lines with the same

indent. Lines with a larger indent will become nested folds. This works well with many programming languages.

Try this by setting the `'foldmethod'` option:

```
:set foldmethod=indent
```

Then you can use the `zm` and `zr` commands to fold more and reduce folding. It's easy to see on this example text:

```
This line is not indented
  This line is indented once
    This line is indented twice
      This line is indented twice
  This line is indented once
This line is not indented
  This line is indented once
  This line is indented once
```

Note that the relation between the amount of indent and the fold depth depends on the `'shiftwidth'` option. Each `'shiftwidth'` worth of indent adds one to the depth of the fold. This is called a fold level.

When you use the `zr` and `zm` commands you actually increase or decrease the `'foldlevel'` option. You could also set it directly:

```
:set foldlevel=3
```

This means that all folds with three times a `'shiftwidth'` indent or more will be closed. The lower the `foldlevel`, the more folds will be closed. When `'foldlevel'` is zero, all folds are closed. `zM` does set `'foldlevel'` to zero. The opposite command `zR` sets `'foldlevel'` to the deepest fold level that is present in the file.

Thus there are two ways to open and close the folds:

(A) By setting the fold level.

This gives a very quick way of "zooming out" to view the structure of the text, move the cursor, and "zoom in" on the text again.

(B) By using `zo` and `zc` commands to open or close specific folds.

This allows opening only those folds that you want to be open, while other folds remain closed.

This can be combined: You can first close most folds by using `zm` a few times and then open a specific fold with `zo`. Or open all folds with `zR` and then close specific folds with `zc`.

But you cannot manually define folds when `'foldmethod'` is "indent", as that would conflict with the relation between the indent and the fold level.

More about folding by indent in the reference manual: [fold-indent](#)

=====

28.6 Folding with markers

Markers in the text are used to specify the start and end of a fold region. This gives precise control over which lines are included in a fold. The disadvantage is that the text needs to be modified.

Try it:

```
:set foldmethod=marker
```

Example text, as it could appear in a C program:

```
/* foobar () {{{ */
int foobar()
{
    /* return a value {{{ */
    return 42;
    /* }}} */
}
/* }}} */
```

Notice that the folded line will display the text before the marker. This is very useful to tell what the fold contains.

It's quite annoying when the markers don't pair up correctly after moving some lines around. This can be avoided by using numbered markers. Example:

```
/* global variables {{{1 */
int varA, varB;

/* functions {{{1 */
/* funcA() {{{2 */
void funcA() {}

/* funcB() {{{2 */
void funcB() {}
/* }}}1 */
```

At every numbered marker a fold at the specified level begins. This will make any fold at a higher level stop here. You can just use numbered start markers to define all folds. Only when you want to explicitly stop a fold before another starts you need to add an end marker.

More about folding with markers in the reference manual: [fold-marker](#)

28.7 Folding by syntax

For each language Vim uses a different syntax file. This defines the colors for various items in the file. If you are reading this in Vim, in a terminal that supports colors, the colors you see are made with the "help" syntax file.

In the syntax files it is possible to add syntax items that have the "fold" argument. These define a fold region. This requires writing a syntax file and adding these items in it. That's not so easy to do. But once it's done, all folding happens automatically.

Here we'll assume you are using an existing syntax file. Then there is nothing more to explain. You can open and close folds as explained above. The folds will be created and deleted automatically when you edit the file.

More about folding by syntax in the reference manual: [fold-syntax](#)

28.8 Folding by expression

This is similar to folding by indent, but instead of using the indent of a line a user function is called to compute the fold level of a line. You can use this for text where something in the text indicates which lines belong together. An example is an e-mail message where the quoted text is indicated by a ">" before the line. To fold these quotes use this:

```
:set foldmethod=expr
:set foldexpr=strlen(substitute(substitute(getline(v:lnum),'\s','','g'),'[^>].*
```

You can try it out on this text:

```
> quoted text he wrote
> quoted text he wrote
> > double quoted text I wrote
> > double quoted text I wrote
```

Explanation for the '**foldexpr**' used in the example (inside out):

getline(v:lnum)	gets the current line
substitute(...,'\s','','g')	removes all white space from the line
substitute(...,'[^>].*','','')	removes everything after leading '>'s
strlen(...)	counts the length of the string, which is the number of '>'s found

Note that a backslash must be inserted before every space, double quote and backslash for the ":set" command. If this confuses you, do

```
:set foldexpr
```

to check the actual resulting value. To correct a complicated expression, use the command-line completion:

```
:set foldexpr=<Tab>
```

Where <Tab> is a real Tab. Vim will fill in the previous value, which you can then edit.

When the expression gets more complicated you should put it in a function and set '**foldexpr**' to call that function.

More about folding by expression in the reference manual: [fold-expr](#)

28.9 Folding unchanged lines

This is useful when you set the '**diff**' option in the same window. The

`vimdiff` command does this for you. Example:

```
:setlocal diff foldmethod=diff scrollbind nowrap foldlevel=1
```

Do this in every window that shows a different version of the same file. You will clearly see the differences between the files, while the text that didn't change is folded.

For more details see [fold-diff](#) .

28.10 Which fold method to use?

All these possibilities make you wonder which method you should choose. Unfortunately, there is no golden rule. Here are some hints.

If there is a syntax file with folding for the language you are editing, that is probably the best choice. If there isn't one, you might try to write it. This requires a good knowledge of search patterns. It's not easy, but when it's working you will not have to define folds manually.

Typing commands to manually fold regions can be used for unstructured text. Then use the `:mkview` command to save and restore your folds.

The marker method requires you to change the file. If you are sharing the files with other people or you have to meet company standards, you might not be allowed to add them.

The main advantage of markers is that you can put them exactly where you want them. That avoids that a few lines are missed when you cut and paste folds. And you can add a comment about what is contained in the fold.

Folding by indent is something that works in many files, but not always very well. Use it when you can't use one of the other methods. However, it is very useful for outlining. Then you specifically use one `'shiftwidth'` for each nesting level.

Folding with expressions can make folds in almost any structured text. It is quite simple to specify, especially if the start and end of a fold can easily be recognized.

If you use the "expr" method to define folds, but they are not exactly how you want them, you could switch to the "manual" method. This will not remove the defined folds. Then you can delete or add folds manually.

Next chapter: [usr_29.txt](#) Moving through programs

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_29.txt For Vim version 9.1. Last change: 2022 Mar 13

VIM USER MANUAL - by Bram Moolenaar

Moving through programs

The creator of Vim is a computer programmer. It's no surprise that Vim contains many features to aid in writing programs. Jump around to find where identifiers are defined and used. Preview declarations in a separate window. There is more in the next chapter.

- 29.1 Using tags
- 29.2 The preview window
- 29.3 Moving through a program
- 29.4 Finding global identifiers
- 29.5 Finding local identifiers

Next chapter: [usr_30.txt](#) Editing programs
Previous chapter: [usr_28.txt](#) Folding
Table of contents: [usr_toc.txt](#)

29.1 Using tags

What is a tag? It is a location where an identifier is defined. An example is a function definition in a C or C++ program. A list of tags is kept in a tags file. This can be used by Vim to directly jump from any place to the tag, the place where an identifier is defined.

To generate the tags file for all C files in the current directory, use the following command:

```
ctags *.c
```

"ctags" is a separate program. Most Unix systems already have it installed. If you do not have it yet, you can find Universal/Exuberant ctags at:

<http://ctags.io>
<http://ctags.sf.net>

Universal ctags is preferred, Exuberant ctags is no longer being developed.

Now when you are in Vim and you want to go to a function definition, you can jump to it by using the following command:

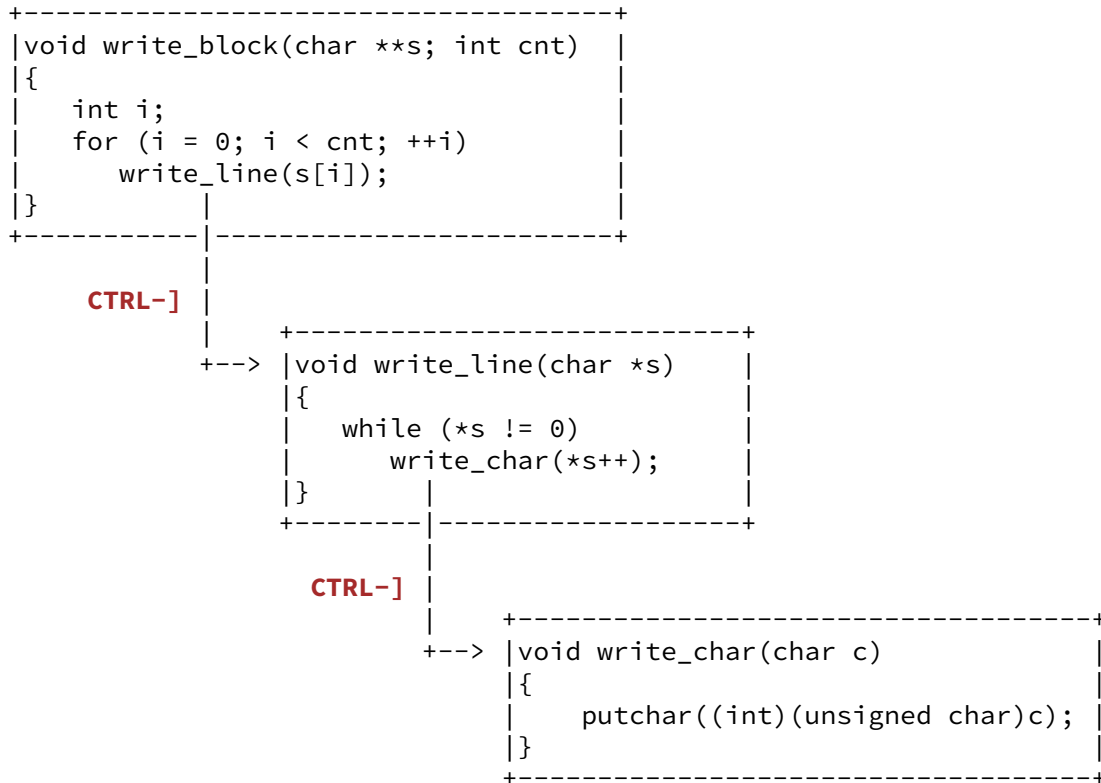
```
:tag startlist
```

This command will find the function "startlist" even if it is in another file.

The **CTRL-]** command jumps to the tag of the word that is under the cursor. This makes it easy to explore a tangle of C code. Suppose, for example, that you are in the function "write_block". You can see that it calls "write_line". But what does "write_line" do? By placing the cursor on the call to "write_line" and pressing **CTRL-]**, you jump to the definition of this function.

The "write_line" function calls "write_char". You need to figure out what

it does. So you position the cursor over the call to "write_char" and press **CTRL-]**. Now you are at the definition of "write_char".



The ":tags" command shows the list of tags that you traversed through:

```

:tags
# TO tag      FROM line  in file/text
1 1 write_line 8 write_block.c
2 1 write_char 7 write_line.c
>
  
```

Now to go back. The **CTRL-T** command goes to the preceding tag. In the example above you get back to the "write_line" function, in the call to "write_char".

This command takes a count argument that indicates how many tags to jump back. You have gone forward, and now back. Let's go forward again. The following command goes to the tag on top of the list:

```
:tag
```

You can prefix it with a count and jump forward that many tags. For example: ":3tag". **CTRL-T** also can be preceded with a count.

These commands thus allow you to go down a call tree with **CTRL-]** and back up again with **CTRL-T**. Use ":tags" to find out where you are.

SPLIT WINDOWS

The ":tag" command replaces the file in the current window with the one

containing the new function. But suppose you want to see not only the old function but also the new one? You can split the window using the `":split"` command followed by the `":tag"` command. Vim has a shorthand command that does both:

```
:stag tagname
```

To split the current window and jump to the tag under the cursor use this command:

```
CTRL-W ]
```

If a count is specified, the new window will be that many lines high.

MORE TAGS FILES

When you have files in many directories, you can create a tags file in each of them. Vim will then only be able to jump to tags within that directory.

To find more tags files, set the `'tags'` option to include all the relevant tags files. Example:

```
:set tags=./tags,../../tags,*/tags
```

This finds a tags file in the same directory as the current file, one directory level higher and in all subdirectories.

This is quite a number of tags files, but it may still not be enough. For example, when editing a file in `"~/proj/src"`, you will not find the tags file `"~/proj/sub/tags"`. For this situation Vim offers to search a whole directory tree for tags files. Example:

```
:set tags=~/proj/**/tags
```

ONE TAGS FILE

When Vim has to search many places for tags files, you can hear the disk rattling. It may get a bit slow. In that case it's better to spend this time while generating one big tags file. You might do this overnight.

This requires the Universal or Exuberant ctags program, mentioned above. It offers an argument to search a whole directory tree:

```
cd ~/proj
ctags -R .
```

The nice thing about this is that Universal/Exuberant ctags recognizes various file types. Thus this doesn't work just for C and C++ programs, also for Eiffel and even Vim scripts. See the ctags documentation to tune this.

Now you only need to tell Vim where your big tags file is:

```
:set tags=~/proj/tags
```

MULTIPLE MATCHES

When a function is defined multiple times (or a method in several classes), the ":tag" command will jump to the first one. If there is a match in the current file, that one is used first.

You can now jump to other matches for the same tag with:

```
:tnext
```

Repeat this to find further matches. If there are many, you can select which one to jump to:

```
:tselect tagname
```

Vim will present you with a list of choices:

```
# pri kind tag          file
1 F  f    mch_init      os_amiga.c
      mch_init()
2 F  f    mch_init      os_mac.c
      mch_init()
3 F  f    mch_init      os_msdos.c
      mch_init(void)
4 F  f    mch_init      os_riscos.c
      mch_init()
Enter nr of choice (<CR> to abort):
```

You can now enter the number (in the first column) of the match that you would like to jump to. The information in the other columns give you a good idea of where the match is defined.

To move between the matching tags, these commands can be used:

```
:tfirst          go to first match
:[count]tprevious go to [count] previous match
:[count]tnext    go to [count] next match
:tlast          go to last match
```

If [count] is omitted then one is used.

GUESSING TAG NAMES

Command line completion is a good way to avoid typing a long tag name. Just type the first bit and press <Tab>:

```
:tag write_<Tab>
```

You will get the first match. If it's not the one you want, press <Tab> until you find the right one.

Sometimes you only know part of the name of a function. Or you have many tags that start with the same string, but end differently. Then you can tell Vim to use a pattern to find the tag.

Suppose you want to jump to a tag that contains "block". First type this:


```
:tag /block
```

Now use command line completion: press `<Tab>`. Vim will find all tags that contain "block" and use the first match.

The "/" before a tag name tells Vim that what follows is not a literal tag name, but a pattern. You can use all the items for search patterns here. For example, suppose you want to select a tag that starts with "write_":

```
:tselect /^write_
```

The "^" specifies that the tag starts with "write_". Otherwise it would also be found halfway a tag name. Similarly "\$" at the end makes sure the pattern matches until the end of a tag.

A TAGS BROWSER

Since `CTRL-]` takes you to the definition of the identifier under the cursor, you can use a list of identifier names as a table of contents. Here is an example.

First create a list of identifiers (this requires Universal or Exuberant ctags):

```
ctags --c-types=f -f functions *.c
```

Now start Vim without a file, and edit this file in Vim, in a vertically split window:

```
vim  
:vsplit functions
```

The window contains a list of all the functions. There is some more stuff, but you can ignore that. Do `":setlocal ts=99"` to clean it up a bit.

In this window, define a mapping:

```
:nnoremap <buffer> <CR> @ye<C-W>w:tag <C-R>"<CR>
```

Move the cursor to the line that contains the function you want to go to. Now press `<Enter>`. Vim will go to the other window and jump to the selected function.

RELATED ITEMS

To make case in tag names be ignored, you can set `'ignorecase'` while leaving `'tagcase'` as "followic", or set `'tagcase'` to "ignore".

The `'tagbsearch'` option tells if the tags file is sorted or not. The default is to assume a sorted tags file, which makes a tags search a lot faster, but doesn't work if the tags file isn't sorted.

The `'taglength'` option can be used to tell Vim the number of significant characters in a tag.

Cscope is a free program. It does not only find places where an identifier is declared, but also where it is used. See [cscope](#) .

29.2 The preview window

When you edit code that contains a function call, you need to use the correct arguments. To know what values to pass you can look at how the function is defined. The tags mechanism works very well for this. Preferably the definition is displayed in another window. For this the preview window can be used.

To open a preview window to display the function "write_char":

```
:ptag write_char
```

Vim will open a window, and jumps to the tag "write_char". Then it takes you back to the original position. Thus you can continue typing without the need to use a **CTRL-W** command.

If the name of a function appears in the text, you can get its definition in the preview window with:

```
CTRL-W }
```

There is a script that automatically displays the text where the word under the cursor was defined. See [CursorHold-example](#) .

To close the preview window use this command:

```
:pclose
```

To edit a specific file in the preview window, use ":pedit". This can be useful to edit a header file, for example:

```
:pedit defs.h
```

Finally, ":psearch" can be used to find a word in the current file and any included files and display the match in the preview window. This is especially useful when using library functions, for which you do not have a tags file. Example:

```
:psearch popen
```

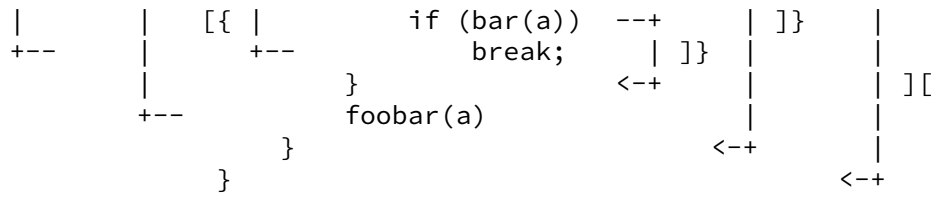
This will show the "stdio.h" file in the preview window, with the function prototype for popen():

```
FILE *popen __P((const char *, const char *));
```

You can specify the height of the preview window, when it is opened, with the **'previewheight'** option.

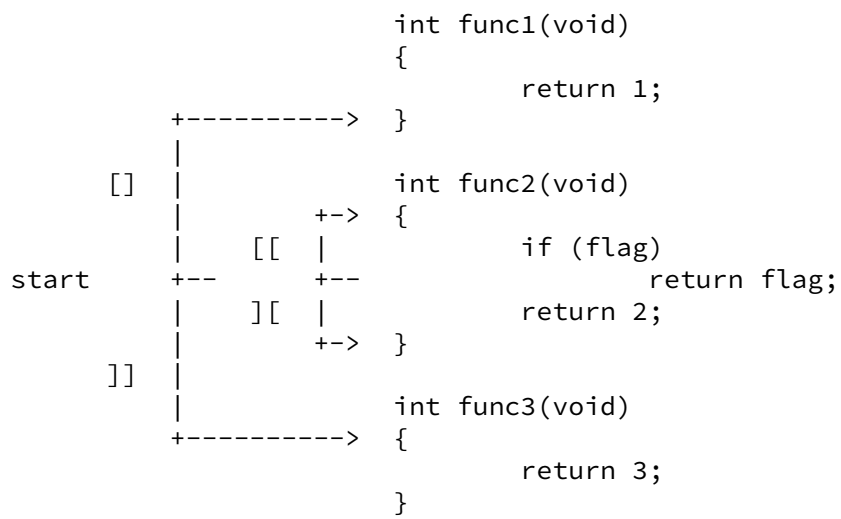
29.3 Moving through a program

Since a program is structured, Vim can recognize items in it. Specific



When writing C++ or Java, the outer `}` block is for the class. The next level of `}` is for a method. When somewhere inside a class use `"[m"` to find the previous start of a method. `"]m"` finds the next start of a method.

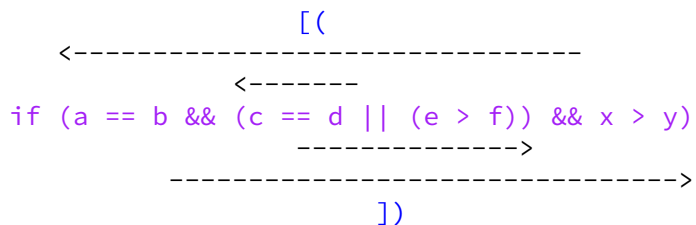
Additionally, `"[]"` moves backward to the end of a function and `"]]"` moves forward to the start of the next function. The end of a function is defined by a `"}"` in the first column.



Don't forget you can also use `"%"` to move between matching `()`, `{ }` and `[]`. That also works when they are many lines apart.

MOVING IN BRACES

The `"[(` and `"])"` commands work similar to `"[{"` and `"}]"`, except that they work on `()` pairs instead of `{ }` pairs.



MOVING IN COMMENTS

To move back to the start of a comment use `"[/`". Move forward to the end of a comment with `"]/"`. This only works for `/* - */` comments.

```

+-->      +--> /*
|          [ / | * A comment about      --+
[ / |      +-- * wonderful life.        | ]/
|          * /                               <--+
|
+--      foo = bar * 3;                    --+
|                                               | ]/
|                                               <--+
|                                               /* a short comment */

```

29.4 Finding global identifiers

You are editing a C program and wonder if a variable is declared as "int" or "unsigned". A quick way to find this is with the "[I" command.

Suppose the cursor is on the word "column". Type:

```
[I
```

Vim will list the matching lines it can find. Not only in the current file, but also in all included files (and files included in them, etc.). The result looks like this:

```

structs.h
1: 29 unsigned column; /* column number */

```

The advantage over using tags or the preview window is that included files are searched. In most cases this results in the right declaration to be found. Also when the tags file is out of date. Also when you don't have tags for the included files.

However, a few things must be right for "[I" to do its work. First of all, the '**include**' option must specify how a file is included. The default value works for C and C++. For other languages you will have to change it.

LOCATING INCLUDED FILES

Vim will find included files in the places specified with the '**path**' option. If a directory is missing, some include files will not be found. You can discover this with this command:

```
:checkpath
```

It will list the include files that could not be found. Also files included by the files that could be found. An example of the output:

```

--- Included files not found in path ---
<io.h>
vim.h -->
  <functions.h>
  <clib/exec_protos.h>

```

The "io.h" file is included by the current file and can't be found. "vim.h" can be found, thus ":checkpath" goes into this file and checks what it includes. The "functions.h" and "clib/exec_protos.h" files, included by

"vim.h" are not found.

Note:

Vim is not a compiler. It does not recognize "#ifdef" statements. This means every "#include" statement is used, also when it comes after "#if NEVER".

To fix the files that could not be found, add a directory to the **'path'** option. A good place to find out about this is the Makefile. Look out for lines that contain "-I" items, like "-I/usr/local/X11". To add this directory use:

```
:set path+=/usr/local/X11
```

When there are many subdirectories, you can use the "*" wildcard. Example:

```
:set path+=/usr/*/include
```

This would find files in "/usr/local/include" as well as "/usr/X11/include".

When working on a project with a whole nested tree of included files, the "**" items is useful. This will search down in all subdirectories. Example:

```
:set path+=/projects/invent/**/include
```

This will find files in the directories:

```
/projects/invent/include  
/projects/invent/main/include  
/projects/invent/main/os/include  
etc.
```

There are even more possibilities. Check out the **'path'** option for info.

If you want to see which included files are actually found, use this command:

```
:checkpath!
```

You will get a (very long) list of included files, the files they include, and so on. To shorten the list a bit, Vim shows "(Already listed)" for files that were found before and doesn't list the included files in there again.

JUMPING TO A MATCH

"[I" produces a list with only one line of text. When you want to have a closer look at the first item, you can jump to that line with the command:

```
[<Tab>
```

You can also use "[**CTRL-I**", since **CTRL-I** is the same as pressing <Tab>.

The list that "[I" produces has a number at the start of each line. When you want to jump to another item than the first one, type the number first:

3[<Tab>

Will jump to the third item in the list. Remember that you can use **CTRL-O** to jump back to where you started from.

RELATED COMMANDS

[i	only lists the first match
]I	only lists items below the cursor
]i	only lists the first item below the cursor

FINDING DEFINED IDENTIFIERS

The "[I" command finds any identifier. To find only macros, defined with "#define" use:

[D

Again, this searches in included files. The **'define'** option specifies what a line looks like that defines the items for "[D". You could change it to make it work with other languages than C or C++.

The commands related to "[D" are:

[d	only lists the first match
]D	only lists items below the cursor
]d	only lists the first item below the cursor

29.5 Finding local identifiers

The "[I" command searches included files. To search in the current file only, and jump to the first place where the word under the cursor is used:

gD

Hint: Goto Definition. This command is very useful to find a variable or function that was declared locally ("static", in C terms). Example (cursor on "counter"):

```
    +--> static int counter = 0;
        |
        | int get_counter(void)
gD    | {
        |     ++counter;
    +--    return counter;
        |
        }
```

To restrict the search even further, and look only in the current function, use this command:

gd

This will go back to the start of the current function and find the first occurrence of the word under the cursor. Actually, it searches backwards to an empty line above a "{" in the first column. From there it searches forward for the identifier. Example (cursor on "idx"):

```
        int find_entry(char *name)
        {
+->         int idx;
gd |         for (idx = 0; idx < table_len; ++idx)
|           if (strcmp(table[idx].name, name) == 0)
+--           return idx;
        }
```

Next chapter: [usr_30.txt](#) Editing programs

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_30.txt For Vim version 9.1. Last change: 2007 Nov 10

VIM USER MANUAL - by Bram Moolenaar

Editing programs

Vim has various commands that aid in writing computer programs. Compile a program and directly jump to reported errors. Automatically set the indent for many languages and format comments.

- 30.1 Compiling
- 30.2 Indenting C files
- 30.3 Automatic indenting
- 30.4 Other indenting
- 30.5 Tabs and spaces
- 30.6 Formatting comments

Next chapter: [usr_31.txt](#) Exploiting the GUI
Previous chapter: [usr_29.txt](#) Moving through programs
Table of contents: [usr_toc.txt](#)

30.1 Compiling

Vim has a set of so called "quickfix" commands. They enable you to compile a program from within Vim and then go through the errors generated and fix them (hopefully). You can then recompile and fix any new errors that are found until finally your program compiles without any error.

The following command runs the program "make" (supplying it with any argument you give) and captures the results:

```
:make {arguments}
```

If errors were generated, they are captured and the editor positions you where the first error occurred.

Take a look at an example ":make" session. (Typical :make sessions generate far more errors and fewer stupid ones.) After typing ":make" the screen looks like this:

```
:!make | &tee /tmp/vim215953.err
gcc -g -Wall -o prog main.c sub.c
main.c: In function 'main':
main.c:6: too many arguments to function 'do_sub'
main.c: At top level:
main.c:10: parse error before '}'
make: *** [prog] Error 1

2 returned
"main.c" 11L, 111C
(3 of 6): too many arguments to function 'do_sub'
Press ENTER or type command to continue
```

From this you can see that you have errors in the file "main.c". When you press <Enter>, Vim displays the file "main.c", with the cursor positioned on line 6, the first line with an error. You did not need to specify the file or the line number, Vim knew where to go by looking in the error messages.

```
+-----+
|int main()
|{
|    int i=3;
cursor -> |    do_sub("foo");
|          ++i;
|          return (0);
|}
|}
|~
| (3 of 12): too many arguments to function 'do_sub'
+-----+
```

The following command goes to where the next error occurs:

```
:cnext
```

Vim jumps to line 10, the last line in the file, where there is an extra '}'.

When there is not enough room, Vim will shorten the error message. To see the whole message use:

```
:cc
```

You can get an overview of all the error messages with the ":clist" command. The output looks like this:

```
:clist
3 main.c: 6:too many arguments to function 'do_sub'
5 main.c: 10:parse error before '}'
```

Only the lines where Vim recognized a file name and line number are listed here. It assumes those are the interesting lines and the rest is just boring messages. However, sometimes unrecognized lines do contain something you want to see. Output from the linker, for example, about an undefined function. To see all the messages add a "!" to the command:

```
:clist!
1 gcc -g -Wall -o prog main.c sub.c
2 main.c: In function 'main':
3 main.c:6: too many arguments to function 'do_sub'
4 main.c: At top level:
5 main.c:10: parse error before '}'
6 make: *** [prog] Error 1
```

Vim will highlight the current error. To go back to the previous error, use:

```
:cprevious
```

Other commands to move around in the error list:

```
:cfirst      to first error
:clast       to last error
:cc 3        to error nr 3
```

USING ANOTHER COMPILER

The name of the program to run when the ":make" command is executed is defined by the '**makeprg**' option. Usually this is set to "make", but Visual C++ users should set this to "nmake" by executing the following command:

```
:set makeprg=nmake
```

You can also include arguments in this option. Special characters need to be escaped with a backslash. Example:

```
:set makeprg=nmake\ -f\ project.mak
```

You can include special Vim keywords in the command specification. The % character expands to the name of the current file. So if you execute the command:

```
:set makeprg=make\ %:S
```

When you are editing main.c, then ":make" executes the following command:

```
make main.c
```

This is not too useful, so you will refine the command a little and use the :r (root) modifier:

```
:set makeprg=make\ %:r:S.o
```

Now the command executed is as follows:

```
make main.o
```

More about these modifiers here: [filename-modifiers](#) .

OLD ERROR LISTS

Suppose you ":make" a program. There is a warning message in one file and an error message in another. You fix the error and use ":make" again to check if it was really fixed. Now you want to look at the warning message. It doesn't show up in the last error list, since the file with the warning wasn't compiled again. You can go back to the previous error list with:

```
:colder
```

Then use ":clist" and ":cc {nr}" to jump to the place with the warning. To go forward to the next error list:

```
:cnewer
```

Vim remembers ten error lists.

SWITCHING COMPILERS

You have to tell Vim what format the error messages are that your compiler produces. This is done with the **'errorformat'** option. The syntax of this option is quite complicated and it can be made to fit almost any compiler. You can find the explanation here: [errorformat](#) .

You might be using various different compilers. Setting the **'makeprg'** option, and especially the **'errorformat'** each time is not easy. Vim offers a simple method for this. For example, to switch to using the Microsoft Visual C++ compiler:

```
:compiler msvc
```

This will find the Vim script for the "msvc" compiler and set the appropriate options.

You can write your own compiler files. See [write-compiler-plugin](#) .

OUTPUT REDIRECTION

The ":make" command redirects the output of the executed program to an error file. How this works depends on various things, such as the **'shell'**. If your ":make" command doesn't capture the output, check the **'makeef'** and **'shellpipe'** options. The **'shellquote'** and **'shellxquote'** options might also matter.

In case you can't get ":make" to redirect the file for you, an alternative is to compile the program in another window and redirect the output into a file. Then have Vim read this file with:

```
:cfile {filename}
```

Jumping to errors will work like with the ":make" command.

30.2 Indenting C style text

A program is much easier to understand when the lines have been properly indented. Vim offers various ways to make this less work. For C or C style programs like Java or C++, set the **'cindent'** option. Vim knows a lot about C programs and will try very hard to automatically set the indent for you. Set the **'shiftwidth'** option to the amount of spaces you want for a deeper level. Four spaces will work fine. One ":set" command will do it:

```
:set cindent shiftwidth=4
```

With this option enabled, when you type something such as "if (x)", the next line will automatically be indented an additional level.

```

Automatic indent   --->   if (flag)
Automatic unindent <--   do_the_work();
Automatic indent   --->   if (other_flag) {
keep indent        --->   do_file();
Automatic unindent <--   do_some_more();
Automatic unindent <--   }

```

When you type something in curly braces (`{}`), the text will be indented at the start and unindented at the end. The unindenting will happen after typing the `}`, since Vim can't guess what you are going to type.

One side effect of automatic indentation is that it helps you catch errors in your code early. When you type a `}` to finish a function, only to find that the automatic indentation gives it more indent than what you expected, there is probably a `}` missing. Use the `%` command to find out which `{` matches the `}` you typed.

A missing `)` and `;` also cause extra indent. Thus if you get more white space than you would expect, check the preceding lines.

When you have code that is badly formatted, or you inserted and deleted lines, you need to re-indent the lines. The `=` operator does this. The simplest form is:

```
==
```

This indents the current line. Like with all operators, there are three ways to use it. In Visual mode `=` indents the selected lines. A useful text object is `a{`. This selects the current `{}` block. Thus, to re-indent the code block the cursor is in:

```
=a{
```

If you have really badly indented code, you can re-indent the whole file with:

```
gg=G
```

However, don't do this in files that have been carefully indented manually. The automatic indenting does a good job, but in some situations you might want to overrule it.

SETTING INDENT STYLE

Different people have different styles of indentation. By default Vim does a pretty good job of indenting in a way that 90% of programmers do. There are different styles, however; so if you want to, you can customize the indentation style with the `'cinoptions'` option.

By default `'cinoptions'` is empty and Vim uses the default style. You can add various items where you want something different. For example, to make curly braces be placed like this:

```

if (flag)
{
    i = 8;

```

```
    j = 0;
}
```

Use this command:

```
:set cinoptions+={2
```

There are many of these items. See [cinoptions-values](#) .

30.3 Automatic indenting

You don't want to switch on the **'cindent'** option manually every time you edit a C file. This is how you make it work automatically:

```
:filetype indent on
```

Actually, this does a lot more than switching on **'cindent'** for C files. First of all, it enables detecting the type of a file. That's the same as what is used for syntax highlighting.

When the filetype is known, Vim will search for an indent file for this type of file. The Vim distribution includes a number of these for various programming languages. This indent file will then prepare for automatic indenting specifically for this file.

If you don't like the automatic indenting, you can switch it off again:

```
:filetype indent off
```

If you don't like the indenting for one specific type of file, this is how you avoid it. Create a file with just this one line:

```
:let b:did_indent = 1
```

Now you need to write this in a file with a specific name:

```
{directory}/indent/{filetype}.vim
```

The `{filetype}` is the name of the file type, such as "cpp" or "java". You can see the exact name that Vim detected with this command:

```
:set filetype
```

In this file the output is:

```
filetype=help
```

Thus you would use "help" for `{filetype}`.

For the `{directory}` part you need to use your runtime directory. Look at the output of this command:

```
set runtimepath
```

Now use the first item, the name before the first comma. Thus if the output

looks like this:

```
runtimepath=~/.vim,/usr/local/share/vim/vim60/runtime,~/.vim/after
```

You use "~/.vim" for {directory}. Then the resulting file name is:

```
~/.vim/indent/help.vim
```

Instead of switching the indenting off, you could write your own indent file. How to do that is explained here: [indent-expression](#) .

30.4 Other indenting

The simplest form of automatic indenting is with the **'autoindent'** option. It uses the indent from the previous line. A bit smarter is the **'smartindent'** option. This is useful for languages where no indent file is available. **'smartindent'** is not as smart as **'cindent'**, but smarter than **'autoindent'**.

With **'smartindent'** set, an extra level of indentation is added for each { and removed for each }. An extra level of indentation will also be added for any of the words in the **'cinwords'** option. Lines that begin with # are treated specially: all indentation is removed. This is done so that preprocessor directives will all start in column 1. The indentation is restored for the next line.

CORRECTING INDENTS

When you are using **'autoindent'** or **'smartindent'** to get the indent of the previous line, there will be many times when you need to add or remove one **'shiftwidth'** worth of indent. A quick way to do this is using the **CTRL-D** and **CTRL-T** commands in Insert mode.

For example, you are typing a shell script that is supposed to look like this:

```
if test -n a; then
    echo a
    echo "-----"
fi
```

Start off by setting these options:

```
:set autoindent shiftwidth=3
```

You start by typing the first line, <Enter> and the start of the second line:

```
if test -n a; then
echo
```

Now you see that you need an extra indent. Type **CTRL-T**. The result:

```
if test -n a; then
    echo
```

The **CTRL-T** command, in Insert mode, adds one **'shiftwidth'** to the indent, no matter where in the line you are.

You continue typing the second line, **<Enter>** and the third line. This time the indent is OK. Then **<Enter>** and the last line. Now you have this:

```
if test -n a; then
    echo a
    echo "-----"
fi
```

To remove the superfluous indent in the last line press **CTRL-D**. This deletes one **'shiftwidth'** worth of indent, no matter where you are in the line.

When you are in Normal mode, you can use the ">>" and "<<" commands to shift lines. ">" and "<" are operators, thus you have the usual three ways to specify the lines you want to indent. A useful combination is:

```
>i{
```

This adds one indent to the current block of lines, inside **{}**. The **{ and }** lines themselves are left unmodified. ">a{" includes them. In this example the cursor is on "printf":

original text	after ">i{"	after ">a{"
<pre>if (flag) { printf("yes"); flag = 0; }</pre>	<pre>if (flag) { printf("yes"); flag = 0; }</pre>	<pre>if (flag) { printf("yes"); flag = 0; }</pre>

30.5 Tabs and spaces

'tabstop' is set to eight by default. Although you can change it, you quickly run into trouble later. Other programs won't know what tabstop value you used. They probably use the default value of eight, and your text suddenly looks very different. Also, most printers use a fixed tabstop value of eight. Thus it's best to keep **'tabstop'** alone. (If you edit a file which was written with a different tabstop setting, see [25.3](#) for how to fix that.)

For indenting lines in a program, using a multiple of eight spaces makes you quickly run into the right border of the window. Using a single space doesn't provide enough visual difference. Many people prefer to use four spaces, a good compromise.

Since a **<Tab>** is eight spaces and you want to use an indent of four spaces, you can't use a **<Tab>** character to make your indent. There are two ways to handle this:

1. Use a mix of **<Tab>** and space characters. Since a **<Tab>** takes the place of eight spaces, you have fewer characters in your file. Inserting a **<Tab>** is quicker than eight spaces. Backspacing works faster as well.
2. Use spaces only. This avoids the trouble with programs that use a different tabstop value.

Fortunately, Vim supports both methods quite well.

SPACES AND TABS

If you are using a combination of tabs and spaces, you just edit normally. The Vim defaults do a fine job of handling things.

You can make life a little easier by setting the **'softtabstop'** option. This option tells Vim to make the `<Tab>` key look and feel as if tabs were set at the value of **'softtabstop'**, but actually use a combination of tabs and spaces.

After you execute the following command, every time you press the `<Tab>` key the cursor moves to the next 4-column boundary:

```
:set softtabstop=4
```

When you start in the first column and press `<Tab>`, you get 4 spaces inserted in your text. The second time, Vim takes out the 4 spaces and puts in a `<Tab>` (thus taking you to column 8). Thus Vim uses as many `<Tab>`s as possible, and then fills up with spaces.

When backspacing it works the other way around. A `<BS>` will always delete the amount specified with **'softtabstop'**. Then `<Tab>`s are used as many as possible and spaces to fill the gap.

The following shows what happens pressing `<Tab>` a few times, and then using `<BS>`. A "." stands for a space and "----->" for a `<Tab>`.

type	result
<code><Tab></code>
<code><Tab><Tab></code>	----->
<code><Tab><Tab><Tab></code>	----->....
<code><Tab><Tab><Tab><BS></code>	----->
<code><Tab><Tab><Tab><BS><BS></code>

An alternative is to use the **'smarttab'** option. When it's set, Vim uses **'shiftwidth'** for a `<Tab>` typed in the indent of a line, and a real `<Tab>` when typed after the first non-blank character. However, `<BS>` doesn't work like with **'softtabstop'**.

JUST SPACES

If you want absolutely no tabs in your file, you can set the **'expandtab'** option:

```
:set expandtab
```

When this option is set, the `<Tab>` key inserts a series of spaces. Thus you get the same amount of white space as if a `<Tab>` character was inserted, but there isn't a real `<Tab>` character in your file.

The backspace key will delete each space by itself. Thus after typing one `<Tab>` you have to press the `<BS>` key up to eight times to undo it. If you are in the indent, pressing **CTRL-D** will be a lot quicker.

CHANGING TABS IN SPACES (AND BACK)

Setting `'expandtab'` does not affect any existing tabs. In other words, any tabs in the document remain tabs. If you want to convert tabs to spaces, use the `":retab"` command. Use these commands:

```
:set expandtab
:%retab
```

Now Vim will have changed all indents to use spaces instead of tabs. However, all tabs that come after a non-blank character are kept. If you want these to be converted as well, add a `!:`

```
:%retab!
```

This is a little bit dangerous, because it can also change tabs inside a string. To check if these exist, you could use this:

```
/"[^"\t]*\t["^"]*
```

It's recommended not to use hard tabs inside a string. Replace them with `"\t"` to avoid trouble.

The other way around works just as well:

```
:set noexpandtab
:%retab!
```

30.6 Formatting comments

One of the great things about Vim is that it understands comments. You can ask Vim to format a comment and it will do the right thing.

Suppose, for example, that you have the following comment:

```
/*
 * This is a test
 * of the text formatting.
 */
```

You then ask Vim to format it by positioning the cursor at the start of the comment and type:

```
gq]/
```

`"gq"` is the operator to format text. `"]/"` is the motion that takes you to the end of a comment. The result is:

```
/*
 * This is a test of the text formatting.
 */
```

Notice that Vim properly handled the beginning of each line.

An alternative is to select the text that is to be formatted in Visual mode

and type "gq".

To add a new line to the comment, position the cursor on the middle line and press "o". The result looks like this:

```
/*
 * This is a test of the text formatting.
 *
 */
```

Vim has automatically inserted a star and a space for you. Now you can type the comment text. When it gets longer than '**textwidth**', Vim will break the line. Again, the star is inserted automatically:

```
/*
 * This is a test of the text formatting.
 * Typing a lot of text here will make Vim
 * break
 */
```

For this to work some flags must be present in '**formatoptions**':

r	insert the star when typing <Enter> in Insert mode
o	insert the star when using "o" or "O" in Normal mode
c	break comment text according to ' textwidth '

See [fo-table](#) for more flags.

DEFINING A COMMENT

The '**comments**' option defines what a comment looks like. Vim distinguishes between a single-line comment and a comment that has a different start, end and middle part.

Many single-line comments start with a specific character. In C++ // is used, in Makefiles #, in Vim scripts ". For example, to make Vim understand C++ comments:

```
:set comments=://
```

The colon separates the flags of an item from the text by which the comment is recognized. The general form of an item in '**comments**' is:

```
{flags}:{text}
```

The {**flags**} part can be empty, as in this case.

Several of these items can be concatenated, separated by commas. This allows recognizing different types of comments at the same time. For example, let's edit an e-mail message. When replying, the text that others wrote is preceded with ">" and "!" characters. This command would work:

```
:set comments=n:>,n:!
```

There are two items, one for comments starting with ">" and one for comments

that start with "!". Both use the flag "n". This means that these comments nest. Thus a line starting with ">" may have another comment after the ">". This allows formatting a message like this:

```
> ! Did you see that site?
> ! It looks really great.
> I don't like it. The
> colors are terrible.
What is the URL of that
site?
```

Try setting **'textwidth'** to a different value, e.g., 80, and format the text by visually selecting it and typing "gq". The result is:

```
> ! Did you see that site? It looks really great.
> I don't like it. The colors are terrible.
What is the URL of that site?
```

You will notice that Vim did not move text from one type of comment to another. The "I" in the second line would have fit at the end of the first line, but since that line starts with "> !" and the second line with ">", Vim knows that this is a different kind of comment.

A THREE PART COMMENT

A C comment starts with "/*", has "*" in the middle and "*/" at the end. The entry in **'comments'** for this looks like this:

```
:set comments=s1:/*,mb:*,ex:*/
```

The start is defined with "s1:/*". The "s" indicates the start of a three-piece comment. The colon separates the flags from the text by which the comment is recognized: "/*". There is one flag: "1". This tells Vim that the middle part has an offset of one space.

The middle part "mb:*" starts with "m", which indicates it is a middle part. The "b" flag means that a blank must follow the text. Otherwise Vim would consider text like "*pointer" also to be the middle of a comment.

The end part "ex:*/" has the "e" for identification. The "x" flag has a special meaning. It means that after Vim automatically inserted a star, typing / will remove the extra space.

For more details see [format-comments](#) .

=====

Next chapter: [usr_31.txt](#) Exploiting the GUI

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Exploiting the GUI

Vim works well in a terminal, but the GUI has a few extra items. A file browser can be used for commands that use a file. A dialog to make a choice between alternatives. Use keyboard shortcuts to access menu items quickly.

- 31.1 The file browser
- 31.2 Confirmation
- 31.3 Menu shortcuts
- 31.4 Vim window position and size
- 31.5 Various

Next chapter: [usr_32.txt](#) The undo tree
Previous chapter: [usr_30.txt](#) Editing programs
Table of contents: [usr_toc.txt](#)

31.1 The file browser

When using the File/Open... menu you get a file browser. This makes it easier to find the file you want to edit. But what if you want to split a window to edit another file? There is no menu entry for this. You could first use Window/Split and then File/Open..., but that's more work.

Since you are typing most commands in Vim, opening the file browser with a typed command is possible as well. To make the split command use the file browser, prepend "browse":

```
:browse split
```

Select a file and then the ":split" command will be executed with it. If you cancel the file dialog nothing happens, the window isn't split.

You can also specify a file name argument. This is used to tell the file browser where to start. Example:

```
:browse split /etc
```

The file browser will pop up, starting in the directory "/etc".

The ":browse" command can be prepended to just about any command that opens a file.

If no directory is specified, Vim will decide where to start the file browser. By default it uses the same directory as the last time. Thus when you used ":browse split" and selected a file in "/usr/local/share", the next time you use a ":browse" it will start in "/usr/local/share" again.

This can be changed with the '**browsedir**' option. It can have one of three values:

last	Use the last directory browsed (default)
buffer	Use the same directory as the current buffer

current use the current directory

For example, when you are in the directory `"/usr"`, editing the file `"/usr/local/share/readme"`, then the command:

```
:set browsedir=buffer
:browse edit
```

Will start the browser in `"/usr/local/share"`. Alternatively:

```
:set browsedir=current
:browse edit
```

Will start the browser in `"/usr"`.

Note:

To avoid using the mouse, most file browsers offer using key presses to navigate. Since this is different for every system, it is not explained here. Vim uses a standard browser when possible, your system documentation should contain an explanation on the keyboard shortcuts somewhere.

When you are not using the GUI version, you could use the file explorer window to select files like in a file browser. However, this doesn't work for the `":browse"` command. See [netrw-browse](#).

31.2 Confirmation

Vim protects you from accidentally overwriting a file and other ways to lose changes. If you do something that might be a bad thing to do, Vim produces an error message and suggests appending `!` if you really want to do it.

To avoid retyping the command with the `!`, you can make Vim give you a dialog. You can then press `"OK"` or `"Cancel"` to tell Vim what you want.

For example, you are editing a file and made changes to it. You start editing another file with:

```
:confirm edit foo.txt
```

Vim will pop up a dialog that looks something like this:

```
+-----+
|      ?   Save changes to "bar.txt"?      |
|      YES   NO           CANCEL           |
+-----+
```

Now make your choice. If you do want to save the changes, select `"YES"`. If you want to lose the changes for ever: `"NO"`. If you forgot what you were doing and want to check what really changed use `"CANCEL"`. You will be back in the same file, with the changes still there.

Just like `":browse"`, the `":confirm"` command can be prepended to most commands

that edit another file. They can also be combined:

```
:confirm browse edit
```

This will produce a dialog when the current buffer was changed. Then it will pop up a file browser to select the file to edit.

Note:

In the dialog you can use the keyboard to select the choice. Typically the <Tab> key and the cursor keys change the choice. Pressing <Enter> selects the choice. This depends on the system though.

When you are not using the GUI, the ":confirm" command works as well. Instead of popping up a dialog, Vim will print the message at the bottom of the Vim window and ask you to press a key to make a choice.

```
:confirm edit main.c
Save changes to "Untitled"?
[Y]es, (N)o, (C)ancel:
```

You can now press the single key for the choice. You don't have to press <Enter>, unlike other typing on the command line.

31.3 Menu shortcuts

The keyboard is used for all Vim commands. The menus provide a simple way to select commands, without knowing what they are called. But you have to move your hand from the keyboard and grab the mouse.

Menus can often be selected with keys as well. This depends on your system, but most often it works this way. Use the <Alt> key in combination with the underlined letter of a menu. For example, <A-w> (<Alt> and w) pops up the Window menu.

In the Window menu, the "split" item has the p underlined. To select it, let go of the <Alt> key and press p.

After the first selection of a menu with the <Alt> key, you can use the cursor keys to move through the menus. <Right> selects a submenu and <left> closes it. <Esc> also closes a menu. <Enter> selects a menu item.

There is a conflict between using the <Alt> key to select menu items, and using <Alt> key combinations for mappings. The '**winaltkeys**' option tells Vim what it should do with the <Alt> key.

The default value "menu" is the smart choice: If the key combination is a menu shortcut it can't be mapped. All other keys are available for mapping.

The value "no" doesn't use any <Alt> keys for the menus. Thus you must use the mouse for the menus, and all <Alt> keys can be mapped.

The value "yes" means that Vim will use any <Alt> keys for the menus. Some <Alt> key combinations may also do other things than selecting a menu.

31.4 Vim window position and size

To see the current Vim window position on the screen use:

```
:winpos
```

This will only work in the GUI. The output may look like this:

```
Window position: X 272, Y 103
```

The position is given in screen pixels. Now you can use the numbers to move Vim somewhere else. For example, to move it to the left a hundred pixels:

```
:winpos 172 103
```

Note:

There may be a small offset between the reported position and where the window moves. This is because of the border around the window. This is added by the window manager.

You can use this command in your startup script to position the window at a specific position.

The size of the Vim window is computed in characters. Thus this depends on the size of the font being used. You can see the current size with this command:

```
:set lines columns
```

To change the size set the **'lines'** and/or **'columns'** options to a new value:

```
:set lines=50  
:set columns=80
```

Obtaining the size works in a terminal just like in the GUI. Setting the size is not possible in most terminals.

You can start the X-Windows version of gvim with an argument to specify the size and position of the window:

```
gvim -geometry {width}x{height}+{x-offset}+{y-offset}
```

{width} and {height} are in characters, {x-offset} and {y-offset} are in pixels. Example:

```
gvim -geometry 80x25+100+300
```

=====

31.5 Various

You can use gvim to edit an e-mail message. In your e-mail program you must select gvim to be the editor for messages. When you try that, you will see that it doesn't work: The mail program thinks that editing is finished, while gvim is still running!

What happens is that gvim disconnects from the shell it was started in. That is fine when you start gvim in a terminal, so that you can do other work

in that terminal. But when you really want to wait for gvim to finish, you must prevent it from disconnecting. The "-f" argument does this:

```
gvim -f file.txt
```

The "-f" stands for foreground. Now Vim will block the shell it was started in until you finish editing and exit.

DELAYED START OF THE GUI

On Unix it's possible to first start Vim in a terminal. That's useful if you do various tasks in the same shell. If you are editing a file and decide you want to use the GUI after all, you can start it with:

```
:gui
```

Vim will open the GUI window and no longer use the terminal. You can continue using the terminal for something else. The "-f" argument is used here to run the GUI in the foreground. You can also use ":gui -f".

THE GVIM STARTUP FILE

When gvim starts, it reads the gvimrc file. That's similar to the vimrc file used when starting Vim. The gvimrc file can be used for settings and commands that are only to be used when the GUI is going to be started. For example, you can set the '**lines**' option to set a different window size:

```
:set lines=55
```

You don't want to do this in a terminal, since its size is fixed (except for an xterm that supports resizing).

The gvimrc file is searched for in the same locations as the vimrc file. Normally its name is "~/.gvimrc" for Unix and "\$VIM/_gvimrc" for MS-Windows. The \$MYGVIMRC environment variable is set to it, thus you can use this command to edit the file, if you have one:

```
:edit $MYGVIMRC
```

If for some reason you don't want to use the normal gvimrc file, you can specify another one with the "-U" argument:

```
gvim -U thisrc ...
```

That allows starting gvim for different kinds of editing. You could set another font size, for example.

To completely skip reading a gvimrc file:

```
gvim -U NONE ...
```

=====

Next chapter: [usr_32.txt](#) The undo tree

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

The undo tree

Vim provides multi-level undo. If you undo a few changes and then make a new change you create a branch in the undo tree. This text is about moving through the branches.

- 32.1 Undo up to a file write
- 32.2 Numbering changes
- 32.3 Jumping around the tree
- 32.4 Time travelling

Next chapter: [usr_40.txt](#) Make new commands
Previous chapter: [usr_31.txt](#) Exploiting the GUI
Table of contents: [usr_toc.txt](#)

32.1 Undo up to a file write

Sometimes you make several changes, and then discover you want to go back to when you have last written the file. You can do that with this command:

```
:earlier 1f
```

The "f" stands for "file" here.

You can repeat this command to go further back in the past. Or use a count different from 1 to go back faster.

If you go back too far, go forward again with:

```
:later 1f
```

Note that these commands really work in time sequence. This matters if you made changes after undoing some changes. It's explained in the next section.

Also **note** that we are talking about text writes here. For writing the undo information in a file see [undo-persistence](#).

32.2 Numbering changes

In section [02.5](#) we only discussed one line of undo/redo. But it is also possible to branch off. This happens when you undo a few changes and then make a new change. The new changes become a branch in the undo tree.

Let's start with the text "one". The first change to make is to append " too". And then move to the first 'o' and change it into 'w'. We then have two changes, numbered 1 and 2, and three states of the text:

```

one
|
change 1
|
one too
|
change 2
|
one two

```

If we now undo one change, back to "one too", and change "one" to "me" we create a branch in the undo tree:

```

one
|
change 1
|
one too
/ \
change 2 change 3
|   |
one two me too

```

You can now use the `u` command to undo. If you do this twice you get to "one". Use `CTRL-R` to redo, and you will go to "one too". One more `CTRL-R` takes you to "me too". Thus undo and redo go up and down in the tree, using the branch that was last used.

What matters here is the order in which the changes are made. Undo and redo are not considered changes in this context. After each change you have a new state of the text.

Note that only the changes are numbered, the text shown in the tree above has no identifier. They are mostly referred to by the number of the change above it. But sometimes by the number of one of the changes below it, especially when moving up in the tree, so that you know which change was just undone.

=====

32.3 Jumping around the tree

So how do you get to "one two" now? You can use this command:

```
:undo 2
```

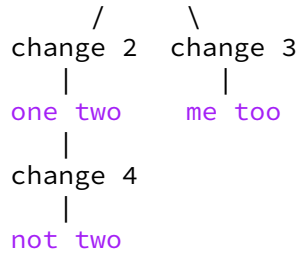
The text is now "one two", you are below change 2. You can use the `:undo` command to jump to below any change in the tree.

Now make another change: change "one" to "not":

```

one
|
change 1
|
one too

```



Now you change your mind and want to go back to "me too". Use the `g-` command. This moves back in time. Thus it doesn't walk the tree upwards or downwards, but goes to the change made before.

You can repeat `g-` and you will see the text change:

```

me too
one two
one too
one

```

Use `g+` to move forward in time:

```

one
one too
one two
me too
not two

```

Using `:undo` is useful if you know what change you want to jump to. `g-` and `g+` are useful if you don't know exactly what the change number is.

You can type a count before `g-` and `g+` to repeat them.

=====

32.4 Time travelling

When you have been working on text for a while the tree grows to become big. Then you may want to go to the text of some minutes ago.

To see what branches there are in the undo tree use this command:

```

:undolist
number changes  time
      3         2  16 seconds ago
      4         3   5 seconds ago

```

Here you can see the number of the leaves in each branch and when the change was made. Assuming we are below change 4, at "not two", you can go back ten seconds with this command:

```

:earlier 10s

```

Depending on how much time you took for the changes you end up at a certain position in the tree. The `:earlier` command argument can be "m" for minutes, "h" for hours and "d" for days. To go all the way back use a big number:

```
:earlier 100d
```

To travel forward in time again use the `:later` command:

```
:later 1m
```

The arguments are "s", "m" and "h", just like with `:earlier` .

If you want even more details, or want to manipulate the information, you can use the `undotree()` function. To see what it returns:

```
:echo undotree()
```

=====

Next chapter: [usr_40.txt](#) Make new commands

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Make new commands

Vim is an extensible editor. You can take a sequence of commands you use often and turn it into a new command. Or redefine an existing command. Autocommands make it possible to execute commands automatically.

- 40.1 Key mapping
- 40.2 Defining command-line commands
- 40.3 Autocommands

Next chapter: [usr_41.txt](#) Write a Vim script
Previous chapter: [usr_32.txt](#) The undo tree
Table of contents: [usr_toc.txt](#)

40.1 Key mapping

A simple mapping was explained in section [05.4](#) . The principle is that one sequence of key strokes is translated into another sequence of key strokes. This is a simple, yet powerful mechanism.

The simplest form is that one key is mapped to a sequence of keys. Since the function keys, except `<F1>`, have no predefined meaning in Vim, these are good choices to map. Example:

```
:map <F2> GoDate: <Esc>:read !date<CR>kJ
```

This shows how three modes are used. After going to the last line with "G", the "o" command opens a new line and starts Insert mode. The text "Date: " is inserted and `<Esc>` takes you out of insert mode.

Notice the use of special keys inside `<>`. This is called angle bracket notation. You type these as separate characters, not by pressing the key itself. This makes the mappings better readable and you can copy and paste the text without problems.

The ":" character takes Vim to the command line. The `:read !date` command reads the output from the "date" command and appends it below the current line. The `<CR>` is required to execute the `:read` command.

At this point of execution the text looks like this:

```
Date:  
Fri Jun 15 12:54:34 CEST 2001
```

Now "kJ" moves the cursor up and joins the lines together.

To decide which key or keys you use for mapping, see [map-which-keys](#) .

MAPPING AND MODES

The `:map` command defines remapping for keys in Normal mode. You can also define mappings for other modes. For example, `:imap` applies to Insert mode.

You can use it to insert a date below the cursor:

```
:imap <F2> <CR>Date: <Esc>:read !date<CR>kJ
```

It looks a lot like the mapping for <F2> in Normal mode, only the start is different. The <F2> mapping for Normal mode is still there. Thus you can map the same key differently for each mode.

Notice that, although this mapping starts in Insert mode, it ends in Normal mode. If you want it to continue in Insert mode, append an "a" to the mapping.

Here is an overview of map commands and in which mode they work:

:map	Normal, Visual and Operator-pending
:vmap	Visual
:nmap	Normal
:omap	Operator-pending
:map!	Insert and Command-line
:imap	Insert
:cmap	Command-line

Operator-pending mode is when you typed an operator character, such as "d" or "y", and you are expected to type the motion command or a text object. Thus when you type "dw", the "w" is entered in operator-pending mode.

Suppose that you want to define <F7> so that the command d<F7> deletes a C program block (text enclosed in curly braces, {}). Similarly y<F7> would yank the program block into the unnamed register. Therefore, what you need to do is to define <F7> to select the current program block. You can do this with the following command:

```
:omap <F7> a{
```

This causes <F7> to perform a select block "a{" in operator-pending mode, just like you typed it. This mapping is useful if typing a { on your keyboard is a bit difficult.

LISTING MAPPINGS

To see the currently defined mappings, use ":map" without arguments. Or one of the variants that include the mode in which they work. The output could look like this:

```
  _g          :call MyGrep(1)<CR>
v <F2>       :s/^/> /<CR>:noh<CR>``
n <F2>       :.,$s/^/> /<CR>:noh<CR>``
  <xHome>    <Home>
  <xEnd>     <End>
```

The first column of the list shows in which mode the mapping is effective. This is "n" for Normal mode, "i" for Insert mode, etc. A blank is used for a mapping defined with ":map", thus effective in both Normal and Visual mode.

One useful purpose of listing the mapping is to check if special keys in `<>` form have been recognized (this only works when color is supported). For example, when `<Esc>` is displayed in color, it stands for the escape character. When it has the same color as the other text, it is five characters.

REMAPPING

The result of a mapping is inspected for other mappings in it. For example, the mappings for `<F2>` above could be shortened to:

```
:map <F2> G<F3>
:imap <F2> <Esc><F3>
:map <F3> oDate: <Esc>:read !date<CR>kJ
```

For Normal mode `<F2>` is mapped to go to the last line, and then behave like `<F3>` was pressed. In Insert mode `<F2>` stops Insert mode with `<Esc>` and then also uses `<F3>`. Then `<F3>` is mapped to do the actual work.

Suppose you hardly ever use Ex mode, and want to use the "Q" command to format text (this was so in old versions of Vim). This mapping will do it:

```
:map Q gq
```

But, in rare cases you need to use Ex mode anyway. Let's map "gQ" to Q, so that you can still go to Ex mode:

```
:map gQ Q
```

What happens now is that when you type "gQ" it is mapped to "Q". So far so good. But then "Q" is mapped to "gq", thus typing "gQ" results in "gq", and you don't get to Ex mode at all.

To avoid keys to be mapped again, use the ":noremap" command:

```
:noremap gQ Q
```

Now Vim knows that the "Q" is not to be inspected for mappings that apply to it. There is a similar command for every mode:

```
:noremap      Normal, Visual and Operator-pending
:vnoremap     Visual
:nnoremap     Normal
:onoremap     Operator-pending
:noremap!    Insert and Command-line
:inoremap     Insert
:cnoremap     Command-line
```

RECURSIVE MAPPING

When a mapping triggers itself, it will run forever. This can be used to repeat an action an unlimited number of times.

For example, you have a list of files that contain a version number in the first line. You edit these files with "vim *.txt". You are now editing the

first file. Define this mapping:

```
:map ,, :s/5.1/5.2/<CR>:wnext<CR>,,
```

Now you type ",,". This triggers the mapping. It replaces "5.1" with "5.2" in the first line. Then it does a ":wnext" to write the file and edit the next one. The mapping ends in ",,". This triggers the same mapping again, thus doing the substitution, etc.

This continues until there is an error. In this case it could be a file where the substitute command doesn't find a match for "5.1". You can then make a change to insert "5.1" and continue by typing ",," again. Or the ":wnext" fails, because you are in the last file in the list.

When a mapping runs into an error halfway, the rest of the mapping is discarded. **CTRL-C** interrupts the mapping (CTRL-Break on MS-Windows).

DELETE A MAPPING

To remove a mapping use the ":unmap" command. Again, the mode the unmapping applies to depends on the command used:

:unmap	Normal, Visual and Operator-pending
:vunmap	Visual
:nunmap	Normal
:ounmap	Operator-pending
:unmap!	Insert and Command-line
:iunmap	Insert
:cunmap	Command-line

There is a trick to define a mapping that works in Normal and Operator-pending mode, but not in Visual mode. First define it for all three modes, then delete it for Visual mode:

```
:map <C-A> /---><CR>  
:vunmap <C-A>
```

Notice that the five characters "<C-A>" stand for the single key **CTRL-A**.

To remove all mappings use the `:mapclear` command. You can guess the variations for different modes by now. Be careful with this command, it can't be undone.

SPECIAL CHARACTERS

The ":map" command can be followed by another command. A | character separates the two commands. This also means that a | character can't be used inside a map command. To include one, use <Bar> (five characters). Example:

```
:map <F8> :write <Bar> !checkin %:S<CR>
```

The same problem applies to the ":unmap" command, with the addition that you have to watch out for trailing white space. These two commands are different:

```
:unmap a | unmap b
:unmap a| unmap b
```

The first command tries to unmap "a ", with a trailing space.

When using a space inside a mapping, use `<Space>` (seven characters):

```
:map <Space> W
```

This makes the spacebar move a blank-separated word forward.

It is not possible to put a comment directly after a mapping, because the " character is considered to be part of the mapping. You can use |", this starts a new, empty command with a comment. Example:

```
:map <Space> W| " Use spacebar to move forward a word
```

MAPPINGS AND ABBREVIATIONS

Abbreviations are a lot like Insert mode mappings. The arguments are handled in the same way. The main difference is the way they are triggered. An abbreviation is triggered by typing a non-word character after the word. A mapping is triggered when typing the last character.

Another difference is that the characters you type for an abbreviation are inserted in the text while you type them. When the abbreviation is triggered these characters are deleted and replaced by what the abbreviation produces. When typing the characters for a mapping, nothing is inserted until you type the last character that triggers it. If the `'showcmd'` option is set, the typed characters are displayed in the last line of the Vim window.

An exception is when a mapping is ambiguous. Suppose you have done two mappings:

```
:imap aa foo
:imap aaa bar
```

Now, when you type "aa", Vim doesn't know if it should apply the first or the second mapping. It waits for another character to be typed. If it is an "a", the second mapping is applied and results in "bar". If it is a space, for example, the first mapping is applied, resulting in "foo", and then the space is inserted.

ADDITIONALLY...

The `<script>` keyword can be used to make a mapping local to a script. See

```
:map-<script> .
```

The `<buffer>` keyword can be used to make a mapping local to a specific buffer. See

```
:map-<buffer>
```

The `<unique>` keyword can be used to make defining a new mapping fail when it already exists. Otherwise a new mapping simply overwrites the old one. See

```
:map-<unique> .
```

To make a key do nothing, map it to `<Nop>` (five characters). This will make the `<F7>` key do nothing at all:

```
:map <F7> <Nop>| map! <F7> <Nop>
```

There must be no space after `<Nop>`.

40.2 Defining command-line commands

The Vim editor enables you to define your own commands. You execute these commands just like any other Command-line mode command.

To define a command, use the `":command"` command, as follows:

```
:command DeleteFirst ldelete
```

Now when you execute the command `":DeleteFirst"` Vim executes `":ldelete"`, which deletes the first line.

Note:

User-defined commands must start with a capital letter. You cannot use `":X"`, `":Next"` and `":Print"`. The underscore cannot be used! You can use digits, but this is discouraged.

To list the user-defined commands, execute the following command:

```
:command
```

Just like with the builtin commands, the user defined commands can be abbreviated. You need to type just enough to distinguish the command from another. Command line completion can be used to get the full name.

NUMBER OF ARGUMENTS

User-defined commands can take a series of arguments. The number of arguments must be specified by the `-nargs` option. For instance, the example `":DeleteFirst"` command takes no arguments, so you could have defined it as follows:

```
:command -nargs=0 DeleteFirst ldelete
```

However, because zero arguments is the default, you do not need to add `"-nargs=0"`. The other values of `-nargs` are as follows:

<code>-nargs=0</code>	No arguments
<code>-nargs=1</code>	One argument
<code>-nargs=*</code>	Any number of arguments
<code>-nargs=?</code>	Zero or one argument
<code>-nargs=+</code>	One or more arguments

USING THE ARGUMENTS

Inside the command definition, the arguments are represented by the `<args>` keyword. For example:

```
:command -nargs=+ Say :echo "<args>"
```

Now when you type

```
:Say Hello World
```

Vim echoes "Hello World". However, if you add a double quote, it won't work. For example:

```
:Say he said "hello"
```

To get special characters turned into a string, properly escaped to use as an expression, use "`<q-args>`":

```
:command -nargs=+ Say :echo <q-args>
```

Now the above `":Say"` command will result in this to be executed:

```
:echo "he said \"hello\""
```

The `<f-args>` keyword contains the same information as the `<args>` keyword, except in a format suitable for use as function call arguments. For example:

```
:command -nargs=* DoIt :call AFunction(<f-args>)  
:DoIt a b c
```

Executes the following command:

```
:call AFunction("a", "b", "c")
```

LINE RANGE

Some commands take a range as their argument. To tell Vim that you are defining such a command, you need to specify a `-range` option. The values for this option are as follows:

<code>-range</code>	Range is allowed; default is the current line.
<code>-range=%</code>	Range is allowed; default is the whole file.
<code>-range={count}</code>	Range is allowed; the last number in it is used as a single number whose default is <code>{count}</code> .

When a range is specified, the keywords `<line1>` and `<line2>` get the values of the first and last line in the range. For example, the following command defines the `SaveIt` command, which writes out the specified range to the file `"save_file"`:

```
:command -range=% SaveIt :<line1>,<line2>write! save_file
```

OTHER OPTIONS

Some of the other options and keywords are as follows:

<code>-count={number}</code>	The command can take a count whose default is <code>{number}</code> . The resulting count can be used through the <code><count></code> keyword.
<code>-bang</code>	You can use a <code>!</code> . If present, using <code><bang></code> will result in a <code>!</code> .
<code>-register</code>	You can specify a register. (The default is the unnamed register.) The register specification is available as <code><reg></code> (a.k.a. <code><register></code>).
<code>-complete={type}</code>	Type of command-line completion used. See <code>:command-completion</code> for the list of possible values.
<code>-bar</code>	The command can be followed by <code> </code> and another command, or <code>"</code> and a comment.
<code>-buffer</code>	The command is only available for the current buffer.

Finally, you have the `<lt>` keyword. It stands for the character `<`. Use this to escape the special meaning of the `<>` items mentioned.

REDEFINING AND DELETING

To redefine the same command use the `!` argument:

```
:command -nargs=+ Say :echo "<args>"  
:command! -nargs=+ Say :echo <q-args>
```

To delete a user command use `":delcommand"`. It takes a single argument, which is the name of the command. Example:

```
:delcommand SaveIt
```

To delete all the user commands:

```
:comclear
```

Careful, this can't be undone!

More details about all this in the reference manual: [user-commands](#) .

=====

40.3 Autocommands

An autocommand is a command that is executed automatically in response to some event, such as a file being read or written or a buffer change. Through the use of autocommands you can train Vim to edit compressed files, for example. That is used in the `gzip` plugin.

Autocommands are very powerful. Use them with care and they will help you avoid typing many commands. Use them carelessly and they will cause a lot of

trouble.

Suppose you want to replace a datestamp on the end of a file every time it is written. First you define a function:

```
:function DateInsert()  
: $delete  
: read !date  
:endfunction
```

You want this function to be called each time, just before a buffer is written to a file. This will make that happen:

```
:autocmd BufWritePre * call DateInsert()
```

"BufWritePre" is the event for which this autocommand is triggered: Just before (pre) writing a buffer to a file. The "*" is a pattern to match with the file name. In this case it matches all files.

With this command enabled, when you do a ":write", Vim checks for any matching BufWritePre autocommands and executes them, and then it performs the ":write".

The general form of the :autocmd command is as follows:

```
:autocmd [group] {events} {file-pattern} [++nested] {command}
```

The [group] name is optional. It is used in managing and calling the commands (more on this later). The {events} parameter is a list of events (comma separated) that trigger the command.

{file-pattern} is a filename, usually with wildcards. For example, using "*.txt" makes the autocommand be used for all files whose name end in ".txt". The optional [++nested] flag allows for nesting of autocommands (see below), and finally, {command} is the command to be executed.

When adding an autocommand the already existing ones remain. To avoid adding the autocommand several times you should use this form:

```
:augroup updateDate  
: autocmd!  
: autocmd BufWritePre * call DateInsert()  
:augroup END
```

This will delete any previously defined autocommand with `:autocmd!` before defining the new one. Groups are explained later.

EVENTS

One of the most useful events is BufReadPost. It is triggered after a new file is being edited. It is commonly used to set option values. For example, you know that "*.gsm" files are GNU assembly language. To get the syntax file right, define this autocommand:

```
:autocmd BufReadPost *.gsm set filetype=asm
```

If Vim is able to detect the type of file, it will set the **'filetype'** option for you. This triggers the Filetype event. Use this to do something when a certain type of file is edited. For example, to load a list of abbreviations for text files:

```
:autocmd FileType text source ~/.vim/abbrevs.vim
```

When starting to edit a new file, you could make Vim insert a skeleton:

```
:autocmd BufNewFile *. [ch] @read ~/skeletons/skel.c
```

See [autocmd-events](#) for a complete list of events.

PATTERNS

The `{file-pattern}` argument can actually be a comma-separated list of file patterns. For example: `*.c,*.h` matches files ending in `.c` and `.h`.

The usual file wildcards can be used. Here is a summary of the most often used ones:

<code>*</code>	Match any character any number of times
<code>?</code>	Match any character once
<code>[abc]</code>	Match the character a, b or c
<code>.</code>	Matches a dot
<code>a{b,c}</code>	Matches "ab" and "ac"

When the pattern includes a slash (/) Vim will compare directory names. Without the slash only the last part of a file name is used. For example, `*.txt` matches `/home/biep/readme.txt`. The pattern `/home/biep/*` would also match it. But `home/foo/*.txt` wouldn't.

When including a slash, Vim matches the pattern against both the full path of the file (`/home/biep/readme.txt`) and the relative path (e.g., `biep/readme.txt`).

Note:

When working on a system that uses a backslash as file separator, such as MS-Windows, you still use forward slashes in autocommands. This makes it easier to write the pattern, since a backslash has a special meaning. It also makes the autocommands portable.

DELETING

To delete an autocommand, use the same command as what it was defined with, but leave out the `{command}` at the end and use a `!`. Example:

```
:autocmd! FileWritePre *
```

This will delete all autocommands for the "FileWritePre" event that use the `"*"` pattern.

LISTING

To list all the currently defined autocommands, use this:

```
:autocmd
```

The list can be very long, especially when filetype detection is used. To list only part of the commands, specify the group, event and/or pattern. For example, to list all BufNewFile autocommands:

```
:autocmd BufNewFile
```

To list all autocommands for the pattern "*.c":

```
:autocmd * *.c
```

Using "*" for the event will list all the events. To list all autocommands for the cprograms group:

```
:autocmd cprograms
```

GROUPS

The {group} item, used when defining an autocommand, groups related autocommands together. This can be used to delete all the autocommands in a certain group, for example.

When defining several autocommands for a certain group, use the ":augroup" command. For example, let's define autocommands for C programs:

```
:augroup cprograms
: autocmd BufReadPost *.c,*.h :set sw=4 sts=4
: autocmd BufReadPost *.cpp :set sw=3 sts=3
:augroup END
```

This will do the same as:

```
:autocmd cprograms BufReadPost *.c,*.h :set sw=4 sts=4
:autocmd cprograms BufReadPost *.cpp :set sw=3 sts=3
```

To delete all autocommands in the "cprograms" group:

```
:autocmd! cprograms
```

NESTING

Generally, commands executed as the result of an autocommand event will not trigger any new events. If you read a file in response to a FileChangedShell event, it will not trigger the autocommands that would set the syntax, for example. To make the events triggered, add the "nested" argument:

```
:autocmd FileChangedShell * ++nested edit
```

EXECUTING AUTOCOMMANDS

It is possible to trigger an autocommand by pretending an event has occurred. This is useful to have one autocommand trigger another one. Example:

```
:autocmd BufReadPost *.new execute "doautocmd BufReadPost " . expand("<afile>:r")
```

This defines an autocommand that is triggered when a new file has been edited. The file name must end in ".new". The ":execute" command uses expression evaluation to form a new command and execute it. When editing the file "tryout.c.new" the executed command will be:

```
:doautocmd BufReadPost tryout.c
```

The expand() function takes the "<afile>" argument, which stands for the file name the autocommand was executed for, and takes the root of the file name with ":r".

":doautocmd" executes on the current buffer. The ":doautoall" command works like "doautocmd" except it executes on all the buffers.

USING NORMAL MODE COMMANDS

The commands executed by an autocommand are Command-line commands. If you want to use a Normal mode command, the ":normal" command can be used.

Example:

```
:autocmd BufReadPost *.log normal G
```

This will make the cursor jump to the last line of *.log files when you start to edit it.

Using the ":normal" command is a bit tricky. First of all, make sure its argument is a complete command, including all the arguments. When you use "i" to go to Insert mode, there must also be a <Esc> to leave Insert mode again. If you use a "/" to start a search pattern, there must be a <CR> to execute it.

The ":normal" command uses all the text after it as commands. Thus there can be no | and another command following. To work around this, put the ":normal" command inside an ":execute" command. This also makes it possible to pass unprintable characters in a convenient way. Example:

```
:autocmd BufReadPost *.chg execute "normal ONew entry:\<Esc>" |  
  \ lread !date
```

This also shows the use of a backslash to break a long command into more lines. This can be used in Vim scripts (not at the command line).

When you want the autocommand do something complicated, which involves jumping around in the file and then returning to the original position, you may want to restore the view on the file. See [restore-position](#) for an example.

IGNORING EVENTS

At times, you will not want to trigger an autocommand. The **'eventignore'** option contains a list of events that will be totally ignored. For example, the following causes events for entering and leaving a window to be ignored:

```
:set eventignore=WinEnter,WinLeave
```

To ignore all events, use the following command:

```
:set eventignore=all
```

To set it back to the normal behavior, make **'eventignore'** empty:

```
:set eventignore=
```

=====

Next chapter: [usr_41.txt](#) Write a Vim script

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Write a Vim script

The Vim script language is used for the startup vimrc file, syntax files, and many other things. This chapter explains the items that can be used in a Vim script. There are a lot of them, therefore this is a long chapter.

- 41.1 Introduction
- 41.2 Variables
- 41.3 Expressions
- 41.4 Conditionals
- 41.5 Executing an expression
- 41.6 Using functions
- 41.7 Defining a function
- 41.8 Lists and Dictionaries
- 41.9 White space
- 41.10 Line continuation
- 41.11 Comments
- 41.12 Fileformat

Next chapter: [usr_42.txt](#) Add new menus
Previous chapter: [usr_40.txt](#) Make new commands
Table of contents: [usr_toc.txt](#)

41.1 Introduction vim-script-intro script

Your first experience with Vim scripts is the vimrc file. Vim reads it when it starts up and executes the commands. You can set options to the values you prefer, define mappings, select plugins and much more. You can use any colon command in it (commands that start with a ":"; these are sometimes referred to as Ex commands or command-line commands).

Syntax files are also Vim scripts. As are files that set options for a specific file type. A complicated macro can be defined by a separate Vim script file. You can think of other uses yourself.

Vim script comes in two flavors: legacy and Vim9 . Since this help file is for new users, we'll teach you the newer and more convenient Vim9 syntax. While legacy script is particularly for Vim, Vim9 script looks more like other languages, such as JavaScript and TypeScript.

To try out Vim script the best way is to edit a script file and source it. Basically:

```
:edit test.vim
[insert the script lines you want]
:w
:source %
```

Let's start with a simple example:

```
vim9script
var i = 1
while i < 5
    echo "count is" i
    i += 1
endwhile
```

The output of the example code is:

```
count is 1
count is 2
count is 3
count is 4
```

In the first line the ``vim9script`` command makes clear this is a new, [Vim9](#) script file. That matters for how the rest of the file is used. It is recommended to put it in the very first line, before any comments.

[vim9-declarations](#)

The ``var i = 1`` command declares the "i" variable and initializes it. The generic form is:

```
var {name} = {expression}
```

In this case the variable name is "i" and the expression is a simple value, the number one.

The ``while`` command starts a loop. The generic form is:

```
while {condition}
    {statements}
endwhile
```

The statements until the matching ``endwhile`` are executed for as long as the condition is true. The condition used here is the expression "i < 5". This is true when the variable i is smaller than five.

Note:

If you happen to write a while loop that keeps on running, you can interrupt it by pressing **CTRL-C** (CTRL-Break on MS-Windows).

The ``echo`` command prints its arguments. In this case the string "count is" and the value of the variable i. Since i is one, this will print:

```
count is 1
```

Then there is the ``i += 1`` command. This does the same thing as "i = i + 1", it adds one to the variable i and assigns the new value to the same variable.

The example was given to explain the commands, but would you really want to make such a loop, it can be written much more compact:

```
for i in range(1, 4)
    echo $"count is {i}"
endfor
```

We won't explain how ``for``, ``range()`` and ``$"string"`` work until later. Follow the links if you are impatient.

TRYING OUT EXAMPLES

You can easily try out most examples in these help files without saving the commands to a file. For example, to try out the "for" loop above do this:

1. position the cursor on the "for"
2. start Visual mode with "v"
3. move down to the "endfor"
4. press colon, then "so" and Enter

After pressing colon you will see `:"<,>"`, which is the range of the Visually selected text.

For some commands it matters they are executed as in `Vim9` script. But typed commands normally use legacy script syntax, such as the example below that causes the E1004 error. For that use this fourth step:

4. press colon, then "vim9 so" and Enter

"vim9" is short for ``vim9cmd``, which is a command modifier to execute the following command in `Vim9` syntax.

Note that this won't work for examples that require a script context.

FOUR KINDS OF NUMBERS

Numbers can be decimal, hexadecimal, octal and binary.

A hexadecimal number starts with "0x" or "0X". For example "0x1f" is decimal 31 and "0x1234" is decimal 4660.

An octal number starts with "0o", "0O". "0o17" is decimal 15.

A binary number starts with "0b" or "0B". For example "0b101" is decimal 5.

A decimal number is just digits. Careful: In legacy script don't put a zero before a decimal number, it will be interpreted as an octal number! That's one reason to use `Vim9` script.

The ``echo`` command evaluates its argument and when it is a number always prints the decimal form. Example:

```
echo 0x7f 0o36
127 30
```

A number is made negative with a minus sign. This also works for hexadecimal, octal and binary numbers:

```
echo -0x7f
-127
```

A minus sign is also used for subtraction. This can sometimes lead to confusion. If we put a minus sign before both numbers we get an error:

```
echo -0x7f -0o36
E1004: White space required before and after '-' at "-0o36"
```

Note: if you are not using a Vim9 script to try out these commands but type them directly, they will be executed as legacy script. Then the echo command sees the second minus sign as subtraction. To get the error, prefix the command with ``vim9cmd``:

```
vim9cmd echo -0x7f -0o36
E1004: White space required before and after '-' at "-0o36"
```

White space in an expression is often required to make sure it is easy to read and avoid errors. Such as thinking that the "-0o36" above makes the number negative, while it is actually seen as a subtraction.

To actually have the minus sign be used for negation, you can put the second expression in parentheses:

```
echo -0x7f (-0o36)
-127 -30
```

41.2 Variables

A variable name consists of ASCII letters, digits and the underscore. It cannot start with a digit. Valid variable names are:

```
counter
_aap3
very_long_variable_name_with_underscores
CamelCaseName
LENGTH
```

Invalid names are "foo.bar" and "6var".

Some variables are global. To see a list of currently defined global variables type this command:

```
:let
```

You can use global variables everywhere. However, it is too easy to use the same name in two unrelated scripts. Therefore variables declared in a script are local to that script. For example, if you have this in "script1.vim":

```
vim9script
var counter = 5
echo counter
5
```

And you try to use the variable in "script2.vim":


```
vim9script
echo counter
E121: Undefined variable: counter
```

Using a script-local variable means you can be sure that it is only changed in that script and not elsewhere.

If you do want to share variables between scripts, use the "g:" prefix and assign the value directly, do not use ``var``. And use a specific name to avoid mistakes. Thus in "script1.vim":

```
vim9script
g:mash_counter = 5
echo g:mash_counter
5
```

And then in "script2.vim":

```
vim9script
echo g:mash_counter
5
```

Global variables can also be accessed on the command line, E.g. typing this:

```
echo g:mash_counter
```

That will not work for a script-local variable.

More about script-local variables here: [script-variable](#) .

There are more kinds of variables, see [internal-variables](#) . The most often used ones are:

b:name	variable local to a buffer
w:name	variable local to a window
g:name	global variable (also in a function)
v:name	variable predefined by Vim

DELETING VARIABLES

Variables take up memory and show up in the output of the ``let`` command. To delete a global variable use the ``unlet`` command. Example:

```
unlet g:counter
```

This deletes the global variable "g:counter" to free up the memory it uses. If you are not sure if the variable exists, and don't want an error message when it doesn't, append !:

```
unlet! g:counter
```

You cannot ``unlet`` script-local variables in [Vim9](#) script, only in legacy script.

When a script has been processed to the end, the local variables declared there will not be deleted. Functions defined in the script can use them. Example:

```
vim9script
var counter = 0
def g:GetCount(): number
    counter += 1
    return counter
enddef
```

Every time you call the function it will return the next count:

```
:echo g:GetCount()
1

:echo g:GetCount()
2
```

If you are worried a script-local variable is consuming too much memory, set it to an empty or null value after you no longer need it. Example:

```
var lines = readfile(...)
...
lines = []
```

Note: below we'll leave out the ``vim9script`` line from examples, so we can concentrate on the relevant commands, but you'll still need to put it at the top of your script file.

STRING VARIABLES AND CONSTANTS

So far only numbers were used for the variable value. Strings can be used as well. Numbers and strings are the basic types of variables that Vim supports. Example:

```
var name = "Peter"
echo name
Peter
```

Every variable has a type. Very often, as in this example, the type is defined by assigning a value. This is called type inference. If you do not want to give the variable a value yet, you need to specify the type:

```
var name: string
var age: number
if male
    name = "Peter"
    age = 42
else
    name = "Elisa"
    age = 45
endif
```

If you make a mistake and try to assign the wrong type of value you'll get an

error:

```
    age = "Peter"  
E1012: Type mismatch; expected number but got string
```

More about types in [41.8](#) .

To assign a string value to a variable, you can use a string constant. There are two types of these. First the string in double quotes, as we used already. If you want to include a double quote inside the string, put a backslash in front of it:

```
    var name = "he is \"Peter\""  
    echo name  
he is "Peter"
```

To avoid the need for backslashes, you can use a string in single quotes:

```
    var name = 'he is "Peter"'  
    echo name  
he is "Peter"
```

Inside a single-quote string all the characters are as they are. Only the single quote itself is special: you need to use two to get one. A backslash is taken literally, thus you can't use it to change the meaning of the character after it:

```
    var name = 'P\e'ter''  
    echo name  
P\e'ter'
```

In double-quote strings it is possible to use special characters. Here are a few useful ones:

<code>\t</code>	<code><Tab></code>
<code>\n</code>	<code><NL></code> , line break
<code>\r</code>	<code><CR></code> , <code><Enter></code>
<code>\e</code>	<code><Esc></code>
<code>\b</code>	<code><BS></code> , backspace
<code>\"</code>	<code>"</code>
<code>\\</code>	<code>\</code> , backslash
<code>\<Esc></code>	<code><Esc></code>
<code>\<C-W></code>	CTRL-W

The last two are just examples. The `"\<name>"` form can be used to include the special key "name".

See [expr-quote](#) for the full list of special items in a string.

41.3 Expressions

Vim has a fairly standard way to handle expressions. You can read the definition here: [expression-syntax](#) . Here we will show the most common

items.

The numbers, strings and variables mentioned above are expressions by themselves. Thus everywhere an expression is expected, you can use a number, string or variable. Other basic items in an expression are:

<code>\$NAME</code>	environment variable
<code>&name</code>	option value
<code>@r</code>	register contents

Examples:

```
echo "The value of 'tabstop' is" &ts
echo "Your home directory is" $HOME
if @a == 'text'
```

The `&name` form can also be used to set an option value, do something and restore the old value. Example:

```
var save_ic = &ic
set noic
s/The Start/The Beginning/
&ic = save_ic
```

This makes sure the "The Start" pattern is used with the **'ignorecase'** option off. Still, it keeps the value that the user had set. (Another way to do this would be to add `"\C"` to the pattern, see [/\C .](#))

MATHEMATICS

It becomes more interesting if we combine these basic items. Let's start with mathematics on numbers:

<code>a + b</code>	add
<code>a - b</code>	subtract
<code>a * b</code>	multiply
<code>a / b</code>	divide
<code>a % b</code>	modulo

The usual precedence is used. Example:

```
echo 10 + 5 * 2
20
```

Grouping is done with parentheses. No surprises here. Example:

```
echo (10 + 5) * 2
30
```

OTHERS

Strings can be concatenated with `".."` (see [expr6](#)). Example:

```
echo "Name: " .. name
Name: Peter
```

When the "echo" command gets multiple arguments, it separates them with a space. In the example the argument is a single expression, thus no space is inserted.

If you don't like the concatenation you can use the "\$string" form, which accepts an expression in curly braces:

```
echo $"Name: {name}"
```

See [interpolated-string](#) for more information.

Borrowed from the C language is the conditional expression:

```
a ? b : c
```

If "a" evaluates to true "b" is used, otherwise "c" is used. Example:

```
var nr = 4
echo nr > 5 ? "nr is big" : "nr is small"
nr is small
```

The three parts of the constructs are always evaluated first, thus you could see it works as:

```
(a) ? (b) : (c)
```

There is also the falsy operator:

```
echo name ?? "No name given"
```

See [??](#) .

41.4 Conditionals

The ``if`` commands executes the following statements, until the matching ``endif``, only when a condition is met. The generic form is:

```
if {condition}
  {statements}
endif
```

Only when the expression `{condition}` evaluates to true or one will the `{statements}` be executed. If they are not executed they must still be valid commands. If they contain garbage, Vim won't be able to find the matching ``endif``.

You can also use ``else``. The generic form for this is:

```
if {condition}
  {statements}
else
  {statements}
```

```
endif
```

The second `{statements}` block is only executed if the first one isn't.

Finally, there is ``elseif``

```
if {condition}
  {statements}
elseif {condition}
  {statements}
endif
```

This works just like using ``else`` and then ``if``, but without the need for an extra ``endif``.

A useful example for your vimrc file is checking the `'term'` option and doing something depending upon its value:

```
if &term == "xterm"
  # Do stuff for xterm
elseif &term == "vt100"
  # Do stuff for a vt100 terminal
else
  # Do something for other terminals
endif
```

This uses `"#"` to start a comment, more about that later.

LOGIC OPERATIONS

We already used some of them in the examples. These are the most often used ones:

<code>a == b</code>	equal to
<code>a != b</code>	not equal to
<code>a > b</code>	greater than
<code>a >= b</code>	greater than or equal to
<code>a < b</code>	less than
<code>a <= b</code>	less than or equal to

The result is true if the condition is met and false otherwise. An example:

```
if v:version >= 800
  echo "congratulations"
else
  echo "you are using an old version, upgrade!"
endif
```

Here `"v:version"` is a variable defined by Vim, which has the value of the Vim version. 800 is for version 8.0, version 8.1 has the value 801. This is useful to write a script that works with multiple versions of Vim.

See `v:version`. You can also check for a specific feature with ``has()`` or a specific patch, see `has-patch`.

The logic operators work both for numbers and strings. When comparing two strings, the mathematical difference is used. This compares byte values, which may not be right for some languages.

If you try to compare a string with a number you will get an error.

For strings there are two more useful items:

<code>str =~ pat</code>	matches with
<code>str !~ pat</code>	does not match with

The left item "str" is used as a string. The right item "pat" is used as a pattern, like what's used for searching. Example:

```
if str =~ " "  
  echo "str contains a space"  
endif  
if str !~ '\.$'  
  echo "str does not end in a full stop"  
endif
```

Notice the use of a single-quote string for the pattern. This is useful, because patterns tend to contain many backslashes and backslashes need to be doubled in a double-quote string.

The match is not anchored, if you want to match the whole string start with "^" and end with "\$".

The **'ignorecase'** option is not used when comparing strings. When you do want to ignore case append "?". Thus "==" compares two strings to be equal while ignoring case. For the full table see [expr==](#) .

MORE LOOPING

The ``while`` command was already mentioned. Two more statements can be used in between the ``while`` and the ``endwhile``:

<code>continue</code>	Jump back to the start of the while loop; the loop continues.
<code>break</code>	Jump forward to the <code>`endwhile`</code> ; the loop is discontinued.

Example:

```
var counter = 1  
while counter < 40  
  if skip_number(counter)  
    continue  
  endif  
  if last_number(counter)  
    break  
  endif
```

```
    sleep 50m
    ++counter
endwhile
```

The ``sleep`` command makes Vim take a nap. The "50m" specifies fifty milliseconds. Another example is ``sleep 4``, which sleeps for four seconds.

``continue`` and ``break`` can also be used in between ``for`` and ``endfor``. Even more looping can be done with the ``for`` command, see below in [41.8](#).

41.5 Executing an expression

So far the commands in the script were executed by Vim directly. The ``execute`` command allows executing the result of an expression. This is a very powerful way to build commands and execute them.

An example is to jump to a tag, which is contained in a variable:

```
execute "tag " .. tag_name
```

The `.."` is used to concatenate the string "tag " with the value of variable "tag_name". Suppose "tag_name" has the value "get_cmd", then the command that will be executed is:

```
tag get_cmd
```

The ``execute`` command can only execute Ex commands. The ``normal`` command executes Normal mode commands. However, its argument is not an expression but the literal command characters. Example:

```
normal gg=G
```

This jumps to the first line with "gg" and formats all lines with the "=" operator and the "G" movement.

To make ``normal`` work with an expression, combine ``execute`` with it. Example:

```
execute "normal " .. count .. "j"
```

This will move the cursor "count" lines down.

Make sure that the argument for ``normal`` is a complete command. Otherwise Vim will run into the end of the argument and silently abort the command. For example, if you start the delete operator, you must give the movement command also. This works:

```
normal d$
```

This does nothing:

```
normal d
```


If you start Insert mode and do not end it with Esc, it will end anyway. This works to insert "new text":

```
execute "normal inew text"
```

If you want to do something after inserting text you do need to end Insert mode:

```
execute "normal inew text\<Esc>b"
```

This inserts "new text" and puts the cursor on the first letter of "text". Notice the use of the special key "\<Esc>". This avoids having to enter a real <Esc> character in your script. That is where `execute` with a double-quote string comes in handy.

If you don't want to execute a string as a command but evaluate it to get the result of the expression, you can use the eval() function:

```
var optname = "path"  
var optvalue = eval('&' .. optname)
```

A "&" character is prepended to "path", thus the argument to eval() is "&path". The result will then be the value of the **'path'** option.

41.6 Using functions

Vim defines many functions and provides a large amount of functionality that way. A few examples will be given in this section. You can find the whole list below: [function-list](#).

A function is called with the parameters in between parentheses, separated by commas. Example:

```
search("Date: ", "W")
```

This calls the search() function, with arguments "Date: " and "W". The search() function uses its first argument as a search pattern and the second one as flags. The "W" flag means the search doesn't wrap around the end of the file.

Using the `call` command is optional in Vim9 script. It is required in legacy script and on the command line:

```
call search("Date: ", "W")
```

A function can be called in an expression. Example:

```
var line = getline(".")  
var repl = substitute(line, '\a', "*", "g")  
setline(".", repl)
```

The getline() function obtains a line from the current buffer. Its argument is a specification of the line number. In this case "." is used, which means

the line where the cursor is.

The `substitute()` function does something similar to the `:substitute` command. The first argument "line" is the string on which to perform the substitution. The second argument '\a' is the pattern, the third "*" is the replacement string. Finally, the last argument "g" is the flags.

The `setline()` function sets the line, specified by the first argument, to a new string, the second argument. In this example the line under the cursor is replaced with the result of the `substitute()`. Thus the effect of the three statements is equal to:

```
:substitute/\a*/g
```

Using the functions becomes interesting when you do more work before and after the `substitute()` call.

FUNCTIONS

function-list

There are many functions. We will mention them here, grouped by what they are used for. You can find an alphabetical list here: [builtin-function-list](#). Use **CTRL-]** on the function name to jump to detailed help on it.

String manipulation:

string-functions

<code>nr2char()</code>	get a character by its number value
<code>list2str()</code>	get a character string from a list of numbers
<code>char2nr()</code>	get number value of a character
<code>str2list()</code>	get list of numbers from a string
<code>str2nr()</code>	convert a string to a Number
<code>str2float()</code>	convert a string to a Float
<code>printf()</code>	format a string according to % items
<code>escape()</code>	escape characters in a string with a '\'
<code>shellescape()</code>	escape a string for use with a shell command
<code>fnameescape()</code>	escape a file name for use with a Vim command
<code>tr()</code>	translate characters from one set to another
<code>strtrans()</code>	translate a string to make it printable
<code>keytrans()</code>	translate internal keycodes to a form that can be used by <code>:map</code>
<code>tolower()</code>	turn a string to lowercase
<code>toupper()</code>	turn a string to uppercase
<code>charclass()</code>	class of a character
<code>match()</code>	position where a pattern matches in a string
<code>matchbufline()</code>	all the matches of a pattern in a buffer
<code>matchend()</code>	position where a pattern match ends in a string
<code>matchfuzzy()</code>	fuzzy matches a string in a list of strings
<code>matchfuzzypos()</code>	fuzzy matches a string in a list of strings
<code>matchstr()</code>	match of a pattern in a string
<code>matchstrlist()</code>	all the matches of a pattern in a List of strings
<code>matchstrpos()</code>	match and positions of a pattern in a string
<code>matchlist()</code>	like <code>matchstr()</code> and also return submatches
<code>stridx()</code>	first index of a short string in a long string
<code>strridx()</code>	last index of a short string in a long string

<code>strlen()</code>	length of a string in bytes
<code>strcharlen()</code>	length of a string in characters
<code>strchars()</code>	number of characters in a string
<code>strutf16len()</code>	number of UTF-16 code units in a string
<code>strwidth()</code>	size of string when displayed
<code>strdisplaywidth()</code>	size of string when displayed, deals with tabs
<code>setcellwidths()</code>	set character cell width overrides
<code>getcellwidths()</code>	get character cell width overrides
<code>reverse()</code>	reverse the order of characters in a string
<code>substitute()</code>	substitute a pattern match with a string
<code>submatch()</code>	get a specific match in ":s" and substitute()
<code>strpart()</code>	get part of a string using byte index
<code>strcharpart()</code>	get part of a string using char index
<code>slice()</code>	take a slice of a string, using char index in Vim9 script
<code>strgetchar()</code>	get character from a string using char index
<code>expand()</code>	expand special keywords
<code>expandcmd()</code>	expand a command like done for <code>:edit</code>
<code>iconv()</code>	convert text from one encoding to another
<code>byteidx()</code>	byte index of a character in a string
<code>byteidxcomp()</code>	like <code>byteidx()</code> but count composing characters
<code>charidx()</code>	character index of a byte in a string
<code>utf16idx()</code>	UTF-16 index of a byte in a string
<code>repeat()</code>	repeat a string multiple times
<code>eval()</code>	evaluate a string expression
<code>execute()</code>	execute an Ex command and get the output
<code>win_execute()</code>	like <code>execute()</code> but in a specified window
<code>trim()</code>	trim characters from a string
<code>gettext()</code>	lookup message translation

List manipulation:

	list-functions
<code>get()</code>	get an item without error for wrong index
<code>len()</code>	number of items in a List
<code>empty()</code>	check if List is empty
<code>insert()</code>	insert an item somewhere in a List
<code>add()</code>	append an item to a List
<code>extend()</code>	append a List to a List
<code>extendnew()</code>	make a new List and append items
<code>remove()</code>	remove one or more items from a List
<code>copy()</code>	make a shallow copy of a List
<code>deepcopy()</code>	make a full copy of a List
<code>filter()</code>	remove selected items from a List
<code>map()</code>	change each List item
<code>mapnew()</code>	make a new List with changed items
<code>foreach()</code>	apply function to List items
<code>reduce()</code>	reduce a List to a value
<code>slice()</code>	take a slice of a List
<code>sort()</code>	sort a List
<code>reverse()</code>	reverse the order of items in a List
<code>uniq()</code>	remove copies of repeated adjacent items
<code>split()</code>	split a String into a List
<code>join()</code>	join List items into a String
<code>range()</code>	return a List with a sequence of numbers
<code>string()</code>	String representation of a List

call()	call a function with List as arguments
index()	index of a value in a List or Blob
indexof()	index in a List or Blob where an expression evaluates to true
max()	maximum value in a List
min()	minimum value in a List
count()	count number of times a value appears in a List
repeat()	repeat a List multiple times
flatten()	flatten a List
flattennew()	flatten a copy of a List

Dictionary manipulation:

dict-functions

get()	get an entry without an error for a wrong key
len()	number of entries in a Dictionary
has_key()	check whether a key appears in a Dictionary
empty()	check if Dictionary is empty
remove()	remove an entry from a Dictionary
extend()	add entries from one Dictionary to another
extendnew()	make a new Dictionary and append items
filter()	remove selected entries from a Dictionary
map()	change each Dictionary entry
mapnew()	make a new Dictionary with changed items
foreach()	apply function to Dictionary items
keys()	get List of Dictionary keys
values()	get List of Dictionary values
items()	get List of Dictionary key-value pairs
copy()	make a shallow copy of a Dictionary
deepcopy()	make a full copy of a Dictionary
string()	String representation of a Dictionary
max()	maximum value in a Dictionary
min()	minimum value in a Dictionary
count()	count number of times a value appears

Floating point computation:

float-functions

float2nr()	convert Float to Number
abs()	absolute value (also works for Number)
round()	round off
ceil()	round up
floor()	round down
trunc()	remove value after decimal point
fmod()	remainder of division
exp()	exponential
log()	natural logarithm (logarithm to base e)
log10()	logarithm to base 10
pow()	value of x to the exponent y
sqrt()	square root
sin()	sine
cos()	cosine
tan()	tangent
asin()	arc sine
acos()	arc cosine
atan()	arc tangent
atan2()	arc tangent
sinh()	hyperbolic sine

cosh()	hyperbolic cosine
tanh()	hyperbolic tangent
isinf()	check for infinity
isnan()	check for not a number
Blob manipulation:	blob-functions
blob2list()	get a list of numbers from a blob
list2blob()	get a blob from a list of numbers
reverse()	reverse the order of numbers in a blob
Other computation:	bitwise-function
and()	bitwise AND
invert()	bitwise invert
or()	bitwise OR
xor()	bitwise XOR
sha256()	SHA-256 hash
rand()	get a pseudo-random number
srand()	initialize seed used by rand()
Variables:	var-functions
instanceof()	check if a variable is an instance of a given class
type()	type of a variable as a number
typename()	type of a variable as text
islocked()	check if a variable is locked
funcref()	get a Funcref for a function reference
function()	get a Funcref for a function name
getbufvar()	get a variable value from a specific buffer
setbufvar()	set a variable in a specific buffer
getwinvar()	get a variable from specific window
gettabvar()	get a variable from specific tab page
gettabwinvar()	get a variable from specific window & tab page
setwinvar()	set a variable in a specific window
settabvar()	set a variable in a specific tab page
settabwinvar()	set a variable in a specific window & tab page
garbagecollect()	possibly free memory
Cursor and mark position:	cursor-functions mark-functions
col()	column number of the cursor or a mark
virtcol()	screen column of the cursor or a mark
line()	line number of the cursor or mark
wincol()	window column number of the cursor
winline()	window line number of the cursor
cursor()	position the cursor at a line/column
screencol()	get screen column of the cursor
screenrow()	get screen row of the cursor
screenpos()	screen row and col of a text character
virtcol2col()	byte index of a text character on screen
getcurpos()	get position of the cursor
getpos()	get position of cursor, mark, etc.
setpos()	set position of cursor, mark, etc.
getmarklist()	list of global/local marks
byte2line()	get line number at a specific byte count
line2byte()	byte count at a specific line

diff_filler()	get the number of filler lines above a line
screenattr()	get attribute at a screen line/row
screenchar()	get character code at a screen line/row
screenchars()	get character codes at a screen line/row
screenstring()	get string of characters at a screen line/row
charcol()	character number of the cursor or a mark
getcharpos()	get character position of cursor, mark, etc.
setcharpos()	set character position of cursor, mark, etc.
getcursorcharpos()	get character position of the cursor
setcursorcharpos()	set character position of the cursor

Working with text in the current buffer:

text-functions

getline()	get a line or list of lines from the buffer
getregion()	get a region of text from the buffer
setline()	replace a line in the buffer
append()	append line or list of lines in the buffer
indent()	indent of a specific line
cindent()	indent according to C indenting
lispindent()	indent according to Lisp indenting
nextnonblank()	find next non-blank line
prevnonblank()	find previous non-blank line
search()	find a match for a pattern
searchpos()	find a match for a pattern
searchcount()	get number of matches before/after the cursor
searchpair()	find the other end of a start/skip/end
searchpairpos()	find the other end of a start/skip/end
searchdecl()	search for the declaration of a name
getcharsearch()	return character search information
setcharsearch()	set character search information

Working with text in another buffer:

getbufline()	get a list of lines from the specified buffer
getbufoneline()	get a one line from the specified buffer
setbufline()	replace a line in the specified buffer
appendbufline()	append a list of lines in the specified buffer
deletebufline()	delete lines from a specified buffer

system-functions file-functions

System functions and manipulation of files:

glob()	expand wildcards
globpath()	expand wildcards in a number of directories
glob2regpat()	convert a glob pattern into a search pattern
findfile()	find a file in a list of directories
finddir()	find a directory in a list of directories
resolve()	find out where a shortcut points to
fnamemodify()	modify a file name
pathshorten()	shorten directory names in a path
simplify()	simplify a path without changing its meaning
executable()	check if an executable program exists
exepath()	full path of an executable program
filereadable()	check if a file can be read
filewritable()	check if a file can be written to
getfperm()	get the permissions of a file
setfperm()	set the permissions of a file

getftype()	get the kind of a file
isabsolutepath()	check if a path is absolute
isdirectory()	check if a directory exists
getfsize()	get the size of a file
getcwd()	get the current working directory
haslocaldir()	check if current window used <code>:lcd</code> or <code>:tcd</code>
tempname()	get the name of a temporary file
mkdir()	create a new directory
chdir()	change current working directory
delete()	delete a file
rename()	rename a file
system()	get the result of a shell command as a string
systemlist()	get the result of a shell command as a list
environ()	get all environment variables
getenv()	get one environment variable
setenv()	set an environment variable
hostname()	name of the system
readfile()	read a file into a List of lines
readblob()	read a file into a Blob
readdir()	get a List of file names in a directory
readdirrex()	get a List of file information in a directory
writefile()	write a List of lines or Blob into a file

Date and Time:

date-functions time-functions

getftime()	get last modification time of a file
localtime()	get current time in seconds
strftime()	convert time to a string
strptime()	convert a date/time string to time
reltime()	get the current or elapsed time accurately
reltimestr()	convert reltime() result to a string
reltimefloat()	convert reltime() result to a Float

Autocmds:

autocmd-functions

autocmd_add()	add a list of autocmds and groups
autocmd_delete()	delete a list of autocmds and groups
autocmd_get()	return a list of autocmds

buffer-functions window-functions arg-functions

Buffers, windows and the argument list:

argc()	number of entries in the argument list
argidx()	current position in the argument list
arglistid()	get id of the argument list
argv()	get one entry from the argument list
bufadd()	add a file to the list of buffers
bufexists()	check if a buffer exists
buflisted()	check if a buffer exists and is listed
bufload()	ensure a buffer is loaded
bufloaded()	check if a buffer exists and is loaded
bufname()	get the name of a specific buffer
bufnr()	get the buffer number of a specific buffer
tabpagebuflist()	return List of buffers in a tab page
tabpagenr()	get the number of a tab page
tabpagewinnr()	like winnr() for a specified tab page
winnr()	get the window number for the current window

bufwinid()	get the window ID of a specific buffer
bufwinnr()	get the window number of a specific buffer
winbufnr()	get the buffer number of a specific window
listener_add()	add a callback to listen to changes
listener_flush()	invoke listener callbacks
listener_remove()	remove a listener callback
win_findbuf()	find windows containing a buffer
win_getid()	get window ID of a window
win_gettype()	get type of window
win_gotoid()	go to window with ID
win_id2tabwin()	get tab and window nr from window ID
win_id2win()	get window nr from window ID
win_move_separator()	move window vertical separator
win_move_statusline()	move window status line
win_splitmove()	move window to a split of another window
getbufinfo()	get a list with buffer information
gettabinfo()	get a list with tab page information
getwininfo()	get a list with window information
getchangelist()	get a list of change list entries
getjumplist()	get a list of jump list entries
swapfilelist()	list of existing swap files in 'directory'
swapinfo()	information about a swap file
swapname()	get the swap file path of a buffer
Command line:	command-line-functions
getcmdcompltype()	get the type of the current command line completion
getcmdline()	get the current command line
getcmdpos()	get position of the cursor in the command line
getcmdscreenpos()	get screen position of the cursor in the command line
setcmdline()	set the current command line
setcmdpos()	set position of the cursor in the command line
getcmdtype()	return the current command-line type
getcmdwintype()	return the current command-line window type
getcompletion()	list of command-line completion matches
fullcommand()	get full command name
Quickfix and location lists:	quickfix-functions
getqflist()	list of quickfix errors
setqflist()	modify a quickfix list
getloclist()	list of location list items
setloclist()	modify a location list
Insert mode completion:	completion-functions
complete()	set found matches
complete_add()	add to found matches
complete_check()	check if completion should be aborted
complete_info()	get current completion information
pumvisible()	check if the popup menu is displayed
pum_getpos()	position and size of popup menu if visible
Folding:	folding-functions
foldclosed()	check for a closed fold at a specific line

foldclosedend()	like foldclosed() but return the last line
foldlevel()	check for the fold level at a specific line
foldtext()	generate the line displayed for a closed fold
foldtextresult()	get the text displayed for a closed fold
Syntax and highlighting:	syntax-functions highlighting-functions
clearmatches()	clear all matches defined by <code>matchadd()</code> and the <code>:match</code> commands
getmatches()	get all matches defined by <code>matchadd()</code> and the <code>:match</code> commands
hlexists()	check if a highlight group exists
hlget()	get highlight group attributes
hlset()	set highlight group attributes
hlID()	get ID of a highlight group
synID()	get syntax ID at a specific position
synIDattr()	get a specific attribute of a syntax ID
synIDtrans()	get translated syntax ID
synstack()	get list of syntax IDs at a specific position
synconcealed()	get info about concealing
diff_hlID()	get highlight ID for diff mode at a position
matchadd()	define a pattern to highlight (a "match")
matchaddpos()	define a list of positions to highlight
matcharg()	get info about <code>:match</code> arguments
matchdelete()	delete a match defined by <code>matchadd()</code> or a <code>:match</code> command
setmatches()	restore a list of matches saved by <code>getmatches()</code>
Spelling:	spell-functions
spellbadword()	locate badly spelled word at or after cursor
spellsuggest()	return suggested spelling corrections
soundfold()	return the sound-a-like equivalent of a word
History:	history-functions
histadd()	add an item to a history
histdel()	delete an item from a history
histget()	get an item from a history
histnr()	get highest index of a history list
Interactive:	interactive-functions
browse()	put up a file requester
browsedir()	put up a directory requester
confirm()	let the user make a choice
getchar()	get a character from the user
getcharstr()	get a character from the user as a string
getcharmod()	get modifiers for the last typed character
getmousepos()	get last known mouse position
getmoushape()	get name of the current mouse shape
echoraw()	output characters as-is
feedkeys()	put characters in the typeahead queue
input()	get a line from the user
inputlist()	let the user pick an entry from a list
inputsecret()	get a line from the user without showing it
inputdialog()	get a line from the user in a dialog

inputsave()	save and clear typeahead
inputrestore()	restore typeahead
GUI:	gui-functions
getfontname()	get name of current font being used
getwinpos()	position of the Vim window
getwinposx()	X position of the Vim window
getwinposy()	Y position of the Vim window
balloon_show()	set the balloon content
balloon_split()	split a message for a balloon
balloon_gettext()	get the text in the balloon
Vim server:	server-functions
serverlist()	return the list of server names
remote_startserver()	run a server
remote_send()	send command characters to a Vim server
remote_expr()	evaluate an expression in a Vim server
server2client()	send a reply to a client of a Vim server
remote_peek()	check if there is a reply from a Vim server
remote_read()	read a reply from a Vim server
foreground()	move the Vim window to the foreground
remote_foreground()	move the Vim server window to the foreground
Window size and position:	window-size-functions
winheight()	get height of a specific window
winwidth()	get width of a specific window
win_screenpos()	get screen position of a window
winlayout()	get layout of windows in a tab page
winrestcmd()	return command to restore window sizes
winsaveview()	get view of current window
winrestview()	restore saved view of current window
Mappings and Menus:	mapping-functions
digraph_get()	get digraph
digraph_getlist()	get all digraph s
digraph_set()	register digraph
digraph_setlist()	register multiple digraph s
hasmapto()	check if a mapping exists
mapcheck()	check if a matching mapping exists
maparg()	get rhs of a mapping
maplist()	get list of all mappings
mapset()	restore a mapping
menu_info()	get information about a menu item
wildmenumode()	check if the wildmode is active
Testing:	test-functions
assert_equal()	assert that two expressions values are equal
assert_equalfile()	assert that two file contents are equal
assert_notequal()	assert that two expressions values are not equal
assert_inrange()	assert that an expression is inside a range
assert_match()	assert that a pattern matches the value
assert_notmatch()	assert that a pattern does not match the value
assert_false()	assert that an expression is false
assert_true()	assert that an expression is true

assert_exception()	assert that a command throws an exception
assert_beeps()	assert that a command beeps
assert_nobeep()	assert that a command does not cause a beep
assert_fails()	assert that a command fails
assert_report()	report a test failure
test_alloc_fail()	make memory allocation fail
test_autochdir()	enable 'autochdir' during startup
test_override()	test with Vim internal overrides
test_garbagecollect_now()	free memory right now
test_garbagecollect_soon()	set a flag to free memory soon
test_getvalue()	get value of an internal variable
test_gui_event()	generate a GUI event for testing
test_ignore_error()	ignore a specific error message
test_mswin_event()	generate an MS-Windows event
test_null_blob()	return a null Blob
test_null_channel()	return a null Channel
test_null_dict()	return a null Dict
test_null_function()	return a null Funcref
test_null_job()	return a null Job
test_null_list()	return a null List
test_null_partial()	return a null Partial function
test_null_string()	return a null String
test_settime()	set the time Vim uses internally
test_setmouse()	set the mouse position
test_feedinput()	add key sequence to input buffer
test_option_not_set()	reset flag indicating option was set
test_refcount()	return an expression's reference count
test_srand_seed()	set the seed value for srand()
test_unknown()	return a value with unknown type
test_void()	return a value with void type

Inter-process communication:

channel-functions

ch_canread()	check if there is something to read
ch_open()	open a channel
ch_close()	close a channel
ch_close_in()	close the in part of a channel
ch_read()	read a message from a channel
ch_readblob()	read a Blob from a channel
ch_readraw()	read a raw message from a channel
ch_sendexpr()	send a JSON message over a channel
ch_sendraw()	send a raw message over a channel
ch_evalexpr()	evaluate an expression over channel
ch_evalraw()	evaluate a raw string over channel
ch_status()	get status of a channel
ch_getbufnr()	get the buffer number of a channel
ch_getjob()	get the job associated with a channel
ch_info()	get channel information
ch_log()	write a message in the channel log file
ch_logfile()	set the channel log file
ch_setoptions()	set the options for a channel
json_encode()	encode an expression to a JSON string
json_decode()	decode a JSON string to Vim types
js_encode()	encode an expression to a JSON string
js_decode()	decode a JSON string to Vim types

err_teapot()	give error 418 or 503
Jobs:	job-functions
job_start()	start a job
job_stop()	stop a job
job_status()	get the status of a job
job_getchannel()	get the channel used by a job
job_info()	get information about a job
job_setoptions()	set options for a job
Signs:	sign-functions
sign_define()	define or update a sign
sign_getdefined()	get a list of defined signs
sign_getplaced()	get a list of placed signs
sign_jump()	jump to a sign
sign_place()	place a sign
sign_placelist()	place a list of signs
sign_undefine()	undefine a sign
sign_unplace()	unplace a sign
sign_unplacelist()	unplace a list of signs
Terminal window:	terminal-functions
term_start()	open a terminal window and run a job
term_list()	get the list of terminal buffers
term_sendkeys()	send keystrokes to a terminal
term_wait()	wait for screen to be updated
term_getjob()	get the job associated with a terminal
term_scrape()	get row of a terminal screen
term_getline()	get a line of text from a terminal
term_getattr()	get the value of attribute <code>{what}</code>
term_getcursor()	get the cursor position of a terminal
term_getscrolled()	get the scroll count of a terminal
term_getaltscreen()	get the alternate screen flag
term_getsize()	get the size of a terminal
term_getstatus()	get the status of a terminal
term_gettitle()	get the title of a terminal
term_gettty()	get the tty name of a terminal
term_setansicolors()	set 16 ANSI colors, used for GUI
term_getansicolors()	get 16 ANSI colors, used for GUI
term_dumpdiff()	display difference between two screen dumps
term_dumpload()	load a terminal screen dump in a window
term_dumpwrite()	dump contents of a terminal screen to a file
term_setkill()	set signal to stop job in a terminal
term_setrestore()	set command to restore a terminal
term_setsize()	set the size of a terminal
term_setapi()	set terminal JSON API function name prefix
Popup window:	popup-window-functions
popup_create()	create popup centered in the screen
popup_atcursor()	create popup just above the cursor position, closes when the cursor moves away
popup_beval()	at the position indicated by <code>v:beval_</code> variables, closes when the mouse moves away
popup_notification()	show a notification for three seconds

popup_dialog()	create popup centered with padding and border
popup_menu()	prompt for selecting an item from a list
popup_hide()	hide a popup temporarily
popup_show()	show a previously hidden popup
popup_move()	change the position and size of a popup
popup_setopts()	override options of a popup
popup_settext()	replace the popup buffer contents
popup_close()	close one popup
popup_clear()	close all popups
popup_filter_menu()	select from a list of items
popup_filter_yesno()	block until 'y' or 'n' is pressed
popup_getoptions()	get current options for a popup
popup_getpos()	get actual position and size of a popup
popup_findecho()	get window ID for popup used for <code>:echowindow</code>
popup_findinfo()	get window ID for popup info window
popup_findpreview()	get window ID for popup preview window
popup_list()	get list of all popup window IDs
popup_locate()	get popup window ID from its screen position
Timers:	timer-functions
timer_start()	create a timer
timer_pause()	pause or unpause a timer
timer_stop()	stop a timer
timer_stopall()	stop all timers
timer_info()	get information about timers
Tags:	tag-functions
taglist()	get list of matching tags
tagfiles()	get a list of tags files
gettagstack()	get the tag stack of a window
settagstack()	modify the tag stack of a window
Prompt Buffer:	promptbuffer-functions
prompt_getprompt()	get the effective prompt text for a buffer
prompt_setcallback()	set prompt callback for a buffer
prompt_setinterrupt()	set interrupt callback for a buffer
prompt_setprompt()	set the prompt text for a buffer
Registers:	register-functions
getreg()	get contents of a register
getreginfo()	get information about a register
getregtype()	get type of a register
setreg()	set contents and type of a register
reg_executing()	return the name of the register being executed
reg_recording()	return the name of the register being recorded
Text Properties:	text-property-functions
prop_add()	attach a property at a position
prop_add_list()	attach a property at multiple positions
prop_clear()	remove all properties from a line or lines
prop_find()	search for a property
prop_list()	return a list of all properties in a line
prop_remove()	remove a property from a line
prop_type_add()	add/define a property type

prop_type_change()	change properties of a type
prop_type_delete()	remove a text property type
prop_type_get()	return the properties of a type
prop_type_list()	return a list of all property types
Sound:	sound-functions
sound_clear()	stop playing all sounds
sound_playevent()	play an event's sound
sound_playfile()	play a sound file
sound_stop()	stop playing a sound
Various:	various-functions
mode()	get current editing mode
state()	get current busy state
visualmode()	last visual mode used
exists()	check if a variable, function, etc. exists
exists_compiled()	like exists() but check at compile time
has()	check if a feature is supported in Vim
changenr()	return number of most recent change
cscope_connection()	check if a cscope connection exists
did_filetype()	check if a FileType autocommand was used
diff()	diff two Lists of strings
eventhandler()	check if invoked by an event handler
getpid()	get process ID of Vim
getscriptinfo()	get list of sourced vim scripts
getimstatus()	check if IME status is active
interrupt()	interrupt script execution
windowsversion()	get MS-Windows version
terminalprops()	properties of the terminal
libcall()	call a function in an external library
libcallnr()	idem, returning a number
undofile()	get the name of the undo file
undotree()	return the state of the undo tree for a buffer
shiftwidth()	effective value of 'shiftwidth'
wordcount()	get byte/word/char count of buffer
luaeval()	evaluate Lua expression
mzeval()	evaluate MzScheme expression
perleval()	evaluate Perl expression (+perl)
py3eval()	evaluate Python expression (+python3)
pyeval()	evaluate Python expression (+python)
pyxeval()	evaluate python_x expression
rubyeval()	evaluate Ruby expression
debugbreak()	interrupt a program being debugged

41.7 Defining a function

Vim enables you to define your own functions. The basic function declaration

begins as follows:

```
def {name}({var1}, {var2}, ...): return-type
  {body}
enddef
```

Note:

Function names must begin with a capital letter.

Let's define a short function to return the smaller of two numbers. It starts with this line:

```
def Min(num1: number, num2: number): number
```

This tells Vim that the function is named "Min", it takes two arguments that are numbers: "num1" and "num2" and returns a number.

The first thing you need to do is to check to see which number is smaller:

```
if num1 < num2
```

Let's assign the variable "smaller" the value of the smallest number:

```
var smaller: number
if num1 < num2
  smaller = num1
else
  smaller = num2
endif
```

The variable "smaller" is a local variable. It is declared to be a number, that way Vim can warn you for any mistakes. Variables used inside a function are local unless prefixed by something like "g:", "w:", or "b:".

Note:

To access a global variable from inside a function you must prepend "g:" to it. Thus "g:today" inside a function is used for the global variable "today", and "today" is another variable, local to the function or the script.

You now use the ``return`` statement to return the smallest number to the user. Finally, you end the function:

```
return smaller
enddef
```

The complete function definition is as follows:

```
def Min(num1: number, num2: number): number
  var smaller: number
  if num1 < num2
    smaller = num1
  else
    smaller = num2
  endif
enddef
```

```
endif
return smaller
enddef
```

Obviously this is a verbose example. You can make it shorter by using two return commands:

```
def Min(num1: number, num2: number): number
  if num1 < num2
    return num1
  endif
  return num2
enddef
```

And if you remember the conditional expression, you need only one line:

```
def Min(num1: number, num2: number): number
  return num1 < num2 ? num1 : num2
enddef
```

A user defined function is called in exactly the same way as a built-in function. Only the name is different. The Min function can be used like this:

```
echo Min(5, 8)
```

Only now will the function be executed and the lines be parsed by Vim. If there are mistakes, like using an undefined variable or function, you will now get an error message. When defining the function these errors are not detected. To get the errors sooner you can tell Vim to compile all the functions in the script:

```
defcompile
```

Compiling functions takes a little time, but does report errors early. You could use `:defcompile` at the end of your script while working on it, and comment it out when everything is fine.

For a function that does not return anything simply leave out the return type:

```
def SayIt(text: string)
  echo text
enddef
```

If you want to return any kind of value, you can use the "any" return type:

```
def GetValue(): any
```

This disables type checking for the return value, use only when needed.

It is also possible to define a legacy function with `function` and `endfunction`. These do not have types and are not compiled. Therefore they execute much slower.

USING A RANGE

A line range can be used with a function call. The function will be called once for every line in the range, with the cursor in that line. Example:

```
def Number()  
  echo "line " .. line(".") .. " contains: " .. getline(".")  
enddef
```

If you call this function with:

```
:10,15Number()
```

The function will be called six times, starting on line 10 and ending on line 15.

LISTING FUNCTIONS

The ``function`` command lists the names and arguments of all user-defined functions:

```
:function  
def <SNR>86_Show(start: string, ...items: list<string>)  
  function GetVimIndent()  
  function SetSyn(name)
```

The "`<SNR>`" prefix means that a function is script-local. `Vim9` functions will start with "def" and include argument and return types. Legacy functions are listed with "function".

To see what a function does, use its name as an argument for ``function``:

```
:function SetSyn  
1   if &syntax == ''  
2     let &syntax = a:name  
3   endif  
  endfunction
```

To see the "Show" function you need to include the script prefix, since multiple "Show" functions can be defined in different scripts. To find the exact name you can use ``function``, but the result may be a very long list. To only get the functions matching a pattern you can use the ``filter`` prefix:

```
:filter Show function  
def <SNR>86_Show(start: string, ...items: list<string>)  
  
:function <SNR>86_Show  
1   echohl Title  
2   echo "start is " .. start  
  etc.
```

DEBUGGING

The line number is useful for when you get an error message or when debugging. See [debug-scripts](#) about debugging mode.

You can also set the **'verbose'** option to 12 or higher to see all function calls. Set it to 15 or higher to see every executed line.

DELETING A FUNCTION

To delete the `SetSyn()` function:

```
:delfunction SetSyn
```

Deleting only works for global functions and functions in legacy script, not for functions defined in a `Vim9` script.

You get an error when the function doesn't exist or cannot be deleted.

FUNCTION REFERENCES

Sometimes it can be useful to have a variable point to one function or another. You can do it with a function reference variable. Often shortened to "funcref". Example:

```
def Right(): string
    return 'Right!'
enddef
def Wrong(): string
    return 'Wrong!'
enddef

var Afunc = g:result == 1 ? Right : Wrong
echo Afunc()
Wrong!
```

This assumes "g:result" is not one. See [Funcref](#) for details.

Note that the name of a variable that holds a function reference must start with a capital. Otherwise it could be confused with the name of a builtin function.

FURTHER READING

Using a variable number of arguments is introduced in section [50.2](#) .

More information about defining your own functions here: [user-functions](#) .

=====

41.8 Lists and Dictionaries

So far we have used the basic types `String` and `Number`. Vim also supports two composite types: `List` and `Dictionary`.

A List is an ordered sequence of items. The items can be any kind of value, thus you can make a List of numbers, a List of Lists and even a List of mixed items. To create a List with three strings:

```
var alist = ['aap', 'noot', 'mies']
```

The List items are enclosed in square brackets and separated by commas. To create an empty List:

```
var alist = []
```

You can add items to a List with the `add()` function:

```
var alist = []
add(alist, 'foo')
add(alist, 'bar')
echo alist
['foo', 'bar']
```

List concatenation is done with `+`:

```
var alist = ['foo', 'bar']
alist = alist + ['and', 'more']
echo alist
['foo', 'bar', 'and', 'more']
```

Or, if you want to extend a List with a function, use `extend()`:

```
var alist = ['one']
extend(alist, ['two', 'three'])
echo alist
['one', 'two', 'three']
```

Notice that using `add()` will have a different effect than `extend()`:

```
var alist = ['one']
add(alist, ['two', 'three'])
echo alist
['one', ['two', 'three']]
```

The second argument of `add()` is added as an item, now you have a nested list.

FOR LOOP

One of the nice things you can do with a List is iterate over it:

```
var alist = ['one', 'two', 'three']
for n in alist
  echo n
endfor
one
two
```

three

This will loop over each element in List "alist", assigning each value to variable "n". The generic form of a for loop is:

```
for {varname} in {list-expression}
  {commands}
endfor
```

To loop a certain number of times you need a List of a specific length. The range() function creates one for you:

```
for a in range(3)
  echo a
endfor
0
1
2
```

Notice that the first item of the List that range() produces is zero, thus the last item is one less than the length of the list. Detail: Internally range() does not actually create the list, so that a large range used in a for loop works efficiently. When used elsewhere, the range is turned into an actual list, which takes more time for a long list.

You can also specify the maximum value, the stride and even go backwards:

```
for a in range(8, 4, -2)
  echo a
endfor
8
6
4
```

A more useful example, looping over all the lines in the buffer:

```
for line in getline(1, 50)
  if line =~ "Date: "
    echo line
  endif
endfor
```

This looks into lines 1 to 50 (inclusive) and echoes any date found in there.

For further reading see [Lists](#) .

DICTIONARIES

A Dictionary stores key-value pairs. You can quickly lookup a value if you know the key. A Dictionary is created with curly braces:

```
var uk2nl = {one: 'een', two: 'twee', three: 'drie'}
```

Now you can lookup words by putting the key in square brackets:

```
echo uk2nl['two']
twee
```

If the key does not have special characters, you can use the dot notation:

```
echo uk2nl.two
twee
```

The generic form for defining a Dictionary is:

```
{<key> : <value>, ...}
```

An empty Dictionary is one without any keys:

```
{}
```

The possibilities with Dictionaries are numerous. There are various functions for them as well. For example, you can obtain a list of the keys and loop over them:

```
for key in keys(uk2nl)
  echo key
endfor
three
one
two
```

You will notice the keys are not ordered. You can sort the list to get a specific order:

```
for key in sort(keys(uk2nl))
  echo key
endfor
one
three
two
```

But you can never get back the order in which items are defined. For that you need to use a List, it stores items in an ordered sequence.

For further reading see [Dictionaries](#) .

41.9 White space

Blank lines are allowed in a script and ignored.

Leading whitespace characters (blanks and TABs) are ignored, except when using `:let-heredoc` without "trim".

Trailing whitespace is often ignored, but not always. One command that includes it is ``map``. You have to watch out for that, it can cause hard to

understand mistakes. A generic solution is to never use trailing white space, unless you really need it.

To include a whitespace character in the value of an option, it must be escaped by a "\" (backslash) as in the following example:

```
:set tags=my\ nice\ file
```

If it would be written as:

```
:set tags=my nice file
```

This will issue an error, because it is interpreted as:

```
:set tags=my
:set nice
:set file
```

Vim9 script is very picky when it comes to white space. This was done intentionally to make sure scripts are easy to read and to avoid mistakes. If you use white space sensibly it will just work. When not you will get an error message telling you where white space is missing or should be removed.

41.10 Line continuation

In legacy Vim script line continuation is done by preceding a continuation line with a backslash:

```
let mylist = [
    \ 'one',
    \ 'two',
    \ ]
```

This requires the '**cpo**' option to exclude the "C" flag. Normally this is done by putting this at the start of the script:

```
let s:save_cpo = &cpo
set cpo&vim
```

And restore the option at the end of the script:

```
let &cpo = s:save_cpo
unlet s:save_cpo
```

A few more details can be found here: [line-continuation](#) .

In Vim9 script the backslash can still be used, but in most places it is not needed:

```
var mylist = [
    'one',
    'two',
    ]
```

Also, the '**cpo**' option does not need to be changed. See [vim9-line-continuation](#) for details.

41.11 Comments

In `Vim9` script the character `#` starts a comment. That character and everything after it until the end-of-line is considered a comment and is ignored, except for commands that don't consider comments, as shown in examples below. A comment can start on any character position on the line, but not when it is part of the command, e.g. inside a string.

The character `"` (the double quote mark) starts a comment in legacy script. This involves some cleverness to make sure double quoted strings are not recognized as comments (just one reason to prefer `Vim9` script).

There is a little "catch" with comments for some commands. Examples:

```
abbrev dev development # shorthand
map <F3> o#include      # insert include
execute cmd             # do it
!ls *.c                # list C files
```

- The abbreviation `'dev'` will be expanded to `'development # shorthand'`.
- The mapping of `<F3>` will actually be the whole line after the `'o# ...'` including the `'# insert include'`.
- The `'execute'` command will give an error.
- The `'!'` command will send everything after it to the shell, most likely causing an error.

There can be no comment after `'map'`, `'abbreviate'`, `'execute'` and `'!'` commands (there are a few more commands with this restriction). For the `'map'`, `'abbreviate'` and `'execute'` commands there is a trick:

```
abbrev dev development|# shorthand
map <F3> o#include|# insert include
execute '!ls *.c'      |# do it
```

With the `'|'` character the command is separated from the next one. And that next command is only a comment. The last command, using `'execute'` is a general solution, it works for all commands that do not accept a comment or a `'|'` to separate the next command.

Notice that there is no white space before the `'|'` in the abbreviation and mapping. For these commands, any character until the end-of-line or `'|'` is included. As a consequence of this behavior, you don't always see that trailing whitespace is included:

```
map <F4> o#include
```

Here it is intended, in other cases it might be accidental. To spot these problems, you can highlight trailing spaces:

```
match Search /\s\+$/
```

For Unix there is one special way to comment a line, that allows making a Vim script executable, and it also works in legacy script:

```
#!/usr/bin/env vim -S
```

```
echo "this is a Vim script"
quit
```

41.12 Fileformat

The end-of-line character depends on the system. For Vim scripts it is recommended to always use the Unix fileformat. Lines are then separated with the Newline character. This also works on any other system. That way you can copy your Vim scripts from MS-Windows to Unix and they still work. See [:source_crnl](#) . To be sure it is set right, do this before writing the file:

```
:setlocal fileformat=unix
```

When using "dos" fileformat, lines are separated with CR-NL, two characters. The CR character causes various problems, better avoid this.

Advance information about writing Vim script is in [usr_50.txt](#) .

Next chapter: [usr_42.txt](#) Add new menus

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_42.txt For Vim version 9.1. Last change: 2008 May 05

VIM USER MANUAL - by Bram Moolenaar

Add new menus

By now you know that Vim is very flexible. This includes the menus used in the GUI. You can define your own menu entries to make certain commands easily accessible. This is for mouse-happy users only.

42.1 Introduction
42.2 Menu commands
42.3 Various
42.4 Toolbar and popup menus

Next chapter: [usr_43.txt](#) Using filetypes
Previous chapter: [usr_41.txt](#) Write a Vim script
Table of contents: [usr_toc.txt](#)

42.1 Introduction

The menus that Vim uses are defined in the file "\$VIMRUNTIME/menu.vim". If you want to write your own menus, you might first want to look through that file.

To define a menu item, use the ":menu" command. The basic form of this command is as follows:

```
:menu {menu-item} {keys}
```

The {menu-item} describes where on the menu to put the item. A typical {menu-item} is "File.Save", which represents the item "Save" under the "File" menu. A dot is used to separate the names. Example:

```
:menu File.Save :update<CR>
```

The ":update" command writes the file when it was modified.

You can add another level: "Edit.Settings.Shiftwidth" defines a submenu "Settings" under the "Edit" menu, with an item "Shiftwidth". You could use even deeper levels. Don't use this too much, you need to move the mouse quite a bit to use such an item.

The ":menu" command is very similar to the ":map" command: the left side specifies how the item is triggered and the right hand side defines the characters that are executed. {keys} are characters, they are used just like you would have typed them. Thus in Insert mode, when {keys} is plain text, that text is inserted.

ACCELERATORS

The ampersand character (&) is used to indicate an accelerator. For instance, you can use Alt-F to select "File" and S to select "Save". (The 'winaltkeys' option may disable this though!). Therefore, the {menu-item} looks like

"&File.&Save". The accelerator characters will be underlined in the menu.

You must take care that each key is used only once in each menu. Otherwise you will not know which of the two will actually be used. Vim doesn't warn you for this.

PRIORITIES

The actual definition of the File.Save menu item is as follows:

```
:menu 10.340 &File.&Save<Tab>:w :confirm w<CR>
```

The number 10.340 is called the priority number. It is used by the editor to decide where it places the menu item. The first number (10) indicates the position on the menu bar. Lower numbered menus are positioned to the left, higher numbers to the right.

These are the priorities used for the standard menus:

```
10      20      40      50      60      70      9999
+-----+
| File  Edit  Tools  Syntax  Buffers  Window      Help  |
+-----+
```

Notice that the Help menu is given a very high number, to make it appear on the far right.

The second number (340) determines the location of the item within the pull-down menu. Lower numbers go on top, higher number on the bottom. These are the priorities in the File menu:

```
10.310  | Open...
10.320  | Split-Open...
10.325  | New
10.330  | Close
10.335  | -----
10.340  | Save
10.350  | Save As...
10.400  | -----
10.410  | Split Diff with
10.420  | Split Patched By
10.500  | -----
10.510  | Print
10.600  | -----
10.610  | Save-Exit
10.620  | Exit
+-----+
```

Notice that there is room in between the numbers. This is where you can insert your own items, if you really want to (it's often better to leave the standard menus alone and add a new menu for your own items).

When you create a submenu, you can add another ".number" to the priority. Thus each name in {menu-item} has its priority number.

SPECIAL CHARACTERS

The `{menu-item}` in this example is `"&File.&Save<Tab>:w"`. This brings up an important point: `{menu-item}` must be one word. If you want to put a dot, space or tabs in the name, you either use the `<>` notation (`<Space>` and `<Tab>`, for instance) or use the backslash (`\`) escape.

```
:menu 10.305 &File.&Do\ It\.\.\. :exit<CR>
```

In this example, the name of the menu item "Do It..." contains a space and the command is `":exit<CR>"`.

The `<Tab>` character in a menu name is used to separate the part that defines the menu name from the part that gives a hint to the user. The part after the `<Tab>` is displayed right aligned in the menu. In the File.Save menu the name used is `"&File.&Save<Tab>:w"`. Thus the menu name is "File.Save" and the hint is `":w"`.

SEPARATORS

The separator lines, used to group related menu items together, can be defined by using a name that starts and ends in a '-'. For example `"-sep-"`. When using several separators the names must be different. Otherwise the names don't matter.

The command from a separator will never be executed, but you have to define one anyway. A single colon will do. Example:

```
:amenu 20.510 Edit.-sep3- :
```

42.2 Menu commands

You can define menu items that exist for only certain modes. This works just like the variations on the `":map"` command:

<code>:menu</code>	Normal, Visual and Operator-pending mode
<code>:nmenu</code>	Normal mode
<code>:vmenu</code>	Visual mode
<code>:omenu</code>	Operator-pending mode
<code>:menu!</code>	Insert and Command-line mode
<code>:imenu</code>	Insert mode
<code>:cmenu</code>	Command-line mode
<code>:tmenu</code>	Terminal mode
<code>:amenu</code>	All modes (except for Terminal mode)

To avoid that the commands of a menu item are being mapped, use the command `":noremenu"`, `":nnoremenu"`, `":anoremenu"`, etc.

USING :AMENU

The `":amenu"` command is a bit different. It assumes that the `{keys}` you

give are to be executed in Normal mode. When Vim is in Visual or Insert mode when the menu is used, Vim first has to go back to Normal mode. `":amenu"` inserts a **CTRL-C** or **CTRL-O** for you. For example, if you use this command:

```
:amenu 90.100 Mine.Find\ Word *
```

Then the resulting menu commands will be:

```
Normal mode:          *
Visual mode:          CTRL-C *
Operator-pending mode: CTRL-C *
Insert mode:          CTRL-O *
Command-line mode:    CTRL-C *
```

When in Command-line mode the **CTRL-C** will abandon the command typed so far. In Visual and Operator-pending mode **CTRL-C** will stop the mode. The **CTRL-O** in Insert mode will execute the command and then return to Insert mode.

CTRL-O only works for one command. If you need to use two or more commands, put them in a function and call that function. Example:

```
:amenu Mine.Next\ File :call <SID>NextFile()<CR>
:function <SID>NextFile()
:  next
:  1/^Code
:endfunction
```

This menu entry goes to the next file in the argument list with `":next"`. Then it searches for the line that starts with "Code".

The `<SID>` before the function name is the script ID. This makes the function local to the current Vim script file. This avoids problems when a function with the same name is defined in another script file. See `<SID>` .

SILENT MENUS

The menu executes the `{keys}` as if you typed them. For a `":"` command this means you will see the command being echoed on the command line. If it's a long command, the hit-Enter prompt will appear. That can be very annoying!

To avoid this, make the menu silent. This is done with the `<silent>` argument. For example, take the call to `NextFile()` in the previous example. When you use this menu, you will see this on the command line:

```
:call <SNR>34_NextFile()
```

To avoid this text on the command line, insert `"<silent>"` as the first argument:

```
:amenu <silent> Mine.Next\ File :call <SID>NextFile()<CR>
```

Don't use `"<silent>"` too often. It is not needed for short commands. If you make a menu for someone else, being able to see the executed command will give him a hint about what he could have typed, instead of using the mouse.

LISTING MENUS

When a menu command is used without a `{keys}` part, it lists the already defined menus. You can specify a `{menu-item}`, or part of it, to list specific menus. Example:

```
:amenu
```

This lists all menus. That's a long list! Better specify the name of a menu to get a shorter list:

```
:amenu Edit
```

This lists only the "Edit" menu items for all modes. To list only one specific menu item for Insert mode:

```
:imenu Edit.Undo
```

Take care that you type exactly the right name. Case matters here. But the '&' for accelerators can be omitted. The `<Tab>` and what comes after it can be left out as well.

DELETING MENUS

To delete a menu, the same command is used as for listing, but with "menu" changed to "unmenu". Thus `":menu"` becomes `":unmenu"`, `":nmenu"` becomes `":nunmenu"`, etc. To delete the "Tools.Make" item for Insert mode:

```
:iunmenu Tools.Make
```

You can delete a whole menu, with all its items, by using the menu name. Example:

```
:aunmenu Syntax
```

This deletes the Syntax menu and all the items in it.

42.3 Various

You can change the appearance of the menus with flags in `'guioptions'`. In the default value they are all included, except "M". You can remove a flag with a command like:

```
:set guioptions-=m
```

m	When removed the menubar is not displayed.
M	When added the default menus are not loaded.
g	When removed the inactive menu items are not made grey but are completely removed. (Does not work on all systems.)

text that explains what a toolbar item will do. For example "Open file". It appears when the mouse pointer is on the item, without moving for a moment. This is very useful if the meaning of the picture isn't that obvious. Example:

```
:tmenu ToolBar.Make Run make in the current directory
```

Note:

Pay attention to the case used. "ToolBar" and "toolbar" are different from "ToolBar"!

To remove a tooltip, use the `:tunmenu` command.

The **'toolbar'** option can be used to display text instead of a bitmap, or both text and a bitmap. Most people use just the bitmap, since the text takes quite a bit of space.

POPUP MENU

The popup menu pops up where the mouse pointer is. On MS-Windows you activate it by clicking the right mouse button. Then you can select an item with the left mouse button. On Unix the popup menu is used by pressing and holding the right mouse button.

The popup menu only appears when the **'mousemodel'** has been set to "popup" or "popup_setpos". The difference between the two is that "popup_setpos" moves the cursor to the mouse pointer position. When clicking inside a selection, the selection will be used unmodified. When there is a selection but you click outside of it, the selection is removed.

There is a separate popup menu for each mode. Thus there are never grey items like in the normal menus.

What is the meaning of life, the universe and everything? [42](#)
Douglas Adams, the only person who knew what this question really was about is now dead, unfortunately. So now you might wonder what the meaning of death is...

=====

Next chapter: [usr_43.txt](#) Using filetypes

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Using filetypes

When you are editing a file of a certain type, for example a C program or a shell script, you often use the same option settings and mappings. You quickly get tired of manually setting these each time. This chapter explains how to do it automatically.

43.1 Plugins for a filetype

43.2 Adding a filetype

Next chapter: [usr_44.txt](#) Your own syntax highlighted
Previous chapter: [usr_42.txt](#) Add new menus
Table of contents: [usr_toc.txt](#)

43.1 Plugins for a filetype

filetype-plugin

How to start using filetype plugins has already been discussed here: [add-filetype-plugin](#) . But you probably are not satisfied with the default settings, because they have been kept minimal. Suppose that for C files you want to set the **'softtabstop'** option to 4 and define a mapping to insert a three-line comment. You do this with only two steps:

1. Create your own runtime directory. On Unix this usually is "`~/vim`". In this directory create the "ftplugin" directory:

```
mkdir ~/.vim
mkdir ~/.vim/ftplugin
```

When you are not on Unix, check the value of the **'runtimepath'** option to see where Vim will look for the "ftplugin" directory:

```
set runtimepath
```

You would normally use the first directory name (before the first comma). You might want to prepend a directory name to the **'runtimepath'** option in your `vimrc` file if you don't like the default value.

2. Create the file "`~/vim/ftplugin/c.vim`", with the contents:

```
setlocal softtabstop=4
noremap <buffer> <LocalLeader>c o/*****<CR><CR>/<Esc>
let b:undo_ftplugin = "setl softtabstop< | unmap <buffer> <LocalLeader>c"
```

Try editing a C file. You should notice that the **'softtabstop'** option is set to 4. But when you edit another file it's reset to the default zero. That is because the `":setlocal"` command was used. This sets the **'softtabstop'** option only locally to the buffer. As soon as you edit another buffer, it will be

set to the value set for that buffer. For a new buffer it will get the default value or the value from the last `":set"` command.

Likewise, the mapping for `"\c"` will disappear when editing another buffer. The `":map <buffer>"` command creates a mapping that is local to the current buffer. This works with any mapping command: `":map!"`, `":vmap"`, etc. The `<LocalLeader>` in the mapping is replaced with the value of the `"maplocalleader"` variable.

The line to set `b:undo_ftplugin` is for when the filetype is set to another value. In that case you will want to undo your preferences. The `b:undo_ftplugin` variable is executed as a command. Watch out for characters with a special meaning inside a string, such as a backslash.

You can find examples for filetype plugins in this directory:

`$VIMRUNTIME/ftplugin/`

More details about writing a filetype plugin can be found here:
[write-plugin](#) .

43.2 Adding a filetype

If you are using a type of file that is not recognized by Vim, this is how to get it recognized. You need a runtime directory of your own. See [your-runtime-dir](#) above.

Create a file `"filetype.vim"` which contains an autocommand for your filetype. (Autocommands were explained in section [40.3](#) .) Example:

```
augroup filetypedetect
au BufNewFile,BufRead *.xyz      setf xyz
augroup END
```

This will recognize all files that end in `".xyz"` as the `"xyz"` filetype. The `":augroup"` commands put this autocommand in the `"filetypedetect"` group. This allows removing all autocommands for filetype detection when doing `":filetype off"`. The `"setf"` command will set the `'filetype'` option to its argument, unless it was set already. This will make sure that `'filetype'` isn't set twice.

You can use many different patterns to match the name of your file. Directory names can also be included. See [autocmd-patterns](#) . For example, the files under `"/usr/share/scripts/"` are all `"ruby"` files, but don't have the expected file name extension. Adding this to the example above:

```
augroup filetypedetect
au BufNewFile,BufRead *.xyz      setf xyz
au BufNewFile,BufRead /usr/share/scripts/*  setf ruby
augroup END
```

However, if you now edit a file `/usr/share/scripts/README.txt`, this is not a ruby file. The danger of a pattern ending in `"*"` is that it quickly matches

too many files. To avoid trouble with this, put the filetype.vim file in another directory, one that is at the end of **'runtimepath'**. For Unix for example, you could use "~/.vim/after/filetype.vim".

You now put the detection of text files in ~/.vim/filetype.vim:

```
augroup filetypedetect
au BufNewFile,BufRead *.txt          setf text
augroup END
```

That file is found in **'runtimepath'** first. Then use this in ~/.vim/after/filetype.vim, which is found last:

```
augroup filetypedetect
au BufNewFile,BufRead /usr/share/scripts/*      setf ruby
augroup END
```

What will happen now is that Vim searches for "filetype.vim" files in each directory in **'runtimepath'**. First ~/.vim/filetype.vim is found. The autocommand to catch *.txt files is defined there. Then Vim finds the filetype.vim file in \$VIMRUNTIME, which is halfway **'runtimepath'**. Finally ~/.vim/after/filetype.vim is found and the autocommand for detecting ruby files in /usr/share/scripts is added.

When you now edit /usr/share/scripts/README.txt, the autocommands are checked in the order in which they were defined. The *.txt pattern matches, thus "setf text" is executed to set the filetype to "text". The pattern for ruby matches too, and the "setf ruby" is executed. But since **'filetype'** was already set to "text", nothing happens here.

When you edit the file /usr/share/scripts/foobar the same autocommands are checked. Only the one for ruby matches and "setf ruby" sets **'filetype'** to ruby.

RECOGNIZING BY CONTENTS

If your file cannot be recognized by its file name, you might be able to recognize it by its contents. For example, many script files start with a line like:

```
#!/bin/xyz
```

To recognize this script create a file "scripts.vim" in your runtime directory (same place where filetype.vim goes). It might look like this:

```
if did_filetype()
  finish
endif
if getline(1) =~ '^#!.*[/\\]xyz\>'
  setf xyz
endif
```

The first check with did_filetype() is to avoid that you will check the contents of files for which the filetype was already detected by the file name. That avoids wasting time on checking the file when the "setf" command won't do anything.

The `scripts.vim` file is sourced by an autocommand in the default `filetype.vim` file. Therefore, the order of checks is:

1. `filetype.vim` files before `$VIMRUNTIME` in `'runtimepath'`
2. first part of `$VIMRUNTIME/filetype.vim`
3. all `scripts.vim` files in `'runtimepath'`
4. remainder of `$VIMRUNTIME/filetype.vim`
5. `filetype.vim` files after `$VIMRUNTIME` in `'runtimepath'`

If this is not sufficient for you, add an autocommand that matches all files and sources a script or executes a function to check the contents of the file.

=====

Next chapter: [usr_44.txt](#) Your own syntax highlighted

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Your own syntax highlighted

Vim comes with highlighting for a couple of hundred different file types. If the file you are editing isn't included, read this chapter to find out how to get this type of file highlighted. Also see [:syn-define](#) in the reference manual.

- 44.1 Basic syntax commands
- 44.2 Keywords
- 44.3 Matches
- 44.4 Regions
- 44.5 Nested items
- 44.6 Following groups
- 44.7 Other arguments
- 44.8 Clusters
- 44.9 Including another syntax file
- 44.10 Synchronizing
- 44.11 Installing a syntax file
- 44.12 Portable syntax file layout

Next chapter: [usr_45.txt](#) Select your language
Previous chapter: [usr_43.txt](#) Using filetypes
Table of contents: [usr_toc.txt](#)

44.1 Basic syntax commands

Using an existing syntax file to start with will save you a lot of time. Try finding a syntax file in `$VIMRUNTIME/syntax` for a language that is similar. These files will also show you the normal layout of a syntax file. To understand it, you need to read the following.

Let's start with the basic arguments. Before we start defining any new syntax, we need to clear out any old definitions:

```
:syntax clear
```

This isn't required in the final syntax file, but very useful when experimenting.

There are more simplifications in this chapter. If you are writing a syntax file to be used by others, read all the way through the end to find out the details.

LISTING DEFINED ITEMS

To check which syntax items are currently defined, use this command:

```
:syntax
```

You can use this to check which items have actually been defined. Quite useful when you are experimenting with a new syntax file. It also shows the colors used for each item, which helps to find out what is what.

To list the items in a specific syntax group use:

```
:syntax list {group-name}
```

This also can be used to list clusters (explained in [44.8](#)). Just include the @ in the name.

MATCHING CASE

Some languages are not case sensitive, such as Pascal. Others, such as C, are case sensitive. You need to tell which type you have with the following commands:

```
:syntax case match  
:syntax case ignore
```

The "match" argument means that Vim will match the case of syntax elements. Therefore, "int" differs from "Int" and "INT". If the "ignore" argument is used, the following are equivalent: "Procedure", "PROCEDURE" and "procedure".

The ":syntax case" commands can appear anywhere in a syntax file and affect the syntax definitions that follow. In most cases, you have only one ":syntax case" command in your syntax file; if you work with an unusual language that contains both case-sensitive and non-case-sensitive elements, however, you can scatter the ":syntax case" command throughout the file.

44.2 Keywords

The most basic syntax elements are keywords. To define a keyword, use the following form:

```
:syntax keyword {group} {keyword} ...
```

The {group} is the name of a syntax group. With the ":highlight" command you can assign colors to a {group}. The {keyword} argument is an actual keyword. Here are a few examples:

```
:syntax keyword xType int long char  
:syntax keyword xStatement if then else endif
```

This example uses the group names "xType" and "xStatement". By convention, each group name is prefixed by the filetype for the language being defined. This example defines syntax for the x language (eXample language without an interesting name). In a syntax file for "csh" scripts the name "cshType" would be used. Thus the prefix is equal to the value of '**filetype**'.

These commands cause the words "int", "long" and "char" to be highlighted one way and the words "if", "then", "else" and "endif" to be highlighted another way. Now you need to connect the x group names to standard Vim names. You do this with the following commands:

```
:highlight link xType Type
:highlight link xStatement Statement
```

This tells Vim to highlight "xType" like "Type" and "xStatement" like "Statement". See [group-name](#) for the standard names.

UNUSUAL KEYWORDS

The characters used in a keyword must be in the **'iskeyword'** option. If you use another character, the word will never match. Vim doesn't give a warning message for this.

The x language uses the '-' character in keywords. This is how it's done:

```
:setlocal iskeyword+--
:syntax keyword xStatement when-not
```

The ":setlocal" command is used to change **'iskeyword'** only for the current buffer. Still it does change the behavior of commands like "w" and "*". If that is not wanted, don't define a keyword but use a match (explained in the next section).

The x language allows for abbreviations. For example, "next" can be abbreviated to "n", "ne" or "nex". You can define them by using this command:

```
:syntax keyword xStatement n[ext]
```

This doesn't match "nextone", keywords always match whole words only.

44.3 Matches

Consider defining something a bit more complex. You want to match ordinary identifiers. To do this, you define a match syntax item. This one matches any word consisting of only lowercase letters:

```
:syntax match xIdentifier /\<\l\+\>/
```

Note:

Keywords overrule any other syntax item. Thus the keywords "if", "then", etc., will be keywords, as defined with the ":syntax keyword" commands above, even though they also match the pattern for xIdentifier.

The part at the end is a pattern, like it's used for searching. The // is used to surround the pattern (like how it's done in a ":substitute" command). You can use any other character, like a plus or a quote.

Now define a match for a comment. In the x language it is anything from # to the end of a line:

```
:syntax match xComment /#.*/
```

Since you can use any search pattern, you can highlight very complex things with a match item. See [pattern](#) for help on search patterns.

44.4 Regions

In the example x language, strings are enclosed in double quotation marks ("). To highlight strings you define a region. You need a region start (double quote) and a region end (double quote). The definition is as follows:

```
:syntax region xString start=/" end=/"
```

The "start" and "end" directives define the patterns used to find the start and end of the region. But what about strings that look like this?

```
"A string with a double quote (\") in it"
```

This creates a problem: The double quotation marks in the middle of the string will end the region. You need to tell Vim to skip over any escaped double quotes in the string. Do this with the skip keyword:

```
:syntax region xString start=/" skip=\/\\" end=/"
```

The double backslash matches a single backslash, since the backslash is a special character in search patterns.

When to use a region instead of a match? The main difference is that a match item is a single pattern, which must match as a whole. A region starts as soon as the "start" pattern matches. Whether the "end" pattern is found or not doesn't matter. Thus when the item depends on the "end" pattern to match, you cannot use a region. Otherwise, regions are often simpler to define. And it is easier to use nested items, as is explained in the next section.

44.5 Nested items

Take a look at this comment:

```
%Get input TODO: Skip white space
```

You want to highlight TODO in big yellow letters, even though it is in a comment that is highlighted blue. To let Vim know about this, you define the following syntax groups:

```
:syntax keyword xTodo TODO contained
:syntax match xComment /*.*/* contains=xTodo
```

In the first line, the "contained" argument tells Vim that this keyword can exist only inside another syntax item. The next line has "contains=xTodo". This indicates that the xTodo syntax element is inside it. The result is that the comment line as a whole is matched with "xComment" and made blue. The word TODO inside it is matched by xTodo and highlighted yellow (highlighting for xTodo was setup for this).

RECURSIVE NESTING

The x language defines code blocks in curly braces. And a code block may contain other code blocks. This can be defined this way:

```
:syntax region xBlock start=/{/ end=}/ / contains=xBlock
```

Suppose you have this text:

```
while i < b {
    if a {
        b = c;
    }
}
```

First a xBlock starts at the { in the first line. In the second line another { is found. Since we are inside a xBlock item, and it contains itself, a nested xBlock item will start here. Thus the "b = c" line is inside the second level xBlock region. Then a } is found in the next line, which matches with the end pattern of the region. This ends the nested xBlock. Because the } is included in the nested region, it is hidden from the first xBlock region. Then at the last } the first xBlock region ends.

KEEPING THE END

Consider the following two syntax items:

```
:syntax region xComment start=%/ end=/$/ contained
:syntax region xPreProc start=#/ end=/$/ contains=xComment
```

You define a comment as anything from % to the end of the line. A preprocessor directive is anything from # to the end of the line. Because you can have a comment on a preprocessor line, the preprocessor definition includes a "contains=xComment" argument. Now look what happens with this text:

```
#define X = Y % Comment text
int foo = 1;
```

What you see is that the second line is also highlighted as xPreProc. The preprocessor directive should end at the end of the line. That is why you have used "end=/\$/". So what is going wrong?

The problem is the contained comment. The comment starts with % and ends at the end of the line. After the comment ends, the preprocessor syntax continues. This is after the end of the line has been seen, so the next line is included as well.

To avoid this problem and to avoid a contained syntax item eating a needed end of line, use the "keepend" argument. This takes care of the double end-of-line matching:

```
:syntax region xComment start=%/ end=/$/ contained
:syntax region xPreProc start=#/ end=/$/ contains=xComment keepend
```

CONTAINING MANY ITEMS

You can use the `contains` argument to specify that everything can be contained. For example:

```
:syntax region xList start=/\[/ end=\/\]/ contains=ALL
```

All syntax items will be contained in this one. It also contains itself, but not at the same position (that would cause an endless loop).

You can specify that some groups are not contained. Thus contain all groups but the ones that are listed:

```
:syntax region xList start=/\[/ end=\/\]/ contains=ALLBUT,xString
```

With the "TOP" item you can include all items that don't have a "contained" argument. "CONTAINED" is used to only include items with a "contained" argument. See `:syn-contains` for the details.

44.6 Following groups

The x language has statements in this form:

```
if (condition) then
```

You want to highlight the three items differently. But "(condition)" and "then" might also appear in other places, where they get different highlighting. This is how you can do this:

```
:syntax match xIf /if/ nextgroup=xIfCondition skipwhite
:syntax match xIfCondition /([^\)]*)/ contained nextgroup=xThen skipwhite
:syntax match xThen /then/ contained
```

The "nextgroup" argument specifies which item can come next. This is not required. If none of the items that are specified are found, nothing happens. For example, in this text:

```
if not (condition) then
```

The "if" is matched by `xIf`. "not" doesn't match the specified nextgroup `xIfCondition`, thus only the "if" is highlighted.

The "skipwhite" argument tells Vim that white space (spaces and tabs) may appear in between the items. Similar arguments are "skipnl", which allows a line break in between the items, and "skipempty", which allows empty lines. Notice that "skipnl" doesn't skip an empty line, something must match after the line break.

44.7 Other arguments

MATCHGROUP

When you define a region, the entire region is highlighted according to the group name specified. To highlight the text enclosed in parentheses () with the group xInside, for example, use the following command:

```
:syntax region xInside start=/(/ end=)/
```

Suppose, that you want to highlight the parentheses differently. You can do this with a lot of convoluted region statements, or you can use the "matchgroup" argument. This tells Vim to highlight the start and end of a region with a different highlight group (in this case, the xParen group):

```
:syntax region xInside matchgroup=xParen start=/(/ end=)/
```

The "matchgroup" argument applies to the start or end match that comes after it. In the previous example both start and end are highlighted with xParen. To highlight the end with xParenEnd:

```
:syntax region xInside matchgroup=xParen start=/(/  
  \ matchgroup=xParenEnd end=)/
```

A side effect of using "matchgroup" is that contained items will not match in the start or end of the region. The example for "transparent" uses this.

TRANSPARENT

In a C language file you would like to highlight the () text after a "while" differently from the () text after a "for". In both of these there can be nested () items, which should be highlighted in the same way. You must make sure the () highlighting stops at the matching). This is one way to do this:

```
:syntax region cWhile matchgroup=cWhile start=/while\s*(/ end=)/  
  \ contains=cCondNest  
:syntax region cFor matchgroup=cFor start=/for\s*(/ end=)/  
  \ contains=cCondNest  
:syntax region cCondNest start=/(/ end=)/ contained transparent
```

Now you can give cWhile and cFor different highlighting. The cCondNest item can appear in either of them, but take over the highlighting of the item it is contained in. The "transparent" argument causes this.

Notice that the "matchgroup" argument has the same group as the item itself. Why define it then? Well, the side effect of using a matchgroup is that contained items are not found in the match with the start item then. This avoids that the cCondNest group matches the (just after the "while" or "for". If this would happen, it would span the whole text until the matching) and the region would continue after it. Now cCondNest only matches after the match with the start pattern, thus after the first (.

OFFSETS

Suppose you want to define a region for the text between (and) after an "if". But you don't want to include the "if" or the (and). You can do this

by specifying offsets for the patterns. Example:

```
:syntax region xCond start=/if\s*(/ms=e+1 end=)/me=s-1
```

The offset for the start pattern is "ms=e+1". "ms" stands for Match Start. This defines an offset for the start of the match. Normally the match starts where the pattern matches. "e+1" means that the match now starts at the end of the pattern match, and then one character further.

The offset for the end pattern is "me=s-1". "me" stands for Match End. "s-1" means the start of the pattern match and then one character back. The result is that in this text:

```
if (foo == bar)
```

Only the text "foo == bar" will be highlighted as xCond.

More about offsets here: [:syn-pattern-offset](#) .

ONELINE

The "oneline" argument indicates that the region does not cross a line boundary. For example:

```
:syntax region xIfThen start=/if/ end=/then/ oneline
```

This defines a region that starts at "if" and ends at "then". But if there is no "then" after the "if", the region doesn't match.

Note:

When using "oneline" the region doesn't start if the end pattern doesn't match in the same line. Without "oneline" Vim does `_not_` check if there is a match for the end pattern. The region starts even when the end pattern doesn't match in the rest of the file.

CONTINUATION LINES AND AVOIDING THEM

Things now become a little more complex. Let's define a preprocessor line. This starts with a # in the first column and continues until the end of the line. A line that ends with \ makes the next line a continuation line. The way you handle this is to allow the syntax item to contain a continuation pattern:

```
:syntax region xPreProc start=/^#/ end=/$/ contains=xLineContinue
:syntax match xLineContinue "\\$" contained
```

In this case, although xPreProc normally matches a single line, the group contained in it (namely xLineContinue) lets it go on for more than one line. For example, it would match both of these lines:

```
#define SPAM spam spam \
        bacon and spam
```

In this case, this is what you want. If it is not what you want, you can call for the region to be on a single line by adding "excludenl" to the contained pattern. For example, you want to highlight "end" in xPreProc, but only at the end of the line. To avoid making the xPreProc continue on the next line, like xLineContinue does, use "excludenl" like this:

```
:syntax region xPreProc start=/^#/ end=/$/
    \ contains=xLineContinue,xPreProcEnd
:syntax match xPreProcEnd excludenl /end$/ contained
:syntax match xLineContinue "\\$" contained
```

"excludenl" must be placed before the pattern. Since "xLineContinue" doesn't have "excludenl", a match with it will extend xPreProc to the next line as before.

44.8 Clusters

One of the things you will notice as you start to write a syntax file is that you wind up generating a lot of syntax groups. Vim enables you to define a collection of syntax groups called a cluster.

Suppose you have a language that contains for loops, if statements, while loops, and functions. Each of them contains the same syntax elements: numbers and identifiers. You define them like this:

```
:syntax match xFor /^for.*/ contains=xNumber,xIdent
:syntax match xIf /^if.*/ contains=xNumber,xIdent
:syntax match xWhile /^while.*/ contains=xNumber,xIdent
```

You have to repeat the same "contains=" every time. If you want to add another contained item, you have to add it three times. Syntax clusters simplify these definitions by enabling you to have one cluster stand for several syntax groups.

To define a cluster for the two items that the three groups contain, use the following command:

```
:syntax cluster xState contains=xNumber,xIdent
```

Clusters are used inside other syntax items just like any syntax group. Their names start with @. Thus, you can define the three groups like this:

```
:syntax match xFor /^for.*/ contains=@xState
:syntax match xIf /^if.*/ contains=@xState
:syntax match xWhile /^while.*/ contains=@xState
```

You can add new group names to this cluster with the "add" argument:

```
:syntax cluster xState add=xString
```

You can remove syntax groups from this list as well:

```
:syntax cluster xState remove=xNumber
```

44.9 Including another syntax file

The C++ language syntax is a superset of the C language. Because you do not want to write two syntax files, you can have the C++ syntax file read in the one for C by using the following command:

```
:runtime! syntax/c.vim
```

The ":runtime!" command searches '**runtimepath**' for all "syntax/c.vim" files. This makes the C parts of the C++ syntax be defined like for C files. If you have replaced the c.vim syntax file, or added items with an extra file, these will be loaded as well.

After loading the C syntax items the specific C++ items can be defined. For example, add keywords that are not used in C:

```
:syntax keyword cppStatement    new delete this friend using
```

This works just like in any other syntax file.

Now consider the Perl language. A Perl script consists of two distinct parts: a documentation section in POD format, and a program written in Perl itself. The POD section starts with "=head" and ends with "=cut".

You want to define the POD syntax in one file, and use it from the Perl syntax file. The ":syntax include" command reads in a syntax file and stores the elements it defined in a syntax cluster. For Perl, the statements are as follows:

```
:syntax include @Pod <sfile>:p:h/pod.vim
:syntax region perlPOD start=/^=head/ end=/^=cut/ contains=@Pod
```

When "=head" is found in a Perl file, the perlPOD region starts. In this region the @Pod cluster is contained. All the items defined as top-level items in the pod.vim syntax files will match here. When "=cut" is found, the region ends and we go back to the items defined in the Perl file.

The ":syntax include" command is clever enough to ignore a ":syntax clear" command in the included file. And an argument such as "contains=ALL" will only contain items defined in the included file, not in the file that includes it.

The "<sfile>:p:h/" part uses the name of the current file (<sfile>), expands it to a full path (:p) and then takes the head (:h). This results in the directory name of the file. This causes the pod.vim file in the same directory to be included.

44.10 Synchronizing

Compilers have it easy. They start at the beginning of a file and parse it straight through. Vim does not have it so easy. It must start in the middle, where the editing is being done. So how does it tell where it is?

The secret is the ":syntax sync" command. This tells Vim how to figure out where it is. For example, the following command tells Vim to scan backward for the beginning or end of a C-style comment and begin syntax coloring from there:

```
:syntax sync ccomment
```

You can tune this processing with some arguments. The "minlines" argument tells Vim the minimum number of lines to look backward, and "maxlines" tells the editor the maximum number of lines to scan.

For example, the following command tells Vim to look at least 10 lines before the top of the screen:

```
:syntax sync ccomment minlines=10 maxlines=500
```

If it cannot figure out where it is in that space, it starts looking farther and farther back until it figures out what to do. But it looks no farther back than 500 lines. (A large "maxlines" slows down processing. A small one might cause synchronization to fail.)

To make synchronizing go a bit faster, tell Vim which syntax items can be skipped. Every match and region that only needs to be used when actually displaying text can be given the "display" argument.

By default, the comment to be found will be colored as part of the Comment syntax group. If you want to color things another way, you can specify a different syntax group:

```
:syntax sync ccomment xAltComment
```

If your programming language does not have C-style comments in it, you can try another method of synchronization. The simplest way is to tell Vim to space back a number of lines and try to figure out things from there. The following command tells Vim to go back 150 lines and start parsing from there:

```
:syntax sync minlines=150
```

A large "minlines" value can make Vim slower, especially when scrolling backwards in the file.

Finally, you can specify a syntax group to look for by using this command:

```
:syntax sync match {sync-group-name}  
 \ grouphere {group-name} {pattern}
```

This tells Vim that when it sees {pattern} the syntax group named {group-name} begins just after the pattern given. The {sync-group-name} is used to give a name to this synchronization specification. For example, the sh scripting language begins an if statement with "if" and ends it with "fi":

```
if [ --f file.txt ] ; then  
    echo "File exists"  
fi
```

To define a "grouphere" directive for this syntax, you use the following command:

```
:syntax sync match shIfSync grouphere shIf "<if>"
```

The "grouphere" argument tells Vim that the pattern ends a group. For example, the end of the if/fi group is as follows:

```
:syntax sync match shIfSync grouphere NONE "\<fi\>"
```

In this example, the NONE tells Vim that you are not in any special syntax region. In particular, you are not inside an if block.

You also can define matches and regions that are with no "grouphere" or "grouphere" arguments. These groups are for syntax groups skipped during synchronization. For example, the following skips over anything inside {}, even if it would normally match another synchronization method:

```
:syntax sync match xSpecial /{.*}/
```

More about synchronizing in the reference manual: [:syn-sync](#) .

44.11 Installing a syntax file

When your new syntax file is ready to be used, drop it in a "syntax" directory in '[runtimepath](#)'. For Unix that would be "`~/vim/syntax`".

The name of the syntax file must be equal to the file type, with ".vim" added. Thus for the x language, the full path of the file would be:

```
~/vim/syntax/x.vim
```

You must also make the file type be recognized. See [43.2](#) .

If your file works well, you might want to make it available to other Vim users. First read the next section to make sure your file works well for others. Then e-mail it to the Vim maintainer: maintainer@vim.org. Also explain how the filetype can be detected. With a bit of luck your file will be included in the next Vim version!

ADDING TO AN EXISTING SYNTAX FILE

We were assuming you were adding a completely new syntax file. When an existing syntax file works, but is missing some items, you can add items in a separate file. That avoids changing the distributed syntax file, which will be lost when installing a new version of Vim.

Write syntax commands in your file, possibly using group names from the existing syntax. For example, to add new variable types to the C syntax file:

```
:syntax keyword cType off_t uint
```

Write the file with the same name as the original syntax file. In this case "c.vim". Place it in a directory near the end of '[runtimepath](#)'. This makes it loaded after the original syntax file. For Unix this would be:

```
~/vim/after/syntax/c.vim
```

44.12 Portable syntax file layout

Wouldn't it be nice if all Vim users exchange syntax files? To make this

possible, the syntax file must follow a few guidelines.

Start with a header that explains what the syntax file is for, who maintains it and when it was last updated. Don't include too much information about changes history, not many people will read it. Example:

```
" Vim syntax file
" Language:      C
" Maintainer:   Bram Moolenaar <Bram@vim.org>
" Last Change:  2001 Jun 18
" Remark:       Included by the C++ syntax.
```

Use the same layout as the other syntax files. Using an existing syntax file as an example will save you a lot of time.

Choose a good, descriptive name for your syntax file. Use lowercase letters and digits. Don't make it too long, it is used in many places: The name of the syntax file "name.vim", '**filetype**', b:current_syntax and the start of each syntax group (nameType, nameStatement, nameString, etc).

Start with a check for "b:current_syntax". If it is defined, some other syntax file, earlier in '**runtimepath**' was already loaded:

```
if exists("b:current_syntax")
  finish
endif
```

Set "b:current_syntax" to the name of the syntax at the end. Don't forget that included files do this too, you might have to reset "b:current_syntax" if you include two files.

Do not include anything that is a user preference. Don't set '**tabstop**', '**expandtab**', etc. These belong in a filetype plugin.

Do not include mappings or abbreviations. Only include setting '**iskeyword**' if it is really necessary for recognizing keywords.

To allow users select their own preferred colors, make a different group name for every kind of highlighted item. Then link each of them to one of the standard highlight groups. That will make it work with every color scheme. If you select specific colors it will look bad with some color schemes. And don't forget that some people use a different background color, or have only eight colors available.

For the linking use "hi def link", so that the user can select different highlighting before your syntax file is loaded. Example:

```
hi def link nameString      String
hi def link nameNumber     Number
hi def link nameCommand    Statement
... etc ...
```

Add the "display" argument to items that are not used when syncing, to speed up scrolling backwards and **CTRL-L**.

Next chapter: [usr_45.txt](#) Select your language

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_45.txt For Vim version 9.1. Last change: 2022 May 13

VIM USER MANUAL - by Bram Moolenaar

Select your language (locale)

The messages in Vim can be given in several languages. This chapter explains how to change which one is used. Also, the different ways to work with files in various languages is explained.

- 45.1 Language for Messages
- 45.2 Language for Menus
- 45.3 Using another encoding
- 45.4 Editing files with a different encoding
- 45.5 Entering language text

Next chapter: [usr_50.txt](#) Advanced Vim script writing
Previous chapter: [usr_44.txt](#) Your own syntax highlighted
Table of contents: [usr_toc.txt](#)

45.1 Language for Messages

When you start Vim, it checks the environment to find out what language you are using. Mostly this should work fine, and you get the messages in your language (if they are available). To see what the current language is, use this command:

```
:language
```

If it replies with "C", this means the default is being used, which is English.

Note:

Using different languages only works when Vim was compiled to handle it. To find out if it works, use the ":version" command and check the output for "+gettext" and "+multi_lang". If they are there, you are OK. If you see "-gettext" or "-multi_lang" you will have to find another Vim.

What if you would like your messages in a different language? There are several ways. Which one you should use depends on the capabilities of your system.

The first way is to set the environment to the desired language before starting Vim. Example for Unix:

```
env LANG=de_DE.ISO_8859-1 vim
```

This only works if the language is available on your system. The advantage is that all the GUI messages and things in libraries will use the right language as well. A disadvantage is that you must do this before starting Vim. If you want to change language while Vim is running, you can use the second method:

```
:language fr_FR.ISO_8859-1
```

This way you can try out several names for your language. You will get an error message when it's not supported on your system. You don't get an error when translated messages are not available. Vim will silently fall back to using English.

To find out which languages are supported on your system, find the directory where they are listed. On my system it is `"/usr/share/locale"`. On some systems it's in `"/usr/lib/locale"`. The manual page for `"setlocale"` should give you a hint where it is found on your system.

Be careful to type the name exactly as it should be. Upper and lowercase matter, and the `'-'` and `'_'` characters are easily confused.

You can also set the language separately for messages, edited text and the time format. See `:language` .

DO-IT-YOURSELF MESSAGE TRANSLATION

If translated messages are not available for your language, you could write them yourself. To do this, get the source code for Vim and the GNU gettext package. After unpacking the sources, instructions can be found in the directory `src/po/README.txt`.

It's not too difficult to do the translation. You don't need to be a programmer. You must know both English and the language you are translating to, of course.

When you are satisfied with the translation, consider making it available to others. Upload it at vim-online (<http://vim.sf.net>) or e-mail it to the Vim maintainer maintainer@vim.org. Or both.

45.2 Language for Menus

The default menus are in English. To be able to use your local language, they must be translated. Normally this is automatically done for you if the environment is set for your language, just like with messages. You don't need to do anything extra for this. But it only works if translations for the language are available.

Suppose you are in Germany, with the language set to German, but prefer to use "File" instead of "Datei". You can switch back to using the English menus this way:

```
:set langmenu=none
```

It is also possible to specify a language:

```
:set langmenu=nl_NL.ISO_8859-1
```

Like above, differences between `"-"` and `"_"` matter. However, upper/lowercase differences are ignored here.

The `'langmenu'` option must be set before the menus are loaded. Once the menus have been defined changing `'langmenu'` has no direct effect. Therefore, put the command to set `'langmenu'` in your vimrc file.

If you really want to switch menu language while running Vim, you can do it

this way:

```
:source $VIMRUNTIME/delmenu.vim
:set langmenu=de_DE.ISO_8859-1
:source $VIMRUNTIME/menu.vim
```

There is one drawback: All menus that you defined yourself will be gone. You will need to redefine them as well.

DO-IT-YOURSELF MENU TRANSLATION

To see which menu translations are available, look in this directory:

`$VIMRUNTIME/lang`

The files are called `menu_{language}.vim`. If you don't see the language you want to use, you can do your own translations. The simplest way to do this is by copying one of the existing language files, and change it.

First find out the name of your language with the `":language"` command. Use this name, but with all letters made lowercase. Then copy the file to your own runtime directory, as found early in `'runtimepath'`. For example, for Unix you would do:

```
!:cp $VIMRUNTIME/lang/menu_ko_kr.euckr.vim ~/.vim/lang/menu_nl_be.iso_8859-1.vim
```

You will find hints for the translation in `"$VIMRUNTIME/lang/README.txt"`.

45.3 Using another encoding

Vim guesses that the files you are going to edit are encoded for your language. For many European languages this is "latin1". Then each byte is one character. That means there are 256 different characters possible. For Asian languages this is not sufficient. These mostly use a double-byte encoding, providing for over ten thousand possible characters. This still isn't enough when a text is to contain several different languages. This is where Unicode comes in. It was designed to include all characters used in commonly used languages. This is the "Super encoding that replaces all others". But it isn't used that much yet.

Fortunately, Vim supports these three kinds of encodings. And, with some restrictions, you can use them even when your environment uses another language than the text.

Nevertheless, when you only edit files that are in the encoding of your language, the default should work fine and you don't need to do anything. The following is only relevant when you want to edit different languages.

USING UNICODE IN THE GUI

The nice thing about Unicode is that other encodings can be converted to it and back without losing information. When you make Vim use Unicode internally, you will be able to edit files in any encoding.

Unfortunately, the number of systems supporting Unicode is still limited.

Thus it's unlikely that your language uses it. You need to tell Vim you want to use Unicode, and how to handle interfacing with the rest of the system.

Let's start with the GUI version of Vim, which is able to display Unicode characters. This should work:

```
:set encoding=utf-8
:set guifont=-misc-fixed-medium-r-normal--18-120-100-100-c-90-iso10646-1
```

The **'encoding'** option tells Vim the encoding of the characters that you use. This applies to the text in buffers (files you are editing), registers, Vim script files, etc. You can regard **'encoding'** as the setting for the internals of Vim.

This example assumes you have this font on your system. The name in the example is for the X Window System. This font is in a package that is used to enhance xterm with Unicode support. If you don't have this font, you might find it here:

<http://www.cl.cam.ac.uk/~mgk25/download/ucs-fonts.tar.gz>

For MS-Windows, some fonts have a limited number of Unicode characters. Try using the "Courier New" font. You can use the Edit/Select Font... menu to select and try out the fonts available. Only fixed-width fonts can be used though. Example:

```
:set guifont=courier_new:h12
```

If it doesn't work well, try getting a fontpack. If Microsoft didn't move it, you can find it here:

<http://www.microsoft.com/typography/fonts/default.aspx>

Now you have told Vim to use Unicode internally and display text with a Unicode font. Typed characters still arrive in the encoding of your original language. This requires converting them to Unicode. Tell Vim the language from which to convert with the **'termencoding'** option. You can do it like this:

```
:let &termencoding = &encoding
:set encoding=utf-8
```

This assigns the old value of **'encoding'** to **'termencoding'** before setting **'encoding'** to utf-8. You will have to try out if this really works for your setup. It should work especially well when using an input method for an Asian language, and you want to edit Unicode text.

USING UNICODE IN A UNICODE TERMINAL

There are terminals that support Unicode directly. The standard xterm that comes with XFree86 is one of them. Let's use that as an example.

First of all, the xterm must have been compiled with Unicode support. See [UTF8-xterm](#) how to check that and how to compile it when needed.

Start the xterm with the "-u8" argument. You might also need so specify a font. Example:

```
xterm -u8 -fn -misc-fixed-medium-r-normal--18-120-100-100-c-90-iso10646-1
```

Now you can run Vim inside this terminal. Set **'encoding'** to "utf-8" as before. That's all.

USING UNICODE IN AN ORDINARY TERMINAL

Suppose you want to work with Unicode files, but don't have a terminal with Unicode support. You can do this with Vim, although characters that are not supported by the terminal will not be displayed. The layout of the text will be preserved.

```
:let &termencoding = &encoding
:set encoding=utf-8
```

This is the same as what was used for the GUI. But it works differently: Vim will convert the displayed text before sending it to the terminal. That avoids that the display is messed up with strange characters.

For this to work the conversion between **'termencoding'** and **'encoding'** must be possible. Vim will convert from latin1 to Unicode, thus that always works. For other conversions the **+iconv** feature is required.

Try editing a file with Unicode characters in it. You will notice that Vim will put a question mark (or underscore or some other character) in places where a character should be that the terminal can't display. Move the cursor to a question mark and use this command:

```
ga
```

Vim will display a line with the code of the character. This gives you a hint about what character it is. You can look it up in a Unicode table. You could actually view a file that way, if you have lots of time at hand.

Note:

Since **'encoding'** is used for all text inside Vim, changing it makes all non-ASCII text invalid. You will notice this when using registers and the **'viminfo'** file (e.g., a remembered search pattern). It's recommended to set **'encoding'** in your vimrc file, and leave it alone.

45.4 Editing files with a different encoding

Suppose you have setup Vim to use Unicode, and you want to edit a file that is in 16-bit Unicode. Sounds simple, right? Well, Vim actually uses utf-8 encoding internally, thus the 16-bit encoding must be converted, since there is a difference between the character set (Unicode) and the encoding (utf-8 or 16-bit).

Vim will try to detect what kind of file you are editing. It uses the encoding names in the **'fileencodings'** option. When using Unicode, the default value is: "ucs-bom,utf-8,latin1". This means that Vim checks the file to see if it's one of these encodings:

ucs-bom	File must start with a Byte Order Mark (BOM). This
---------	--

	allows detection of 16-bit, 32-bit and utf-8 Unicode encodings.
utf-8	utf-8 Unicode. This is rejected when a sequence of bytes is illegal in utf-8.
latin1	The good old 8-bit encoding. Always works.

When you start editing that 16-bit Unicode file, and it has a BOM, Vim will detect this and convert the file to utf-8 when reading it. The **'fileencoding'** option (without `s` at the end) is set to the detected value. In this case it is "utf-16le". That means it's Unicode, 16-bit and little-endian. This file format is common on MS-Windows (e.g., for registry files).

When writing the file, Vim will compare **'fileencoding'** with **'encoding'**. If they are different, the text will be converted.

An empty value for **'fileencoding'** means that no conversion is to be done. Thus the text is assumed to be encoded with **'encoding'**.

If the default **'fileencodings'** value is not good for you, set it to the encodings you want Vim to try. Only when a value is found to be invalid will the next one be used. Putting "latin1" first doesn't work, because it is never illegal. An example, to fall back to Japanese when the file doesn't have a BOM and isn't utf-8:

```
:set fileencodings=ucs-bom,utf-8,sjis
```

See [encoding-values](#) for suggested values. Other values may work as well. This depends on the conversion available.

FORCING AN ENCODING

If the automatic detection doesn't work you must tell Vim what encoding the file is. Example:

```
:edit ++enc=koi8-r russian.txt
```

The "++enc" part specifies the name of the encoding to be used for this file only. Vim will convert the file from the specified encoding, Russian in this example, to **'encoding'**. **'fileencoding'** will also be set to the specified encoding, so that the reverse conversion can be done when writing the file.

The same argument can be used when writing the file. This way you can actually use Vim to convert a file. Example:

```
:write ++enc=utf-8 russian.txt
```

Note:

Conversion may result in lost characters. Conversion from an encoding to Unicode and back is mostly free of this problem, unless there are illegal characters. Conversion from Unicode to other encodings often loses information when there was more than one language in the file.

45.5 Entering language text

Computer keyboards don't have much more than a hundred keys. Some languages

have thousands of characters, Unicode has over hundred thousand. So how do you type these characters?

First of all, when you don't use too many of the special characters, you can use digraphs. This was already explained in [24.9](#).

When you use a language that uses many more characters than keys on your keyboard, you will want to use an Input Method (IM). This requires learning the translation from typed keys to resulting character. When you need an IM you probably already have one on your system. It should work with Vim like with other programs. For details see [mbyte-XIM](#) for the X Window system and [mbyte-IME](#) for MS-Windows.

KEYMAPS

For some languages the character set is different from latin, but uses a similar number of characters. It's possible to map keys to characters. Vim uses keymaps for this.

Suppose you want to type Hebrew. You can load the keymap like this:

```
:set keymap=hebrew
```

Vim will try to find a keymap file for you. This depends on the value of `'encoding'`. If no matching file was found, you will get an error message.

Now you can type Hebrew in Insert mode. In Normal mode, and when typing a ":" command, Vim automatically switches to English. You can use this command to switch between Hebrew and English:

```
CTRL-^
```

This only works in Insert mode and Command-line mode. In Normal mode it does something completely different (jumps to alternate file).

The usage of the keymap is indicated in the mode message, if you have the `'showmode'` option set. In the GUI Vim will indicate the usage of keymaps with a different cursor color.

You can also change the usage of the keymap with the `'iminsert'` and `'imsearch'` options.

To see the list of mappings, use this command:

```
:lmap
```

To find out which keymap files are available, in the GUI you can use the Edit/Keymap menu. Otherwise you can use this command:

```
:echo globpath(&rtp, "keymap/*.vim")
```

DO-IT-YOURSELF KEYMAPS

You can create your own keymap file. It's not very difficult. Start with a keymap file that is similar to the language you want to use. Copy it to the "keymap" directory in your runtime directory. For example, for Unix, you would use the directory `"~/vim/keymap"`.

The name of the keymap file must look like this:

`keymap/{name}.vim`

or

`keymap/{name}_{encoding}.vim`

`{name}` is the name of the keymap. Chose a name that is obvious, but different from existing keymaps (unless you want to replace an existing keymap file).

`{name}` cannot contain an underscore. Optionally, add the encoding used after an underscore. Examples:

`keymap/hebrew.vim`

`keymap/hebrew_utf-8.vim`

The contents of the file should be self-explanatory. Look at a few of the keymaps that are distributed with Vim. For the details, see [mbyte-keymap](#) .

LAST RESORT

If all other methods fail, you can enter any character with **CTRL-V**:

encoding	type	range
8-bit	CTRL-V 123	decimal 0-255
8-bit	CTRL-V x a1	hexadecimal 00-ff
16-bit	CTRL-V u 013b	hexadecimal 0000-ffff
31-bit	CTRL-V U 001303a4	hexadecimal 00000000-7fffffff

Don't type the spaces. See [i_CTRL-V_digit](#) for the details.

=====

Next chapter: [usr_50.txt](#) Advanced Vim script writing

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_50.txt For Vim version 9.1. Last change: 2022 Jun 20

VIM USER MANUAL - by Bram Moolenaar

Advanced Vim script writing

- 50.1 Exceptions
- 50.2 Function with variable number of arguments
- 50.3 Restoring the view

Next chapter: [usr_51.txt](#) Create a plugin
Previous chapter: [usr_45.txt](#) Select your language (local)
Table of contents: [usr_toc.txt](#)

50.1 Exceptions

Let's start with an example:

```
try
  read ~/templates/pascal.tpl
catch /E484:/
  echo "Sorry, the Pascal template file cannot be found."
endtry
```

The ``read`` command will fail if the file does not exist. Instead of generating an error message, this code catches the error and gives the user a message with more information.

For the commands in between ``try`` and ``endtry`` errors are turned into exceptions. An exception is a string. In the case of an error the string contains the error message. And every error message has a number. In this case, the error we catch contains "E484:". This number is guaranteed to stay the same (the text may change, e.g., it may be translated).

Besides being able to give a nice error message, Vim will also continue executing commands after the ``:endtry``. Otherwise, once an uncaught error is encountered, execution of the script/function/mapping will be aborted.

When the ``read`` command causes another error, the pattern "E484:" will not match in it. Thus this exception will not be caught and result in the usual error message and execution is aborted.

You might be tempted to do this:

```
try
  read ~/templates/pascal.tpl
catch
  echo "Sorry, the Pascal template file cannot be found."
endtry
```

This means all errors are caught. But then you will not see an error that would indicate a completely different problem, such as "E21: Cannot make

changes, **'modifiable'** is off". Think twice before you catch any error!

Another useful mechanism is the ``finally`` command:

```
var tmp = tempname()
try
  exe "!.,$write " .. tmp
  exe "!filter " .. tmp
  !,$delete
  exe "!:,$read " .. tmp
finally
  delete(tmp)
endtry
```

This filters the lines from the cursor until the end of the file through the "filter" command, which takes a file name argument. No matter if the filtering works, if something goes wrong in between ``try`` and ``finally`` or the user cancels the filtering by pressing **CTRL-C**, the ``delete(tmp)`` call is always executed. This makes sure you don't leave the temporary file behind.

The ``finally`` does not catch the exception, the error will still abort further execution.

More information about exception handling can be found in the reference manual: [exception-handling](#) .

50.2 Function with variable number of arguments

Vim enables you to define functions that have a variable number of arguments. The following command, for instance, defines a function that must have 1 argument (start) and can have up to 20 additional arguments:

```
def Show(start: string, ...items: list<string>)
```

The variable "items" will be a list in the function containing the extra arguments. You can use it like any list, for example:

```
def Show(start: string, ...items: list<string>)
  echohl Title
  echo "start is " .. start
  echohl None
  for index in range(len(items))
    echon $" Arg {index} is {items[index]}"
  endfor
  echo
enddef
```

You can call it like this:

```
Show('Title', 'one', 'two', 'three')
start is Title Arg 0 is one Arg 1 is two Arg 2 is three
```

This uses the ``echohl`` command to specify the highlighting used for the

following ``echo`` command. ``echohl None`` stops it again. The ``echon`` command works like ``echo``, but doesn't output a line break.

If you call it with one argument the "items" list will be empty. ``range(len(items))`` returns a list with the indexes, what ``for`` loops over, we'll explain that further down.

50.3 Restoring the view

Sometimes you want to jump around, make a change and then go back to the same position and view. For example to change something in the file header. This can be done with two functions:

```
var view = winsaveview()
# Move around, make changes
winrestview(view)
```

Next chapter: [usr_51.txt](#) Create a plugin

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Write plugins

Plugins can be used to define settings for a specific type of file, syntax highlighting and many other things. This chapter explains how to write the most common Vim plugins.

- 51.1 Writing a generic plugin
- 51.2 Writing a filetype plugin
- 51.3 Writing a compiler plugin
- 51.4 Distributing Vim scripts

Next chapter: [usr_52.txt](#) Write large plugins
Previous chapter: [usr_50.txt](#) Advanced Vim script writing
Table of contents: [usr_toc.txt](#)

51.1 Writing a generic plugin

write-plugin

You can write a Vim script in such a way that many people can use it. This is called a plugin. Vim users can drop your script in their plugin directory and use its features right away [add-plugin](#) .

There are actually two types of plugins:

- global plugins: For all types of files.
- filetype plugins: Only for files of a specific type.

In this section the first type is explained. Most items are also relevant for writing filetype plugins. The specifics for filetype plugins are in the next section [write-filetype-plugin](#) .

We will use [Vim9](#) syntax here, the recommended way to write new plugins. Make sure the file starts with the ``vim9script`` command.

NAME

First of all you must choose a name for your plugin. The features provided by the plugin should be clear from its name. And it should be unlikely that someone else writes a plugin with the same name but which does something different.

A script that corrects typing mistakes could be called "typecorrect.vim". We will use it here as an example.

For the plugin to work for everybody, it should follow a few guidelines. This will be explained step-by-step. The complete example plugin is at the end.

BODY

Let's start with the body of the plugin, the lines that do the actual work:

```
12     iabbrev teh the
13     iabbrev otehr other
14     iabbrev wnat want
15     iabbrev synchronisation
16         \ synchronization
```

The actual list should be much longer, of course.

The line numbers have only been added to explain a few things, don't put them in your plugin file!

FIRST LINE

```
1     vim9script noclear
```

You need to use ``vim9script`` as the very first command. Best is to put it in the very first line.

The script we are writing will have a ``finish`` command to bail out when it is loaded a second time. To avoid that the items defined in the script are lost the "noclear" argument is used. More info about this at [vim9-reload](#).

HEADER

You will probably add new corrections to the plugin and soon have several versions lying around. And when distributing this file, people will want to know who wrote this wonderful plugin and where they can send remarks. Therefore, put a header at the top of your plugin:

```
2     # Vim global plugin for correcting typing mistakes
3     # Last Change: 2021 Dec 30
4     # Maintainer:  Bram Moolenaar <Bram@vim.org>
```

About copyright and licensing: Since plugins are very useful and it's hardly worth restricting their distribution, please consider making your plugin either public domain or use the Vim [license](#). A short [note](#) about this near the top of the plugin should be sufficient. Example:

```
5     # License:      This file is placed in the public domain.
```

NOT LOADING

It is possible that a user doesn't always want to load this plugin. Or the system administrator has dropped it in the system-wide plugin directory, but a user has their own plugin they want to use. Then the user must have a chance to disable loading this specific plugin. These lines will make it possible:


```

7     if exists("g:loaded_typecorrect")
8         finish
9     endif
10    g:loaded_typecorrect = 1

```

This also avoids that when the script is loaded twice it would pointlessly redefine functions and cause trouble for autocommands that are added twice.

The name is recommended to start with "g:loaded_" and then the file name of the plugin, literally. The "g:" is prepended to make the variable global, so that other places can check whether its functionality is available. Without "g:" it would be local to the script.

Using ``finish`` stops Vim from reading the rest of the file, it's much quicker than using if-endif around the whole file, since Vim would still need to parse the commands to find the ``endif``.

MAPPING

Now let's make the plugin more interesting: We will add a mapping that adds a correction for the word under the cursor. We could just pick a key sequence for this mapping, but the user might already use it for something else. To allow the user to define which keys a mapping in a plugin uses, the `<Leader>` item can be used:

```

20    map <unique> <Leader>a <Plug>TypecorrAdd;

```

The "`<Plug>TypecorrAdd;`" thing will do the work, more about that further on.

The user can set the "g:mapleader" variable to the key sequence that they want plugin mappings to start with. Thus if the user has done:

```

g:mapleader = "_"

```

the mapping will define "_a". If the user didn't do this, the default value will be used, which is a backslash. Then a map for "\a" will be defined.

Note that `<unique>` is used, this will cause an error message if the mapping already happened to exist. `:map-<unique>`

But what if the user wants to define their own key sequence? We can allow that with this mechanism:

```

19    if !hasmapto('<Plug>TypecorrAdd;')
20        map <unique> <Leader>a <Plug>TypecorrAdd;
21    endif

```

This checks if a mapping to "`<Plug>TypecorrAdd;`" already exists, and only defines the mapping from "`<Leader>a`" if it doesn't. The user then has a chance of putting this in their vimrc file:

```

map ,c <Plug>TypecorrAdd;

```

Then the mapped key sequence will be ",c" instead of "_a" or "\a".

PIECES

If a script gets longer, you often want to break up the work in pieces. You can use functions or mappings for this. But you don't want these functions and mappings to interfere with the ones from other scripts. For example, you could define a function `Add()`, but another script could try to define the same function. To avoid this, we define the function local to the script. Fortunately, in [Vim9](#) script this is the default. In a legacy script you would need to prefix the name with "s:".

We will define a function that adds a new typing correction:

```
28     def Add(from: string, correct: bool)
29         var to = input($"type the correction for {from}: ")
30         exe $":iabbrev {from} {to}"
...
34     enddef
```

Now we can call the function `Add()` from within this script. If another script also defines `Add()`, it will be local to that script and can only be called from that script. There can also be a global `g:Add()` function, which is again another function.

`<SID>` can be used with mappings. It generates a script ID, which identifies the current script. In our typing correction plugin we use it like this:

```
22     noremap <unique> <script> <Plug>TypecorrAdd; <SID>Add
...
26     noremap <SID>Add :call <SID>Add(expand("<cword>"), true)<CR>
```

Thus when a user types "\a", this sequence is invoked:

```
\a -> <Plug>TypecorrAdd; -> <SID>Add -> :call <SID>Add(...)
```

If another script also maps `<SID>Add`, it will get another script ID and thus define another mapping.

Note that instead of `Add()` we use `<SID>Add()` here. That is because the mapping is typed by the user, thus outside of the script context. The `<SID>` is translated to the script ID, so that Vim knows in which script to look for the `Add()` function.

This is a bit complicated, but it's required for the plugin to work together with other plugins. The basic rule is that you use `<SID>Add()` in mappings and `Add()` in other places (the script itself, autocommands, user commands).

We can also add a menu entry to do the same as the mapping:

```
24     noremenu <script> Plugin.Add\ Correction <SID>Add
```

The "Plugin" menu is recommended for adding menu items for plugins. In this

case only one item is used. When adding more items, creating a submenu is recommended. For example, "Plugin.CVS" could be used for a plugin that offers CVS operations "Plugin.CVS.checkin", "Plugin.CVS.checkout", etc.

Note that in line 28 ":noremap" is used to avoid that any other mappings cause trouble. Someone may have remapped ":call", for example. In line 24 we also use ":noremap", but we do want "<SID>Add" to be remapped. This is why "<script>" is used here. This only allows mappings which are local to the script. `:map-<script>` The same is done in line 26 for ":noremenu".
`:menu-<script>`

<SID> AND <Plug> using-<Plug>

Both <SID> and <Plug> are used to avoid that mappings of typed keys interfere with mappings that are only to be used from other mappings. **Note** the difference between using <SID> and <Plug>:

<Plug> is visible outside of the script. It is used for mappings which the user might want to map a key sequence to. <Plug> is a special code that a typed key will never produce.

To make it very unlikely that other plugins use the same sequence of characters, use this structure: <Plug> scriptname mapname

In our example the scriptname is "Typecorr" and the mapname is "Add".

We add a semicolon as the terminator. This results in "<Plug>TypecorrAdd;". Only the first character of scriptname and mapname is uppercase, so that we can see where mapname starts.

<SID> is the script ID, a unique identifier for a script. Internally Vim translates <SID> to "<SNR>123_", where "123" can be any number. Thus a function "<SID>Add()" will have a name "<SNR>11_Add()" in one script, and "<SNR>22_Add()" in another. You can see this if you use the ":function" command to get a list of functions. The translation of <SID> in mappings is exactly the same, that's how you can call a script-local function from a mapping.

USER COMMAND

Now let's add a user command to add a correction:

```
36   if !exists(":Correct")
37       command -nargs=1 Correct :call Add(<q-args>, false)
38   endif
```

The user command is defined only if no command with the same name already exists. Otherwise we would get an error here. Overriding the existing user command with ":command!" is not a good idea, this would probably make the user wonder why the command they defined themselves doesn't work. `:command`
If it did happen you can find out who to blame with:

`verbose command Correct`

SCRIPT VARIABLES

When a variable starts with "s:" it is a script variable. It can only be used inside a script. Outside the script it's not visible. This avoids trouble with using the same variable name in different scripts. The variables will be kept as long as Vim is running. And the same variables are used when sourcing the same script again. `s:var`

The nice thing about `Vim9` script is that variables are local to the script by default. You can prepend "s:" if you like, but you do not need to. And functions in the script can also use the script variables without a prefix (they must be declared before the function for this to work).

Script-local variables can also be used in functions, autocommands and user commands that are defined in the script. Thus they are the perfect way to share information between parts of your plugin, without it leaking out. In our example we can add a few lines to count the number of corrections:

```
17     var count = 4
...
28     def Add(from: string, correct: bool)
...
32         count += 1
33         echo "you now have " .. count .. " corrections"
34     enddef
```

"count" is declared and initialized to 4 in the script itself. When later the `Add()` function is called, it increments "count". It doesn't matter from where the function was called, since it has been defined in the script, it will use the local variables from this script.

THE RESULT

Here is the resulting complete example:

```
1     vim9script noclear
2     # Vim global plugin for correcting typing mistakes
3     # Last Change: 2021 Dec 30
4     # Maintainer:  Bram Moolenaar <Bram@vim.org>
5     # License:     This file is placed in the public domain.
6
7     if exists("g:loaded_typecorrect")
8         finish
9     endif
10    g:loaded_typecorrect = 1
11
12    iabbrev teh the
13    iabbrev otehr other
14    iabbrev wnat want
15    iabbrev synchronisation
16           \ synchronization
17    var count = 4
18
```

```

19   if !hasmapto('<Plug>TypecorrAdd;')
20     map <unique> <Leader>a <Plug>TypecorrAdd;
21   endif
22   noremap <unique> <script> <Plug>TypecorrAdd; <SID>Add
23
24   noremenu <script> Plugin.Add\ Correction      <SID>Add
25
26   noremap <SID>Add  :call <SID>Add(expand("<cword>"), true)<CR>
27
28   def Add(from: string, correct: bool)
29     var to = input("type the correction for " .. from .. ": ")
30     exe ":iabbrev " .. from .. " " .. to
31     if correct | exe "normal viws\C-R\" \|b\e" | endif
32     count += 1
33     echo "you now have " .. count .. " corrections"
34   enddef
35
36   if !exists(":"Correct")
37     command -nargs=1 Correct call Add(<q-args>, false)
38   endif

```

Line 31 wasn't explained yet. It applies the new correction to the word under the cursor. The `:normal` command is used to use the new abbreviation. **Note** that mappings and abbreviations are expanded here, even though the function was called from a mapping defined with `:"noremap"`.

DOCUMENTATION

[write-local-help](#)

It's a good idea to also write some documentation for your plugin. Especially when its behavior can be changed by the user. See [add-local-help](#) for how they are installed.

Here is a simple example for a plugin help file, called "typecorrect.txt":

```

1   *typecorrect.txt*      Plugin for correcting typing mistakes
2
3   If you make typing mistakes, this plugin will have them corrected
4   automatically.
5
6   There are currently only a few corrections. Add your own if you like.
7
8   Mappings:
9   <Leader>a or <Plug>TypecorrAdd;
10      Add a correction for the word under the cursor.
11
12   Commands:
13   :Correct {word}
14      Add a correction for {word}.
15
16                                          *typecorrect-settings*
17   This plugin doesn't have any settings.

```

The first line is actually the only one for which the format matters. It will

be extracted from the help file to be put in the "LOCAL ADDITIONS:" section of help.txt [local-additions](#) . The first "*" must be in the first column of the first line. After adding your help file do ":help" and check that the entries line up nicely.

You can add more tags inside ** in your help file. But be careful not to use existing help tags. You would probably use the name of your plugin in most of them, like "typecorrect-settings" in the example.

Using references to other parts of the help in || is recommended. This makes it easy for the user to find associated help.

SUMMARY

plugin-special

Summary of special things to use in a plugin:

var name	Variable local to the script.
<SID>	Script-ID, used for mappings and functions local to the script.
hasmapto()	Function to test if the user already defined a mapping for functionality the script offers.
<Leader>	Value of "mapleader", which the user defines as the keys that plugin mappings start with.
map <unique>	Give a warning if a mapping already exists.
noremap <script>	Use only mappings local to the script, not global mappings.
exists(":Cmd")	Check if a user command already exists.

51.2 Writing a filetype plugin [write-filetype-plugin](#) [ftplugin](#)

A filetype plugin is like a global plugin, except that it sets options and defines mappings for the current buffer only. See [add-filetype-plugin](#) for how this type of plugin is used.

First read the section on global plugins above [51.1](#) . All that is said there also applies to filetype plugins. There are a few extras, which are explained here. The essential thing is that a filetype plugin should only have an effect on the current buffer.

DISABLING

If you are writing a filetype plugin to be used by many people, they need a chance to disable loading it. Put this at the top of the plugin:

```
# Only do this when not done yet for this buffer
```

```
if exists("b:did_ftplugin")
  finish
endif
b:did_ftplugin = 1
```

This also needs to be used to avoid that the same plugin is executed twice for the same buffer (happens when using an ":edit" command without arguments).

Now users can disable loading the default plugin completely by making a filetype plugin with only these lines:

```
vim9script
b:did_ftplugin = 1
```

This does require that the filetype plugin directory comes before \$VIMRUNTIME in **'runtimepath'**!

If you do want to use the default plugin, but overrule one of the settings, you can write the different setting in a script:

```
setlocal textwidth=70
```

Now write this in the "after" directory, so that it gets sourced after the distributed "vim.vim" ftplugin **after-directory** . For Unix this would be "~/.vim/after/ftplugin/vim.vim". **Note** that the default plugin will have set "b:did_ftplugin", it is ignored here.

OPTIONS

To make sure the filetype plugin only affects the current buffer use the

```
setlocal
```

command to set options. And only set options which are local to a buffer (see the help for the option to check that). When using `:setlocal`` for global options or options local to a window, the value will change for many buffers, and that is not what a filetype plugin should do.

When an option has a value that is a list of flags or items, consider using "+=" and "-=" to keep the existing value. Be aware that the user may have changed an option value already. First resetting to the default value and then changing it is often a good idea. Example:

```
setlocal formatoptions& formatoptions+=ro
```

MAPPINGS

To make sure mappings will only work in the current buffer use the

```
map <buffer>
```

command. This needs to be combined with the two-step mapping explained above.

An example of how to define functionality in a filetype plugin:

```
if !hasmapto('<Plug>JavaImport;')
  map <buffer> <unique> <LocalLeader>i <Plug>JavaImport;
endif
noremap <buffer> <unique> <Plug>JavaImport; oimport ""<Left><Esc>
```

`hasmapto()` is used to check if the user has already defined a map to `<Plug>JavaImport;`. If not, then the filetype plugin defines the default mapping. This starts with `<LocalLeader>`, which allows the user to select the key(s) they want filetype plugin mappings to start with. The default is a backslash.

"`<unique>`" is used to give an error message if the mapping already exists or overlaps with an existing mapping.

`:noremap` is used to avoid that any other mappings that the user has defined interferes. You might want to use `:noremap <script>` to allow remapping mappings defined in this script that start with `<SID>`.

The user must have a chance to disable the mappings in a filetype plugin, without disabling everything. Here is an example of how this is done for a plugin for the mail filetype:

```
# Add mappings, unless the user didn't want this.
if !exists("g:no_plugin_maps") && !exists("g:no_mail_maps")
  # Quote text by inserting "> "
  if !hasmapto('<Plug>MailQuote;')
    vmap <buffer> <LocalLeader>q <Plug>MailQuote;
    nmap <buffer> <LocalLeader>q <Plug>MailQuote;
  endif
  vnoremap <buffer> <Plug>MailQuote; :s/^/> /<CR>
  nnoremap <buffer> <Plug>MailQuote; :.,$s/^/> /<CR>
endif
```

Two global variables are used:

<code>g:no_plugin_maps</code>	disables mappings for all filetype plugins
<code>g:no_mail_maps</code>	disables mappings for the "mail" filetype

USER COMMANDS

To add a user command for a specific file type, so that it can only be used in one buffer, use the `--buffer` argument to `:command`. Example:

```
command -buffer Make make %:r.s
```

VARIABLES

A filetype plugin will be sourced for each buffer of the type it's for. Local script variables will be shared between all invocations. Use local buffer variables `b:var` if you want a variable specifically for one buffer.

FUNCTIONS

When defining a function, this only needs to be done once. But the filetype plugin will be sourced every time a file with this filetype will be opened. This construct makes sure the function is only defined once:

```
if !exists("*Func")
  def Func(arg)
    ...
  enddef
endif
```

Don't forget to use "noclear" with the ``vim9script`` command to avoid that the function is deleted when the script is sourced a second time.

UNDO

`undo_indent` `undo_ftplugin`

When the user does `":setfiletype xyz"` the effect of the previous filetype should be undone. Set the `b:undo_ftplugin` variable to the commands that will undo the settings in your filetype plugin. Example:

```
b:undo_ftplugin = "setlocal fo< com< tw< commentstring<"
                  \ .. "| unlet b:match_ignorecase b:match_words b:match_skip"
```

Using `":setlocal"` with `"<"` after the option name resets the option to its global value. That is mostly the best way to reset the option value.

For undoing the effect of an indent script, the `b:undo_indent` variable should be set accordingly.

Both these variables use legacy script syntax, not `Vim9` syntax.

FILE NAME

The filetype must be included in the file name `ftplugin-name`. Use one of these three forms:

```
.../ftplugin/stuff.vim
.../ftplugin/stuff_foo.vim
.../ftplugin/stuff/bar.vim
```

"stuff" is the filetype, "foo" and "bar" are arbitrary names.

FILETYPE DETECTION

`plugin-filetype`

If your filetype is not already detected by Vim, you should create a filetype detection snippet in a separate file. It is usually in the form of an autocommand that sets the filetype when the file name matches a pattern. Example:

```
au BufNewFile,BufRead *.foo          setlocal filetype=foofoo
```

Write this single-line file as "ftdetect/foofoo.vim" in the first directory that appears in **'runtimepath'**. For Unix that would be "~/.vim/ftdetect/foofoo.vim". The convention is to use the name of the filetype for the script name.

You can make more complicated checks if you like, for example to inspect the contents of the file to recognize the language. Also see [new-filetype](#) .

SUMMARY

[ftplugin-special](#)

Summary of special things to use in a filetype plugin:

<code><LocalLeader></code>	Value of "maplocalleader", which the user defines as the keys that filetype plugin mappings start with.
<code>map <buffer></code>	Define a mapping local to the buffer.
<code>noremap <script></code>	Only remap mappings defined in this script that start with <code><SID></code> .
<code>setlocal</code>	Set an option for the current buffer only.
<code>command -buffer</code>	Define a user command local to the buffer.
<code>exists("*s:Func")</code>	Check if a function was already defined.

Also see [plugin-special](#) , the special things used for all plugins.

51.3 Writing a compiler plugin

[write-compiler-plugin](#)

A compiler plugin sets options for use with a specific compiler. The user can load it with the `:compiler` command. The main use is to set the **'errorformat'** and **'makeprg'** options.

Easiest is to have a look at examples. This command will edit all the default compiler plugins:

```
next $VIMRUNTIME/compiler/*.vim
```

Type ``:next`` to go to the next plugin file.

There are two special items about these files. First is a mechanism to allow a user to overrule or add to the default file. The default files start with:

```
vim9script
if exists("g:current_compiler")
  finish
endif
g:current_compiler = "mine"
```

When you write a compiler file and put it in your personal runtime directory (e.g., ~/.vim/compiler for Unix), you set the "current_compiler" variable to

make the default file skip the settings.

`:CompilerSet`

The second mechanism is to use `":set"` for `":compiler!"` and `":setlocal"` for `":compiler"`. Vim defines the `":CompilerSet"` user command for this. However, older Vim versions don't, thus your plugin should define it then. This is an example:

```
if exists(":CompilerSet") != 2
  command -nargs=* CompilerSet setlocal <args>
endif
CompilerSet errorformat&          " use the default 'errorformat'
CompilerSet makeprg=nmake
```

When you write a compiler plugin for the Vim distribution or for a system-wide runtime directory, use the mechanism mentioned above. When `"current_compiler"` was already set by a user plugin nothing will be done.

When you write a compiler plugin to overrule settings from a default plugin, don't check `"current_compiler"`. This plugin is supposed to be loaded last, thus it should be in a directory at the end of `'runtimepath'`. For Unix that could be `~/.vim/after/compiler`.

51.4 Distributing Vim scripts

`distribute-script`

Vim users will look for scripts on the Vim website: <http://www.vim.org>. If you made something that is useful for others, share it!

Another place is github. But there you need to know where to find it! The advantage is that most plugin managers fetch plugins from github. You'll have to use your favorite search engine to find them.

Vim scripts can be used on any system. However, there might not be a tar or gzip command. If you want to pack files together and/or compress them the "zip" utility is recommended.

For utmost portability use Vim itself to pack scripts together. This can be done with the `Vimball` utility. See `vimball`.

It's good if you add a line to allow automatic updating. See `glvs-plugins`.

Next chapter: `usr_52.txt` Write large plugins

Copyright: see `manual-copyright` vim:tw=78:ts=8:noet:ft=help:norl:

[usr_52.txt](#) For [Vim version 9.1](#). Last change: 2022 Jun 04

VIM USER MANUAL - by Bram Moolenaar

Write larger plugins

When plugins do more than simple things, they tend to grow big. This file explains how to make sure they still load fast and how to split them up in smaller parts.

- [52.1](#) Export and import
- [52.2](#) Autoloading
- [52.3](#) Autoloading without import/export
- [52.4](#) Other mechanisms to use
- [52.5](#) Using a Vim9 script from legacy script

Next chapter: [usr_90.txt](#) Installing Vim
Previous chapter: [usr_51.txt](#) Create a plugin
Table of contents: [usr_toc.txt](#)

[52.1](#) Export and import

Vim9 script was designed to make it easier to write large Vim scripts. It looks more like other script languages, especially Typescript. Also, functions are compiled into instructions that can be executed quickly. This makes Vim9 script a lot faster, up to a 100 times.

The basic idea is that a script file has items that are private, only used inside the script file, and items that are exported, which can be used by scripts that import them. That makes very clear what is defined where.

Let's start with an example, a script that exports one function and has one private function:

```
vim9script

export def GetMessage(count: string): string
    var nr = str2nr(count)
    var result = $"To {nr} we say "
    result .= GetReply(nr)
    return result
enddef

def GetReply(nr: number): string
    if nr == 42
        return 'yes'
    elseif nr = 22
        return 'maybe'
    else
        return 'no'
    endif
enddef
```

The ``vim9script`` command is required, ``export`` only works in a `Vim9` script.

The ``export def GetMessage(...`` line starts with ``export``, meaning that this function can be called by other scripts. The line ``def GetReply(...`` does not start with ``export``, this is a script-local function, it can only be used inside this script file.

Now about the script where this is imported. In this example we use this layout, which works well for a plugin below the "pack" directory:

```
.../plugin/theplugin.vim
.../lib/getmessage.vim
```

Assuming the "..." directory has been added to `'runtimepath'`, Vim will look for plugins in the "plugin" directory and source "theplugin.vim". Vim does not recognize the "lib" directory, you can put any scripts there.

The above script that exports `GetMessage()` goes in `lib/getmessage.vim`. The `GetMessage()` function is used in `plugin/theplugin.vim`:

```
vim9script

import "../lib/getmessage.vim"
command -nargs=1 ShowMessage echomsg getmessage.GetMessage(<f-args>)
```

The ``import`` command uses a relative path, it starts with `"../"`, which means to go one directory up. For other kinds of paths see the ``:import`` command.

How we can try out the command that the plugin provides:

```
ShowMessage 1
To 1 we say no

ShowMessage 22
To 22 we say maybe
```

Notice that the function `GetMessage()` is prefixed with the imported script name "getmessage". That way, for every imported function used, you know what script it was imported from. If you import several scripts each of them could define a `GetMessage()` function:

```
vim9script

import "../lib/getmessage.vim"
import "../lib/getother.vim"
command -nargs=1 ShowMessage echomsg getmessage.GetMessage(<f-args>)
command -nargs=1 ShowOther echomsg getother.GetMessage(<f-args>)
```

If the imported script name is long or you use it in many places, you can shorten it by adding an "as" argument:

```
import "../lib/getmessage.vim" as msg
command -nargs=1 ShowMessage echomsg msg.GetMessage(<f-args>)
```

RELOADING

One thing to keep in mind: the imported "lib/getmessage.vim" script will be sourced only once. When it is imported a second time sourcing it will be skipped, since the items in it have already been created. It does not matter if this import command is in another script, or in the same script that is sourced again.

This is efficient when using a plugin, but when still developing a plugin it means that changing "lib/getmessage.vim" after it has been imported will have no effect. You need to quit Vim and start it again. (Rationale: the items defined in the script could be used in a compiled function, sourcing the script again may break those functions).

USING GLOBALS

Sometimes you will want to use global variables or functions, so that they can be used anywhere. A good example is a global variable that passes a preference to a plugin. To avoid other scripts using the same name, use a prefix that is very unlikely to be used elsewhere. For example, if you have a "mytags" plugin, you could use:

```
g:mytags_location = '$HOME/project'  
g:mytags_style = 'fast'
```

52.2 Autoloading

After splitting your large script into pieces, all the lines will still be loaded and executed the moment the script is used. Every ``import`` loads the imported script to find the items defined there. Although that is good for finding errors early, it also takes time. Which is wasted if the functionality is not often used.

Instead of having ``import`` load the script immediately, it can be postponed until needed. Using the example above, only one change needs to be made in the plugin/theplugin.vim script:

```
import autoload "./lib/getmessage.vim"
```

Nothing in the rest of the script needs to change. However, the types will not be checked. Not even the existence of the GetMessage() function is checked until it is used. You will have to decide what is more important for your script: fast startup or getting errors early. You can also add the "autoload" argument later, after you have checked everything works.

AUTOLOAD DIRECTORY

Another form is to use autoload with a script name that is not an absolute or relative path:

```
import autoload "monthlib.vim"
```

This will search for the script "monthlib.vim" in the autoload directories of `'runtimepath'`. With Unix one of the directories often is `"~/vim/autoload"`. It will also search under `'packpath'`, under "start".

The main advantage of this is that this script can be easily shared with other scripts. You do need to make sure that the script name is unique, since Vim will search all the "autoload" directories in 'runtimepath', and if you are using several plugins with a plugin manager, it may add a directory to 'runtimepath', each of which might have an "autoload" directory.

Without autoload:

```
import "monthlib.vim"
```

Vim will search for the script "monthlib.vim" in the import directories of 'runtimepath'. Note that in this case adding or removing "autoload" changes where the script is found. With a relative or absolute path the location does not change.

52.3 Autoloading without import/export

write-library-script

A mechanism from before import/export is still useful and some users may find it a bit simpler. The idea is that you call a function with a special name. That function is then in an autoload script. We will call that one script a library script.

The autoload mechanism is based on a function name that has "#" characters:

```
mylib#myfunction(arg)
```

Vim will recognize the function name by the embedded "#" character and when it is not defined yet search for the script "autoload/mylib.vim" in 'runtimepath'. That script must define the "mylib#myfunction()" function. Obviously the name "mylib" is the part before the "#" and is used as the name of the script, adding ".vim".

You can put many other functions in the mylib.vim script, you are free to organize your functions in library scripts. But you must use function names where the part before the '#' matches the script name. Otherwise Vim would not know what script to load. This is where it differs from the import/export mechanism.

If you get really enthusiastic and write lots of library scripts, you may want to use subdirectories. Example:

```
netlib#ftp#read('somefile')
```

Here the script name is taken from the function name up to the last "#". The "#" in the middle are replaced by a slash, the last one by ".vim". Thus you get "netlib/ftp.vim". For Unix the library script used for this could be:

```
~/vim/autoload/netlib/ftp.vim
```

Where the function is defined like this:

```
def netlib#ftp#read(fname: string)
```



```
        # Read the file fname through ftp
    enddef
```

Notice that the name the function is defined with is exactly the same as the name used for calling the function. And the part before the last '#' exactly matches the subdirectory and script name.

You can use the same mechanism for variables:

```
var weekdays = dutch#weekdays
```

This will load the script "autoload/dutch.vim", which should contain something like:

```
var dutch#weekdays = ['zondag', 'maandag', 'dinsdag', 'woensdag',
    \ 'donderdag', 'vrijdag', 'zaterdag']
```

Further reading: [autoload](#) .

52.4 Other mechanisms to use

Some may find the use of several files a hassle and prefer to keep everything together in one script. To avoid this resulting in slow startup there is a mechanism that only defines a small part and postpones the rest to when it is actually used. [write-plugin-quickload](#)

The basic idea is that the plugin is loaded twice. The first time user commands and mappings are defined that offer the functionality. The second time the functions that implement the functionality are defined.

It may sound surprising that quickload means loading a script twice. What we mean is that it loads quickly the first time, postponing the bulk of the script to the second time, which only happens when you actually use it. When you always use the functionality it actually gets slower!

This uses a FuncUndefined autocommand. This works differently from the [autoload](#) functionality explained above.

The following example shows how it's done:

```
" Vim global plugin for demonstrating quick loading
" Last Change: 2005 Feb 25
" Maintainer:  Bram Moolenaar <Bram@vim.org>
" License:     This file is placed in the public domain.

if !exists(":did_load")
    command -nargs=* BNRead call BufNetRead(<f-args>)
    map <F19> :call BufNetWrite('something')<CR>

    let s:did_load = 1
    exe 'au FuncUndefined BufNet* source ' .. expand('<sfile>')
    finish
endif
```

```

function BufNetRead(...)
    echo 'BufNetRead(' .. string(a:000) .. ')'
    " read functionality here
endfunction

function BufNetWrite(...)
    echo 'BufNetWrite(' .. string(a:000) .. ')'
    " write functionality here
endfunction

```

When the script is first loaded "s:did_load" is not set. The commands between the "if" and "endif" will be executed. This ends in a `:finish` command, thus the rest of the script is not executed.

The second time the script is loaded "s:did_load" exists and the commands after the "endif" are executed. This defines the (possible long) BufNetRead() and BufNetWrite() functions.

If you drop this script in your plugin directory Vim will execute it on startup. This is the sequence of events that happens:

1. The "BNRead" command is defined and the <F19> key is mapped when the script is sourced at startup. A `FuncUndefined` autocommand is defined. The `:finish` command causes the script to terminate early.
2. The user types the BNRead command or presses the <F19> key. The BufNetRead() or BufNetWrite() function will be called.
3. Vim can't find the function and triggers the `FuncUndefined` autocommand event. Since the pattern "BufNet*" matches the invoked function, the command "source fname" will be executed. "fname" will be equal to the name of the script, no matter where it is located, because it comes from expanding "<sfile>" (see `expand()`).
4. The script is sourced again, the "s:did_load" variable exists and the functions are defined.

Notice that the functions that are loaded afterwards match the pattern in the `FuncUndefined` autocommand. You must make sure that no other plugin defines functions that match this pattern.

52.5 Using a Vim9 script from legacy script source-vim9-script

In some cases you have a legacy Vim script where you want to use items from a Vim9 script. For example in your `.vimrc` you want to initialize a plugin. The best way to do this is to use `:import`. For example:

```

import 'myNicePlugin.vim'
call myNicePlugin.NiceInit('today')

```

This finds the exported function "NiceInit" in the Vim9 script file and makes it available as script-local item "myNicePlugin.NiceInit". `:import` always

uses the script namespace, even when "s:" is not given. If "myNicePlugin.vim" was already sourced it is not sourced again.

Besides avoiding putting any items in the global namespace (where name clashes can cause unexpected errors), this also means the script is sourced only once, no matter how many times items from it are imported.

In some cases, e.g. for testing, you may just want to source the Vim9 script. That is OK, but then only global items will be available. The Vim9 script will have to make sure to use a unique name for these global items. Example:

```
source ~/.vim/extra/myNicePlugin.vim
call g:NicePluginTest()
```

=====

Next chapter: [usr_90.txt](#) Installing Vim

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

usr_90.txt For Vim version 9.1. Last change: 2022 May 13

VIM USER MANUAL - by Bram Moolenaar

Installing Vim

install

Before you can use Vim you have to install it. Depending on your system it's simple or easy. This chapter gives a few hints and also explains how upgrading to a new version is done.

- 90.1 Unix
- 90.2 MS-Windows
- 90.3 Upgrading
- 90.4 Common installation issues
- 90.5 Uninstalling Vim

Previous chapter: [usr_52.txt](#) Write plugins using Vim9 script
Table of contents: [usr_toc.txt](#)

90.1 Unix

First you have to decide if you are going to install Vim system-wide or for a single user. The installation is almost the same, but the directory where Vim is installed in differs.

For a system-wide installation the base directory `"/usr/local"` is often used. But this may be different for your system. Try finding out where other packages are installed.

When installing for a single user, you can use your home directory as the base. The files will be placed in subdirectories like `"bin"` and `"shared/vim"`.

FROM A PACKAGE

You can get precompiled binaries for many different UNIX systems. There is a long list with links on this page:

<http://www.vim.org/binaries.html>

Volunteers maintain the binaries, so they are often out of date. It is a good idea to compile your own UNIX version from the source. Also, creating the editor from the source allows you to control which features are compiled. This does require a compiler though.

If you have a Linux distribution, the `"vi"` program is probably a minimal version of Vim. It doesn't do syntax highlighting, for example. Try finding another Vim package in your distribution, or search on the web site.

FROM SOURCES

To compile and install Vim, you will need the following:

- A C compiler (GCC preferred)
- The GZIP program (you can get it from www.gnu.org)
- The Vim source and runtime archives

To get the Vim archives, look in this file for a mirror near you, this should provide the fastest download:

```
ftp://ftp.vim.org/pub/vim/MIRRORS
```

Or use the home site ftp.vim.org, if you think it's fast enough. Go to the "unix" directory and you'll find a list of files there. The version number is embedded in the file name. You will want to get the most recent version.

You can get the files for Unix in one big archive that contains everything:

```
vim-8.2.tar.bz2
```

You need the bzip2 program to uncompress it.

COMPILING

First create a top directory to work in, for example:

```
mkdir ~/vim
cd ~/vim
```

Then unpack the archives there. You can unpack it like this:

```
tar xf path/vim-8.2.tar.bz2
```

If your tar command doesn't support bz2 directly:

```
bzip2 -d -c path/vim-8.2.tar.bz2 | tar xf -
```

Change "path" to where you have downloaded the file.

If you are satisfied with getting the default features, and your environment is setup properly, you should be able to compile Vim with just this:

```
cd vim82/src
make
```

The make program will run configure and compile everything. Further on we will explain how to compile with different features.

If there are errors while compiling, carefully look at the error messages. There should be a hint about what went wrong. Hopefully you will be able to correct it. You might have to disable some features to make Vim compile. Look in the Makefile for specific hints for your system.

TESTING

Now you can check if compiling worked OK:

```
make test
```

This will run a sequence of test scripts to verify that Vim works as expected. Vim will be started many times and all kinds of text and messages flash by. If it is alright you will finally see:

```
test results:  
ALL DONE
```

If you get "TEST FAILURE" some test failed. If there are one or two messages about failed tests, Vim might still work, but not perfectly. If you see a lot of error messages or Vim doesn't finish until the end, there must be something wrong. Either try to find out yourself, or find someone who can solve it. You could look in the [maillist-archive](#) for a solution. If everything else fails, you could ask in the vim [maillist](#) if someone can help you.

INSTALLING

[install-home](#)

If you want to install in your home directory, edit the Makefile and search for a line:

```
#prefix = $(HOME)
```

Remove the # at the start of the line.

When installing for the whole system, Vim has most likely already selected a good installation directory for you. You can also specify one, see below. You need to become root for the following.

To install Vim do:

```
make install
```

That should move all the relevant files to the right place. Now you can try running vim to verify that it works. Use two simple tests to check if Vim can find its runtime files:

```
:help  
:syntax enable
```

If this doesn't work, use this command to check where Vim is looking for the runtime files:

```
:echo $VIMRUNTIME
```

You can also start Vim with the "-V" argument to see what happens during startup:

```
vim -V
```

Don't forget that the user manual assumes you Vim in a certain way. After installing Vim, follow the instructions at [not-compatible](#) to make Vim work as assumed in this manual.

SELECTING FEATURES

Vim has many ways to select features. One of the simple ways is to edit the Makefile. There are many directions and examples. Often you can enable or disable a feature by uncommenting a line.

An alternative is to run "configure" separately. This allows you to specify configuration options manually. The disadvantage is that you have to figure out what exactly to type.

Some of the most interesting configure arguments follow. These can also be enabled from the Makefile.

<code>--prefix={directory}</code>	Top directory where to install Vim.
<code>--with-features=tiny</code>	Compile with some features disabled.
<code>--with-features=normal</code>	Compile with more features enabled.
<code>--with-features=huge</code>	Compile with most features enabled. See +feature-list for which feature is enabled in which case.
<code>--enable-perlinterp</code>	Enable the Perl interface. There are similar arguments for ruby, python and tcl.
<code>--disable-gui</code>	Do not compile the GUI interface.
<code>--without-x</code>	Do not compile X-windows features. When both of these are used, Vim will not connect to the X server, which makes startup faster.

To see the whole list use:

```
./configure --help
```

You can find a bit of explanation for each feature, and links for more information here: [feature-list](#) .

For the adventurous, edit the file "feature.h". You can also change the source code yourself!

90.2 MS-Windows

There are two ways to install the Vim program for Microsoft Windows. You can uncompress several archives, or use a self-installing big archive. Most users with fairly recent computers will prefer the second method. For the first one, you will need:

- An archive with binaries for Vim.
- The Vim runtime archive.
- A program to unpack the zip files.

To get the Vim archives, look in this file for a mirror near you, this should provide the fastest download:

```
ftp://ftp.vim.org/pub/vim/MIRRORS
```


Or use the home site ftp.vim.org, if you think it's fast enough. Go to the "pc" directory and you'll find a list of files there. The version number is embedded in the file name. You will want to get the most recent version. We will use "82" here, which is version 8.2.

gvim82.exe The self-installing archive.

This is all you need for the second method. Just launch the executable, and follow the prompts.

For the first method you must choose one of the binary archives. These are available:

gvim82.zip The normal MS-Windows GUI version.
gvim82ole.zip The MS-Windows GUI version with OLE support.
 Uses more memory, supports interfacing with
 other OLE applications.
vim82w32.zip 32 bit MS-Windows console version.

You only need one of them. Although you could install both a GUI and a console version. You always need to get the archive with runtime files.

vim82rt.zip The runtime files.

Use your un-zip program to unpack the files. For example, using the "unzip" program:

```
cd c:\
unzip path\gvim82.zip
unzip path\vim82rt.zip
```

This will unpack the files in the directory "c:\vim\vim82". If you already have a "vim" directory somewhere, you will want to move to the directory just above it.

Now change to the "vim\vim82" directory and run the install program:

```
install
```

Carefully look through the messages and select the options you want to use. If you finally select "do it" the install program will carry out the actions you selected.

The install program doesn't move the runtime files. They remain where you unpacked them.

In case you are not satisfied with the features included in the supplied binaries, you could try compiling Vim yourself. Get the source archive from the same location as where the binaries are. You need a compiler for which a makefile exists. Microsoft Visual C, MinGW and Cygwin compilers can be used. Check the file src/INSTALLpc.txt for hints.

90.3 Upgrading

If you are running one version of Vim and want to install another, here is what to do.

UNIX

When you type "make install" the runtime files will be copied to a directory which is specific for this version. Thus they will not overwrite a previous version. This makes it possible to use two or more versions next to each other.

The executable "vim" will overwrite an older version. If you don't care about keeping the old version, running "make install" will work fine. You can delete the old runtime files manually. Just delete the directory with the version number in it and all files below it. Example:

```
rm -rf /usr/local/share/vim/vim74
```

There are normally no changed files below this directory. If you did change the "filetype.vim" file, for example, you better merge the changes into the new version before deleting it.

If you are careful and want to try out the new version for a while before switching to it, install the new version under another name. You need to specify a configure argument. For example:

```
./configure --with-vim-name=vim8
```

Before running "make install", you could use "make -n install" to check that no valuable existing files are overwritten.

When you finally decide to switch to the new version, all you need to do is to rename the binary to "vim". For example:

```
mv /usr/local/bin/vim8 /usr/local/bin/vim
```

MS-WINDOWS

Upgrading is mostly equal to installing a new version. Just unpack the files in the same place as the previous version. A new directory will be created, e.g., "vim82", for the files of the new version. Your runtime files, vimrc file, viminfo, etc. will be left alone.

If you want to run the new version next to the old one, you will have to do some handwork. Don't run the install program, it will overwrite a few files of the old version. Execute the new binaries by specifying the full path. The program should be able to automatically find the runtime files for the right version. However, this won't work if you set the \$VIMRUNTIME variable somewhere.

If you are satisfied with the upgrade, you can delete the files of the previous version. See [90.5](#) .

90.4 Common installation issues

This section describes some of the common problems that occur when installing

Vim and suggests some solutions. It also contains answers to many installation questions.

Q: I Do Not Have Root Privileges. How Do I Install Vim? (Unix)

Use the following configuration command to install Vim in a directory called \$HOME/vim:

```
./configure --prefix=$HOME
```

This gives you a personal copy of Vim. You need to put \$HOME/bin in your path to execute the editor. Also see [install-home](#) .

Q: The Colors Are Not Right on My Screen. (Unix)

Check your terminal settings by using the following command in a shell:

```
echo $TERM
```

If the terminal type listed is not correct, fix it. For more hints, see [06.2](#) . Another solution is to always use the GUI version of Vim, called gvim. This avoids the need for a correct terminal setup.

Q: My Backspace And Delete Keys Don't Work Right

The definition of what key sends what code is very unclear for backspace <BS> and Delete keys. First of all, check your \$TERM setting. If there is nothing wrong with it, try this:

```
:set t_kb=^V<BS>
:set t_kD=^V<Del>
```

In the first line you need to press **CTRL-V** and then hit the backspace key. In the second line you need to press **CTRL-V** and then hit the Delete key. You can put these lines in your vimrc file, see [05.1](#) . A disadvantage is that it won't work when you use another terminal some day. Look here for alternate solutions: [:fixdel](#) .

Q: I Am Using RedHat Linux. Can I Use the Vim That Comes with the System?

By default RedHat installs a minimal version of Vim. Check your RPM packages for something named "Vim-enhanced-version.rpm" and install that.

Q: How Do I Turn Syntax Coloring On? How do I make plugins work?

Use the example vimrc script. You can find an explanation on how to use it here: [not-compatible](#) .

See chapter 6 for information about syntax highlighting: [usr_06.txt](#) .

Q: What Is a Good vimrc File to Use?

See the www.vim.org Web site for several good examples.

Q: Where Do I Find a Good Vim Plugin?

See the Vim-online site: <http://vim.sf.net>. Many users have uploaded useful Vim scripts and plugins there.

Q: Where Do I Find More Tips?

See the Vim-online site: <http://vim.sf.net>. There is an archive with hints from Vim users. You might also want to search in the [maillist-archive](#) .

90.5 Uninstalling Vim

In the unlikely event you want to uninstall Vim completely, this is how you do it.

UNIX

When you installed Vim as a package, check your package manager to find out how to remove the package again.

If you installed Vim from sources you can use this command:

```
make uninstall
```

However, if you have deleted the original files or you used an archive that someone supplied, you can't do this. Do delete the files manually, here is an example for when `"/usr/local"` was used as the root:

```
rm -rf /usr/local/share/vim/vim82
rm /usr/local/bin/evim
rm /usr/local/bin/evim
rm /usr/local/bin/ex
rm /usr/local/bin/gvim
rm /usr/local/bin/gvim
rm /usr/local/bin/gvimdiff
rm /usr/local/bin/rgvim
rm /usr/local/bin/rgvim
rm /usr/local/bin/rvim
rm /usr/local/bin/rvim
rm /usr/local/bin/view
rm /usr/local/bin/vim
rm /usr/local/bin/vimdiff
rm /usr/local/bin/vimtutor
```

```
rm /usr/local/bin/xxd
rm /usr/local/man/man1/evim.1
rm /usr/local/man/man1/ex.1
rm /usr/local/man/man1/gvim.1
rm /usr/local/man/man1/gvimdiff.1
rm /usr/local/man/man1/rgvim.1
rm /usr/local/man/man1/rvim.1
rm /usr/local/man/man1/view.1
rm /usr/local/man/man1/vim.1
rm /usr/local/man/man1/vimdiff.1
rm /usr/local/man/man1/vimtutor.1
rm /usr/local/man/man1/xxd.1
```

MS-WINDOWS

If you installed Vim with the self-installing archive you can run the "uninstall-gui" program located in the same directory as the other Vim programs, e.g. "c:\vim\vim82". You can also launch it from the Start menu if installed the Vim entries there. This will remove most of the files, menu entries and desktop shortcuts. Some files may remain however, as they need a Windows restart before being deleted.

You will be given the option to remove the whole "vim" directory. It probably contains your vimrc file and other runtime files that you created, so be careful.

Else, if you installed Vim with the zip archives, the preferred way is to use the "uninstall" program. You can find it in the same directory as the "install" program, e.g., "c:\vim\vim82". This should also work from the usual "install/remove software" page.

However, this only removes the registry entries for Vim. You have to delete the files yourself. Simply select the directory "vim\vim82" and delete it recursively. There should be no files there that you changed, but you might want to check that first.

The "vim" directory probably contains your vimrc file and other runtime files that you created. You might want to keep that.

=====

Table of contents: [usr_toc.txt](#)

Copyright: see [manual-copyright](#) vim:tw=78:ts=8:noet:ft=help:norl:

Introduction to Vim

[ref](#) [reference](#)

1. Introduction	intro
2. Vim on the internet	internet
3. Credits	credits
4. Notation	notation
5. Modes, introduction	vim-modes-intro
6. Switching from mode to mode	mode-switching
7. The window contents	window-contents
8. Definitions	definitions

1. Introduction

[intro](#)

Vim stands for Vi IMproved. It used to be Vi IMitation, but there are so many improvements that a name change was appropriate. Vim is a text editor which includes almost all the commands from the Unix program "Vi" and a lot of new ones. It is very useful for editing programs and other plain text.

All commands are given with the keyboard. This has the advantage that you can keep your fingers on the keyboard and your eyes on the screen. For those who want it, there is mouse support and a GUI version with scrollbars and menus (see [gui.txt](#)).

An overview of this manual can be found in the file "help.txt", [help.txt](#). It can be accessed from within Vim with the [<Help>](#) or [<F1>](#) key and with the [:help](#) command (just type [":help"](#), without the bars or quotes).

The ['helpfile'](#) option can be set to the name of the help file, in case it is not located in the default place. You can jump to subjects like with tags: Use [CTRL-\]](#) to jump to a subject under the cursor, use [CTRL-T](#) to jump back.

The differences between Vi and Vim are mentioned in [vi_diff.txt](#).

This manual refers to Vim on various machines. There may be small differences between different computers and terminals. Besides the remarks given in this document, there is a separate document for each supported system, see [sys-file-list](#).

[pronounce](#)

Vim is pronounced as one word, like Jim, not vi-ai-em. It's written with a capital, since it's a name, again like Jim. The GUI version of Vim is written "gVim" (or "GVim" when at the beginning of a sentence).

This manual is a reference for all the Vim commands and options. This is not an introduction to the use of Vi or Vim, it gets a bit complicated here and there. For beginners, there is a hands-on [tutor](#). To learn using Vim, read the user manual [usr_toc.txt](#).

[book](#) [books](#)

Most books on Vi and Vim contain a section for beginners. Others are spending more words on specific functionality. You can find an overview of Vim books here:

http://iccf-holland.org/vim_books.html

2. Vim on the internet

internet

[www](#) [WWW](#) [faq](#) [FAQ](#) [distribution](#) [download](#)

The Vim pages contain the most recent information about Vim. They also contain links to the most recent version of Vim. The FAQ is a list of Frequently Asked Questions. Read this if you have problems.

Vim home page: <https://www.vim.org/>
Vim FAQ: https://vimhelp.org/vim_faq.txt.html
Downloading: <https://www.vim.org/download.php>

Asking questions, finding answers: <https://vi.stackexchange.com/>
"Vi and Vim Stack Exchange is a question and answer site for people using the vi and Vim families of text editors"

Usenet News group where Vim is discussed: [news](#) [usenet](#)
[comp.editors](#)

This group is also for other editors. If you write about Vim, don't forget to mention that.

You can access it here:

<https://groups.google.com/forum/#!topic/comp.editors>

[mail-list](#) [maillist](#)

There are several mailing lists for Vim:

[<vim@vim.org>](mailto:vim@vim.org)

[vim-use](#) [vim_use](#)

For discussions about using existing versions of Vim: Useful mappings, questions, answers, where to get a specific version, etc. There are quite a few people watching this list and answering questions, also for beginners. Don't hesitate to ask your question here.

[<vim-dev@vim.org>](mailto:vim-dev@vim.org)

[vim-dev](#) [vim_dev](#) [vimdev](#)

For discussions about changing Vim: New features, porting, patches, beta-test versions, etc.

[<vim-announce@vim.org>](mailto:vim-announce@vim.org)

[vim-announce](#) [vim_announce](#)

Announcements about new versions of Vim; also for beta-test versions and ports to different systems. This is a read-only list.

[<vim-mac@vim.org>](mailto:vim-mac@vim.org)

[vim-mac](#) [vim_mac](#)

For discussions about using and improving the Macintosh version of Vim.

[<vim-security@googlegroups.com>](mailto:vim-security@googlegroups.com)

[vim-security](#)

This list is for (privately) discussing security relevant issues of Vim.

See <http://www.vim.org/maillist.php> for the latest information.

NOTE:

- Anyone can see the archive, e.g. on Google groups. Search this if you have questions, except for the vim-security list.

- You can only send messages to these lists if you have subscribed!
- The first message is moderated, thus it may take a few hours to show up.
- You need to send the messages from the same location as where you subscribed from (to avoid spam mail).

subscribe-maillist

If you want to join, send a message to
[<vim-subscribe@vim.org>](mailto:vim-subscribe@vim.org)

Make sure that your "From:" address is correct. Then the list server will give you help on how to subscribe.

maillist-archive

For more information and archives look on the Vim maillist page:
<http://www.vim.org/maillist.php>

Bug reports:

[bugs](#) [bug-reports](#) [bugreport.vim](#)

There are three ways to report bugs:

1. For issues with runtime files, look in the header for an email address or any other way to report it to the maintainer.
2. Open an issue on GitHub: <https://github.com/vim/vim/issues>
The text will be forwarded to the vim-dev maillist.
3. Send bug reports to: Vim Developers [<vim-dev@vim.org>](mailto:vim-dev@vim.org)
This is a maillist, you need to become a member first and many people will see the message. If you don't want that, e.g. because it is a security issue, please contact any of the current Vim maintainers
<https://github.com/orgs/vim/people> (but not Bram or the vim-dev ML).
In the future, a proper process for handling security issues will be setup.

Please be brief; all the time that is spent on answering mail is subtracted from the time that is spent on improving Vim! Always give a reproducible example and try to find out which settings or other things trigger the bug.

Preferably start Vim with:

```
vim --clean -u reproduce.vim
```

Where reproduce.vim is a script that reproduces the problem. Try different machines, if relevant (is this an MS-Windows specific bug perhaps?).

Send me patches if you can! If you create a pull request on <https://github.com/vim/vim> then the automated checks will run and report any obvious problems. But you can also send the patch by email (use an attachment to avoid white space changes).

It will help to include information about the version of Vim you are using and your setup. You can get the information with this command:

```
:so $VIMRUNTIME/bugreport.vim
```

This will create a file "bugreport.txt" in the current directory, with a lot of information of your environment. Before sending this out, check if it doesn't contain any confidential information!

If Vim crashes, please try to find out where. You can find help on this here:
[debug.txt](#) .

In case of doubt or when you wonder if the problem has already been fixed but you can't find a fix for it, become a member of the vim-dev maillist and ask your question there. [maillist](#)

[year-2000](#) [Y2K](#)

Since Vim internally doesn't use dates for editing, there is no year 2000 problem to worry about. Vim does use the time in the form of seconds since January 1st 1970. It is used for a time-stamp check of the edited file and the swap file, which is not critical and should only cause warning messages.

There might be a year 2038 problem, when the seconds don't fit in a 32 bit int anymore. This depends on the compiler, libraries and operating system. Specifically, `time_t` and the `ctime()` function are used. And the `time_t` is stored in four bytes in the swap file. But that's only used for printing a file date/time for recovery, it will never affect normal editing.

The Vim `strftime()` function directly uses the `strftime()` system function. `localtime()` uses the `time()` system function. `getftime()` uses the time returned by the `stat()` system function. If your system libraries are year 2000 compliant, Vim is too.

The user may create scripts for Vim that use external commands. These might introduce Y2K problems, but those are not really part of Vim itself.

3. Credits [credits](#) [author](#)

Most of Vim was created by Bram Moolenaar [<Bram@vim.org>](mailto:Bram@vim.org) [Bram-Moolenaar](#) .

Parts of the documentation come from several Vi manuals, written by:

- W.N. Joy
- Alan P.W. Hewett
- Mark Horton

The Vim editor is based on Stevie and includes (ideas from) other software, worked on by the people mentioned here. Other people helped by sending me patches, suggestions and giving feedback about what is good and bad in Vim.

Vim would never have become what it is now, without the help of these people!

Ron Aaron	Win32 GUI changes
Mohsin Ahmed	encryption
Zoltan Arpadffy	work on VMS port
Tony Andrews	Stevie
Gert van Antwerpen	changes for DJGPP on MS-DOS
Berkeley DB(3)	ideas for swap file implementation
Keith Bostic	Nvi
Walter Briscoe	Makefile updates, various patches
Ralf Brown	SPAWN library for MS-DOS
Robert Colon	many useful remarks
Marcin Dalecki	GTK+ GUI port, toolbar icons, <code>gettext()</code>
Kayhan Demirel	sent me news in Uganda
Chris & John Downey	xvi (ideas for multi-windows version)
Henk Elbers	first VMS port

Daniel Elstner	GTK+ 2 port
Eric Fischer	Mac port, ' cindent ', and other improvements
Benji Fisher	Answering lots of user questions
Bill Foster	Athena GUI port (later removed)
Google	Let Bram work on Vim one day a week
Loic Grenie	xvim (ideas for multi windows version)
Sven Guckes	Vim promoter and previous WWW page maintainer Sven-Guckes
Darren Hiebert	Exuberant ctags
Jason Hildebrand	GTK+ 2 port
Bruce Hunsaker	improvements for VMS port
Andy Kahn	Cscope support, GTK+ GUI port
Oezguer Kesim	Maintainer of Vim Mailing Lists
Axel Kielhorn	work on the Macintosh port
Steve Kirkendall	Elvis
Roger Knobbe	original port to Windows NT
Sergey Laskavy	Vim's help from Moscow
Felix von Leitner	Previous maintainer of Vim Mailing Lists
David Leonard	Port of Python extensions to Unix
Avner Lottem	Edit in right-to-left windows
Flemming Madsen	X11 client-server, various features and patches
Tony Mechelynck	answers many user questions
Paul Moore	Python interface extensions, many patches
Katsuhito Nagano	Work on multibyte versions
Sung-Hyun Nam	Work on multibyte versions
Vince Negri	Win32 GUI and generic console enhancements
Steve Oualline	Author of the first Vim book frombook
Dominique Pelle	Valgrind reports and many fixes
A.Politz	Many bug reports and some fixes
George V. Reilly	Win32 port, Win32 GUI start-off
Stephen Riehm	bug collector
Stefan Roemer	various patches and help to users
Ralf Schandl	IBM OS/390 port
Olaf Seibert	DICE and BeBox version, regexp improvements
Mortaza Shiran	Farsi patches
Peter da Silva	termlib
Paul Slotman	OS/2 port
Henry Spencer	regular expressions
Dany St-Amant	Macintosh port
Tim Thompson	Stevie
G. R. (Fred) Walter	Stevie
Sven Verdoolaege	Perl interface
Robert Webb	Command-line completion, GUI versions, and lots of patches
Ingo Wilken	Tcl interface
Mike Williams	PostScript printing
Juergen Weigert	Lattice version, AUX improvements, UNIX and MS-DOS ports, autoconf
Stefan 'Sec' Zehl	Maintainer of vim.org
Yasuhiro Matsumoto	many MS-Windows improvements
Ken Takata	fixes and features
Kazunobu Kuriyama	GTK 3
Christian Brabandt	many fixes, features, user support, etc.
Yegappan Lakshmanan	many quickfix features

I wish to thank all the people that sent me bug reports and suggestions. The list is too long to mention them all here. Vim would not be the same without the ideas from all these people: They keep Vim alive!

love peace friendship gross-national-happiness

In this documentation there are several references to other versions of Vi:

- Vi** **vi**
"the original". Without further remarks this is the version of Vi that appeared in Sun OS 4.x. `":version"` returns "Version 3.7, 6/7/85". Sometimes other versions are referred to. Only runs under Unix. Source code is now available under a BSD-style license. More information on Vi can be found through:
<http://ex-vi.sourceforge.net/>
- Posix**
From the IEEE standard 1003.2, Part 2: Shell and utilities. Generally known as "Posix". This is a textual description of how Vi is supposed to work. See [posix-compliance](#) .
- Nvi**
The "New" Vi. The version of Vi that comes with BSD 4.4 and FreeBSD. Very good compatibility with the original Vi, with a few extensions. The version used is 1.79. `":version"` returns "Version 1.79 (10/23/96)". There has been no release the last few years, although there is a development version 1.81. Source code is freely available.
- Elvis**
Another Vi clone, made by Steve Kirkendall. Very compact but isn't as flexible as Vim. Development has stalled, Elvis has left the building! Source code is freely available.
- Neovim**
A Vim clone. Forked the Vim source in 2014 and went a different way. Very much bound to github and has many more dependencies, making development more complex and limiting portability. Code has been refactored, resulting in patches not being exchangeable with Vim. Supports a remote GUI and integration with scripting languages.

4. Notation

notation

When syntax highlighting is used to read this, text that is not typed literally is often highlighted with the Special group. These are items in `[]`, `{}` and `<>`, and **CTRL-X**.

Note that Vim uses all possible characters in commands. Sometimes the `[]`, `{}` and `<>` are part of what you type, the context should make this clear.

`[]` Characters in square brackets are optional.

`[count]` **count** `[count]`
An optional number that may precede the command to multiply or iterate the command. If no number is given, a count of one

is used, unless otherwise noted. **Note** that in this manual the [count] is not mentioned in the description of the command, but only in the explanation. This was done to make the commands easier to look up. If the 'showcmd' option is on, the (partially) entered count is shown at the bottom of the window. You can use to erase the last digit (N).

[**"x**] An optional register designation where text can be stored. See registers . The x is a single character between 'a' and 'z' or 'A' and 'Z' or "'", and in some cases (with the put command) between '0' and '9', '%', '#', or others. The uppercase and lowercase letter designate the same register, but the lowercase letter is used to overwrite the previous register contents, while the uppercase letter is used to append to the previous register contents. Without the ""x" or with "" the stored text is put into the unnamed register.

{ } Curly braces denote parts of the command which must appear, but which can take a number of different values. The differences between Vim and Vi are also given in curly braces (this will be clear from the context).

{char1-char2} A single character from the range char1 to char2. For example: {a-z} is a lowercase letter. Multiple ranges may be concatenated. For example, {a-zA-Z0-9} is any alphanumeric character.

{motion} A command that moves the cursor. These are explained in motion.txt . Examples:

w	to start of next word
b	to begin of current word
4j	four lines down
/The<CR>	to next occurrence of "The"

This is used after an operator command to move over the text that is to be operated upon.

- If the motion includes a count and the operator also has a count, the two counts are multiplied. For example: "2d3w" deletes six words.
- The motion can be backwards, e.g. "db" to delete to the start of the word.
- The motion can also be a mouse click. The mouse is not supported in every terminal though.
- The ":omap" command can be used to map characters while an operator is pending.
- Ex commands can be used to move the cursor. This can be used to call a function that does some complicated motion. The motion is always characterwise exclusive, no matter what ":" command is used. This means it's impossible to include the last character of a line without the line break (unless 'virtualedit' is set).

<Space>	space	32	space
<lt>	less-than	< 60	*<lt>*
<Bslash>	backslash	\ 92	backslash <Bslash>
<Bar>	vertical bar	124	<Bar>
	delete	127	
<CSI>	command sequence intro	ALT-Esc 155	<CSI>
<xCSI>	CSI when typed in the GUI		<xCSI>
<EOL>	end-of-line (can be <CR>, <NL> or <CR><NL>, depends on system and 'fileformat')		<EOL>
<Up>	cursor-up		cursor-up cursor_up
<Down>	cursor-down		cursor-down cursor_down
<Left>	cursor-left		cursor-left cursor_left
<Right>	cursor-right		cursor-right cursor_right
<S-Up>	shift-cursor-up		
<S-Down>	shift-cursor-down		
<S-Left>	shift-cursor-left		
<S-Right>	shift-cursor-right		
<C-Left>	control-cursor-left		
<C-Right>	control-cursor-right		
<F1> - <F12>	function keys 1 to 12		function_key function-key
<S-F1> - <S-F12>	shift-function keys 1 to 12		<S-F1>
<Help>	help key		
<Undo>	undo key		
<Insert>	insert key		
<Home>	home		home
<End>	end		end
<PageUp>	page-up		page_up page-up
<PageDown>	page-down		page_down page-down
<kHome>	keypad home (upper left)		keypad-home
<kEnd>	keypad end (lower left)		keypad-end
<kPageUp>	keypad page-up (upper right)		keypad-page-up
<kPageDown>	keypad page-down (lower right)		keypad-page-down
<kPlus>	keypad +		keypad-plus
<kMinus>	keypad -		keypad-minus
<kMultiply>	keypad *		keypad-multiply
<kDivide>	keypad /		keypad-divide
<kEnter>	keypad Enter		keypad-enter
<kPoint>	keypad Decimal point		keypad-point
<k0> - <k9>	keypad 0 to 9		keypad-0 keypad-9
<S-...>	shift-key		shift <S-
<C-...>	control-key		control ctrl <C-
<M-...>	alt-key or meta-key		meta alt <M-
<A-...>	same as <M-...>		<A-
<D-...>	command-key (Mac) / super (GTK)		<D-
<t_xx>	key with "xx" entry in termcap		

Note: The shifted cursor keys, the help key, and the undo key are only available on a few terminals. On the Amiga, shifted function key 10 produces a code (CSI) that is also used by key sequences. It will be recognized only after typing another key.

Note: There are two codes for the delete key. 127 is the decimal ASCII value for the delete key, which is always recognized. Some delete keys send another value, in which case this value is obtained from the termcap entry "kD". Both values have the same effect. Also see `:fixdel`.

Note: The keypad keys are used in the same way as the corresponding "normal" keys. For example, `<kHome>` has the same effect as `<Home>`. If a keypad key sends the same raw key code as its non-keypad equivalent, it will be recognized as the non-keypad code. For example, when `<kHome>` sends the same code as `<Home>`, when pressing `<kHome>` Vim will think `<Home>` was pressed. Mapping `<kHome>` will not work then.

`<>`

Examples are often given in the `<>` notation. Sometimes this is just to make clear what you need to type, but often it can be typed literally, e.g., with the `":map"` command. The rules are:

1. Any printable characters are typed directly, except backslash and '<'
2. A backslash is represented with `"\\"`, double backslash, or `"<Bslash>"`.
3. A real '<' is represented with `"\<"` or `"<lt>"`. When there is no confusion possible, a '<' can be used directly.
4. `"<key>"` means the special key typed. This is the notation explained in the table above. A few examples:

<code><Esc></code>	Escape key
<code><C-G></code>	CTRL-G
<code><Up></code>	cursor up key
<code><C-LeftMouse></code>	Control- left mouse click
<code><S-F11></code>	Shifted function key 11
<code><M-a></code>	Meta- a ('a' with bit 8 set)
<code><M-A></code>	Meta- A ('A' with bit 8 set)
<code><t_kd></code>	"kd" termcap entry (cursor down key)

Although you can specify `<M-{char}>` with `{char}` being a multibyte character, Vim may not be able to know what byte sequence that is and then it won't work.

If you want to use the full `<>` notation in Vim, you have to make sure the '<' flag is excluded from `'cptions'` (when `'compatible'` is not set, it already is by default).

```
:set cpo==<
```

The `<>` notation uses `<lt>` to escape the special meaning of key names. Using a backslash also works, but only when `'cptions'` does not include the 'B' flag.

Examples for mapping **CTRL-H** to the six characters `"<Home>"`:

```
:imap <C-H> \<Home>
:imap <C-H> <lt>Home>
```

The first one only works when the 'B' flag is not in `'cptions'`. The second one always works.

To get a literal `"<lt>"` in a mapping:

```
:map <C-L> <lt>lt>
```

For mapping, abbreviation and menu commands you can then copy-paste the examples and use them directly. Or type them literally, including the '<' and '>' characters. This does NOT work for other commands, like `":set"` and `":autocmd"`!

The notation can be used in a double quoted strings, using "\<" at the start, e.g. "\<C-Space>". This results in a special key code. To convert this back to readable text use ``keytrans()``.

5. Modes, introduction

[vim-modes-intro](#) [vim-modes](#)

Vim has seven BASIC modes:

	Normal	Normal-mode	command-mode
Normal mode	In Normal mode you can enter all the normal editor commands. If you start the editor you are in this mode (unless you have set the <code>'insertmode'</code> option, see below). This is also known as command mode.		
Visual mode	This is like Normal mode, but the movement commands extend a highlighted area. When a non-movement command is used, it is executed for the highlighted area. See Visual-mode . If the <code>'showmode'</code> option is on "-- VISUAL --" is shown at the bottom of the window.		
Select mode	This looks most like the MS-Windows selection mode. Typing a printable character deletes the selection and starts Insert mode. See Select-mode . If the <code>'showmode'</code> option is on "-- SELECT --" is shown at the bottom of the window.		
Insert mode	In Insert mode the text you type is inserted into the buffer. See Insert-mode . If the <code>'showmode'</code> option is on "-- INSERT --" is shown at the bottom of the window.		
Command-line mode Cmdline mode	In Command-line mode (also called Cmdline mode) you can enter one line of text at the bottom of the window. This is for the Ex commands, ":", the pattern search commands, "?" and "/", and the filter command, "!". Cmdline-mode		
Ex mode	Like Command-line mode, but after entering a command you remain in Ex mode. Very limited editing of the command line. Ex-mode		
Terminal-Job mode	Interacting with a job in a terminal window. Typed keys go to the job and the job output is displayed in the terminal window. See terminal about how to switch to other modes.		

There are seven ADDITIONAL modes. These are variants of the BASIC modes:

	Operator-pending	Operator-pending-mode
Operator-pending mode	This is like Normal mode, but after an operator command has started, and Vim is waiting for a <code>{motion}</code> to specify the text that the operator will work on.	

Replace mode Replace mode is a special case of Insert mode. You can do the same things as in Insert mode, but for each character you enter, one character of the existing text is deleted. See [Replace-mode](#) . If the '**showmode**' option is on "-- REPLACE --" is shown at the bottom of the window.

Virtual Replace mode Virtual Replace mode is similar to Replace mode, but instead of file characters you are replacing screen real estate. See [Virtual-Replace-mode](#) . If the '**showmode**' option is on "-- VREPLACE --" is shown at the bottom of the window.

Insert Normal mode Entered when **CTRL-O** is typed in Insert mode (see [i_CTRL-O](#)). This is like Normal mode, but after executing one command Vim returns to Insert mode. If the '**showmode**' option is on "-- (insert) --" is shown at the bottom of the window.

Terminal-Normal mode Using Normal mode in a terminal window. Making changes is impossible. Use an insert command, such as "a" or "i", to return to Terminal-Job mode.

Insert Visual mode Entered when starting a Visual selection from Insert mode, e.g., by using **CTRL-O** and then "v", "V" or **CTRL-V**. When the Visual selection ends, Vim returns to Insert mode. If the '**showmode**' option is on "-- (insert) VISUAL --" is shown at the bottom of the window.

Insert Select mode Entered when starting Select mode from Insert mode. E.g., by dragging the mouse or <S-Right>. When the Select mode ends, Vim returns to Insert mode. If the '**showmode**' option is on "-- (insert) SELECT --" is shown at the bottom of the window.

===== mode-switching

6. Switching from mode to mode

If for any reason you do not know which mode you are in, you can always get back to Normal mode by typing <Esc> twice. This doesn't work for Ex mode though, use ":visual". You will know you are back in Normal mode when you see the screen flash or hear the bell after you type <Esc>. However, when pressing <Esc> after using **CTRL-O** in Insert mode you get a beep but you are still in Insert mode, type <Esc> again.

	i_esc						
TO mode	Normal	Visual	Select	Insert	Replace	Cmd-line	Ex
FROM mode							
Normal		v V ^V	*4	*1	R gR	: / ? !	Q
Visual	*2		^G	c C	--	:	--

Select	*5	^O ^G		*6	--	--	--
Insert	<Esc>	--	--		<Insert>	--	--
Replace	<Esc>	--	--	<Insert>		--	--
Command-line	*3	--	--	:start	--	--	--
Ex	:vi	--	--	--	--	--	--

-- not possible

- *1 Go from Normal mode to Insert mode by giving the command "i", "I", "a", "A", "o", "O", "c", "C", "s" or "S".
- *2 Go from Visual mode to Normal mode by giving a non-movement command, which causes the command to be executed, or by hitting <Esc> "v", "V" or "**CTRL-V**" (see v_v), which just stops Visual mode without side effects.
- *3 Go from Command-line mode to Normal mode by:
 - Hitting <CR> or <NL>, which causes the entered command to be executed.
 - Deleting the complete line (e.g., with **CTRL-U**) and giving a final <BS>.
 - Hitting **CTRL-C** or <Esc>, which quits the command-line without executing the command.
 In the last case <Esc> may be the character defined with the 'wildchar' option, in which case it will start command-line completion. You can ignore that and type <Esc> again.
- *4 Go from Normal to Select mode by:
 - use the mouse to select text while 'selectmode' contains "mouse"
 - use a non-printable command to move the cursor while keeping the Shift key pressed, and the 'selectmode' option contains "key"
 - use "v", "V" or "**CTRL-V**" while 'selectmode' contains "cmd"
 - use "gh", "gH" or "g **CTRL-H**" g_CTRL-H
- *5 Go from Select mode to Normal mode by using a non-printable command to move the cursor, without keeping the Shift key pressed.
- *6 Go from Select mode to Insert mode by typing a printable character. The selection is deleted and the character is inserted.

If the 'insertmode' option is on, editing a file will start in Insert mode.

CTRL-_CTRL-N i_CTRL-_CTRL-N c_CTRL-_CTRL-N v_CTRL-_CTRL-N
 Additionally the command CTRL-\ **CTRL-N** or <C-\><C-N> can be used to go to Normal mode from any other mode. This can be used to make sure Vim is in Normal mode, without causing a beep like <Esc> would. However, this does not work in Ex mode. When used after a command that takes an argument, such as f or m , the timeout set with 'timeoutlen' applies.

When focus is in a terminal window, CTRL-\ **CTRL-N** goes to Normal mode until an edit command is entered, see t_CTRL-_CTRL-N .

CTRL-_CTRL-G i_CTRL-_CTRL-G c_CTRL-_CTRL-G v_CTRL-_CTRL-G
 The command CTRL-\ **CTRL-G** or <C-\><C-G> can be used to go to Insert mode when 'insertmode' is set. Otherwise it goes to Normal mode. This can be used to make sure Vim is in the mode indicated by 'insertmode', without knowing in what mode Vim currently is.

- Q mode-Ex Ex-mode Ex EX E501
 Switch to "Ex" mode. This is a bit like typing ":" commands one after another, except:
- You don't have to keep pressing ":".
 - The screen doesn't get updated after each command.

- There is no normal command-line editing.
- Mappings and abbreviations are not used.

In fact, you are editing the lines with the "standard" line-input editing commands (or <BS> to erase, **CTRL-U** to kill the whole line). Vim will enter this mode by default if it's invoked as "ex" on the command-line or the `-e` command line argument was used. Use the `":vi"` command `:visual` to exit "Ex" mode. **Note:** In older versions of Vim "Q" formatted text, that is now done with `gq`. But if you use the `vimrc_example.vim` script or `defaults.vim`, "Q" works like "gq". Except for Select mode.

`gQ` `gQ` Switch to "Ex" mode like with "Q", but really behave like typing ":" commands after another. All command line editing, completion etc. is available. Use the ``:vi`` command (``:visual``) to exit "Ex" mode.

7. The window contents `window-contents`

In Normal mode and Insert/Replace mode the screen window will show the current contents of the buffer: What You See Is What You Get. There are two exceptions:

- When the `'coptions'` option contains '\$', and the change is within one line, the text is not directly deleted, but a '\$' is put at the last deleted character.
- When inserting text in one window, other windows on the same text are not updated until the insert is finished.

Lines longer than the window width will wrap, unless the `'wrap'` option is off (see below). The `'linebreak'` option can be set to wrap at a blank character.

If the window has room after the last line of the buffer, Vim will show '~' in the first column of the last lines in the window, like this:

```
+-----+
|some line|
|last line|
|~        |
|~        |
+-----+
```

Thus the '~' lines indicate that the end of the buffer was reached.

If the last line in a window doesn't fit, Vim will indicate this with a '@' in the first column of the last lines in the window, like this:

```
+-----+
|first line|
|second line|
|@         |
+-----+
```

```
|@                               |
+-----+

```

Thus the '@' lines indicate that there is a line that doesn't fit in the window.

When the "lastline" flag is present in the **'display'** option, you will not see '@' characters at the left side of window. If the last line doesn't fit completely, only the part that fits is shown, and the last three characters of the last line are replaced with "@@@", like this:

```
+-----+
|first line      |
|second line    |
|a very long line that d|
|oesn't fit in the wi@@@|
+-----+

```

If there is a single line that is too long to fit in the window, this is a special situation. Vim will show only part of the line, around where the cursor is. There are no special characters shown, so that you can edit all parts of this line.

The '@' occasion in the **'highlight'** option can be used to set special highlighting for the '@' and '~' characters. This makes it possible to distinguish them from real characters in the buffer.

The **'showbreak'** option contains the string to put in front of wrapped lines.

wrap-off

If the **'wrap'** option is off, long lines will not wrap. Only the part that fits on the screen is shown. If the cursor is moved to a part of the line that is not shown, the screen is scrolled horizontally. The advantage of this method is that columns are shown as they are and lines that cannot fit on the screen can be edited. The disadvantage is that you cannot see all the characters of a line at once. The **'sidescroll'** option can be set to the minimal number of columns to scroll.

All normal ASCII characters are displayed directly on the screen. The <Tab> is replaced with the number of spaces that it represents. Other non-printing characters are replaced with "^{char}", where {char} is the non-printing character with 64 added. Thus character 7 (bell) will be shown as "^G". Characters between 127 and 160 are replaced with "~{char}", where {char} is the character with 64 subtracted. These characters occupy more than one position on the screen. The cursor can only be positioned on the first one.

If you set the **'number'** option, all lines will be preceded with their number. Tip: If you don't like wrapping lines to mix with the line numbers, set the **'showbreak'** option to eight spaces:

```
":set showbreak=\\ \\ \\ \\ \\ \\ \\ "
```

If you set the **'list'** option, <Tab> characters will not be shown as several spaces, but as "^I". A '\$' will be placed at the end of the line, so you can find trailing blanks.

In Command-line mode only the command-line itself is shown correctly. The display of the buffer contents is updated as soon as you go back to Command mode.

The last line of the window is used for status and other messages. The status messages will only be used if an option is on:

status message	option	default	Unix default
current mode	'showmode'	on	on
command characters	'showcmd'	on	off
cursor position	'ruler'	off	off

The current mode is "-- INSERT --" or "-- REPLACE --", see 'showmode'. The command characters are those that you typed but were not used yet.

If you have a slow terminal you can switch off the status messages to speed up editing:

```
:set nosc noru nosm
```

If there is an error, an error message will be shown for at least one second (in reverse video).

Some commands show how many lines were affected. Above which threshold this happens can be controlled with the 'report' option (default 2).

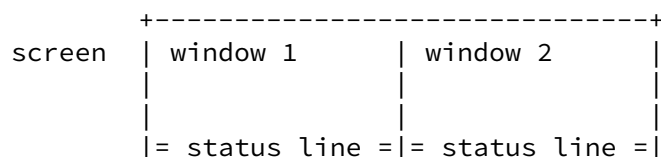
On the Amiga Vim will run in a CLI window. The name Vim and the full name of the current file name will be shown in the title bar. When the window is resized, Vim will automatically redraw the window. You may make the window as small as you like, but if it gets too small not a single line will fit in it. Make it at least 40 characters wide to be able to read most messages on the last line.

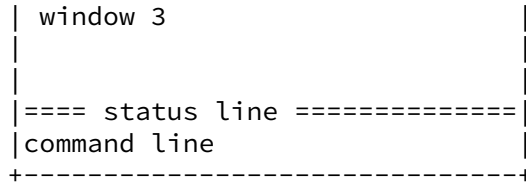
On most Unix systems, resizing the window is recognized and handled correctly by Vim.

8. Definitions definitions

buffer	Contains lines of text, usually read from a file.
screen	The whole area that Vim uses to work in. This can be a terminal emulator window. Also called "the Vim window".
window	A view on a buffer. There can be multiple windows for one buffer.

A screen contains one or more windows, separated by status lines and with the command line at the bottom.





The command line is also used for messages. It scrolls up the screen when there is not enough room in the command line.

A difference is made between four types of lines:

- buffer lines The lines in the buffer. This is the same as the lines as they are read from/written to a file. They can be thousands of characters long.
- logical lines The buffer lines with folding applied. Buffer lines in a closed fold are changed to a single logical line: "+-- 99 lines folded". They can be thousands of characters long.
- window lines The lines displayed in a window: A range of logical lines with wrapping, line breaks, etc. applied. They can only be as long as the width of the window allows, longer lines are wrapped or truncated.
- screen lines The lines of the screen that Vim uses. Consists of the window lines of all windows, with status lines and the command line added. They can only be as long as the width of the screen allows. When the command line gets longer it wraps and lines are scrolled to make room.

buffer lines	logical lines	window lines	screen lines
1. one	1. one	1. +-- folded	1. +-- folded
2. two	2. +-- folded	2. five	2. five
3. three	3. five	3. six	3. six
4. four	4. six	4. seven	4. seven
5. five	5. seven		5. === status line ===
6. six			6. aaa
7. seven			7. bbb
			8. ccc ccc c
1. aaa	1. aaa	1. aaa	9. cc
2. bbb	2. bbb	2. bbb	10. ddd
3. ccc ccc ccc	3. ccc ccc ccc	3. ccc ccc c	11. ~
4. ddd	4. ddd	4. cc	12. === status line ===
		5. ddd	13. (command line)
		6. ~	

=====

Help on help files

helphelp

- 1. Help commands [online-help](#)
- 2. Translated help files [help-translated](#)
- 3. Writing help files [help-writing](#)

=====
1. Help commands

[online-help](#)

`<Help>` or `help <Help> :h :help <F1> i_<F1> i_<Help>`
`:h[elp]`

Open a window and display the help file in read-only mode. If there is a help window open already, use that one. Otherwise, if the current window uses the full width of the screen or is at least 80 characters wide, the help window will appear just above the current window. Otherwise the new window is put at the very top.

The '[helplang](#)' option is used to select a language, if the main help file is available in several languages.

`:h[elp] {subject}` Like `:help`, additionally jump to the tag `{subject}`. For example:

`:help options`

`{subject}` can include wildcards such as `"*`, `"?"` and `"[a-z]"`:

`:help z?` jump to help for any "z" command
`:help z.` jump to the help for "z."

But when a tag exists it is taken literally:

`:help :?` jump to help for "?:"

If there is no full match for the pattern, or there are several matches, the "best" match will be used. A sophisticated algorithm is used to decide which match is better than another one. These items are considered in the computation:

- A match with same case is much better than a match with different case.
- A match that starts after a non-alphanumeric character is better than a match in the middle of a word.
- A match at or near the beginning of the tag is better than a match further on.
- The more alphanumeric characters match, the better.
- The shorter the length of the match, the better.

The **'helplang'** option is used to select a language, if the `{subject}` is available in several languages. To find a tag in a specific language, append "@ab", where "ab" is the two-letter language code. See [help-translated](#) .

Note that the longer the `{subject}` you give, the less matches will be found. You can get an idea how this all works by using commandline completion (type **CTRL-D** after `:"help subject"` `c_CTRL-D`).

If there are several matches, you can have them listed by hitting **CTRL-D**. Example:

```
:help cont<Ctrl-D>
```

Instead of typing `:"help CTRL-V"` to search for help for **CTRL-V** you can type:

```
:help ^V
```

This also works together with other characters, for example to find help for **CTRL-V** in Insert mode:

```
:help i^V
```

It is also possible to first do `:"help"` and then use `:"tag {pattern}"` in the help window. The `:"tnext"` command can then be used to jump to other matches, `"tselect"` to list matches and choose one.

```
:help index  
:tselect /.mode
```

When there is no argument you will see matches for "help", to avoid listing all possible matches (that would be very slow).

The number of matches displayed is limited to 300.

The `:"help"` command can be followed by `'|'` and another command, but you don't need to escape the `'|'` inside a help command. So these both work:

```
:help |  
:help k| only
```

Note that a space before the `'|'` is seen as part of the `:"help"` argument.

You can also use `<NL>` or `<CR>` to separate the help command from a following command. You need to type **CTRL-V** first to insert the `<NL>` or `<CR>`. Example:

```
:help so<C-V><CR>only
```

`:h[elp]! [subject]`

Like `:"help"`, but in non-English help files prefer to find a tag in a file with the same language as the current file. See [help-translated](#) .

`:helpc[lose]`

`:helpc` `:helpclose`
Close one help window, if there is one. Vim will try to restore the window layout (including cursor position) to the same layout it was before opening the help window initially. This might cause

triggering several autocommands.

:helpg **:helpgrep**

:helpg[rep] {pattern}[@xx]

Search all help text files and make a list of lines in which **{pattern}** matches. Jumps to the first match. The optional **[@xx]** specifies that only matches in the "xx" language are to be found.

You can navigate through the matches with the **quickfix** commands, e.g., **:cnext** to jump to the next one. Or use **:cwindow** to get the list of matches in the quickfix window.

{pattern} is used as a Vim regexp **pattern** .

'**ignorecase**' is not used, add "**\c**" to ignore case.

Example for case sensitive search:

:helpgrep Uganda

Example for case ignoring search:

:helpgrep uganda\c

Example for searching in French help:

:helpgrep backspace@fr

The pattern does not support line breaks, it must match within one line. You can use **:grep** instead, but then you need to get the list of help files in a complicated way.

Cannot be followed by another command, everything is used as part of the pattern. But you can use **:execute** when needed.

Compressed help files will not be searched (Fedora compresses the help files).

:lh **:lhhelpgrep**

:lh[elpgrep] {pattern}[@xx]

Same as "**:helpgrep**", except the location list is used instead of the quickfix list. If the help window is already opened, then the location list for that window is used. Otherwise, a new help window is opened and the location list for that window is set. The location list for the current window is not changed then.

:exu **:exusage**

:exu[sage]

Show help on Ex commands. Added to simulate the Nvi command.

:viu **:viusage**

:viu[sage]

Show help on Normal mode commands. Added to simulate the Nvi command.

When no argument is given to **:help** the file given with the '**helpfile**' option will be opened. Otherwise the specified tag is searched for in all "doc/tags" files in the directories specified in the '**runtimepath**' option.

If you would like to open the help in the current window, see this tip:

help-curwin .

The initial height of the help window can be set with the **'helpheight'** option (default 20).

help-buffer-options

When the help buffer is created, several local options are set to make sure the help text is displayed as it was intended:

'iskeyword'	nearly all ASCII chars except ' ', '*', '' and ' '
'foldmethod'	"manual"
'tabstop'	8
'arabic'	off
'binary'	off
'buflisted'	off
'cursorbind'	off
'diff'	off
'foldenable'	off
'list'	off
'modifiable'	off
'number'	off
'relativenumber'	off
'rightleft'	off
'scrollbind'	off
'spell'	off

Jump to specific subjects by using tags. This can be done in two ways:

- Use the **"CTRL-]"** command while standing on the name of a command or option. This only works when the tag is a keyword. "**<C-Leftmouse>**" and "**g<LeftMouse>**" work just like **"CTRL-]"**.
- use the **":ta {subject}"** command. This also works with non-keyword characters.

Use **CTRL-T** or **CTRL-O** to jump back.

Use **":q"** to close the help window.

If there are several matches for an item you are looking for, this is how you can jump to each one of them:

1. Open a help window
2. Use the **":tag"** command with a slash prepended to the tag. E.g.:
:tag /min
3. Use **":tnext"** to jump to the next matching tag.

It is possible to add help files for plugins and other items. You don't need to change the distributed help files for that. See [add-local-help](#) .

To write a local help file, see [write-local-help](#) .

Note that the title lines from the local help files are automagically added to the "LOCAL ADDITIONS" section in the "help.txt" help file [local-additions](#) . This is done when viewing the file in Vim, the file itself is not changed. It is done by going through all help files and obtaining the first line of each file. The files in \$VIMRUNTIME/doc are skipped.

help-xterm-window

If you want to have the help in another xterm window, you could use this command:

```
:!xterm -e vim +help &
```

```
                :helpfind  :helpf
:helpf[ind]      Like :help , but use a dialog to enter the argument.
                  Only for backwards compatibility. It now executes the
                  ToolBar.FindHelp menu entry instead of using a builtin
                  dialog. {only when compiled with |+GUI_GTK|}
```

```
                :helpt     :helptags
                E150  E151  E152  E153  E154  E670
:helpt[ags] [++t] {dir}
                  Generate the help tags file(s) for directory {dir}.
                  When {dir} is ALL then all "doc" directories in
                  'runtimepath' will be used.
```

All "*.txt" and "*.??x" files in the directory and sub-directories are scanned for a help tag definition in between stars. The "*.??x" files are for translated docs, they generate the "tags-??" file, see [help-translated](#) . The generated tags files are sorted.

When there are duplicates an error message is given. An existing tags file is silently overwritten.

The optional "++t" argument forces adding the "help-tags" tag. This is also done when the {dir} is equal to \$VIMRUNTIME/doc.

To rebuild the help tags in the runtime directory (requires write permission there):

```
:helptags $VIMRUNTIME/doc
```

2. Translated help files

[help-translated](#)

It is possible to add translated help files, next to the original English help files. Vim will search for all help in "doc" directories in 'runtimepath'. This is only available when compiled with the `+multi_lang` feature.

At this moment translations are available for:

- Chinese - multiple authors
- French - translated by David Blanchet
- Italian - translated by Antonio Colombo
- Japanese - multiple authors
- Polish - translated by Mikolaj Machowski
- Russian - translated by Vassily Ragosin

See the Vim website to find them: <http://www.vim.org/translations.php>

A set of translated help files consists of these files:

- help.abx
- howto.abx
- ...

tags-ab

"ab" is the two-letter language code. Thus for Italian the names are:

```
help.itx
howto.itx
...
tags-it
```

The **'helplang'** option can be set to the preferred language(s). The default is set according to the environment. Vim will first try to find a matching tag in the preferred language(s). English is used when it cannot be found.

To find a tag in a specific language, append "@ab" to a tag, where "ab" is the two-letter language code. Example:

```
:he user-manual@it
:he user-manual@en
```

The first one finds the Italian user manual, even when **'helplang'** is empty. The second one finds the English user manual, even when **'helplang'** is set to "it".

When using command-line completion for the ":help" command, the "@en" extension is only shown when a tag exists for multiple languages. When the tag only exists for English "@en" is omitted. When the first candidate has an "@ab" extension and it matches the first language in **'helplang'** "@ab" is also omitted.

When using **CTRL-]** or ":help!" in a non-English help file Vim will try to find the tag in the same language. If not found then **'helplang'** will be used to select a language.

Help files must use latin1 or utf-8 encoding. Vim assumes the encoding is utf-8 when finding non-ASCII characters in the first line. Thus you must translate the header with "For Vim version".

The same encoding must be used for the help files of one language in one directory. You can use a different encoding for different languages and use a different encoding for help files of the same language but in a different directory.

Hints for translators:

- Do not translate the tags. This makes it possible to use **'helplang'** to specify the preferred language. You may add new tags in your language.
- When you do not translate a part of a file, add tags to the English version, using the "tag@en" notation.
- Make a package with all the files and the tags file available for download. Users can drop it in one of the "doc" directories and start use it. Report this to Bram, so that he can add a link on www.vim.org.
- Use the **:helptags** command to generate the tags files. It will find all languages in the specified directory.

=====
3. Writing help files help-writing

For ease of use, a Vim help file for a plugin should follow the format of the standard Vim help files, except for the first line. If you are writing a new help file it's best to copy one of the existing files and use it as a template.

The first line in a help file should have the following format:

```
plugin_name.txt      {short description of the plugin}
```

The first field is a help tag where `":help plugin_name"` will jump to. The remainder of the line, after a Tab, describes the plugin purpose in a short way. This will show up in the "LOCAL ADDITIONS" section of the main help file. Check there that it shows up properly: [local-additions](#) .

If you want to add a version number or last modification date, put it in the second line, right aligned.

At the bottom of the help file, place a Vim modeline to set the `'textwidth'` and `'tabstop'` options and the `'filetype'` to "help". Never set a global option in such a modeline, that can have undesired consequences.

TAGS

To define a help tag, place the name between asterisks (`*tag-name*`). The tag-name should be different from all the Vim help tag names and ideally should begin with the name of the Vim plugin. The tag name is usually right aligned on a line.

When referring to an existing help tag and to create a hot-link, place the name between two bars (`|`) eg. [help-writing](#) .

When referring to a Vim command and to create a hot-link, place the name between two backticks, eg. inside ``:filetype``. You will see this is highlighted as a command, like a code block (see below).

When referring to a Vim option in the help file, place the option name between two single quotes, eg. `'statusline'`

HIGHLIGHTING

To define a column heading, use a tilde character at the end of the line. This will highlight the column heading in a different color. E.g.

Column heading

To separate sections in a help file, place a series of '=' characters in a line starting from the first column. The section separator line is highlighted differently.

To quote a block of ex-commands verbatim, place a greater than (>) character at the end of the line before the block and a less than (<) character as the first non-blank on a line following the block. Any line starting in column 1

also implicitly stops the block of ex-commands before it. E.g.

```
function Example_Func()  
    echo "Example"  
endfunction
```

The following are highlighted differently in a Vim help file:

- a special key name expressed either in <> notation as in <PageDown>, or as a Ctrl character as in **CTRL-X**
- anything between {braces}, e.g. {lhs} and {rhs}

The word "Note", "Notes" and similar automatically receive distinctive highlighting. So do these:

```
*Todo    something to do  
*Error   something wrong
```

You can find the details in \$VIMRUNTIME/syntax/help.vim

GENDER NEUTRAL LANGUAGE

gender-neutral inclusion

Vim is for everybody, no matter race, gender or anything. For new or updated help text, gender neutral language is recommended. Some of the help text is many years old and there is no need to change it. We do not make any assumptions about the gender of the user, no matter how the text is phrased. The goal is that the reader understands how Vim works, the exact wording is secondary.

Many online technical style guides include sections about gender neutral language. Here are a few:

```
https://developers.google.com/style/pronouns  
https://techwhirl.com/gender-neutral-technical-writing/  
https://www.skillsyouneed.com/write/gender-neutral-language.html  
https://ualr.edu/writingcenter/avoid-sexist-language/
```

Note: gender neutral language does not require using singular "they".

VIM REFERENCE MANUAL by Bram Moolenaar

index

This file contains a list of all commands for each mode, with a tag and a short description. The lists are sorted on ASCII value.

Tip: When looking for certain functionality, use a search command. E.g., to look for deleting something, use: "/delete".

- 1. Insert mode insert-index
- 2. Normal mode normal-index
 - 2.1. Text objects objects
 - 2.2. Window commands CTRL-W
 - 2.3. Square bracket commands [
 - 2.4. Commands starting with 'g' g
 - 2.5. Commands starting with 'z' z
 - 2.6. Operator-pending mode operator-pending-index
- 3. Visual mode visual-index
- 4. Command-line editing ex-edit-index
- 5. Terminal-Job mode terminal-job-index
- 6. EX commands ex-cmd-index

For an overview of options see [option-list](#) .
 For an overview of built-in functions see [functions](#) .
 For a list of Vim variables see [vim-variable](#) .
 For a complete listing of all help items see [help-tags](#) .



1. Insert mode insert-index

tag	char	action in Insert mode
i_CTRL-@	CTRL-@	insert previously inserted text and stop insert
i_CTRL-A	CTRL-A	insert previously inserted text
	CTRL-B	not used i_CTRL-B-gone
i_CTRL-C	CTRL-C	quit insert mode, without checking for abbreviation, unless ' insertmode ' set.
i_CTRL-D	CTRL-D	delete one shiftwidth of indent in the current line
i_CTRL-E	CTRL-E	insert the character which is below the cursor
	CTRL-F	not used (but by default it's in ' cinkeys ' to re-indent the current line)
i_CTRL-G_j	CTRL-G CTRL-J	line down, to column where inserting started
i_CTRL-G_j	CTRL-G j	line down, to column where inserting started
i_CTRL-G_j	CTRL-G <Down>	line down, to column where inserting started
i_CTRL-G_k	CTRL-G CTRL-K	line up, to column where inserting started
i_CTRL-G_k	CTRL-G k	line up, to column where inserting started
i_CTRL-G_k	CTRL-G <Up>	line up, to column where inserting started
i_CTRL-G_u	CTRL-G u	start new undoable edit
i_CTRL-G_U	CTRL-G U	don't break undo with next cursor movement

i_<BS>	<BS>	delete character before the cursor
i_digraph	{char1}<BS>{char2}	enter digraph (only when 'digraph' option set)
i_CTRL-H	CTRL-H	same as <BS>
i_<Tab>	<Tab>	insert a <Tab> character
i_CTRL-I	CTRL-I	same as <Tab>
i_<NL>	<NL>	same as <CR>
i_CTRL-J	CTRL-J	same as <CR>
i_CTRL-K	CTRL-K {char1} {char2}	enter digraph
i_CTRL-L	CTRL-L	when 'insertmode' set: Leave Insert mode
i_<CR>	<CR>	begin new line
i_CTRL-M	CTRL-M	same as <CR>
i_CTRL-N	CTRL-N	find next match for keyword in front of the cursor
i_CTRL-O	CTRL-O	execute a single command and return to insert mode
i_CTRL-P	CTRL-P	find previous match for keyword in front of the cursor
i_CTRL-Q	CTRL-Q	same as CTRL-V , unless used for terminal control flow
i_CTRL-SHIFT-Q	CTRL-SHIFT-Q {char}	like CTRL-Q unless <code>modifyOtherKeys</code> is active
i_CTRL-R	CTRL-R {register}	insert the contents of a register
i_CTRL-R_CTRL-R	CTRL-R CTRL-R {register}	insert the contents of a register literally
i_CTRL-R_CTRL-O	CTRL-R CTRL-O {register}	insert the contents of a register literally and don't auto-indent
i_CTRL-R_CTRL-P	CTRL-R CTRL-P {register}	insert the contents of a register literally and fix indent.
i_CTRL-T	CTRL-S CTRL-T	not used or used for terminal control flow insert one shiftwidth of indent in current line
i_CTRL-U	CTRL-U	delete all entered characters in the current line
i_CTRL-V	CTRL-V {char}	insert next non-digit literally
i_CTRL-SHIFT-V	CTRL-SHIFT-V {char}	like CTRL-V unless <code>modifyOtherKeys</code> is active
i_CTRL-V_digit	CTRL-V {number}	insert three digit decimal number as a single byte.
i_CTRL-W	CTRL-W	delete word before the cursor
i_CTRL-X	CTRL-X {mode}	enter CTRL-X sub mode, see <code>i_CTRL-X_index</code>
i_CTRL-Y	CTRL-Y	insert the character which is above the cursor
i_CTRL-Z	CTRL-Z	when 'insertmode' set: suspend Vim
i_<Esc>	<Esc>	end insert mode (unless 'insertmode' set)
i_CTRL-[CTRL-[same as <Esc>
i_CTRL-_CTRL-N	CTRL-\ CTRL-N	go to Normal mode
i_CTRL-_CTRL-G	CTRL-\ CTRL-G	go to mode specified with 'insertmode'
	CTRL-\ a - z	reserved for extensions
	CTRL-\ others	not used
i_CTRL-]	CTRL-]	trigger abbreviation

i_CTRL-^	CTRL-^	toggle use of <code>:lmap</code> mappings
i_CTRL-_	CTRL-_	When ' allowrevins ' set: change language (Hebrew, Farsi) {only when compiled with the <code>+rightleft</code> feature}
	<Space> to '~'	not used, except '0' and '^' followed by CTRL-D
i_0_CTRL-D	0 CTRL-D	delete all indent in the current line
i_^_CTRL-D	^ CTRL-D	delete all indent in the current line, restore it in the next line
i_		delete character under the cursor
	Meta characters (0x80 to 0xff, 128 to 255)	not used
i_<Left>	<Left>	cursor one character left
i_<S-Left>	<S-Left>	cursor one word left
i_<C-Left>	<C-Left>	cursor one word left
i_<Right>	<Right>	cursor one character right
i_<S-Right>	<S-Right>	cursor one word right
i_<C-Right>	<C-Right>	cursor one word right
i_<Up>	<Up>	cursor one line up
i_<S-Up>	<S-Up>	same as <PageUp>
i_<Down>	<Down>	cursor one line down
i_<S-Down>	<S-Down>	same as <PageDown>
i_<Home>	<Home>	cursor to start of line
i_<C-Home>	<C-Home>	cursor to start of file
i_<End>	<End>	cursor past end of line
i_<C-End>	<C-End>	cursor past end of file
i_<PageUp>	<PageUp>	one screenful backward
i_<PageDown>	<PageDown>	one screenful forward
i_<F1>	<F1>	same as <Help>
i_<Help>	<Help>	stop insert mode and display help window
i_<Insert>	<Insert>	toggle Insert/Replace mode
i_<LeftMouse>	<LeftMouse>	cursor at mouse click
i_<ScrollWheelDown>	<ScrollWheelDown>	move window three lines down
i_<S-ScrollWheelDown>	<S-ScrollWheelDown>	move window one page down
i_<ScrollWheelUp>	<ScrollWheelUp>	move window three lines up
i_<S-ScrollWheelUp>	<S-ScrollWheelUp>	move window one page up
i_<ScrollWheelLeft>	<ScrollWheelLeft>	move window six columns left
i_<S-ScrollWheelLeft>	<S-ScrollWheelLeft>	move window one page left
i_<ScrollWheelRight>	<ScrollWheelRight>	move window six columns right
i_<S-ScrollWheelRight>	<S-ScrollWheelRight>	move window one page right

commands in **CTRL-X** submode

[i_CTRL-X_index](#)

i_CTRL-X_CTRL-D	CTRL-X CTRL-D	complete defined identifiers
i_CTRL-X_CTRL-E	CTRL-X CTRL-E	scroll up
i_CTRL-X_CTRL-F	CTRL-X CTRL-F	complete file names
i_CTRL-X_CTRL-I	CTRL-X CTRL-I	complete identifiers
i_CTRL-X_CTRL-K	CTRL-X CTRL-K	complete identifiers from dictionary
i_CTRL-X_CTRL-L	CTRL-X CTRL-L	complete whole lines

i_CTRL-X_CTRL-N	CTRL-X CTRL-N	next completion
i_CTRL-X_CTRL-O	CTRL-X CTRL-O	omni completion
i_CTRL-X_CTRL-P	CTRL-X CTRL-P	previous completion
i_CTRL-X_CTRL-S	CTRL-X CTRL-S	spelling suggestions
i_CTRL-X_CTRL-T	CTRL-X CTRL-T	complete identifiers from thesaurus
i_CTRL-X_CTRL-Y	CTRL-X CTRL-Y	scroll down
i_CTRL-X_CTRL-U	CTRL-X CTRL-U	complete with ' completefunc '
i_CTRL-X_CTRL-V	CTRL-X CTRL-V	complete like in : command line
i_CTRL-X_CTRL-Z	CTRL-X CTRL-Z	stop completion, keeping the text as-is
i_CTRL-X_CTRL-]	CTRL-X CTRL-]	complete tags
i_CTRL-X_s	CTRL-X s	spelling suggestions

commands in completion mode (see [popupmenu-keys](#))

complete_CTRL-E	CTRL-E	stop completion and go back to original text
complete_CTRL-Y	CTRL-Y	accept selected match and stop completion
	CTRL-L	insert one character from the current match
	<CR>	insert currently selected match
	<BS>	delete one character and redo search
	CTRL-H	same as <BS>
	<Up>	select the previous match
	<Down>	select the next match
	<PageUp>	select a match several entries back
	<PageDown>	select a match several entries forward
	other	stop completion and insert the typed character

2. Normal mode

[normal-index](#)

CHAR	any non-blank character
WORD	a sequence of non-blank characters
N	a number entered before the command
{motion}	a cursor movement command
Nmove	the text that is moved over with a {motion}
SECTION	a section that possibly starts with '}' instead of '{'

note: 1 = cursor movement command; 2 = can be undone/redone

tag	char	note	action in Normal mode
	CTRL-@		not used
CTRL-A	CTRL-A	2	add N to number at/after cursor
CTRL-B	CTRL-B	1	scroll N screens Backwards
CTRL-C	CTRL-C		interrupt current (search) command
CTRL-D	CTRL-D		scroll Down N lines (default: half a screen)
CTRL-E	CTRL-E		scroll N lines upwards (N lines Extra)
CTRL-F	CTRL-F	1	scroll N screens Forward
CTRL-G	CTRL-G		display current file name and position
<BS>	<BS>	1	same as "h"
CTRL-H	CTRL-H	1	same as "h"
<Tab>	<Tab>	1	go to N newer entry in jump list
CTRL-I	CTRL-I	1	same as <Tab>
<NL>	<NL>	1	same as "j"
CTRL-J	CTRL-J	1	same as "j"

	CTRL-K	not used
CTRL-L	CTRL-L	redraw screen
<CR>	<CR>	1 cursor to the first CHAR N lines lower
CTRL-M	CTRL-M	1 same as <CR>
CTRL-N	CTRL-N	1 same as "j"
CTRL-O	CTRL-O	1 go to N older entry in jump list
CTRL-P	CTRL-P	1 same as "k"
	CTRL-Q	not used, or used for terminal control flow
CTRL-R	CTRL-R	2 redo changes which were undone with 'u'
	CTRL-S	not used, or used for terminal control flow
CTRL-T	CTRL-T	jump to N older Tag in tag list
CTRL-U	CTRL-U	scroll N lines Upwards (default: half a screen)
CTRL-V	CTRL-V	start blockwise Visual mode
CTRL-W	CTRL-W {char}	window commands, see CTRL-W
CTRL-X	CTRL-X	2 subtract N from number at/after cursor
CTRL-Y	CTRL-Y	scroll N lines downwards
CTRL-Z	CTRL-Z	suspend program (or start new shell)
	CTRL-[<Esc>	not used
CTRL-_CTRL-N	CTRL-\ CTRL-N	go to Normal mode (no-op)
CTRL-_CTRL-G	CTRL-\ CTRL-G	go to mode specified with ' insertmode '
	CTRL-\ a - z	reserved for extensions
	CTRL-\ others	not used
CTRL-]	CTRL-]	:ta to indent under cursor
CTRL-^	CTRL-^	edit Nth alternate file (equivalent to ":e #N")
	CTRL-_	not used
<Space>	<Space>	1 same as "l"
!	!{motion}{filter}	2 filter Nmove text through the {filter} command
!!	!!{filter}	2 filter N lines through the {filter} command
quote	"{register}	use {register} for next delete, yank or put ({.%#;} only work with put)
#	#	1 search backward for the Nth occurrence of the ident under the cursor
\$	\$	1 cursor to the end of Nth next line
%	%	1 find the next (curly/square) bracket on this line and go to its match, or go to matching comment bracket, or go to matching preprocessor directive.
N%	{count}%	1 go to N percentage in the file
&	&	2 repeat last :s
'	'{a-zA-Z0-9}	1 cursor to the first CHAR on the line with mark {a-zA-Z0-9}
''	''	1 cursor to the first CHAR of the line where the cursor was before the latest jump.
'('(1 cursor to the first CHAR on the line of the start of the current sentence
')	')	1 cursor to the first CHAR on the line of the end of the current sentence
'<	'<	1 cursor to the first CHAR of the line where highlighted area starts/started in the

		current buffer.
'>	'>	1 cursor to the first CHAR of the line where highlighted area ends/ended in the current buffer.
'['	'['	1 cursor to the first CHAR on the line of the start of last operated text or start of put text
']'	']'	1 cursor to the first CHAR on the line of the end of last operated text or end of put text
'{'	'{'	1 cursor to the first CHAR on the line of the start of the current paragraph
'}'	'}'	1 cursor to the first CHAR on the line of the end of the current paragraph
((1 cursor N sentences backward
))	1 cursor N sentences forward
star	*	1 search forward for the Nth occurrence of the ident under the cursor
+	+	1 same as <CR>
,	,	1 repeat latest f, t, F or T in opposite direction N times
-	-	1 cursor to the first CHAR N lines higher
.	.	2 repeat last change with count replaced with N
/	/ {pattern} <CR>	1 search forward for the Nth occurrence of {pattern}
/ <CR>	/ <CR>	1 search forward for {pattern} of last search
0	0	1 cursor to the first char of the line
count	1	prepend to command to give a count
count	2	"
count	3	"
count	4	"
count	5	"
count	6	"
count	7	"
count	8	"
count	9	"
:	:	1 start entering an Ex command
N:	{count}:	start entering an Ex command with range from current line to N-1 lines down
;	;	1 repeat latest f, t, F or T N times
<	< {motion}	2 shift Nmove lines one 'shiftwidth' leftwards
<<	<<	2 shift N lines one 'shiftwidth' leftwards
=	= {motion}	2 filter Nmove lines through "indent"
==	==	2 filter N lines through "indent"
>	> {motion}	2 shift Nmove lines one 'shiftwidth' rightwards
>>	>>	2 shift N lines one 'shiftwidth' rightwards
?	? {pattern} <CR>	1 search backward for the Nth previous occurrence of {pattern}
? <CR>	? <CR>	1 search backward for {pattern} of last search
@	@ {a-z}	2 execute the contents of register {a-z} N times

@:	@:	repeat the previous ":" command N times
@@	@@	2 repeat the previous @{a-z} N times
A	A	2 append text after the end of the line N times
B	B	1 cursor N WORDS backward
C	["x]C	2 change from the cursor position to the end of the line, and N-1 more lines [into register x]; synonym for "c\$"
D	["x]D	2 delete the characters under the cursor until the end of the line and N-1 more lines [into register x]; synonym for "d\$"
E	E	1 cursor forward to the end of WORD N
F	F{char}	1 cursor to the Nth occurrence of {char} to the left
G	G	1 cursor to line N, default last line
H	H	1 cursor to line N from top of screen
I	I	2 insert text before the first CHAR on the line N times
J	J	2 Join N lines; default is 2
K	K	lookup Keyword under the cursor with 'keywordprg'
L	L	1 cursor to line N from bottom of screen
M	M	1 cursor to middle line of screen
N	N	1 repeat the latest '/' or '?' N times in opposite direction
O	O	2 begin a new line above the cursor and insert text, repeat N times
P	["x]P	2 put the text [from register x] before the cursor N times
Q	Q	switch to "Ex" mode
R	R	2 enter replace mode: overtype existing characters, repeat the entered text N-1 times
S	["x]S	2 delete N lines [into register x] and start insert; synonym for "cc".
T	T{char}	1 cursor till after Nth occurrence of {char} to the left
U	U	2 undo all latest changes on one line
V	V	start linewise Visual mode
W	W	1 cursor N WORDS forward
X	["x]X	2 delete N characters before the cursor [into register x]
Y	["x]Y	yank N lines [into register x]; synonym for "yy"
ZZ	ZZ	write if buffer changed and close window
ZQ	ZQ	close window without writing
[[{char}	square bracket command (see [below)
]] {char}	square bracket command (see] below)
^	^	1 cursor to the first CHAR of the line
⏪	⏪	1 cursor to the first CHAR N - 1 lines lower
`	`{a-zA-Z0-9}	1 cursor to the mark {a-zA-Z0-9}
⏩	⏩	1 cursor to the start of the current sentence
⏪	⏪	1 cursor to the end of the current sentence
⏩	⏩	1 cursor to the start of the highlighted area

<code>`></code>	<code>`></code>	1	cursor to the end of the highlighted area
<code>`[</code>	<code>`[</code>	1	cursor to the start of last operated text or start of putted text
<code>`]</code>	<code>`]</code>	1	cursor to the end of last operated text or end of putted text
<code>``</code>	<code>``</code>	1	cursor to the position before latest jump
<code>`{</code>	<code>`{</code>	1	cursor to the start of the current paragraph
<code>`}</code>	<code>`}</code>	1	cursor to the end of the current paragraph
<code>a</code>	<code>a</code>	2	append text after the cursor N times
<code>b</code>	<code>b</code>	1	cursor N words backward
<code>c</code>	<code>["x]c{motion}</code>	2	delete Nmove text [into register x] and start insert
<code>cc</code>	<code>["x]cc</code>	2	delete N lines [into register x] and start insert
<code>d</code>	<code>["x]d{motion}</code>	2	delete Nmove text [into register x]
<code>dd</code>	<code>["x]dd</code>	2	delete N lines [into register x]
<code>do</code>	<code>do</code>	2	same as <code>":diffget"</code>
<code>dp</code>	<code>dp</code>	2	same as <code>":diffput"</code>
<code>e</code>	<code>e</code>	1	cursor forward to the end of word N
<code>f</code>	<code>f{char}</code>	1	cursor to Nth occurrence of <code>{char}</code> to the right
<code>g</code>	<code>g{char}</code>		extended commands, see <code>g</code> below
<code>h</code>	<code>h</code>	1	cursor N chars to the left
<code>i</code>	<code>i</code>	2	insert text before the cursor N times
<code>j</code>	<code>j</code>	1	cursor N lines downward
<code>k</code>	<code>k</code>	1	cursor N lines upward
<code>l</code>	<code>l</code>	1	cursor N chars to the right
<code>m</code>	<code>m{A-Za-z}</code>		set mark <code>{A-Za-z}</code> at cursor position
<code>n</code>	<code>n</code>	1	repeat the latest <code>'/'</code> or <code>'?'</code> N times
<code>o</code>	<code>o</code>	2	begin a new line below the cursor and insert text, repeat N times
<code>p</code>	<code>["x]p</code>	2	put the text [from register x] after the cursor N times
<code>q</code>	<code>q{0-9a-zA-Z"}</code>		record typed characters into named register <code>{0-9a-zA-Z"}</code> (uppercase to append) (while recording) stops recording
<code>q</code>	<code>q</code>		edit : command-line in command-line window
<code>q:</code>	<code>q:</code>		edit / command-line in command-line window
<code>q/</code>	<code>q/</code>		edit ? command-line in command-line window
<code>q?</code>	<code>q?</code>		edit ? command-line in command-line window
<code>r</code>	<code>r{char}</code>	2	replace N chars with <code>{char}</code>
<code>s</code>	<code>["x]s</code>	2	(substitute) delete N characters [into register x] and start insert
<code>t</code>	<code>t{char}</code>	1	cursor till before Nth occurrence of <code>{char}</code> to the right
<code>u</code>	<code>u</code>	2	undo changes
<code>v</code>	<code>v</code>		start characterwise Visual mode
<code>w</code>	<code>w</code>	1	cursor N words forward
<code>x</code>	<code>["x]x</code>	2	delete N characters under and after the cursor [into register x]
<code>y</code>	<code>["x]y{motion}</code>		yank Nmove text [into register x]
<code>yy</code>	<code>["x]yy</code>		yank N lines [into register x]
<code>z</code>	<code>z{char}</code>		commands starting with <code>'z'</code> , see <code>z</code> below
<code>{</code>	<code>{</code>	1	cursor N paragraphs backward
<code>bar</code>	<code> </code>	1	cursor to column N

}	}	1	cursor N paragraphs forward
~	~	2	' tildeop ' off: switch case of N characters under cursor and move the cursor N characters to the right
~	~{motion}		' tildeop ' on: switch case of Nmove text
<C-End>	<C-End>	1	same as "G"
<C-Home>	<C-Home>	1	same as "gg"
<C-Left>	<C-Left>	1	same as "b"
<C-LeftMouse>	<C-LeftMouse>		":ta" to the keyword at the mouse click
<C-Right>	<C-Right>	1	same as "w"
<C-RightMouse>	<C-RightMouse>		same as " CTRL-T "
<C-Tab>	<C-Tab>		same as "g<Tab>"
	["x]	2	same as "x"
N	{count}		remove the last digit from {count}
<Down>	<Down>	1	same as "j"
<End>	<End>	1	same as "\$"
<F1>	<F1>		same as <Help>
<Help>	<Help>		open a help window
<Home>	<Home>	1	same as "0"
<Insert>	<Insert>	2	same as "i"
<Left>	<Left>	1	same as "h"
<LeftMouse>	<LeftMouse>	1	move cursor to the mouse click position
<MiddleMouse>	<MiddleMouse>	2	same as "gP" at the mouse click position
<PageDown>	<PageDown>		same as CTRL-F
<PageUp>	<PageUp>		same as CTRL-B
<Right>	<Right>	1	same as "l"
<RightMouse>	<RightMouse>		start Visual mode, move cursor to the mouse click position
<S-Down>	<S-Down>	1	same as CTRL-F
<S-Left>	<S-Left>	1	same as "b"
<S-LeftMouse>	<S-LeftMouse>		same as "*" at the mouse click position
<S-Right>	<S-Right>	1	same as "w"
<S-RightMouse>	<S-RightMouse>		same as "#" at the mouse click position
<S-Up>	<S-Up>	1	same as CTRL-B
<Undo>	<Undo>	2	same as "u"
<Up>	<Up>	1	same as "k"
<ScrollWheelDown>	<ScrollWheelDown>		move window three lines down
<S-ScrollWheelDown>	<S-ScrollWheelDown>		move window one page down
<ScrollWheelUp>	<ScrollWheelUp>		move window three lines up
<S-ScrollWheelUp>	<S-ScrollWheelUp>		move window one page up
<ScrollWheelLeft>	<ScrollWheelLeft>		move window six columns left
<S-ScrollWheelLeft>	<S-ScrollWheelLeft>		move window one page left
<ScrollWheelRight>	<ScrollWheelRight>		move window six columns right
<S-ScrollWheelRight>	<S-ScrollWheelRight>		move window one page right

2.1 Text objects

objects

These can be used after an operator or in Visual mode to select an object.

tag	command	action in op-pending and Visual mode
v_aquote	a"	double quoted string
v_a'	a'	single quoted string

v_a(a(same as ab
v_a)	a)	same as ab
v_a<	a<	"a <>" from '<' to the matching '>'
v_a>	a>	same as a<
v_aB	aB	"a Block" from "[{" to "]"}" (with brackets)
v_aW	aW	"a WORD" (with white space)
v_a[a["a []" from '[' to the matching ']'
v_a]	a]	same as a[
v_a`	a`	string in backticks
v_ab	ab	"a block" from "[" to "]"" (with braces)
v_ap	ap	"a paragraph" (with white space)
v_as	as	"a sentence" (with white space)
v_at	at	"a tag block" (with white space)
v_aw	aw	"a word" (with white space)
v_a{	a{	same as aB
v_a}	a}	same as aB
v_iquote	i"	double quoted string without the quotes
v_i'	i'	single quoted string without the quotes
v_i(i(same as ib
v_i)	i)	same as ib
v_i<	i<	"inner <>" from '<' to the matching '>'
v_i>	i>	same as i<
v_iB	iB	"inner Block" from "[{" and "]"}"
v_iW	iW	"inner WORD"
v_i[i["inner []" from '[' to the matching ']'
v_i]	i]	same as i[
v_i`	i`	string in backticks without the backticks
v_ib	ib	"inner block" from "[" to "]""
v_ip	ip	"inner paragraph"
v_is	is	"inner sentence"
v_it	it	"inner tag block"
v_iw	iw	"inner word"
v_i{	i{	same as ib
v_i}	i}	same as ib

2.2 Window commands

CTRL-W

tag	command	action in Normal mode
CTRL-W_CTRL-B	CTRL-W CTRL-B	same as " CTRL-W b"
CTRL-W_CTRL-C	CTRL-W CTRL-C	same as " CTRL-W c"
CTRL-W_CTRL-D	CTRL-W CTRL-D	same as " CTRL-W d"
CTRL-W_CTRL-F	CTRL-W CTRL-F	same as " CTRL-W f"
	CTRL-W CTRL-G	same as " CTRL-W g .."
CTRL-W_CTRL-H	CTRL-W CTRL-H	same as " CTRL-W h"
CTRL-W_CTRL-I	CTRL-W CTRL-I	same as " CTRL-W i"
CTRL-W_CTRL-J	CTRL-W CTRL-J	same as " CTRL-W j"
CTRL-W_CTRL-K	CTRL-W CTRL-K	same as " CTRL-W k"
CTRL-W_CTRL-L	CTRL-W CTRL-L	same as " CTRL-W l"
CTRL-W_CTRL-N	CTRL-W CTRL-N	same as " CTRL-W n"
CTRL-W_CTRL-O	CTRL-W CTRL-O	same as " CTRL-W o"
CTRL-W_CTRL-P	CTRL-W CTRL-P	same as " CTRL-W p"
CTRL-W_CTRL-Q	CTRL-W CTRL-Q	same as " CTRL-W q"

CTRL-W_CTRL-R	CTRL-W CTRL-R	same as " CTRL-W r "
CTRL-W_CTRL-S	CTRL-W CTRL-S	same as " CTRL-W s "
CTRL-W_CTRL-T	CTRL-W CTRL-T	same as " CTRL-W t "
CTRL-W_CTRL-V	CTRL-W CTRL-V	same as " CTRL-W v "
CTRL-W_CTRL-W	CTRL-W CTRL-W	same as " CTRL-W w "
CTRL-W_CTRL-X	CTRL-W CTRL-X	same as " CTRL-W x "
CTRL-W_CTRL-Z	CTRL-W CTRL-Z	same as " CTRL-W z "
CTRL-W_CTRL-]	CTRL-W CTRL-]	same as " CTRL-W] "
CTRL-W_CTRL-^	CTRL-W CTRL-^	same as " CTRL-W ^ "
CTRL-W_CTRL-_	CTRL-W CTRL-_	same as " CTRL-W _ "
CTRL-W_+	CTRL-W +	increase current window height N lines
CTRL-W_-	CTRL-W -	decrease current window height N lines
CTRL-W_:	CTRL-W :	same as <code>:</code> , edit a command line
CTRL-W_<	CTRL-W <	decrease current window width N columns
CTRL-W_=	CTRL-W =	make all windows the same height & width
CTRL-W_>	CTRL-W >	increase current window width N columns
CTRL-W_H	CTRL-W H	move current window to the far left
CTRL-W_J	CTRL-W J	move current window to the very bottom
CTRL-W_K	CTRL-W K	move current window to the very top
CTRL-W_L	CTRL-W L	move current window to the far right
CTRL-W_P	CTRL-W P	go to preview window
CTRL-W_R	CTRL-W R	rotate windows upwards N times
CTRL-W_S	CTRL-W S	same as " CTRL-W s "
CTRL-W_T	CTRL-W T	move current window to a new tab page
CTRL-W_W	CTRL-W W	go to N previous window (wrap around)
CTRL-W_]	CTRL-W]	split window and jump to tag under cursor
CTRL-W_^	CTRL-W ^	split current window and edit alternate file N
CTRL-W__	CTRL-W _	set current window height to N (default: very high)
CTRL-W_b	CTRL-W b	go to bottom window
CTRL-W_c	CTRL-W c	close current window (like <code>:close</code>)
CTRL-W_d	CTRL-W d	split window and jump to definition under the cursor
CTRL-W_f	CTRL-W f	split window and edit file name under the cursor
CTRL-W_F	CTRL-W F	split window and edit file name under the cursor and jump to the line number following the file name.
CTRL-W_g_CTRL-]	CTRL-W g CTRL-]	split window and do <code>:tjump</code> to tag under cursor
CTRL-W_g]	CTRL-W g]	split window and do <code>:tselect</code> for tag under cursor
CTRL-W_g}	CTRL-W g }	do a <code>:ptjump</code> to the tag under the cursor
CTRL-W_gf	CTRL-W g f	edit file name under the cursor in a new tab page
CTRL-W_gF	CTRL-W g F	edit file name under the cursor in a new tab page and jump to the line number following the file name.
CTRL-W_gt	CTRL-W g t	same as <code>`gt`</code> : go to next tab page
CTRL-W_gT	CTRL-W g T	same as <code>`gT`</code> : go to previous tab page
CTRL-W_g<Tab>	CTRL-W g <Tab>	same as <code>g<Tab></code> : go to last accessed tab page.
CTRL-W_h	CTRL-W h	go to Nth left window (stop at first window)

CTRL-W_i	CTRL-W i	split window and jump to declaration of identifier under the cursor
CTRL-W_j	CTRL-W j	go N windows down (stop at last window)
CTRL-W_k	CTRL-W k	go N windows up (stop at first window)
CTRL-W_l	CTRL-W l	go to Nth right window (stop at last window)
CTRL-W_n	CTRL-W n	open new window, N lines high
CTRL-W_o	CTRL-W o	close all but current window (like <code>:only</code>)
CTRL-W_p	CTRL-W p	go to previous (last accessed) window
CTRL-W_q	CTRL-W q	quit current window (like <code>:quit</code>)
CTRL-W_r	CTRL-W r	rotate windows downwards N times
CTRL-W_s	CTRL-W s	split current window in two parts, new window N lines high
CTRL-W_t	CTRL-W t	go to top window
CTRL-W_v	CTRL-W v	split current window vertically, new window N columns wide
CTRL-W_w	CTRL-W w	go to N next window (wrap around)
CTRL-W_x	CTRL-W x	exchange current window with window N (default: next window)
CTRL-W_z	CTRL-W z	close preview window
CTRL-W_bar	CTRL-W 	set window width to N columns
CTRL-W_}	CTRL-W }	show tag under cursor in preview window
CTRL-W_<Down>	CTRL-W <Down>	same as " CTRL-W j "
CTRL-W_<Up>	CTRL-W <Up>	same as " CTRL-W k "
CTRL-W_<Left>	CTRL-W <Left>	same as " CTRL-W h "
CTRL-W_<Right>	CTRL-W <Right>	same as " CTRL-W l "

2.3 Square bracket commands

[]

tag	char	note	action in Normal mode
[_CTRL-D	[CTRL-D		jump to first #define found in current and included files matching the word under the cursor, start searching at beginning of current file
[_CTRL-I	[CTRL-I		jump to first line in current and included files that contains the word under the cursor, start searching at beginning of current file
[#	[#	1	cursor to N previous unmatched #if, #else or #ifdef
['	['	1	cursor to previous lowercase mark, on first non-blank
[([(1	cursor N times back to unmatched '('
[*star	[*star	1	same as "[/]"
[`	[`	1	cursor to previous lowercase mark
[/	[/	1	cursor to N previous start of a C comment
[D	[D		list all defines found in current and included files matching the word under the cursor, start searching at beginning of current file
[I	[I		list all lines found in current and included files that contain the word under the cursor, start searching at beginning of

		current file
[P	[P	2 same as "[p"
[[[[1 cursor N sections backward
]]]]	1 cursor N SECTIONS backward
[c	[c	1 cursor N times backwards to start of change
[d	[d	show first #define found in current and included files matching the word under the cursor, start searching at beginning of current file
[f	[f	same as "gf"
[i	[i	show first line found in current and included files that contains the word under the cursor, start searching at beginning of current file
[m	[m	1 cursor N times back to start of member function
[p	[p	2 like "P", but adjust indent to current line
[s	[s	1 move to the previous misspelled word
[z	[z	1 move to start of open fold
[{	[{	1 cursor N times back to unmatched '{'
[<MiddleMouse>	[<MiddleMouse>	2 same as "[p"
] _CTRL-D] CTRL-D	jump to first #define found in current and included files matching the word under the cursor, start searching at cursor position
] _CTRL-I] CTRL-I	jump to first line in current and included files that contains the word under the cursor, start searching at cursor position
]#]#	1 cursor to N next unmatched #endif or #else
]']'	1 cursor to next lowercase mark, on first non-blank
])])	1 cursor N times forward to unmatched ')'
]star]*	1 same as "]/"
]`]`	1 cursor to next lowercase mark
] /] /	1 cursor to N next end of a C comment
]D]D	list all #defines found in current and included files matching the word under the cursor, start searching at cursor position
]I]I	list all lines found in current and included files that contain the word under the cursor, start searching at cursor position
]P]P	2 same as "[p"
] [] [1 cursor N SECTIONS forward
]]]]	1 cursor N sections forward
]c]c	1 cursor N times forward to start of change
]d]d	show first #define found in current and included files matching the word under the cursor, start searching at cursor position
]f]f	same as "gf"
]i]i	show first line found in current and included files that contains the word under the cursor, start searching at cursor position

```

]m          ]m          1  cursor N times forward to end of member
                    function
]p          ]p          2  like "p", but adjust indent to current line
]s          ]s          1  move to next misspelled word
]z          ]z          1  move to end of open fold
]}          ]}          1  cursor N times forward to unmatched '}'
]<MiddleMouse> ]<MiddleMouse> 2  same as "]p"

```

2.4 Commands starting with 'g'

g

tag	char	note	action in Normal mode
g_CTRL-A	g CTRL-A		only when compiled with MEM_PROFILE defined: dump a memory profile
g_CTRL-G	g CTRL-G		show information about current cursor position
g_CTRL-H	g CTRL-H		start Select block mode
g_CTRL-]	g CTRL-]		:tjump to the tag under the cursor
g#	g#	1	like "#", but without using "\<" and "\>"
g\$	g\$	1	when 'wrap' off go to rightmost character of the current line that is on the screen; when 'wrap' on go to the rightmost character of the current screen line
g&	g&	2	repeat last ":s" on all lines
g'	g'{mark}	1	like ' but without changing the jumplist
g`	g`{mark}	1	like ` but without changing the jumplist
gstar	g*	1	like "*", but without using "\<" and "\>"
g+	g+		go to newer text state N times
g,	g,	1	go to N newer position in change list
g-	g-		go to older text state N times
g0	g0	1	when 'wrap' off go to leftmost character of the current line that is on the screen; when 'wrap' on go to the leftmost character of the current screen line
g8	g8		print hex value of bytes used in UTF-8 character under the cursor
g;	g;	1	go to N older position in change list
g<	g<		display previous command output
g?	g?	2	Rot13 encoding operator
g?g?	g??	2	Rot13 encode current line
g?g?	g?g?	2	Rot13 encode current line
gD	gD	1	go to definition of word under the cursor in current file
gE	gE	1	go backwards to the end of the previous WORD
gH	gH		start Select line mode
gI	gI	2	like "I", but always start in column 1
gJ	gJ	2	join lines without inserting space
gN	gN	1,2	find the previous match with the last used search pattern and Visually select it
gP	["x]gP	2	put the text [from register x] before the cursor N times, leave the cursor after it
gQ	gQ		switch to "Ex" mode with Vim editing

gR	gR	2	enter Virtual Replace mode
gT	gT		go to the previous tab page
gU	gU{motion}	2	make Nmove text uppercase
gV	gV		don't reselect the previous Visual area when executing a mapping or menu in Select mode
g]	g]		:tselect on the tag under the cursor
g^	g^	1	when 'wrap' off go to leftmost non-white character of the current line that is on the screen; when 'wrap' on go to the leftmost non-white character of the current screen line
g_	g_	1	cursor to the last CHAR N - 1 lines lower
ga	ga		print ascii value of character under the cursor
gd	gd	1	go to definition of word under the cursor in current function
ge	ge	1	go backwards to the end of the previous word
gf	gf		start editing the file whose name is under the cursor
gF	gF		start editing the file whose name is under the cursor and jump to the line number following the filename.
gg	gg	1	cursor to line N, default first line
gh	gh		start Select mode
gi	gi	2	like "i", but first move to the '^' mark
gj	gj	1	like "j", but when 'wrap' on go N screen lines down
gk	gk	1	like "k", but when 'wrap' on go N screen lines up
gm	gm	1	go to character at middle of the screenline
gM	gM	1	go to character at middle of the text line
gn	gn	1,2	find the next match with the last used search pattern and Visually select it
go	go	1	cursor to byte N in the buffer
gp	["x]gp	2	put the text [from register x] after the cursor N times, leave the cursor after it
gq	gq{motion}	2	format Nmove text
gr	gr{char}	2	virtual replace N chars with {char}
gs	gs		go to sleep for N seconds (default 1)
gt	gt		go to the next tab page
gu	gu{motion}	2	make Nmove text lowercase
gV	gV		reselect the previous Visual area
gw	gw{motion}	2	format Nmove text and keep cursor
netrw-gx	gx		execute application for file name under the cursor (only with netrw plugin)
g@	g@{motion}		call 'operatorfunc'
g~	g~{motion}	2	swap case for Nmove text
g<Down>	g<Down>	1	same as "gj"
g<End>	g<End>	1	same as "g\$"
g<Home>	g<Home>	1	same as "g0"
g<LeftMouse>	g<LeftMouse>		same as <C-LeftMouse>
	g<MiddleMouse>		same as <C-MiddleMouse>

g<RightMouse>	g<RightMouse>	same as <C-RightMouse>
g<Tab>	g<Tab>	go to the last accessed tab page.
z<Up>	g<Up>	1 same as "gk"

2.5 Commands starting with 'z'

z

tag	char	note	action in Normal mode
z<CR>	z<CR>		redraw, cursor line to top of window, cursor on first non-blank
zN<CR>	z{height}<CR>		redraw, make window {height} lines high
z+	z+		cursor on line N (default line below window), otherwise like "z<CR>"
z-	z-		redraw, cursor line at bottom of window, cursor on first non-blank
z.	z.		redraw, cursor line to center of window, cursor on first non-blank
z=	z=		give spelling suggestions
zA	zA		open a closed fold or close an open fold recursively
zC	zC		close folds recursively
zD	zD		delete folds recursively
zE	zE		eliminate all folds
zF	zF		create a fold for N lines
zG	zG		temporarily mark word as correctly spelled
zH	zH		when 'wrap' off scroll half a screenwidth to the right
zL	zL		when 'wrap' off scroll half a screenwidth to the left
zM	zM		set 'foldlevel' to zero
zN	zN		set 'foldenable'
zO	zO		open folds recursively
zR	zR		set 'foldlevel' to the deepest fold
zW	zW		temporarily mark word as incorrectly spelled
zX	zX		re-apply 'foldlevel'
z^	z^		cursor on line N (default line above window), otherwise like "z-"
za	za		open a closed fold, close an open fold
zb	zb		redraw, cursor line at bottom of window
zc	zc		close a fold
zd	zd		delete a fold
ze	ze		when 'wrap' off scroll horizontally to position the cursor at the end (right side) of the screen
zf	zf{motion}		create a fold for Nmove text
zg	zg		permanently mark word as correctly spelled
zh	zh		when 'wrap' off scroll screen N characters to the right
zi	zi		toggle 'foldenable'
zj	zj	1	move to the start of the next fold
zk	zk	1	move to the end of the previous fold
zl	zl		when 'wrap' off scroll screen N characters to the left

zm	zm	subtract one from 'foldlevel'
zn	zn	reset 'foldenable'
zo	zo	open fold
zp	zp	paste in block-mode without trailing spaces
zP	zP	paste in block-mode without trailing spaces
zr	zr	add one to 'foldlevel'
zs	zs	when 'wrap' off scroll horizontally to position the cursor at the start (left side) of the screen
zt	zt	redraw, cursor line at top of window
zuw	zuw	undo zw
zug	zug	undo zg
zuW	zuW	undo zW
zuG	zuG	undo zG
zv	zv	open enough folds to view the cursor line
zw	zw	permanently mark word as incorrectly spelled
zx	zx	re-apply 'foldlevel' and do "zv"
zy	zy	yank without trailing spaces
zz	zz	redraw, cursor line at center of window
z<Left>	z<Left>	same as "zh"
z<Right>	z<Right>	same as "zl"

2.6 Operator-pending mode

[operator-pending-index](#)

These can be used after an operator, but before a {motion} has been entered.

tag	char	action in Operator-pending mode
o_v	v	force operator to work characterwise
o_V	V	force operator to work linewise
o_CTRL-V	CTRL-V	force operator to work blockwise

3. Visual mode

[visual-index](#)

Most commands in Visual mode are the same as in Normal mode. The ones listed here are those that are different.

tag	command	note	action in Visual mode
v_CTRL-_CTRL-N	CTRL-\ CTRL-N		stop Visual mode
v_CTRL-_CTRL-G	CTRL-\ CTRL-G		go to mode specified with 'insertmode'
v_CTRL-A	CTRL-A	2	add N to number in highlighted text
v_CTRL-C	CTRL-C		stop Visual mode
v_CTRL-G	CTRL-G		toggle between Visual mode and Select mode
v_<BS>	<BS>	2	Select mode: delete highlighted area
v_CTRL-H	CTRL-H	2	same as <BS>
v_CTRL-O	CTRL-O		switch from Select to Visual mode for one command
v_CTRL-V	CTRL-V		make Visual mode blockwise or stop Visual mode
v_CTRL-X	CTRL-X	2	subtract N from number in highlighted text
v_<Esc>	<Esc>		stop Visual mode

<code>v_CTRL-]</code>	CTRL-]	jump to highlighted tag
<code>v_!</code>	<code>!{filter}</code>	2 filter the highlighted lines through the external command <code>{filter}</code>
<code>v_:</code>	:	start a command-line with the highlighted lines as a range
<code>v_<</code>	<	2 shift the highlighted lines one 'shiftwidth' left
<code>v_=</code>	=	2 filter the highlighted lines through the external program given with the 'equalprg' option
<code>v_></code>	>	2 shift the highlighted lines one 'shiftwidth' right
<code>v_b_A</code>	A	2 block mode: append same text in all lines, after the highlighted area
<code>v_C</code>	C	2 delete the highlighted lines and start insert
<code>v_D</code>	D	2 delete the highlighted lines
<code>v_b_I</code>	I	2 block mode: insert same text in all lines, before the highlighted area
<code>v_J</code>	J	2 join the highlighted lines
<code>v_K</code>	K	run 'keywordprg' on the highlighted area
<code>v_O</code>	O	move horizontally to other corner of area
<code>v_P</code>	P	replace highlighted area with register contents; registers are unchanged
	Q	does not start Ex mode
<code>v_R</code>	R	2 delete the highlighted lines and start insert
<code>v_S</code>	S	2 delete the highlighted lines and start insert
<code>v_U</code>	U	2 make highlighted area uppercase
<code>v_V</code>	V	make Visual mode linewise or stop Visual mode
<code>v_X</code>	X	2 delete the highlighted lines
<code>v_Y</code>	Y	yank the highlighted lines
<code>v_aquote</code>	<code>a"</code>	extend highlighted area with a double quoted string
<code>v_a'</code>	<code>a'</code>	extend highlighted area with a single quoted string
<code>v_a(</code>	<code>a(</code>	same as <code>ab</code>
<code>v_a)</code>	<code>a)</code>	same as <code>ab</code>
<code>v_a<</code>	<code>a<</code>	extend highlighted area with a <code><></code> block
<code>v_a></code>	<code>a></code>	same as <code>a<</code>
<code>v_aB</code>	<code>aB</code>	extend highlighted area with a <code>{ }</code> block
<code>v_aW</code>	<code>aW</code>	extend highlighted area with "a WORD"
<code>v_a[</code>	<code>a[</code>	extend highlighted area with a <code>[]</code> block
<code>v_a]</code>	<code>a]</code>	same as <code>a[</code>
<code>v_a`</code>	<code>a`</code>	extend highlighted area with a backtick quoted string
<code>v_ab</code>	<code>ab</code>	extend highlighted area with a <code>()</code> block
<code>v_ap</code>	<code>ap</code>	extend highlighted area with a paragraph
<code>v_as</code>	<code>as</code>	extend highlighted area with a sentence
<code>v_at</code>	<code>at</code>	extend highlighted area with a tag block
<code>v_aw</code>	<code>aw</code>	extend highlighted area with "a word"
<code>v_a{</code>	<code>a{</code>	same as <code>aB</code>

v_a}	a}	same as aB
v_c	c	2 delete highlighted area and start insert
v_d	d	2 delete highlighted area
v_g_CTRL-A	g CTRL-A	2 add N to number in highlighted text
v_g_CTRL-X	g CTRL-X	2 subtract N from number in highlighted text
v_gJ	gJ	2 join the highlighted lines without inserting spaces
v_gq	gq	2 format the highlighted lines
v_gv	gv	exchange current and previous highlighted area
v_iquote	i"	extend highlighted area with a double quoted string (without quotes)
v_i'	i'	extend highlighted area with a single quoted string (without quotes)
v_i(i(same as iB
v_i)	i)	same as iB
v_i<	i<	extend highlighted area with inner <> block
v_i>	i>	same as i<
v_iB	iB	extend highlighted area with inner {} block
v_iW	iW	extend highlighted area with "inner WORD"
v_i[i[extend highlighted area with inner [] block
v_i]	i]	same as i[
v_i`	i`	extend highlighted area with a backtick quoted string (without the backticks)
v_ib	ib	extend highlighted area with inner () block
v_ip	ip	extend highlighted area with inner paragraph
v_is	is	extend highlighted area with inner sentence
v_it	it	extend highlighted area with inner tag block
v_iw	iw	extend highlighted area with "inner word"
v_i{	i{	same as iB
v_i}	i}	same as iB
v_o	o	move cursor to other corner of area
v_p	p	replace highlighted area with register contents; deleted text in unnamed register
v_r	r	2 replace highlighted area with a character
v_s	s	2 delete highlighted area and start insert
v_u	u	2 make highlighted area lowercase
v_v	v	make Visual mode characterwise or stop Visual mode
v_x	x	2 delete the highlighted area
v_y	y	yank the highlighted area
v_~	~	2 swap case for the highlighted area

4. Command-line editing

[ex-edit-index](#)

Get to the command-line with the ':', '!', '/' or '?' commands.

Normal characters are inserted at the current cursor position.

"Completion" below refers to context-sensitive completion. It will complete file names, tags, commands etc. as appropriate.

tag	command	action in Command-line editing mode
-----	---------	-------------------------------------

	CTRL-@	not used
--	--------	----------

c_CTRL-A	CTRL-A	do completion on the pattern in front of the cursor and insert all matches
c_CTRL-B	CTRL-B	cursor to begin of command-line
c_CTRL-C	CTRL-C	same as <Esc>
c_CTRL-D	CTRL-D	list completions that match the pattern in front of the cursor
c_CTRL-E	CTRL-E	cursor to end of command-line
'cedit'	CTRL-F	default value for 'cedit': opens the command-line window; otherwise not used
c_CTRL-G	CTRL-G	next match when 'incsearch' is active
c_<BS>	<BS>	delete the character in front of the cursor
c_digraph	{char1} <BS> {char2}	enter digraph when 'digraph' is on
c_CTRL-H	CTRL-H	same as <BS>
c_<Tab>	<Tab>	if 'wildchar' is <Tab>: Do completion on the pattern in front of the cursor
c_<S-Tab>	<S-Tab>	same as CTRL-P
c_wildchar	'wildchar'	Do completion on the pattern in front of the cursor (default: <Tab>)
c_CTRL-I	CTRL-I	same as <Tab>
c_<NL>	<NL>	same as <CR>
c_CTRL-J	CTRL-J	same as <CR>
c_CTRL-K	CTRL-K {char1} {char2}	enter digraph
c_CTRL-L	CTRL-L	do completion on the pattern in front of the cursor and insert the longest common part
c_<CR>	<CR>	execute entered command
c_CTRL-M	CTRL-M	same as <CR>
c_CTRL-N	CTRL-N	after using 'wildchar' with multiple matches: go to next match, otherwise: recall older command-line from history.
	CTRL-O	not used
c_CTRL-P	CTRL-P	after using 'wildchar' with multiple matches: go to previous match, otherwise: recall older command-line from history.
c_CTRL-Q	CTRL-Q	same as CTRL-V , unless it's used for terminal control flow
c_CTRL-R	CTRL-R {regname}	insert the contents of a register or object under the cursor as if typed
c_CTRL-R_CTRL-R	CTRL-R CTRL-R {regname}	
c_CTRL-R_CTRL-O	CTRL-R CTRL-O {regname}	insert the contents of a register or object under the cursor literally
	CTRL-S	not used, or used for terminal control flow
c_CTRL-T	CTRL-T	previous match when 'incsearch' is active
c_CTRL-U	CTRL-U	remove all characters
c_CTRL-V	CTRL-V	insert next non-digit literally, insert three digit decimal number as a single byte.
c_CTRL-W	CTRL-W	delete the word in front of the cursor
	CTRL-X	not used (reserved for completion)
	CTRL-Y	copy (yank) modeless selection
	CTRL-Z	not used (reserved for suspend)
c_<Esc>	<Esc>	abandon command-line without executing it

c_CTRL-[CTRL-[same as <Esc>
c_CTRL-_CTRL-N	CTRL-\ CTRL-N	go to Normal mode, abandon command-line
c_CTRL-_CTRL-G	CTRL-\ CTRL-G	go to mode specified with 'insertmode', abandon command-line
	CTRL-\ a - d	reserved for extensions
c_CTRL-_e	CTRL-\ e {expr}	replace the command line with the result of {expr}
	CTRL-\ f - z	reserved for extensions
	CTRL-\ others	not used
c_CTRL-]	CTRL-]	trigger abbreviation
c_CTRL-^	CTRL-^	toggle use of :lmap mappings
c_CTRL-_	CTRL-_	when 'allowrevins' set: change language (Hebrew, Farsi)
c_		delete the character under the cursor
c_<Left>	<Left>	cursor left
c_<S-Left>	<S-Left>	cursor one word left
c_<C-Left>	<C-Left>	cursor one word left
c_<Right>	<Right>	cursor right
c_<S-Right>	<S-Right>	cursor one word right
c_<C-Right>	<C-Right>	cursor one word right
c_<Up>	<Up>	recall previous command-line from history that matches pattern in front of the cursor
c_<S-Up>	<S-Up>	recall previous command-line from history
c_<Down>	<Down>	recall next command-line from history that matches pattern in front of the cursor
c_<S-Down>	<S-Down>	recall next command-line from history
c_<Home>	<Home>	cursor to start of command-line
c_<End>	<End>	cursor to end of command-line
c_<PageDown>	<PageDown>	same as <S-Down>
c_<PageUp>	<PageUp>	same as <S-Up>
c_<Insert>	<Insert>	toggle insert/overstrike mode
c_<LeftMouse>	<LeftMouse>	cursor at mouse click

commands in wildmenu mode (see 'wildmenu')

<Up>	move up to parent / select the previous match
<Down>	move down to submenu / select the next match
<Left>	select the previous match / move up to parent
<Right>	select the next match / move down to submenu
<CR>	move into submenu when doing menu completion
CTRL-E	stop completion and go back to original text
CTRL-Y	accept selected match and stop completion
other	stop completion and insert the typed character

commands in wildmenu mode with 'wildoptions' set to "pum"

<PageUp>	select a match several entries back
<PageDown>	select a match several entries forward

5. Terminal-Job mode

[terminal-job-index](#)

Most Normal mode commands except for window commands (CTRL-W) do not work in

a terminal window. Switch to Terminal-Normal mode to use them. This assumes `'termwinkey'` is not set.

tag	char	action in Terminal-Job mode
t_CTRL-_CTRL-N	CTRL-\ CTRL-N	switch to Terminal-Normal mode
t_CTRL-W_N	CTRL-W N	switch to Terminal-Normal mode
t_CTRL-W_:	CTRL-W :	enter an Ex command
t_CTRL-W_.	CTRL-W .	type CTRL-W in the terminal
t_CTRL-W_	CTRL-W CTRL-\	send a CTRL-\ to the job in the terminal
t_CTRL-W_quote	CTRL-W " {register}	paste register in the terminal
t_CTRL-W_CTRL-C	CTRL-W CTRL-C	forcefully ends the job
t_CTRL-W_CTRL-W	CTRL-W CTRL-W	move focus to the next window
t_CTRL-W_gt	CTRL-W gt	go to next tabpage, same as `gt`
t_CTRL-W_gT	CTRL-W gT	go to previous tabpage, same as `gT`

You found it, Arthur! holy-grail :smile

6. EX commands ex-cmd-index :index

This is a brief but complete listing of all the ":" commands, without mentioning any arguments. The optional part of the command name is inside []. The commands are sorted on the non-optional part of their name.

tag	command	action
:	:	nothing
:range	:{range}	go to last line in {range}
:!	:!	filter lines or execute an external command
:!!	:!!	repeat last "!" command
:#	:#	same as ":number"
:&	:&	repeat last ":substitute"
:star	:*	use the last Visual area, like :<,>
:<	:<	shift lines one 'shiftwidth' left
:='	:='	print the last line number
:>	:>	shift lines one 'shiftwidth' right
:@	:@	execute contents of a register
:@@	:@@	repeat the previous "@"
:2match	:2mat[ch]	define a second match to highlight
:3match	:3mat[ch]	define a third match to highlight
:Next	:N[ext]	go to previous file in the argument list
:Print	:P[rint]	print lines
:X	:X	ask for encryption key
:append	:a[ppend]	append text
:abbreviate	:ab[breviate]	enter abbreviation
:abclear	:abc[lear]	remove all abbreviations
:aboveleft	:abo[veleft]	make split window appear left or above
:all	:al[l]	open a window for each file in the argument list
:amenu	:am[enu]	enter new menu item for all modes
:anoremenu	:an[oremenu]	enter a new menu for all modes that will not be remapped

<code>:args</code>	<code>:ar[gs]</code>	print the argument list
<code>:argadd</code>	<code>:arga[dd]</code>	add items to the argument list
<code>:argdedupe</code>	<code>:argded[upe]</code>	remove duplicates from the argument list
<code>:argdelete</code>	<code>:argd[elete]</code>	delete items from the argument list
<code>:argedit</code>	<code>:arge[dit]</code>	add item to the argument list and edit it
<code>:argdo</code>	<code>:argdo</code>	do a command on all items in the argument list
<code>:argglobal</code>	<code>:argg[lobal]</code>	define the global argument list
<code>:arglocal</code>	<code>:argl[ocal]</code>	define a local argument list
<code>:argument</code>	<code>:argu[ment]</code>	go to specific file in the argument list
<code>:ascii</code>	<code>:as[ci]</code>	print ascii value of character under the cursor
<code>:autocmd</code>	<code>:au[tocmd]</code>	enter or show autocommands
<code>:augroup</code>	<code>:aug[roup]</code>	select the autocommand group to use
<code>:aunmenu</code>	<code>:aun[menu]</code>	remove menu for all modes
<code>:buffer</code>	<code>:b[uffer]</code>	go to specific buffer in the buffer list
<code>:bNext</code>	<code>:bN[ext]</code>	go to previous buffer in the buffer list
<code>:ball</code>	<code>:ba[ll]</code>	open a window for each buffer in the buffer list
<code>:badd</code>	<code>:bad[d]</code>	add buffer to the buffer list
<code>:balt</code>	<code>:balt</code>	like <code>":badd"</code> but also set the alternate file
<code>:bdelete</code>	<code>:bd[elete]</code>	remove a buffer from the buffer list
<code>:behave</code>	<code>:be[have]</code>	set mouse and selection behavior
<code>:belowright</code>	<code>:bel[owright]</code>	make split window appear right or below
<code>:bfirst</code>	<code>:bf[irst]</code>	go to first buffer in the buffer list
<code>:blast</code>	<code>:bl[ast]</code>	go to last buffer in the buffer list
<code>:bmodified</code>	<code>:bm[odified]</code>	go to next buffer in the buffer list that has been modified
<code>:bnext</code>	<code>:bn[ext]</code>	go to next buffer in the buffer list
<code>:botright</code>	<code>:bo[tright]</code>	make split window appear at bottom or far right
<code>:bprevious</code>	<code>:bp[revious]</code>	go to previous buffer in the buffer list
<code>:brewind</code>	<code>:br[ewind]</code>	go to first buffer in the buffer list
<code>:break</code>	<code>:brea[k]</code>	break out of while loop
<code>:breakadd</code>	<code>:breaka[dd]</code>	add a debugger breakpoint
<code>:breakdel</code>	<code>:breakd[el]</code>	delete a debugger breakpoint
<code>:breaklist</code>	<code>:breakl[ist]</code>	list debugger breakpoints
<code>:browse</code>	<code>:bro[wse]</code>	use file selection dialog
<code>:bufdo</code>	<code>:bufdo</code>	execute command in each listed buffer
<code>:buffers</code>	<code>:buffers</code>	list all files in the buffer list
<code>:bunload</code>	<code>:bun[load]</code>	unload a specific buffer
<code>:bwipeout</code>	<code>:bw[ipeout]</code>	really delete a buffer
<code>:change</code>	<code>:c[hange]</code>	replace a line or series of lines
<code>:cNext</code>	<code>:cN[ext]</code>	go to previous error
<code>:cNfile</code>	<code>:cNf[ile]</code>	go to last error in previous file
<code>:cabbrev</code>	<code>:ca[bbrev]</code>	like <code>":abbreviate"</code> but for Command-line mode
<code>:cabclear</code>	<code>:cabclear</code>	clear all abbreviations for Command-line mode
<code>:cabove</code>	<code>:cabo[ve]</code>	go to error above current line
<code>:caddbuffer</code>	<code>:cad[dbuffer]</code>	add errors from buffer
<code>:caddexpr</code>	<code>:cadde[xpr]</code>	add errors from expr
<code>:caddfile</code>	<code>:caddf[ile]</code>	add error message to current quickfix list
<code>:cafter</code>	<code>:caf[ter]</code>	go to error after current cursor
<code>:call</code>	<code>:cal[l]</code>	call a function
<code>:catch</code>	<code>:cat[ch]</code>	part of a <code>:try</code> command
<code>:cbefore</code>	<code>:cbef[ore]</code>	go to error before current cursor
<code>:cbelow</code>	<code>:cbel[ow]</code>	go to error below current line
<code>:cbottom</code>	<code>:cbo[ttom]</code>	scroll to the bottom of the quickfix window
<code>:cbuffer</code>	<code>:cb[uffer]</code>	parse error messages and jump to first error

<code>:cc</code>	<code>:cc</code>	go to specific error
<code>:cclose</code>	<code>:ccl[ose]</code>	close quickfix window
<code>:cd</code>	<code>:cd</code>	change directory
<code>:cdo</code>	<code>:cdo</code>	execute command in each valid error list entry
<code>:cfdo</code>	<code>:cfd[o]</code>	execute command in each file in error list
<code>:center</code>	<code>:ce[nter]</code>	format lines at the center
<code>:cexpr</code>	<code>:cex[pr]</code>	read errors from expr and jump to first
<code>:cfile</code>	<code>:cf[ile]</code>	read file with error messages and jump to first
<code>:cfirst</code>	<code>:cfir[st]</code>	go to the specified error, default first one
<code>:cgetbuffer</code>	<code>:cgetb[uffer]</code>	get errors from buffer
<code>:cgetexpr</code>	<code>:cgete[xpr]</code>	get errors from expr
<code>:cgetfile</code>	<code>:cg[etfile]</code>	read file with error messages
<code>:changes</code>	<code>:changes</code>	print the change list
<code>:chdir</code>	<code>:chd[ir]</code>	change directory
<code>:checkpath</code>	<code>:che[ckpath]</code>	list included files
<code>:checktime</code>	<code>:checkt[ime]</code>	check timestamp of loaded buffers
<code>:chistory</code>	<code>:chi[story]</code>	list the error lists
<code>:class</code>	<code>:class</code>	start of a class specification
<code>:clast</code>	<code>:cla[st]</code>	go to the specified error, default last one
<code>:clearjumps</code>	<code>:cle[arjumps]</code>	clear the jump list
<code>:clist</code>	<code>:cl[ist]</code>	list all errors
<code>:close</code>	<code>:clo[se]</code>	close current window
<code>:cmap</code>	<code>:cm[ap]</code>	like <code>":map"</code> but for Command-line mode
<code>:cmapclear</code>	<code>:cmapc[lear]</code>	clear all mappings for Command-line mode
<code>:cmenu</code>	<code>:cme[nu]</code>	add menu for Command-line mode
<code>:cnext</code>	<code>:cn[ext]</code>	go to next error
<code>:cnewer</code>	<code>:cnew[er]</code>	go to newer error list
<code>:cnfile</code>	<code>:cnf[ile]</code>	go to first error in next file
<code>:cnoremap</code>	<code>:cno[remap]</code>	like <code>":noremap"</code> but for Command-line mode
<code>:cnoreabbrev</code>	<code>:cnorea[bbrev]</code>	like <code>":noreabbrev"</code> but for Command-line mode
<code>:cnoremenu</code>	<code>:cnoreme[nu]</code>	like <code>":noremenu"</code> but for Command-line mode
<code>:copy</code>	<code>:co[py]</code>	copy lines
<code>:colder</code>	<code>:col[der]</code>	go to older error list
<code>:colorscheme</code>	<code>:colo[rscheme]</code>	load a specific color scheme
<code>:command</code>	<code>:com[mand]</code>	create user-defined command
<code>:comclear</code>	<code>:comc[lear]</code>	clear all user-defined commands
<code>:compiler</code>	<code>:comp[iler]</code>	do settings for a specific compiler
<code>:continue</code>	<code>:con[tinue]</code>	go back to <code>:while</code>
<code>:confirm</code>	<code>:conf[irm]</code>	prompt user when confirmation required
<code>:const</code>	<code>:cons[t]</code>	create a variable as a constant
<code>:copen</code>	<code>:cope[n]</code>	open quickfix window
<code>:cprevious</code>	<code>:cp[revious]</code>	go to previous error
<code>:cpfile</code>	<code>:cpf[ile]</code>	go to last error in previous file
<code>:cquit</code>	<code>:cq[uit]</code>	quit Vim with an error code
<code>:crewind</code>	<code>:cr[ewind]</code>	go to the specified error, default first one
<code>:cscope</code>	<code>:cs[cope]</code>	execute cscope command
<code>:cstag</code>	<code>:cst[ag]</code>	use cscope to jump to a tag
<code>:cunmap</code>	<code>:cu[nmap]</code>	like <code>":unmap"</code> but for Command-line mode
<code>:cunabbrev</code>	<code>:cuna[bbrev]</code>	like <code>":unabbrev"</code> but for Command-line mode
<code>:cunmenu</code>	<code>:cunme[nu]</code>	remove menu for Command-line mode
<code>:cwindow</code>	<code>:cw[indow]</code>	open or close quickfix window
<code>:delete</code>	<code>:d[elete]</code>	delete lines
<code>:debug</code>	<code>:deb[ug]</code>	run a command in debugging mode
<code>:debuggreedy</code>	<code>:debugg[reedy]</code>	read debug mode commands from normal input

<code>:def</code>	<code>:def</code>	define a Vim9 user function
<code>:defcompile</code>	<code>:defc[ompile]</code>	compile Vim9 user functions in current script
<code>:defer</code>	<code>:defer</code>	call function when current function is done
<code>:delcommand</code>	<code>:delc[ommand]</code>	delete user-defined command
<code>:delfunction</code>	<code>:delf[unction]</code>	delete a user function
<code>:delmarks</code>	<code>:delm[arks]</code>	delete marks
<code>:diffupdate</code>	<code>:dif[fupdate]</code>	update 'diff' buffers
<code>:diffget</code>	<code>:diffg[et]</code>	remove differences in current buffer
<code>:diffoff</code>	<code>:diffo[ff]</code>	switch off diff mode
<code>:diffpatch</code>	<code>:diffp[atc]</code>	apply a patch and show differences
<code>:diffput</code>	<code>:diffpu[t]</code>	remove differences in other buffer
<code>:diffsplit</code>	<code>:diffs[plit]</code>	show differences with another file
<code>:diffthis</code>	<code>:diffthis</code>	make current window a diff window
<code>:digraphs</code>	<code>:dig[raphs]</code>	show or enter digraphs
<code>:display</code>	<code>:di[splay]</code>	display registers
<code>:disassemble</code>	<code>:disa[ssemble]</code>	disassemble Vim9 user function
<code>:djump</code>	<code>:dj[ump]</code>	jump to #define
<code>:dl</code>	<code>:dl</code>	short for :delete with the 'l' flag
<code>:dlist</code>	<code>:dli[st]</code>	list #defines
<code>:doautocmd</code>	<code>:do[autocmd]</code>	apply autocommands to current buffer
<code>:doautoall</code>	<code>:doautoa[ll]</code>	apply autocommands for all loaded buffers
<code>:dp</code>	<code>:d[elete]p</code>	short for :delete with the 'p' flag
<code>:drop</code>	<code>:dr[op]</code>	jump to window editing file or edit file in current window
<code>:dsearch</code>	<code>:ds[earch]</code>	list one #define
<code>:dsplit</code>	<code>:dsp[lit]</code>	split window and jump to #define
<code>:edit</code>	<code>:e[dit]</code>	edit a file
<code>:earlier</code>	<code>:ea[rlier]</code>	go to older change, undo
<code>:echo</code>	<code>:ec[ho]</code>	echoes the result of expressions
<code>:echoconsole</code>	<code>:echoc[onsole]</code>	like <code>:echomsg</code> but write to stdout
<code>:echoerr</code>	<code>:echoe[rr]</code>	like <code>:echo</code> , show like an error and use history
<code>:echohl</code>	<code>:echoh[l]</code>	set highlighting for echo commands
<code>:echomsg</code>	<code>:echom[sg]</code>	same as <code>:echo</code> , put message in history
<code>:echon</code>	<code>:echon</code>	same as <code>:echo</code> , but without <code><EOL></code>
<code>:echowindow</code>	<code>:echow[indow]</code>	same as <code>:echomsg</code> , but use a popup window
<code>:else</code>	<code>:el[se]</code>	part of an <code>:if</code> command
<code>:elseif</code>	<code>:elsei[f]</code>	part of an <code>:if</code> command
<code>:emenu</code>	<code>:em[enu]</code>	execute a menu by name
<code>:endclass</code>	<code>:endclass</code>	end of a class specification
<code>:enddef</code>	<code>:enddef</code>	end of a user function started with <code>:def</code>
<code>:endif</code>	<code>:en[dif]</code>	end previous <code>:if</code>
<code>:endfor</code>	<code>:endfo[r]</code>	end previous <code>:for</code>
<code>:endfunction</code>	<code>:endf[unction]</code>	end of a user function started with <code>:function</code>
<code>:endtry</code>	<code>:endt[ry]</code>	end previous <code>:try</code>
<code>:endwhile</code>	<code>:endw[hile]</code>	end previous <code>:while</code>
<code>:enew</code>	<code>:ene[w]</code>	edit a new, unnamed buffer
<code>:eval</code>	<code>:ev[al]</code>	evaluate an expression and discard the result
<code>:ex</code>	<code>:ex</code>	same as <code>":edit"</code>
<code>:execute</code>	<code>:exe[cute]</code>	execute result of expressions
<code>:exit</code>	<code>:exi[t]</code>	same as <code>":xit"</code>
<code>:export</code>	<code>:exp[ort]</code>	Vim9: export an item from a script
<code>:exusage</code>	<code>:exu[sage]</code>	overview of Ex commands
<code>:file</code>	<code>:f[ile]</code>	show or set the current file name
<code>:files</code>	<code>:files</code>	list all files in the buffer list

<code>:filetype</code>	<code>:filet[ype]</code>	switch file type detection on/off
<code>:filter</code>	<code>:filt[er]</code>	filter output of following command
<code>:find</code>	<code>:fin[d]</code>	find file in ' path ' and edit it
<code>:final</code>	<code>:final</code>	declare an immutable variable in Vim9
<code>:finally</code>	<code>:fina[lly]</code>	part of a <code>:try</code> command
<code>:finish</code>	<code>:fini[sh]</code>	quit sourcing a Vim script
<code>:first</code>	<code>:fir[st]</code>	go to the first file in the argument list
<code>:fixdel</code>	<code>:fix[del]</code>	set key code of <code></code>
<code>:fold</code>	<code>:fo[ld]</code>	create a fold
<code>:foldclose</code>	<code>:foldc[lose]</code>	close folds
<code>:folddoopen</code>	<code>:foldd[oopen]</code>	execute command on lines not in a closed fold
<code>:folddoclosed</code>	<code>:folddoc[losed]</code>	execute command on lines in a closed fold
<code>:foldopen</code>	<code>:foldo[pen]</code>	open folds
<code>:for</code>	<code>:for</code>	for loop
<code>:function</code>	<code>:fu[nction]</code>	define a user function
<code>:global</code>	<code>:g[lobal]</code>	execute commands for matching lines
<code>:goto</code>	<code>:go[to]</code>	go to byte in the buffer
<code>:grep</code>	<code>:gr[ep]</code>	run ' grepprg ' and jump to first match
<code>:grepadd</code>	<code>:grepa[dd]</code>	like <code>:grep</code> , but append to current list
<code>:gui</code>	<code>:gu[i]</code>	start the GUI
<code>:gvim</code>	<code>:gv[im]</code>	start the GUI
<code>:hardcopy</code>	<code>:ha[rdcopy]</code>	send text to the printer
<code>:help</code>	<code>:h[elp]</code>	open a help window
<code>:helpclose</code>	<code>:helpc[lose]</code>	close one help window
<code>:helpfind</code>	<code>:helpf[ind]</code>	dialog to open a help window
<code>:helpgrep</code>	<code>:helpg[rep]</code>	like <code>":grep"</code> but searches help files
<code>:helptags</code>	<code>:helpt[ags]</code>	generate help tags for a directory
<code>:highlight</code>	<code>:hi[ghlight]</code>	specify highlighting methods
<code>:hide</code>	<code>:hid[e]</code>	hide current buffer for a command
<code>:history</code>	<code>:his[tory]</code>	print a history list
<code>:horizontal</code>	<code>:hor[izontal]</code>	following window command work horizontally
<code>:insert</code>	<code>:i[nsert]</code>	insert text
<code>:iabbrev</code>	<code>:ia[bbrev]</code>	like <code>":abbrev"</code> but for Insert mode
<code>:iabclear</code>	<code>:iabc[lear]</code>	like <code>":abclear"</code> but for Insert mode
<code>:if</code>	<code>:if</code>	execute commands when condition met
<code>:ijump</code>	<code>:ij[ump]</code>	jump to definition of identifier
<code>:ilist</code>	<code>:il[ist]</code>	list lines where identifier matches
<code>:imap</code>	<code>:im[ap]</code>	like <code>":map"</code> but for Insert mode
<code>:imapclear</code>	<code>:imapc[lear]</code>	like <code>":mapclear"</code> but for Insert mode
<code>:imenu</code>	<code>:ime[nu]</code>	add menu for Insert mode
<code>:import</code>	<code>:imp[ort]</code>	Vim9: import an item from another script
<code>:inoremap</code>	<code>:ino[remap]</code>	like <code>":noremap"</code> but for Insert mode
<code>:inoreabbrev</code>	<code>:inorea[bbrev]</code>	like <code>":noreabbrev"</code> but for Insert mode
<code>:inoremenu</code>	<code>:inoreme[nu]</code>	like <code>":noremenu"</code> but for Insert mode
<code>:intro</code>	<code>:int[ro]</code>	print the introductory message
<code>:isearch</code>	<code>:is[earch]</code>	list one line where identifier matches
<code>:isplit</code>	<code>:isp[lit]</code>	split window and jump to definition of identifier
<code>:iunmap</code>	<code>:iu[nmap]</code>	like <code>":unmap"</code> but for Insert mode
<code>:iunabbrev</code>	<code>:iuna[bbrev]</code>	like <code>":unabbrev"</code> but for Insert mode
<code>:iunmenu</code>	<code>:iunme[nu]</code>	remove menu for Insert mode
<code>:join</code>	<code>:j[oin]</code>	join lines
<code>:jumps</code>	<code>:ju[mps]</code>	print the jump list
<code>:k</code>	<code>:k</code>	set a mark

<code>:keepalt</code>	<code>:keepa[lt]</code>	following command keeps the alternate file
<code>:keepmarks</code>	<code>:kee[pmarks]</code>	following command keeps marks where they are
<code>:keepjumps</code>	<code>:keepj[umps]</code>	following command keeps jumplist and marks
<code>:keeppatterns</code>	<code>:keep[atterns]</code>	following command keeps search pattern history
<code>:lNext</code>	<code>:lN[ext]</code>	go to previous entry in location list
<code>:lNfile</code>	<code>:lNf[ile]</code>	go to last entry in previous file
<code>:list</code>	<code>:l[ist]</code>	print lines
<code>:labove</code>	<code>:lab[ove]</code>	go to location above current line
<code>:laddexpr</code>	<code>:lad[dexpr]</code>	add locations from expr
<code>:laddbuffer</code>	<code>:laddb[uffer]</code>	add locations from buffer
<code>:laddfile</code>	<code>:laddf[ile]</code>	add locations to current location list
<code>:lafter</code>	<code>:laf[ter]</code>	go to location after current cursor
<code>:last</code>	<code>:la[st]</code>	go to the last file in the argument list
<code>:language</code>	<code>:lan[guage]</code>	set the language (locale)
<code>:later</code>	<code>:lat[er]</code>	go to newer change, redo
<code>:lbefore</code>	<code>:lbef[ore]</code>	go to location before current cursor
<code>:lbelow</code>	<code>:lbel[ow]</code>	go to location below current line
<code>:lbottom</code>	<code>:lbo[ttom]</code>	scroll to the bottom of the location window
<code>:lbuffer</code>	<code>:lb[uffer]</code>	parse locations and jump to first location
<code>:lcd</code>	<code>:lc[d]</code>	change directory locally
<code>:lchdir</code>	<code>:lch[dir]</code>	change directory locally
<code>:lclose</code>	<code>:lcl[ose]</code>	close location window
<code>:lscope</code>	<code>:lcs[cope]</code>	like <code>":cscope"</code> but uses location list
<code>:ldo</code>	<code>:ld[o]</code>	execute command in valid location list entries
<code>:lfdo</code>	<code>:lfd[o]</code>	execute command in each file in location list
<code>:left</code>	<code>:le[ft]</code>	left align lines
<code>:leftabove</code>	<code>:lefta[bove]</code>	make split window appear left or above
<code>:legacy</code>	<code>:leg[acy]</code>	make following command use legacy script syntax
<code>:let</code>	<code>:let</code>	assign a value to a variable or option
<code>:lexpr</code>	<code>:lex[pr]</code>	read locations from expr and jump to first
<code>:lfile</code>	<code>:lf[ile]</code>	read file with locations and jump to first
<code>:lfirst</code>	<code>:lfir[st]</code>	go to the specified location, default first one
<code>:lgetbuffer</code>	<code>:lgetb[uffer]</code>	get locations from buffer
<code>:lgetexpr</code>	<code>:lgete[xpr]</code>	get locations from expr
<code>:lgetfile</code>	<code>:lg[etfile]</code>	read file with locations
<code>:lgrep</code>	<code>:lgr[ep]</code>	run 'grep' and jump to first match
<code>:lgrepadd</code>	<code>:lgrepa[dd]</code>	like <code>:grep</code> , but append to current list
<code>:lhelpgrep</code>	<code>:lh[elpgrep]</code>	like <code>":helpgrep"</code> but uses location list
<code>:lhistory</code>	<code>:lhi[story]</code>	list the location lists
<code>:ll</code>	<code>:ll</code>	go to specific location
<code>:llast</code>	<code>:lla[st]</code>	go to the specified location, default last one
<code>:llist</code>	<code>:lli[st]</code>	list all locations
<code>:lmake</code>	<code>:lmak[e]</code>	execute external command 'makeprg' and parse error messages
<code>:lmap</code>	<code>:lm[ap]</code>	like <code>":map!"</code> but includes Lang-Arg mode
<code>:lmapclear</code>	<code>:lmapc[lear]</code>	like <code>":mapclear!"</code> but includes Lang-Arg mode
<code>:lnext</code>	<code>:lne[xt]</code>	go to next location
<code>:lnewer</code>	<code>:lnew[er]</code>	go to newer location list
<code>:lnfile</code>	<code>:lnf[ile]</code>	go to first location in next file
<code>:lnoremap</code>	<code>:ln[oremap]</code>	like <code>":noremap!"</code> but includes Lang-Arg mode
<code>:loadkeymap</code>	<code>:loadk[eymap]</code>	load the following keymaps until EOF
<code>:loadview</code>	<code>:lo[adview]</code>	load view for current window from a file
<code>:lockmarks</code>	<code>:loc[kmarks]</code>	following command keeps marks where they are
<code>:lockvar</code>	<code>:lockv[ar]</code>	lock variables

<code>:lolder</code>	<code>:lol[der]</code>	go to older location list
<code>:lopen</code>	<code>:lope[n]</code>	open location window
<code>:lprevious</code>	<code>:lp[revious]</code>	go to previous location
<code>:lpfile</code>	<code>:lpf[file]</code>	go to last location in previous file
<code>:lrewind</code>	<code>:lr[ewind]</code>	go to the specified location, default first one
<code>:ls</code>	<code>:ls</code>	list all buffers
<code>:ltag</code>	<code>:lt[ag]</code>	jump to tag and add matching tags to the location list
<code>:lunmap</code>	<code>:lu[nmap]</code>	like <code>":unmap!"</code> but includes Lang-Arg mode
<code>:lua</code>	<code>:lua</code>	execute Lua command
<code>:luado</code>	<code>:luad[o]</code>	execute Lua command for each line
<code>:luafile</code>	<code>:luaf[ile]</code>	execute Lua script file
<code>:lvimgrep</code>	<code>:lv[imgrep]</code>	search for pattern in files
<code>:lvimgrepadd</code>	<code>:lvimgrepa[dd]</code>	like <code>:vimgrep</code> , but append to current list
<code>:lwindow</code>	<code>:lw[indow]</code>	open or close location window
<code>:move</code>	<code>:m[ove]</code>	move lines
<code>:mark</code>	<code>:ma[rk]</code>	set a mark
<code>:make</code>	<code>:mak[e]</code>	execute external command 'makeprg' and parse error messages
<code>:map</code>	<code>:map</code>	show or enter a mapping
<code>:mapclear</code>	<code>:mapc[lear]</code>	clear all mappings for Normal and Visual mode
<code>:marks</code>	<code>:marks</code>	list all marks
<code>:match</code>	<code>:mat[ch]</code>	define a match to highlight
<code>:menu</code>	<code>:me[nu]</code>	enter a new menu item
<code>:menutranslate</code>	<code>:menut[ranslate]</code>	add a menu translation item
<code>:messages</code>	<code>:mes[sages]</code>	view previously displayed messages
<code>:mkexrc</code>	<code>:mk[exrc]</code>	write current mappings and settings to a file
<code>:mksession</code>	<code>:mks[ession]</code>	write session info to a file
<code>:mkspell</code>	<code>:mksp[ell]</code>	produce .spl spell file
<code>:mkvimrc</code>	<code>:mkv[imrc]</code>	write current mappings and settings to a file
<code>:mkview</code>	<code>:mkvie[w]</code>	write view of current window to a file
<code>:mode</code>	<code>:mod[e]</code>	show or change the screen mode
<code>:mzscheme</code>	<code>:mz[scheme]</code>	execute MzScheme command
<code>:mzfile</code>	<code>:mzf[ile]</code>	execute MzScheme script file
<code>:nbclose</code>	<code>:nbc[lose]</code>	close the current Netbeans session
<code>:nbkey</code>	<code>:nb[key]</code>	pass a key to Netbeans
<code>:nbstart</code>	<code>:nbs[art]</code>	start a new Netbeans session
<code>:next</code>	<code>:n[ext]</code>	go to next file in the argument list
<code>:new</code>	<code>:new</code>	create a new empty window
<code>:nmap</code>	<code>:nm[ap]</code>	like <code>":map"</code> but for Normal mode
<code>:nmapclear</code>	<code>:nmapc[lear]</code>	clear all mappings for Normal mode
<code>:nmenu</code>	<code>:nme[nu]</code>	add menu for Normal mode
<code>:nnoremap</code>	<code>:nn[oremap]</code>	like <code>":noremap"</code> but for Normal mode
<code>:nnoremenu</code>	<code>:nnoreme[nu]</code>	like <code>":noremenu"</code> but for Normal mode
<code>:noautocmd</code>	<code>:noa[utocmd]</code>	following commands don't trigger autocommands
<code>:noremap</code>	<code>:no[remap]</code>	enter a mapping that will not be remapped
<code>:nohlsearch</code>	<code>:noh[lsearch]</code>	suspend 'hlsearch' highlighting
<code>:noreabbrev</code>	<code>:norea[bbrev]</code>	enter an abbreviation that will not be remapped
<code>:noremenu</code>	<code>:noreme[nu]</code>	enter a menu that will not be remapped
<code>:normal</code>	<code>:norm[al]</code>	execute Normal mode commands
<code>:noswapfile</code>	<code>:nos[wapfile]</code>	following commands don't create a swap file
<code>:number</code>	<code>:nu[mber]</code>	print lines with line number
<code>:nunmap</code>	<code>:nun[map]</code>	like <code>":unmap"</code> but for Normal mode

<code>:nunmenu</code>	<code>:nunme[nu]</code>	remove menu for Normal mode
<code>:oldfiles</code>	<code>:ol[dfiles]</code>	list files that have marks in the viminfo file
<code>:open</code>	<code>:o[pen]</code>	start open mode (not implemented)
<code>:omap</code>	<code>:om[ap]</code>	like <code>":map"</code> but for Operator-pending mode
<code>:omapclear</code>	<code>:omapc[lear]</code>	remove all mappings for Operator-pending mode
<code>:omenu</code>	<code>:ome[nu]</code>	add menu for Operator-pending mode
<code>:only</code>	<code>:on[ly]</code>	close all windows except the current one
<code>:onoremap</code>	<code>:ono[remap]</code>	like <code>":noremap"</code> but for Operator-pending mode
<code>:onoremenu</code>	<code>:onoreme[nu]</code>	like <code>":noremenu"</code> but for Operator-pending mode
<code>:options</code>	<code>:opt[ions]</code>	open the options-window
<code>:ounmap</code>	<code>:ou[nmap]</code>	like <code>":unmap"</code> but for Operator-pending mode
<code>:ounmenu</code>	<code>:ounme[nu]</code>	remove menu for Operator-pending mode
<code>:ownsyntax</code>	<code>:ow[nsyntax]</code>	set new local syntax highlight for this window
<code>:packadd</code>	<code>:pa[ckadd]</code>	add a plugin from 'packpath'
<code>:packloadall</code>	<code>:packl[oadall]</code>	load all packages under 'packpath'
<code>:pclose</code>	<code>:pc[lose]</code>	close preview window
<code>:pedit</code>	<code>:ped[it]</code>	edit file in the preview window
<code>:perl</code>	<code>:pe[rl]</code>	execute Perl command
<code>:print</code>	<code>:p[rint]</code>	print lines
<code>:profdel</code>	<code>:profd[el]</code>	stop profiling a function or script
<code>:profile</code>	<code>:prof[ile]</code>	profiling functions and scripts
<code>:promptfind</code>	<code>:pro[mptfind]</code>	open GUI dialog for searching
<code>:promptrepl</code>	<code>:promptr[ep]</code>	open GUI dialog for search/replace
<code>:perldo</code>	<code>:perld[o]</code>	execute Perl command for each line
<code>:pop</code>	<code>:po[p]</code>	jump to older entry in tag stack
<code>:popup</code>	<code>:popu[p]</code>	popup a menu by name
<code>:ppop</code>	<code>:pp[op]</code>	<code>":pop"</code> in preview window
<code>:preserve</code>	<code>:pre[serve]</code>	write all text to swap file
<code>:previous</code>	<code>:prev[ious]</code>	go to previous file in argument list
<code>:psearch</code>	<code>:ps[earch]</code>	like <code>":ijump"</code> but shows match in preview window
<code>:ptag</code>	<code>:pt[ag]</code>	show tag in preview window
<code>:ptNext</code>	<code>:ptN[ext]</code>	<code>:tNext</code> in preview window
<code>:ptfirst</code>	<code>:ptf[irst]</code>	<code>:trewind</code> in preview window
<code>:ptjump</code>	<code>:ptj[ump]</code>	<code>:tjump</code> and show tag in preview window
<code>:ptlast</code>	<code>:ptl[ast]</code>	<code>:tlast</code> in preview window
<code>:ptnext</code>	<code>:ptn[ext]</code>	<code>:tnext</code> in preview window
<code>:ptprevious</code>	<code>:ptp[revious]</code>	<code>:tprevious</code> in preview window
<code>:ptrewind</code>	<code>:ptr[ewind]</code>	<code>:trewind</code> in preview window
<code>:ptselect</code>	<code>:pts[elect]</code>	<code>:tselect</code> and show tag in preview window
<code>:public</code>	<code>:public</code>	prefix for a class or object member
<code>:put</code>	<code>:pu[t]</code>	insert contents of register in the text
<code>:pwd</code>	<code>:pw[d]</code>	print current directory
<code>:py3</code>	<code>:py3</code>	execute Python 3 command
<code>:python3</code>	<code>:python3</code>	same as <code>:py3</code>
<code>:py3do</code>	<code>:py3d[o]</code>	execute Python 3 command for each line
<code>:py3file</code>	<code>:py3f[ile]</code>	execute Python 3 script file
<code>:python</code>	<code>:py[thon]</code>	execute Python command
<code>:pydo</code>	<code>:pyd[o]</code>	execute Python command for each line
<code>:pyfile</code>	<code>:pyf[ile]</code>	execute Python script file
<code>:pyx</code>	<code>:pyx</code>	execute <code>python_x</code> command
<code>:pythonx</code>	<code>:pythonx</code>	same as <code>:pyx</code>
<code>:pyxd</code>	<code>:pyxd[o]</code>	execute <code>python_x</code> command for each line
<code>:pyxfile</code>	<code>:pyxfile</code>	execute <code>python_x</code> script file
<code>:quit</code>	<code>:q[uit]</code>	quit current window (when one window quit Vim)

<code>:quitall</code>	<code>:quita[ll]</code>	quit Vim
<code>:qall</code>	<code>:qa[ll]</code>	quit Vim
<code>:read</code>	<code>:r[ead]</code>	read file into the text
<code>:recover</code>	<code>:rec[over]</code>	recover a file from a swap file
<code>:redo</code>	<code>:red[o]</code>	redo one undone change
<code>:redir</code>	<code>:redi[r]</code>	redirect messages to a file or register
<code>:redraw</code>	<code>:redr[aw]</code>	force a redraw of the display
<code>:redrawstatus</code>	<code>:redraws[tatus]</code>	force a redraw of the status line(s)
<code>:redrawtabline</code>	<code>:redrawt[abline]</code>	force a redraw of the tabline
<code>:registers</code>	<code>:reg[isters]</code>	display the contents of registers
<code>:resize</code>	<code>:res[ize]</code>	change current window height
<code>:retab</code>	<code>:ret[ab]</code>	change tab size
<code>:return</code>	<code>:retu[rn]</code>	return from a user function
<code>:rewind</code>	<code>:rew[ind]</code>	go to the first file in the argument list
<code>:right</code>	<code>:ri[ght]</code>	right align text
<code>:rightbelow</code>	<code>:rightb[elow]</code>	make split window appear right or below
<code>:ruby</code>	<code>:rub[y]</code>	execute Ruby command
<code>:rubydo</code>	<code>:rubyd[o]</code>	execute Ruby command for each line
<code>:rubyfile</code>	<code>:rubyf[ile]</code>	execute Ruby script file
<code>:rundo</code>	<code>:rund[o]</code>	read undo information from a file
<code>:runtime</code>	<code>:ru[n]time</code>	source vim scripts in 'runtimepath'
<code>:rviminfo</code>	<code>:rv[im]info</code>	read from viminfo file
<code>:substitute</code>	<code>:s[ub]stitute</code>	find and replace text
<code>:sNext</code>	<code>:sN[ext]</code>	split window and go to previous file in argument list
<code>:sandbox</code>	<code>:san[d]box</code>	execute a command in the sandbox
<code>:sargument</code>	<code>:sa[r]gument</code>	split window and go to specific file in argument list
<code>:sall</code>	<code>:sal[l]</code>	open a window for each file in argument list
<code>:saveas</code>	<code>:sav[e]as</code>	save file under another name.
<code>:sbuffer</code>	<code>:sb[uff]er</code>	split window and go to specific file in the buffer list
<code>:sbNext</code>	<code>:sbN[ext]</code>	split window and go to previous file in the buffer list
<code>:sball</code>	<code>:sba[ll]</code>	open a window for each file in the buffer list
<code>:sbfirst</code>	<code>:sbf[ir]st</code>	split window and go to first file in the buffer list
<code>:sblast</code>	<code>:sbl[ast]</code>	split window and go to last file in buffer list
<code>:sbmodified</code>	<code>:sbm[od]ified</code>	split window and go to modified file in the buffer list
<code>:sbnext</code>	<code>:sbn[ext]</code>	split window and go to next file in the buffer list
<code>:sbprevious</code>	<code>:sbp[re]vious</code>	split window and go to previous file in the buffer list
<code>:sbrewind</code>	<code>:sbr[ew]ind</code>	split window and go to first file in the buffer list
<code>:scriptnames</code>	<code>:scr[ipt]names</code>	list names of all sourced Vim scripts
<code>:scriptencoding</code>	<code>:scripte[n]coding</code>	encoding used in sourced Vim script
<code>:scriptversion</code>	<code>:scriptv[er]sion</code>	version of Vim script used
<code>:scscope</code>	<code>:scs[co]pe</code>	split window and execute cscope command
<code>:set</code>	<code>:se[t]</code>	show or set options
<code>:setfiletype</code>	<code>:setf[ile]type</code>	set 'filetype' , unless it was set already
<code>:setglobal</code>	<code>:setg[lob]al</code>	show global values of options

<code>:setlocal</code>	<code>:setl[ocal]</code>	show or set options locally
<code>:sfind</code>	<code>:sf[ind]</code>	split current window and edit file in 'path'
<code>:sfirst</code>	<code>:sfir[st]</code>	split window and go to first file in the argument list
<code>:shell</code>	<code>:sh[ell]</code>	escape to a shell
<code>:simalt</code>	<code>:sim[alt]</code>	Win32 GUI: simulate Windows ALT key
<code>:sign</code>	<code>:sig[n]</code>	manipulate signs
<code>:silent</code>	<code>:sil[ent]</code>	run a command silently
<code>:sleep</code>	<code>:sl[eeep]</code>	do nothing for a few seconds
<code>:sleep!</code>	<code>:sl[eeep]!</code>	do nothing for a few seconds, without the cursor visible
<code>:slast</code>	<code>:sla[st]</code>	split window and go to last file in the argument list
<code>:smagic</code>	<code>:sm[agic]</code>	:substitute with 'magic'
<code>:smap</code>	<code>:smap</code>	like ":map" but for Select mode
<code>:smapclear</code>	<code>:smapc[lear]</code>	remove all mappings for Select mode
<code>:smenu</code>	<code>:sme[nu]</code>	add menu for Select mode
<code>:smile</code>	<code>:smi[le]</code>	make the user happy
<code>:snext</code>	<code>:sn[ext]</code>	split window and go to next file in the argument list
<code>:snomagic</code>	<code>:sno[magic]</code>	:substitute with 'nomagic'
<code>:snoremap</code>	<code>:snor[emap]</code>	like ":noremap" but for Select mode
<code>:snoremenu</code>	<code>:snoreme[nu]</code>	like ":noremenu" but for Select mode
<code>:sort</code>	<code>:sor[t]</code>	sort lines
<code>:source</code>	<code>:so[urce]</code>	read Vim or Ex commands from a file
<code>:spelledump</code>	<code>:speld[ump]</code>	split window and fill with all correct words
<code>:spellgood</code>	<code>:spe[llgood]</code>	add good word for spelling
<code>:spellinfo</code>	<code>:spelli[nfo]</code>	show info about loaded spell files
<code>:spellrare</code>	<code>:spellra[re]</code>	add rare word for spelling
<code>:spellrepall</code>	<code>:spellr[epall]</code>	replace all bad words like last z=
<code>:spellundo</code>	<code>:spellu[ndo]</code>	remove good or bad word
<code>:spellwrong</code>	<code>:spellw[rong]</code>	add spelling mistake
<code>:split</code>	<code>:sp[lit]</code>	split current window
<code>:sprevious</code>	<code>:spr[evious]</code>	split window and go to previous file in the argument list
<code>:srewind</code>	<code>:sre[wind]</code>	split window and go to first file in the argument list
<code>:stop</code>	<code>:st[op]</code>	suspend the editor or escape to a shell
<code>:stag</code>	<code>:sta[g]</code>	split window and jump to a tag
<code>:startinsert</code>	<code>:star[tinsert]</code>	start Insert mode
<code>:startgreplace</code>	<code>:startg[replace]</code>	start Virtual Replace mode
<code>:startreplace</code>	<code>:startr[eplace]</code>	start Replace mode
<code>:static</code>	<code>:static</code>	prefix for a class member or function
<code>:stopinsert</code>	<code>:stopi[nsert]</code>	stop Insert mode
<code>:stjump</code>	<code>:stj[ump]</code>	do ":tjump" and split window
<code>:stselect</code>	<code>:sts[elect]</code>	do ":tselect" and split window
<code>:sunhide</code>	<code>:sun[hide]</code>	same as ":unhide"
<code>:sunmap</code>	<code>:sunm[ap]</code>	like ":unmap" but for Select mode
<code>:sunmenu</code>	<code>:sunme[nu]</code>	remove menu for Select mode
<code>:suspend</code>	<code>:sus[pend]</code>	same as ":stop"
<code>:sview</code>	<code>:sv[iew]</code>	split window and edit file read-only
<code>:swapname</code>	<code>:sw[apname]</code>	show the name of the current swap file
<code>:syntax</code>	<code>:sy[ntax]</code>	syntax highlighting
<code>:syntime</code>	<code>:synti[me]</code>	measure syntax highlighting speed

<code>:syncbind</code>	<code>:sync[bind]</code>	sync scroll binding
<code>:t</code>	<code>:t</code>	same as <code>":copy"</code>
<code>:tNext</code>	<code>:tN[ext]</code>	jump to previous matching tag
<code>:tabNext</code>	<code>:tabN[ext]</code>	go to previous tab page
<code>:tabclose</code>	<code>:tabc[lose]</code>	close current tab page
<code>:tabdo</code>	<code>:tabdo</code>	execute command in each tab page
<code>:tabedit</code>	<code>:tabe[dit]</code>	edit a file in a new tab page
<code>:tabfind</code>	<code>:tabf[ind]</code>	find file in 'path' , edit it in a new tab page
<code>:tabfirst</code>	<code>:tabfir[st]</code>	go to first tab page
<code>:tablast</code>	<code>:tabl[ast]</code>	go to last tab page
<code>:tabmove</code>	<code>:tabm[ove]</code>	move tab page to other position
<code>:tabnew</code>	<code>:tabnew</code>	edit a file in a new tab page
<code>:tabnext</code>	<code>:tabn[ext]</code>	go to next tab page
<code>:tabonly</code>	<code>:tabo[nly]</code>	close all tab pages except the current one
<code>:tabprevious</code>	<code>:tabp[revious]</code>	go to previous tab page
<code>:tabrewind</code>	<code>:tabr[ewind]</code>	go to first tab page
<code>:tabs</code>	<code>:tabs</code>	list the tab pages and what they contain
<code>:tab</code>	<code>:tab</code>	create new tab when opening new window
<code>:tag</code>	<code>:ta[g]</code>	jump to tag
<code>:tags</code>	<code>:tags</code>	show the contents of the tag stack
<code>:tcd</code>	<code>:tc[d]</code>	change directory for tab page
<code>:tchdir</code>	<code>:tch[dir]</code>	change directory for tab page
<code>:tcl</code>	<code>:tcl</code>	execute Tcl command
<code>:tcldo</code>	<code>:tcldo[o]</code>	execute Tcl command for each line
<code>:tclfile</code>	<code>:tclf[ile]</code>	execute Tcl script file
<code>:tearoff</code>	<code>:te[aroff]</code>	tear-off a menu
<code>:terminal</code>	<code>:ter[minal]</code>	open a terminal window
<code>:tfirst</code>	<code>:tf[irst]</code>	jump to first matching tag
<code>:throw</code>	<code>:th[row]</code>	throw an exception
<code>:tjump</code>	<code>:tj[ump]</code>	like <code>":tselect"</code> , but jump directly when there is only one match
<code>:tlast</code>	<code>:tl[ast]</code>	jump to last matching tag
<code>:tlmenu</code>	<code>:tlm[enu]</code>	add menu for Terminal-Job mode
<code>:tlnoemenu</code>	<code>:tln[oremenu]</code>	like <code>":noemenu"</code> but for Terminal-Job mode
<code>:tlunmenu</code>	<code>:tlu[nmenu]</code>	remove menu for Terminal-Job mode
<code>:tmapclear</code>	<code>:tmapc[lear]</code>	remove all mappings for Terminal-Job mode
<code>:tmap</code>	<code>:tma[p]</code>	like <code>":map"</code> but for Terminal-Job mode
<code>:tmenu</code>	<code>:tm[enu]</code>	define menu tooltip
<code>:tnext</code>	<code>:tn[ext]</code>	jump to next matching tag
<code>:tnoremap</code>	<code>:tno[remap]</code>	like <code>":noremap"</code> but for Terminal-Job mode
<code>:topleft</code>	<code>:to[pleft]</code>	make split window appear at top or far left
<code>:tprevious</code>	<code>:tp[revious]</code>	jump to previous matching tag
<code>:trewind</code>	<code>:tr[ewind]</code>	jump to first matching tag
<code>:try</code>	<code>:try</code>	execute commands, abort on error or exception
<code>:tselect</code>	<code>:ts[elect]</code>	list matching tags and select one
<code>:tunmap</code>	<code>:tunma[p]</code>	like <code>":unmap"</code> but for Terminal-Job mode
<code>:tunmenu</code>	<code>:tu[nmenu]</code>	remove menu tooltip
<code>:undo</code>	<code>:u[ndo]</code>	undo last change(s)
<code>:undojoin</code>	<code>:undoj[oin]</code>	join next change with previous undo block
<code>:undolist</code>	<code>:undol[ist]</code>	list leafs of the undo tree
<code>:unabbreviate</code>	<code>:una[bbreviate]</code>	remove abbreviation
<code>:unhide</code>	<code>:unh[ide]</code>	open a window for each loaded file in the buffer list
<code>:unlet</code>	<code>:unl[et]</code>	delete variable

<code>:unlockvar</code>	<code>:unlo[ckvar]</code>	unlock variables
<code>:unmap</code>	<code>:unm[ap]</code>	remove mapping
<code>:unmenu</code>	<code>:unme[nu]</code>	remove menu
<code>:unsilent</code>	<code>:uns[ilent]</code>	run a command not silently
<code>:update</code>	<code>:up[date]</code>	write buffer if modified
<code>:vglobal</code>	<code>:v[global]</code>	execute commands for not matching lines
<code>:var</code>	<code>:var</code>	variable declaration in Vim9
<code>:version</code>	<code>:ve[rsion]</code>	print version number and other info
<code>:verbose</code>	<code>:verb[ose]</code>	execute command with 'verbose' set
<code>:vertical</code>	<code>:vert[ical]</code>	make following command split vertically
<code>:vim9cmd</code>	<code>:vim9[cmd]</code>	make following command use Vim9 script syntax
<code>:vim9script</code>	<code>:vim9s[cript]</code>	indicates Vim9 script file
<code>:vimgrep</code>	<code>:vim[grep]</code>	search for pattern in files
<code>:vimgrepadd</code>	<code>:vimgrepa[dd]</code>	like <code>:vimgrep</code> , but append to current list
<code>:visual</code>	<code>:vi[sual]</code>	same as <code>":edit"</code> , but turns off "Ex" mode
<code>:viusage</code>	<code>:viu[sage]</code>	overview of Normal mode commands
<code>:view</code>	<code>:vie[w]</code>	edit a file read-only
<code>:vmap</code>	<code>:vm[ap]</code>	like <code>":map"</code> but for Visual+Select mode
<code>:vmapclear</code>	<code>:vmapc[lear]</code>	remove all mappings for Visual+Select mode
<code>:vmenu</code>	<code>:vme[nu]</code>	add menu for Visual+Select mode
<code>:vnew</code>	<code>:vne[w]</code>	create a new empty window, vertically split
<code>:vnoremap</code>	<code>:vn[oremap]</code>	like <code>":noremap"</code> but for Visual+Select mode
<code>:vnoremenu</code>	<code>:vnoreme[nu]</code>	like <code>":noremenu"</code> but for Visual+Select mode
<code>:vsplit</code>	<code>:vs[plit]</code>	split current window vertically
<code>:vunmap</code>	<code>:vu[nmap]</code>	like <code>":unmap"</code> but for Visual+Select mode
<code>:vunmenu</code>	<code>:vunme[nu]</code>	remove menu for Visual+Select mode
<code>:windo</code>	<code>:windo</code>	execute command in each window
<code>:write</code>	<code>:w[rite]</code>	write to a file
<code>:wNext</code>	<code>:wN[ext]</code>	write to a file and go to previous file in argument list
<code>:wall</code>	<code>:wa[ll]</code>	write all (changed) buffers
<code>:while</code>	<code>:wh[ile]</code>	execute loop for as long as condition met
<code>:winsize</code>	<code>:wi[nsize]</code>	get or set window size (obsolete)
<code>:wincmd</code>	<code>:winc[md]</code>	execute a Window (CTRL-W) command
<code>:winpos</code>	<code>:winp[os]</code>	get or set window position
<code>:wnext</code>	<code>:wn[ext]</code>	write to a file and go to next file in argument list
<code>:wprevious</code>	<code>:wp[revious]</code>	write to a file and go to previous file in argument list
<code>:wq</code>	<code>:wq</code>	write to a file and quit window or Vim
<code>:wqall</code>	<code>:wqa[ll]</code>	write all changed buffers and quit Vim
<code>:wundo</code>	<code>:wu[ndo]</code>	write undo information to a file
<code>:wviminfo</code>	<code>:wv[iminfo]</code>	write to viminfo file
<code>:xit</code>	<code>:x[it]</code>	write if buffer changed and close window
<code>:xall</code>	<code>:xa[ll]</code>	same as <code>":wqall"</code>
<code>:xmapclear</code>	<code>:xmapc[lear]</code>	remove all mappings for Visual mode
<code>:xmap</code>	<code>:xm[ap]</code>	like <code>":map"</code> but for Visual mode
<code>:xmenu</code>	<code>:xme[nu]</code>	add menu for Visual mode
<code>:xrestore</code>	<code>:xr[estore]</code>	restores the X server connection
<code>:xnoremap</code>	<code>:xn[oremap]</code>	like <code>":noremap"</code> but for Visual mode
<code>:xnoremenu</code>	<code>:xnoreme[nu]</code>	like <code>":noremenu"</code> but for Visual mode
<code>:xunmap</code>	<code>:xu[nmap]</code>	like <code>":unmap"</code> but for Visual mode
<code>:xunmenu</code>	<code>:xunme[nu]</code>	remove menu for Visual mode
<code>:yank</code>	<code>:y[ank]</code>	yank lines into a register

```
:z  
:~
```

```
:z  
:~
```

```
print some lines  
repeat last ":substitute"
```

VIM REFERENCE MANUAL by Bram Moolenaar

How to ...

[howdoi](#) [how-do-i](#) [howto](#) [how-to](#)

tutor	get started
:quit	exit? I'm trapped, help me!
initialization	initialize Vim
vimrc-intro	write a Vim script file (vimrc)
suspend	suspend Vim
usr_11.txt	recover after a crash
07.4	keep a backup of my file when writing over it
usr_07.txt	edit files
23.4	edit binary files
usr_24.txt	insert text
deleting	delete text
usr_04.txt	change text
04.5	copy and move text
usr_25.txt	format text
30.6	format comments
30.2	indent C programs
25.3	automatically set indent
usr_26.txt	repeat commands
02.5	undo and redo
usr_03.txt	move around
word-motions	word motions
left-right-motions	left-right motions
up-down-motions	up-down motions
object-motions	text-object motions
various-motions	various motions
object-select	text-object selection
'whichwrap'	move over line breaks
'virtualedit'	move to where there is no text
usr_27.txt	specify pattern for searches
tags-and-searches	do tags and special searches
29.4	search in include'd files used to find variables, functions, or macros
K	look up manual for the keyword under cursor
03.7	scroll
'sidescroll'	scroll horizontally/sideways
'scrolloff'	set visible context lines
mode-switching	change modes
04.4	use Visual mode
'insertmode'	start Vim in Insert mode
40.1	map keys

24.7	create abbreviations
ins-expandtab	expand a tab to spaces in Insert mode
i_CTRL-R	insert contents of a register in Insert mode
24.3	complete words in Insert mode
25.1	break a line before it gets too long
20.1	do command-line editing
20.3	do command-line completion
'cmdheight'	increase the height of command-line
10.3	specify command-line ranges
40.3	specify commands to be executed automatically before/after reading/writing entering/leaving a buffer/window
'autowrite'	write automatically
30.1	speedup edit-compile-edit cycle or compile and fix errors within Vim
options	set options
auto-setting	set options automatically
term-dependent-settings	set options depending on terminal name
save-settings	save settings
:quote	comment my .vim files
'helpheight'	change the default help height
'highlight'	set various highlighting modes
'title'	set the window title
'icon'	set window icon title
'report'	avoid seeing the change messages on every line
'shortmess'	avoid <code>hit-enter</code> prompts
mouse-using	use mouse with Vim
usr_08.txt	manage multiple windows and buffers
gui.txt	use the gui
You can't! (yet)	do dishes using Vim
usr_06.txt	switch on syntax highlighting
2html.vim	convert a colored file to HTML
less	use Vim like less or more with syntax highlighting

Tips and ideas for using Vim

tips

These are just a few that we thought would be helpful for many users. You can find many more tips on the wiki. The URL can be found on <http://www.vim.org>

Don't forget to browse the user manual, it also contains lots of useful tips [usr_toc.txt](#) .

Editing C programs	C-editing
Finding where identifiers are used	ident-search
Switching screens in an xterm	xterm-screens
Scrolling in Insert mode	scroll-insert
Smooth scrolling	scroll-smooth
Correcting common typing mistakes	type-mistakes
Counting words, lines, etc.	count-items
Restoring the cursor position	restore-position
Renaming files	rename-files
Change a name in multiple files	change-name
Speeding up external commands	speed-up
Useful mappings	useful-mappings
Compressing the help files	gzip-helpfile
Executing shell commands in a window	shell-window
Hex editing	hex-editing
Using <> notation in autocommands	autocmd-<>
Highlighting matching parens	match-parens
Opening help in the current window	help-curwin

=====
Editing C programs

C-editing

There are quite a few features in Vim to help you edit C program files. Here is an overview with tags to jump to:

usr_29.txt	Moving through programs chapter in the user manual.
usr_30.txt	Editing programs chapter in the user manual.
C-indenting	Automatically set the indent of a line while typing text.
=	Re-indent a few lines.
format-comments	Format comments.
:checkpath	Show all recursively included files.
[i	Search for identifier under cursor in current and included files.
[_CTRL-I	Jump to match for "[i"
[I	List all lines in current and included files where identifier under the cursor matches.
[d	Search for define under cursor in current and included

files.

<code>CTRL-]</code>	Jump to tag under cursor (e.g., definition of a function).
<code>CTRL-T</code>	Jump back to before a CTRL-] command.
<code>:tselect</code>	Select one tag out of a list of matching tags.
<code>gd</code>	Go to Declaration of local variable under cursor.
<code>gD</code>	Go to Declaration of global variable under cursor.
<code>gf</code>	Go to file name under the cursor.
<code>%</code>	Go to matching <code>()</code> , <code>{}</code> , <code>[]</code> , <code>/* */</code> , <code>#if</code> , <code>#else</code> , <code>#endif</code> .
<code>[/</code>	Go to previous start of comment.
<code>]/</code>	Go to next end of comment.
<code>[#</code>	Go back to unclosed <code>#if</code> , <code>#ifdef</code> , or <code>#else</code> .
<code>]#</code>	Go forward to unclosed <code>#else</code> or <code>#endif</code> .
<code>[(</code>	Go back to unclosed <code>'(</code>
<code>)</code>	Go forward to unclosed <code>')</code>
<code>[{</code>	Go back to unclosed <code>'{</code>
<code>}]</code>	Go forward to unclosed <code>}'</code>
<code>v_ab</code>	Select "a block" from <code>"[(</code> to <code>)]"</code> , including braces
<code>v_ib</code>	Select "inner block" from <code>"[(</code> to <code>)]"</code>
<code>v_aB</code>	Select "a block" from <code>"[{</code> to <code>}]"</code> , including brackets
<code>v_iB</code>	Select "inner block" from <code>"[{</code> to <code>}]"</code>

=====

Finding where identifiers are used

`ident-search`

You probably already know that `tags` can be used to jump to the place where a function or variable is defined. But sometimes you wish you could jump to all the places where a function or variable is being used. This is possible in two ways:

1. Using the `:grep` command. This should work on most Unix systems, but can be slow (it reads all files) and only searches in one directory.
2. Using ID utils. This is fast and works in multiple directories. It uses a database to store locations. You will need some additional programs for this to work. And you need to keep the database up to date.

Using the GNU id-tools:

What you need:

- The GNU id-tools installed (`mkid` is needed to create ID and `lid` is needed to use the macros).
- An identifier database file called "ID" in the current directory. You can create it with the shell command `"mkid file1 file2 .."`.

Put this in your `.vimrc`:

```
map _u :call ID_search()<Bar>execute "/\<" .. g:word .. "\>"<CR>
map _n :n<Bar>execute "/\<" .. g:word .. "\>"<CR>
```

```
function! ID_search()
  let g:word = expand("<cword>")
```

```

let x = system("lid --key=none " .. g:word)
let x = substitute(x, "\n", " ", "g")
execute "next " .. x
endfun

```

To use it, place the cursor on a word, type "_u" and vim will load the file that contains the word. Search for the next occurrence of the word in the same file with "n". Go to the next file with "_n".

This has been tested with id-utils-3.2 (which is the name of the id-tools archive file on your closest gnu-ftp-mirror).

[the idea for this comes from Andreas Kutschera]

=====

Switching screens in an xterm xterm-screens xterm-save-screen

(From comp.editors, by Juergen Weigert, in reply to a question)

```

:> Another question is that after exiting vim, the screen is left as it
:> was, i.e. the contents of the file I was viewing (editing) was left on
:> the screen. The output from my previous like "ls" were lost,
:> ie. no longer in the scrolling buffer. I know that there is a way to
:> restore the screen after exiting vim or other vi like editors,
:> I just don't know how. Helps are appreciated. Thanks.
:
:I imagine someone else can answer this. I assume though that vim and vi do
:the same thing as each other for a given xterm setup.

```

They not necessarily do the same thing, as this may be a termcap vs. terminfo problem. You should be aware that there are two databases for describing attributes of a particular type of terminal: termcap and terminfo. This can cause differences when the entries differ AND when of the programs in question one uses terminfo and the other uses termcap (also see [+terminfo](#)).

In your particular problem, you are looking for the control sequences `^[[?47h` and `^[[?47l`. These switch between xterms alternate and main screen buffer. As a quick workaround a command sequence like

```

echo -n "^[[?47h"; vim ... ; echo -n "^[[?47l"

```

may do what you want. (My notation `^[` means the ESC character, further down you'll see that the databases use `\E` instead).

On startup, vim echoes the value of the termcap variable `ti` (terminfo: `smcup`) to the terminal. When exiting, it echoes `te` (terminfo: `rmcup`). Thus these two variables are the correct place where the above mentioned control sequences should go.

Compare your xterm termcap entry (found in `/etc/termcap`) with your xterm terminfo entry (retrieved with `"infocmp -C xterm"`). Both should contain entries similar to:

```

:te=\E[2J\E[?47l\E8:ti=\E\E[?47h:

```

PS: If you find any difference, someone (your sysadmin?) should better check


```
:%s/^//n          lines
:%s/the/&/gn       "the" anywhere
:%s/\/&/gn    "the" as a word
```

You might want to reset **'hlsearch'** or do `":nohlsearch"`.
Add the 'e' flag if you don't want an error when there are no matches.

An alternative is using `v_g_CTRL-G` in Visual mode.

If you want to find matches in multiple files use `:vimgrep` .

If you want to count bytes, you can use this: count-bytes

```
Visually select the characters (block is also possible)
Use "y" to yank the characters
Use the strlen() function:
:echo strlen(@)
```

A line break is counted for one byte.

===== restore-position

Sometimes you want to write a mapping that makes a change somewhere in the file and restores the cursor position, without scrolling the text. For example, to change the date mark in a file:

```
:map <F2> msHmtgg/Last [cC]hange:\s*/e+1<CR>"_D"=strftime("%Y %b %d")<CR>p'tzt`s
```

Breaking up saving the position:

```
ms      store cursor position in the 's' mark
H       go to the first line in the window
mt      store this position in the 't' mark
```

Breaking up restoring the position:

```
't      go to the line previously at the top of the window
zt      scroll to move this line to the top of the window
`s      jump to the original position of the cursor
```

For something more advanced see `winsaveview()` and `winrestview()` .

===== rename-files

Say I have a directory with the following files in them (directory picked at random :-):

```
buffer.c
charset.c
digraph.c
...
```

and I want to rename *.c *.bla. I'd do it like this:

```
$ vim
```

```
:r !ls *.c
:%s/\(.*\).c/mv & \1.bla
:w !sh
:q!
```

Change a name in multiple files

change-name

Example for using a script file to change a name in several files:

Create a file "subs.vim" containing substitute commands and a :update command:

```
:%s/Jones/Smith/g
:%s/Allen/Peter/g
:update
```

Execute Vim on all files you want to change, and source the script for each argument:

```
vim *.let
argdo source subs.vim
```

See `:argdo` .

Speeding up external commands

speed-up

In some situations, execution of an external command can be very slow. This can also slow down wildcard expansion on Unix. Here are a few suggestions to increase the speed.

If your .cshrc (or other file, depending on the shell used) is very long, you should separate it into a section for interactive use and a section for non-interactive use (often called secondary shells). When you execute a command from Vim like "!:ls", you do not need the interactive things (for example, setting the prompt). Put the stuff that is not needed after these lines:

```
if ($?prompt == 0) then
    exit 0
endif
```

Another way is to include the "-f" flag in the 'shell' option, e.g.:

```
:set shell=csh\ -f
```

(the backslash is needed to include the space in the option).

This will make csh completely skip the use of the .cshrc file. This may cause some things to stop working though.

Useful mappings

useful-mappings

Here are a few mappings that some people like to use.

map-backtick

```
:map ' `
```

Make the single quote work like a backtick. Puts the cursor on the column of a mark, instead of going to the first non-blank character in the line.

emacs-keys

For Emacs-style editing on the command-line:

```
" start of line
:cnoremap <C-A>      <Home>
" back one character
:cnoremap <C-B>      <Left>
" delete character under cursor
:cnoremap <C-D>      <Del>
" end of line
:cnoremap <C-E>      <End>
" forward one character
:cnoremap <C-F>      <Right>
" recall newer command-line
:cnoremap <C-N>      <Down>
" recall previous (older) command-line
:cnoremap <C-P>      <Up>
" back one word
:cnoremap <Esc><C-B>  <S-Left>
" forward one word
:cnoremap <Esc><C-F>  <S-Right>
```

NOTE: This requires that the '<' flag is excluded from **'cptions'**. <>

format-bullet-list

This mapping will format any bullet list. It requires that there is an empty line above and below each list entry. The expression commands are used to be able to give comments to the parts of the mapping.

```
:let m =      ":map _f :set ai<CR>"      " need 'autoindent' set
:let m ..= "{0<Esc>"                        " add empty line above item
:let m ..= "}}{}^W"                          " move to text after bullet
:let m ..= "i      <CR>      <Esc>"          " add space for indent
:let m ..= "gq}"                              " format text after the bullet
:let m ..= "{dd"                              " remove the empty line
:let m ..= "5lDJ"                             " put text after bullet
:execute m                                     |" define the mapping
```

(<> notation <> . **Note** that this is all typed literally. ^W is "^" "W", not **CTRL-W**. You can copy/paste this into Vim if '<' is not included in **'cptions'**.)

Note that the last comment starts with |", because the ":execute" command doesn't accept a comment directly.

You also need to set **'textwidth'** to a non-zero value, e.g.,

```
:set tw=70
```

A mapping that does about the same, but takes the indent for the list from the

first line (Note: this mapping is a single long line with a lot of spaces):
:map _f :set ai<CR>}}{a

collapse

These two mappings reduce a sequence of empty (;b) or blank (;n) lines into a single line

```
:map ;b GoZ<Esc>:g/^$/././-j<CR>Gdd
:map ;n GoZ<Esc>:g/^[ <Tab>]*$/.//[^ <Tab>]/-j<CR>Gdd
```

=====
Compressing the help files

gzip-helpfile

For those of you who are really short on disk space, you can compress the help files and still be able to view them with Vim. This makes accessing the help files a bit slower and requires the "gzip" program.

- (1) Compress all the help files: "gzip doc/*.txt".
- (2) Edit "doc/tags" and change the ".txt" to ".txt.gz":
:%s=(\t.*\.txt)\t=1.gz\t=
- (3) Add this line to your vimrc:
set helpfile={dirname}/help.txt.gz

Where {dirname} is the directory where the help files are. The `gzip` plugin will take care of decompressing the files. You must make sure that \$VIMRUNTIME is set to where the other Vim files are, when they are not in the same location as the compressed "doc" directory. See `$VIMRUNTIME` .

=====
Executing shell commands in a window

shell-window

See `terminal` .

Another solution is splitting your terminal screen or display window with the "splitvt" program. You can probably find it on some ftp server. The person that knows more about this is Sam Lantinga <slouken@cs.ucdavis.edu>.

Another alternative is the "window" command, found on BSD Unix systems, which supports multiple overlapped windows. Or the "screen" program, found at www.uni-erlangen.de, which supports a stack of windows.

=====
Hex editing

hex-editing using-xxd

See section [23.4](#) of the user manual.

If one has a particular extension that one uses for binary files (such as exe, bin, etc), you may find it helpful to automate the process with the following bit of autocmds for your `<.vimrc>`. Change that "*.bin" to whatever comma-separated list of extension(s) you find yourself wanting to edit:

```
" vim -b : edit binary using xxd-format!
```

```

augroup Binary
  autocmd!
  autocmd BufReadPre *.bin set binary
  autocmd BufReadPost *.bin
    \ if &binary
    \ |   execute "silent %!xxd -c 32"
    \ |   set filetype=xxd
    \ |   redraw
    \ | endif
  autocmd BufWritePre *.bin
    \ if &binary
    \ |   let s:view = winsaveview()
    \ |   execute "silent %!xxd -r -c 32"
    \ | endif
  autocmd BufWritePost *.bin
    \ if &binary
    \ |   execute "silent %!xxd -c 32"
    \ |   set nomodified
    \ |   call winrestview(s:view)
    \ |   redraw
    \ | endif
augroup END

```

Using <> notation in autocommands

autocmd-<>

The <> notation is not recognized in the argument of an :autocmd. To avoid having to use special characters, you could use a self-destroying mapping to get the <> notation and then call the mapping from the autocmd. Example:

map-self-destroy

```

" This is for automatically adding the name of the file to the menu list.
" It uses a self-destroying mapping!
" 1. use a line in the buffer to convert the 'dots' in the file name to \.
" 2. store that in register 'i'
" 3. add that name to the Buffers menu list
" WARNING: this does have some side effects, like overwriting the
" current register contents and removing any mapping for the "i" command.
"

```

```

autocmd BufNewFile,BufReadPre * nmap i :nunmap i<CR>0<C-R>%<Esc>:.g/\./s/\./\./g<CR>0"9y
autocmd BufNewFile,BufReadPre * normal i

```

Another method, perhaps better, is to use the ":execute" command. In the string you can use the <> notation by preceding it with a backslash. Don't forget to double the number of existing backslashes and put a backslash before 'i'.

```

autocmd BufNewFile,BufReadPre * exe "normal 0<C-R>%<Esc>:.g/\./s/\./\./g<CR>0\"9y

```

For a real buffer menu, user functions should be used (see :function), but then the <> notation isn't used, which defeats using it as an example here.

Highlighting matching parens

match-parens

This example shows the use of a few advanced tricks:

- using the `CursorMoved` autocommand event
- using `searchpairpos()` to find a matching paren
- using `synID()` to detect whether the cursor is in a string or comment
- using `:match` to highlight something
- using a `pattern` to match a specific position in the file.

This should be put in a Vim script file, since it uses script-local variables. It skips matches in strings or comments, unless the cursor started in string or comment. This requires syntax highlighting.

A slightly more advanced version is used in the `matchparen` plugin.

```
let s:paren_hl_on = 0
function s:Highlight_Matching_Paren()
  if s:paren_hl_on
    match none
    let s:paren_hl_on = 0
  endif

  let c_lnum = line('.')
  let c_col = col('.')

  let c = getline(c_lnum)[c_col - 1]
  let plist = split(&matchpairs, ':\|,')
  let i = index(plist, c)
  if i < 0
    return
  endif
  if i % 2 == 0
    let s_flags = 'nW'
    let c2 = plist[i + 1]
  else
    let s_flags = 'nbW'
    let c2 = c
    let c = plist[i - 1]
  endif
  if c == '['
    let c = '\['
    let c2 = '\]'
  endif
  let s_skip = 'synIDattr(synID(line("."), col("."), 0), "name") ' ..
    \ '=~? "string\|comment"'
  execute 'if' s_skip '| let s_skip = 0 | endif'

  let [m_lnum, m_col] = searchpairpos(c, '', c2, s_flags, s_skip)

  if m_lnum > 0 && m_lnum >= line('w0') && m_lnum <= line('w$')
    exe 'match Search /\(\\%' .. c_lnum .. '\\%' .. c_col ..
      \ 'c\)\|(\\%' .. m_lnum .. '\\%' .. m_col .. 'c\)/'
    let s:paren_hl_on = 1
  endif
endfunction
```

```
autocmd CursorMoved,CursorMovedI * call s:Highlight_Matching_Paren()
autocmd InsertEnter * match none
```

Opening help in the current window

`help-curwin`

By default, help is displayed in a split window. If you prefer it opens in the current window, try this custom `:HelpCurwin` command:

```
command -bar -nargs=? -complete=help HelpCurwin execute s:HelpCurwin(<q-args>)
let s:did_open_help = v:false

function s:HelpCurwin(subject) abort
  let mods = 'silent noautocmd keepalt'
  if !s:did_open_help
    execute mods .. ' help'
    execute mods .. ' helpclose'
    let s:did_open_help = v:true
  endif
  if !getcompletion(a:subject, 'help')->empty()
    execute mods .. ' edit ' .. &helpfile
    set buftype=help
  endif
  return 'help ' .. a:subject
endfunction
```


This file contains an alphabetical list of messages and error messages that Vim produces. You can use this if you don't understand what the message means. It is not complete though.

1. Old messages `:messages`
2. Error messages `error-messages`
3. Messages `messages`

-
1. Old messages `:messages` `:mes` `message-history`

The `":messages"` command can be used to view previously given messages. This is especially useful when messages have been overwritten or truncated. This depends on the `'shortmess'` option.

- | | |
|---------------------------------------|---|
| <code>:mes[sages]</code> | Show all messages. |
| <code>:{count}mes[sages]</code> | Show the <code>{count}</code> most recent messages. |
| <code>:mes[sages] clear</code> | Clear all messages. |
| <code>:{count}mes[sages] clear</code> | Clear messages, keeping only the <code>{count}</code> most recent ones. |

The number of remembered messages is fixed at 200.

The `"g<"` command can be used to see the last page of previous command output. This is especially useful if you accidentally typed `<Space>` at the hit-enter prompt. You are then back at the hit-enter prompt and can then scroll further back.

Note: If the output has been stopped with `"q"` at the more prompt, it will only be displayed up to this point.

The previous command output is cleared when another command produces output. The `"g<"` output is not redirected.

If you are using translated messages, the first printed line tells who maintains the messages or the translations. You can use this to contact the maintainer when you spot a mistake.

If you want to find help on a specific (error) message, use the ID at the start of the message. For example, to get help on the message:

`E72: Close error on swap file`

or (translated):

E72: Errore durante chiusura swap file

Use:

`:help E72`

If you are lazy, it also works without the shift key:

`:help e72`

The number in this ID has no meaning.

=====
2. Error messages error-messages errors

When an error message is displayed, but it is removed before you could read it, you can see it again with:

`:echo errmsg`

Or view a list of recent messages with:

`:messages`

See `:messages`` above.

LIST OF MESSAGES

E222	E228	E232	E292	E293	E298	E304	E316
E317	E318	E320	E322	E323	E341	E356	E438
E439	E440	E473	E570				

Add to read buffer

makemap: Illegal mode

Cannot create BalloonEval with both message and callback

block was not locked

Didn't get block nr {N}?

ml_upd_block0(): Didn't get block 0??

pointer block id wrong {N}

Updated too many blocks?

get_varp ERROR

u_undo: line numbers wrong

undo list corrupt

undo line missing

ml_get: cannot find line {N} in buffer {nr} {name}

line number out of range: {N} past the end

line count wrong in block {N}

Internal error: lalloc(0,)

Internal error: {function}

Internal error in regexp

fatal error in cs_manage_matches

Invalid count for del_bytes(): {N}

E340 E685 internal-error

This is an internal error. If you can reproduce it, please send in a bug report, see [bugs](#) .

ATTENTION

Found a swap file by the name ...

See [ATTENTION](#) .

E92

Buffer {N} not found

The buffer you requested does not exist. This can also happen when you have wiped out a buffer which contains a mark or is referenced in another way.

`:bwipeout`

E95

Buffer with this name already exists

You cannot have two buffers with exactly the same name. This includes the path leading to the file.

E1513

Cannot switch buffer. 'winfixbuf' is enabled

If a window has `'winfixbuf'` enabled, you cannot change that window's current buffer. You need to set `'nowinfixbuf'` before continuing. You may use `[!]` to force the window to switch buffers, if your command supports it.

E72

Close error on swap file

The `swap-file` , that is used to keep a copy of the edited text, could not be closed properly. Mostly harmless.

E169

Command too recursive

This happens when an Ex command executes an Ex command that executes an Ex command, etc. The limit is 200 or the value of `'maxfuncdepth'`, whatever is larger. When it's more there probably is an endless loop. Probably a `:execute` or `:source` command is involved.

E254

Cannot allocate color {name}

The color name `{name}` is unknown. See [gui-colors](#) for a list of colors that are available on most systems.

E1244

Bad color string: {str}

The provided color did not conform to the pattern `#rrggbb`

E458

Cannot allocate colormap entry, some colors may be incorrect

This means that there are not enough colors available for Vim. It will still run, but some of the colors will not appear in the specified color. Try stopping other applications that use many colors, or start them after starting

gvim.

Browsers are known to consume a lot of colors. You can avoid this with netscape by telling it to use its own colormap:

```
netscape -install
```

Or tell it to limit to a certain number of colors (64 should work well):

```
netscape -ncols 64
```

This can also be done with a line in your Xdefaults file:

```
Netscape*installColormap: Yes
```

or

```
Netscape*maxImageColors: 64
```

E79

Cannot expand wildcards

A filename contains a strange combination of characters, which causes Vim to attempt expanding wildcards but this fails. This does NOT mean that no matching file names could be found, but that the pattern was illegal.

E459

Cannot go back to previous directory

While expanding a file name, Vim failed to go back to the previously used directory. All file names being used may be invalid now! You need to have execute permission on the current directory.

E190 E212

Cannot open "{filename}" for writing Can't open file for writing

For some reason the file you are writing to cannot be created or overwritten. The reason could be that you do not have permission to write in the directory or the file name is not valid.

E166

Can't open linked file for writing

You are trying to write to a file which can't be overwritten, and the file is a link (either a hard link or a symbolic link). Writing might still be possible if the directory that contains the link or the file is writable, but Vim now doesn't know if you want to delete the link and write the file in its place, or if you want to delete the file itself and write the new file in its place. If you really want to write the file under this name, you have to manually delete the link or the file, or change the permissions so that Vim can overwrite.

E46

Cannot change read-only variable "{name}"

You are trying to assign a value to an argument of a function `a:var` or a Vim internal variable `v:var` which is read-only.

E90

Cannot unload last buffer

Vim always requires one buffer to be loaded, otherwise there would be nothing to display in the window.

E40

Can't open errorfile <filename>

When using the ":make" or ":grep" commands: The file used to save the error messages or grep output cannot be opened. This can have several causes:

- **'shellredir'** has a wrong value.
- The shell changes directory, causing the error file to be written in another directory. This could be fixed by changing **'makeef'**, but then the make command is still executed in the wrong directory.
- **'makeef'** has a wrong value.
- The **'grepprg'** or **'makeprg'** could not be executed. This cannot always be detected (especially on MS-Windows). Check your \$PATH.

Can't open file C:\TEMP\VIoD243.TMP

On MS-Windows, this message appears when the output of an external command was to be read, but the command didn't run successfully. This can be caused by many things. Check the **'shell'**, **'shellquote'**, **'shellxquote'**, **'shellslash'** and related options. It might also be that the external command was not found, there is no different error message for that.

E12

Command not allowed from exrc/vimrc in current dir or tag search

Some commands are not allowed for security reasons. These commands mostly come from a .exrc or .vimrc file in the current directory, or from a tags file. Also see **'secure'**.

E74

Command too complex

A mapping resulted in a very long command string. Could be caused by a mapping that indirectly calls itself.

CONVERSION ERROR

When writing a file and the text "CONVERSION ERROR" appears, this means that some bits were lost when converting text from the internally used UTF-8 to the format of the file. The file will not be marked unmodified. If you care about the loss of information, set the **'fileencoding'** option to another value that can handle the characters in the buffer and write again. If you don't care, you can abandon the buffer or reset the **'modified'** option. If there is a backup file, when **'writebackup'** or **'backup'** is set, it will not be deleted, so you can move it back into place if you want to discard the changes.

E302

Could not rename swap file

When the file name changes, Vim tries to rename the `swap-file` as well. This failed and the old swap file is now still used. Mostly harmless.

E43 E44

Damaged match string
Corrupted regexp program

Something inside Vim went wrong and resulted in a corrupted regexp. If you know how to reproduce this problem, please report it. [bugs](#)

E208 E209 E210

Error writing to "{filename}"
Error closing "{filename}"
Error reading "{filename}"

This occurs when Vim is trying to rename a file, but a simple change of file name doesn't work. Then the file will be copied, but somehow this failed. The result may be that both the original file and the destination file exist and the destination file may be incomplete.

Vim: Error reading input, exiting...

This occurs when Vim cannot read typed characters while input is required. Vim got stuck, the only thing it can do is exit. This can happen when both stdin and stderr are redirected and executing a script that doesn't exit Vim.

E47

Error while reading errorfile

Reading the error file was not possible. This is NOT caused by an error message that was not recognized.

E80

Error while writing

Writing a file was not completed successfully. The file is probably incomplete.

E13 E189

File exists (add ! to override)
"{filename}" exists (add ! to override)

You are protected from accidentally overwriting a file. When you want to write anyway, use the same command, but add a "!" just after the command. Example:

```
:w /tmp/test  
changes to:  
:w! /tmp/test
```

E768

Swap file exists: {filename} (:silent! overrides)

You are protected from overwriting a file that is being edited by Vim. This

happens when you use `":w! filename"` and a swapfile is found.

- If the swapfile was left over from an old crashed edit session you may want to delete the swapfile. Edit `{filename}` to find out information about the swapfile.

- If you want to write anyway prepend `":silent!"` to the command. For example:
`:silent! w! /tmp/test`

The special command is needed, since you already added the `!` for overwriting an existing file.

E139

File is loaded in another buffer

You are trying to write a file under a name which is also used in another buffer. This would result in two versions of the same file.

E142

File not written: Writing is disabled by 'write' option

The `'write'` option is off. This makes all commands that try to write a file generate this message. This could be caused by a `-m` commandline argument. You can switch the `'write'` option on with `":set write"`.

E25

GUI cannot be used: Not enabled at compile time

You are running a version of Vim that doesn't include the GUI code. Therefore `"gvim"` and `":gui"` don't work.

E49

Invalid scroll size

This is caused by setting an invalid value for the `'scroll'`, `'scrolljump'` or `'scrolloff'` options.

E17

"{filename}" is a directory

You tried to write a file with the name of a directory. This is not possible. You probably need to append a file name.

E19

Mark has invalid line number

You are using a mark that has a line number that doesn't exist. This can happen when you have a mark in another file, and some other program has deleted lines from it.

E219 E220

Missing {.
Missing }.

Using a `{}` construct in a file name, but there is a `{ without a matching }` or the other way around. It should be used like this: `{foo,bar}`. This matches `"foo"` and `"bar"`.

E315

`ml_get: invalid lnum: {number}`

This is an internal Vim error. Please try to find out how it can be reproduced, and submit a bug report [bugreport.vim](#) .

E173

`{number} more files to edit`

You are trying to exit, while the last item in the argument list has not been edited. This protects you from accidentally exiting when you still have more files to work on. See [argument-list](#) . If you do want to exit, just do it again and it will work.

E23 E194

`No alternate file`
`No alternate file name to substitute for '#'`

The alternate file is not defined yet. See [alternate-file](#) .

E32

`No file name`

The current buffer has no name. To write it, use `":w fname"`. Or give the buffer a name with `":file fname"`.

E141

`No file name for buffer {number}`

One of the buffers that was changed does not have a file name. Therefore it cannot be written. You need to give the buffer a file name:

```
:buffer {number}  
:file {filename}
```

E33

`No previous substitute regular expression`

When using the '~' character in a pattern, it is replaced with the previously used pattern in a `":substitute"` command. This fails when no such command has been used yet. See [/~](#) . This also happens when using `":s/pat/%/"`, where the "%" stands for the previous substitute string.

E35

`No previous regular expression`

When using an empty search pattern, the previous search pattern is used. But that is not possible if there was no previous search.

E24

`No such abbreviation`

You have used an `":unabbreviate"` command with an argument which is not an existing abbreviation. All variations of this command give the same message:

":cunabbrev", ":iunabbrev", etc. Check for trailing white space.

/dev/dsp: No such file or directory

Only given for GTK GUI with Gnome support. Gnome tries to use the audio device and it isn't present. You can ignore this error.

E31

No such mapping

You have used an ":unmap" command with an argument which is not an existing mapping. All variations of this command give the same message: ":cunmap", ":unmap!", etc. A few hints:

- Check for trailing white space.
- If the mapping is buffer-local you need to use ":unmap <buffer>".
:map-<buffer>

E37 E89

No write since last change (add ! to override)

No write since last change for buffer {N} (add ! to override)

You are trying to `abandon` a file that has changes. Vim protects you from losing your work. You can either write the changed file with ":w", or, if you are sure, `abandon` it anyway, and lose all the changes. This can be done by adding a '!' character just after the command you used. Example:

```
:e other_file
```

changes to:

```
:e! other_file
```

E162

No write since last change for buffer "{name}"

This appears when you try to exit Vim while some buffers are changed. You will either have to write the changed buffer (with `:w`), or use a command to abandon the buffer forcefully, e.g., with ":qa!". Careful, make sure you don't throw away changes you really want to keep. You might have forgotten about a buffer, especially when `'hidden'` is set.

[No write since last change]

This appears when executing a shell command while at least one buffer was changed. To avoid the message reset the `'warn'` option.

E38

Null argument

Something inside Vim went wrong and resulted in a NULL pointer. If you know how to reproduce this problem, please report it. [bugs](#)

E41 E82 E83 E342

Out of memory!

Out of memory! (allocating {number} bytes)

```
Cannot allocate any buffer, exiting...
Cannot allocate buffer, using other one...
```

Oh, oh. You must have been doing something complicated, or some other program is consuming your memory. Be careful! Vim is not completely prepared for an out-of-memory situation. First make sure that any changes are saved. Then try to solve the memory shortage. To stay on the safe side, exit Vim and start again.

If this happens while Vim is still initializing, editing files is very unlikely to work, therefore Vim will exit with value 123.

Buffers are only partly kept in memory, thus editing a very large file is unlikely to cause an out-of-memory situation. Undo information is completely in memory, you can reduce that with these options:

- **'undolevels'** Set to a low value, or to -1 to disable undo completely. This helps for a change that affects all lines.
- **'undoreload'** Set to zero to disable.

E339

Pattern too long

This happens on systems with 16 bit ints: The compiled regexp pattern is longer than about 65000 characters. Try using a shorter pattern. It also happens when the offset of a rule doesn't fit in the space available. Try simplifying the pattern.

E45

'readonly' option is set (add ! to override)

You are trying to write a file that was marked as read-only. To write the file anyway, either reset the **'readonly'** option, or add a **'!'** character just after the command you used. Example:

```
:w
changes to:
:w!
```

E294 E295 E301

```
Read error in swap file
Seek error in swap file read
Oops, lost the swap file!!!
```

Vim tried to read text from the **swap-file**, but something went wrong. The text in the related buffer may now be corrupted! Check carefully before you write a buffer. You may want to write it in another file and check for differences.

E192

Recursive use of **:normal** too deep

You are using a **":normal"** command, whose argument again uses a **":normal"** command in a recursive way. This is restricted to **'maxmapdepth'** levels. This example illustrates how to get this message:

```
:map gq :normal gq<CR>
```

If you type "gq", it will execute this mapping, which will call "gq" again.

E22

Scripts nested too deep

Scripts can be read with the "-s" command-line argument and with the `:source!` command. The script can then again read another script. This can continue for about 14 levels. When more nesting is done, Vim assumes that there is a recursive loop and stops with this error message.

E319

Sorry, the command is not available in this version

You have used a command that is not present in the version of Vim you are using. When compiling Vim, many different features can be enabled or disabled. This depends on how big Vim has chosen to be and the operating system. See `+feature-list` for when which feature is available. The `:version` command shows which feature Vim was compiled with.

E300

Swap file already exists (symlink attack?)

This message appears when Vim is trying to open a swap file and finds it already exists or finds a symbolic link in its place. This shouldn't happen, because Vim already checked that the file doesn't exist. Either someone else opened the same file at exactly the same moment (very unlikely) or someone is attempting a symlink attack (could happen when editing a file in /tmp or when `'directory'` starts with "/tmp", which is a bad choice).

E432

Tags file not sorted: {file name}

Vim (and Vi) expect tags files to be sorted in ASCII order. Binary searching can then be used, which is a lot faster than a linear search. If your tags files are not properly sorted, reset the `'tagbsearch'` option. This message is only given when Vim detects a problem when searching for a tag. Sometimes this message is not given, even though the tags file is not properly sorted.

E424

Too many different highlighting attributes in use

Vim can only handle about 223 different kinds of highlighting. If you run into this limit, you have used too many `:highlight` commands with different arguments. A `":highlight link"` is not counted.

E77

Too many file names

When expanding file names, more than one match was found. Only one match is allowed for the command that was used.

E303

Unable to open swap file for "{filename}", recovery impossible

Vim was not able to create a swap file. You can still edit the file, but if Vim unexpectedly exits the changes will be lost. And Vim may consume a lot of memory when editing a big file. You may want to change the **'directory'** option to avoid this error. This error is not given when **'directory'** is empty. See [swap-file](#) .

E140

Use **!** to write partial buffer

When using a range to write part of a buffer, it is unusual to overwrite the original file. It is probably a mistake (e.g., when Visual mode was active when using `":w"`), therefore Vim requires using a **!** after the command, e.g.: `":3,10w!"`.

Warning: Cannot convert string "<Key>Escape,_Key_Cancel" to type VirtualBinding

Messages like this appear when starting up. This is not a Vim problem, your X11 configuration is wrong. You can find a hint on how to solve this here: <http://groups.yahoo.com/group/solarisonintel/message/12179>.
[this URL is no longer valid]

W10

Warning: Changing a readonly file

The file is read-only and you are making a change to it anyway. You can use the [FileChangedRO](#) autocommand event to avoid this message (the autocommand must reset the **'readonly'** option). See **'modifiable'** to completely disallow making changes to a file. This message is only given for the first change after **'readonly'** has been set.

W13

Warning: File "{filename}" has been created after editing started

You are editing a file in Vim when it didn't exist, but it does exist now. You will have to decide if you want to keep the version in Vim or the newly created file. This message is not given when **'buftype'** is not empty.

W11

Warning: File "{filename}" has changed since editing started

The file which you have started editing has got another timestamp and the contents changed (more precisely: When reading the file again with the current option settings and autocommands you would end up with different text). This probably means that some other program changed the file. You will have to find out what happened, and decide which version of the file you want to keep. Set the **'autoread'** option if you want to do this automatically. This message is not given when **'buftype'** is not empty. Also see the [FileChangedShell](#) autocommand.

There is one situation where you get this message even though there is nothing wrong: If you save a file in Windows on the day the daylight saving time

starts. It can be fixed in one of these ways:

- Add this line in your autoexec.bat:

```
SET TZ=-1
```

Adjust the "-1" for your time zone.

- Disable "automatically adjust clock for daylight saving changes".
- Just write the file again the next day. Or set your clock to the next day, write the file twice and set the clock back.

If you get W11 all the time, you may need to disable "Acronis Active Protection" or register Vim as a trusted service/application.

W12

Warning: File "{filename}" has changed and the buffer was changed in Vim as well

Like the above, and the buffer for the file was changed in this Vim as well. You will have to decide if you want to keep the version in this Vim or the one on disk. This message is not given when 'buftype' is not empty.

W16

Warning: Mode of file "{filename}" has changed since editing started

When the timestamp for a buffer was changed and the contents are still the same but the mode (permissions) have changed. This usually occurs when checking out a file from a version control system, which causes the read-only bit to be reset. It should be safe to reload the file. Set 'autoread' to automatically reload the file.

E211

File "{filename}" no longer available

The file which you have started editing has disappeared, or is no longer accessible. Make sure you write the buffer somewhere to avoid losing changes. This message is not given when 'buftype' is not empty.

W14

Warning: List of file names overflow

You must be using an awful lot of buffers. It's now possible that two buffers have the same number, which causes various problems. You might want to exit Vim and restart it.

E931

Buffer cannot be registered

Out of memory or a duplicate buffer number. May happen after W14. Looking up a buffer will not always work, better restart Vim.

E296 E297

Seek error in swap file write
Write error in swap file

This mostly happens when the disk is full. Vim could not write text into the `swap-file`. It's not directly harmful, but when Vim unexpectedly exits some text may be lost without recovery being possible. Vim might run out of memory

when this problem persists.

```
                                connection-refused
Xlib: connection to "<machine-name:0.0" refused by server
```

This happens when Vim tries to connect to the X server, but the X server does not allow a connection. The connection to the X server is needed to be able to restore the title and for the xterm clipboard support. Unfortunately this error message cannot be avoided, except by disabling the `+xterm_clipboard` and `+X11` features.

```
                                E10
\\ should be followed by /, ? or &
```

A command line started with a backslash or the range of a command contained a backslash in a wrong place. This is often caused by command-line continuation being disabled. Remove the 'C' flag from the '`cptions`' option to enable it. Or use `":set nocp"`.

```
                                E471
Argument required
```

This happens when an Ex command with mandatory argument(s) was executed, but no argument has been specified.

```
                                E474  E475  E983
Invalid argument
Invalid argument: {arg}
Duplicate argument: {arg}
```

An Ex command or function has been executed, but an invalid argument has been specified.

```
                                E488
Trailing characters
Trailing characters: {text}
```

An argument has been added to an Ex command that does not permit one. Or the argument has invalid characters and has not been recognized.

```
                                E477  E478
No ! allowed
Don't panic!
```

You have added a "!" after an Ex command that doesn't permit one.

```
                                E481
No range allowed
```

A range was specified for an Ex command that doesn't permit one. See [cmdline-ranges](#) .

```
                                E482  E483
Can't create file {filename}
```

Can't get temp file name

Vim cannot create a temporary file.

E484 E485

Can't open file {filename}
Can't read file {filename}

Vim cannot read a temporary file. Especially on Windows, this can be caused by wrong escaping of special characters for cmd.exe; the approach was changed with patch 7.3.443. Try using `shellescape()` for all shell arguments given to `system()`, or explicitly add escaping with `^`. Also see `'shellxquote'` and `'shellxescape'`.

E464

Ambiguous use of user-defined command

There are two user-defined commands with a common name prefix, and you used Command-line completion to execute one of them. `user-cmd-ambiguous`

Example:

```
:command MyCommand1 echo "one"  
:command MyCommand2 echo "two"  
:MyCommand
```

E492

Not an editor command

You tried to execute a command that is neither an Ex command nor a user-defined command.

E943

Command table needs to be updated, run 'make cmdidxs'

This can only happen when changing the source code, when adding a command in `src/ex_cmds.h`. The lookup table then needs to be updated, by running:

```
make cmdidxs
```

E928 E889

E928: String required
E889: Number required

These happen when a value or expression is used that does not have the expected type.

=====
3. Messages messages

This is an (incomplete) overview of various messages that Vim gives:

```
hit-enter  press-enter  hit-return  
press-return  hit-enter-prompt
```

Press ENTER or type command to continue

This message is given when there is something on the screen for you to read, and the screen is about to be redrawn:

- After executing an external command (e.g., "!!ls" and "=").
 - Something is displayed on the status line that is longer than the width of the window, or runs into the **'showcmd'** or **'ruler'** output.
- > Press **<Enter>** or **<Space>** to redraw the screen and continue, without that key being used otherwise.
- > Press **':'** or any other Normal mode command character to start that command. **Note** that after an external command some special keys, such as the cursor keys, may not work normally, because the terminal is still set to a state for executing the external command.
- > Press **'k'**, **<Up>**, **'u'**, **'b'** or **'g'** to scroll back in the messages. This works the same way as at the **more-prompt**. Only works when **'compatible'** is off and **'more'** is on.
- > Pressing **'j'**, **'f'**, **'d'** or **<Down>** is ignored when messages scrolled off the top of the screen, **'compatible'** is off and **'more'** is on, to avoid that typing one **'j'** or **'f'** too many causes the messages to disappear.
- > Press **<C-Y>** to copy (yank) a modeless selection to the clipboard register.
- > Use a menu. The characters defined for Cmdline-mode are used.
- > When **'mouse'** contains the **'r'** flag, clicking the left mouse button works like pressing **<Space>**. This makes it impossible to select text though.
- > For the GUI clicking the left mouse button in the last line works like pressing **<Space>**.

If you accidentally hit **<Enter>** or **<Space>** and you want to see the displayed text then use **g<**. This only works when **'more'** is set.

To reduce the number of hit-enter prompts:

- Set **'cmdheight'** to 2 or higher.
- Add flags to **'shortmess'**.
- Reset **'showcmd'** and/or **'ruler'**.
- Make sure **':echo'** text is shorter than or equal to **v:echospace** screen cells.

If your script causes the hit-enter prompt and you don't know why, you may find the **v:scrollstart** variable useful.

Also see **'mouse'**. The hit-enter message is highlighted with the **hl-Question** group.

```
more-prompt pager
-- More --
-- More -- SPACE/d/j: screen/page/line down, b/u/k: up, q: quit
```

This message is given when the screen is filled with messages. It is only given when the **'more'** option is on. It is highlighted with the **hl-MoreMsg** group.

Type	effect
<CR> or <NL> or j or <Down>	one more line
d	down a page (half a screen)
<Space> or f or <PageDown>	down a screen

G	down all the way, until the hit-enter prompt
<BS> or k or <Up>	one line back
u	up a page (half a screen)
b or <PageUp>	back a screen
g	back to the start
q, <Esc> or CTRL-C	stop the listing
:	stop the listing and enter a command-line
<C-Y>	yank (copy) a modeless selection to the clipboard ("* and "+" registers)
{menu-entry}	what the menu is defined to in Cmdline-mode.
<LeftMouse>	next page (*)

Any other key causes the meaning of the keys to be displayed.

- (*) Clicking the left mouse button only works:
 - For the GUI: in the last line of the screen.
 - When 'r' is included in '**mouse**' (but then selecting text won't work).

Note: The typed key is directly obtained from the terminal, it is not mapped and typeahead is ignored.

The `g<` command can be used to see the last page of previous command output. This is especially useful if you accidentally typed `<Space>` at the hit-enter prompt.

VIM REFERENCE MANUAL by Bram Moolenaar

quotes

Here are some nice quotes about Vim that I collected from news and mail.

vim (vim) noun - Ebullient vitality and energy. [Latin, accusative of vis, strength] (Dictionary)

Vim is so much better than vi that a great many of my old vi :map's became immediately obsolete! (Tony Nugent, Australia)

Coming with a very GUI mindset from Windows, I always thought of people using Vi as some kind of outer space alien in human clothes. Once I tried I really got addicted by its power and now I found myself typing Vim keypresses in the oddest places! That's why I would like to see Vim embedded in every application which deals with text editing. (José Fonseca)

I was a 12-year emacs user who switched to Vim about a year ago after finally giving up on the multiple incompatible versions, flaky contributed packages, disorganized keystrokes, etc. And it was one of the best moves I ever made. (Joel Burton)

Although all of the programs were used during the preparation of the new and revised material, most of the editing was done with Vim versions 4.5 and 5.0 under GNU-Linux (Redhat 4.2). (Arnold Robbins, Israel, author of "Learning the Vi editor")

Out of all the open software i've ever seen and used, and i've seen a lot, Vim is the best, most useful and highest quality to work with, second only to the linux kernel itself. (Peter Jay Salzman)

It's well worth noting that the entirety of SourceForge was written using Vim and its nifty PHP syntax highlighting. I think the entire SF.net tech staff uses Vim and we're all excited to have you aboard! (Tim Perdue)

Vim is one of a select bunch of tools for which I have no substitute. It is a brilliant piece of work! (Biju Chacko)

A previous girlfriend of mine switched to emacs. Needless to say, the relationship went nowhere. (Geoffrey Mann)

I rarely think about Vim, in the same way that I guess a fish rarely thinks about water. It's the environment in which everything else happens. I'm a fairly busy system administrator working on a lot of different platforms. Vim is the only thing that's consistent across all my systems, and it's just about the only thing that doesn't break from time to time. When a new system comes in the door without Vim, I install it right away. Great to have a tool that's the same everywhere, that's completely reliable, so I can ignore it and think about other things. (Pete Schaeffer)

Having recently succeeded in running Vim via telnet through a Nokia Communicator, I can now report that it works nicely on a Palm Pilot too. (Allan Kelly, Scotland)

You've done a tremendous job with 'VIM', Bram! The more I use it, the more impressed I get (I am an old 'vi' die hard who once started out with early versions of 'emacs' in the late 1970's and was relieved by finding 'vi' in the first UNIX I came across in 1983). In my opinion, it's about time 'VIM' replace 'emacs' as the standard for top editors. (Bo Thide', Sweden)

I love and use Vim heavily too. (Larry Wall)

Vi is like a Ferrari, if you're a beginner, it handles like a bitch, but once you get the hang of it, it's small, powerful and FAST! (Unknown)
Vim is like a new model Ferrari, and sounds like one too - "VIIIIIIIMMM!" (Stephen Riehm, Germany)

Schon bei Nutzung eines Bruchteils der Vim-Funktionen wird der Benutzer recht schnell die Vorzuege dieses Editors kennen- und schaeetzenlernen.
Translated: Even when only using a fraction of Vim-functions, the user will quickly get used to and appreciate the advantages of this editor. (Garry Glendown, conclusion of an article on Vim in iX magazine 9/1998)

I've recently acquired the O'Reilly book on Vi (it also discusses Vim in-depth), and I'm amazed at just how powerful this application is. (Jeffrey Rankin)

This guide was written using the Windows 9.x distribution of gvim, which is quite possibly the greatest thing to come along since God created the naked girl. (Michael DiBernardo)

Boy, I thought I knew almost everything about Vim, but every time I browse the online documentation, I hit upon a minor but cool aspect of a Vim feature that I didn't know before! I must say the documentation is one the finest I've ever seen in a product -- even better than most commercial products. (Gautam Mudunuri)

Vim 4.5 is really a fantastic editor. It has sooooo many features and more importantly, the defaults are so well thought out that you really don't have to change anything!! Words cannot express my amazement and gratitude to the creators of Vim. Keep it up. (Vikas, USA)

I wonder how long it will be before people will refer to other Vi editors as Vim clones? (Darren Hiebert)

I read about [\[auto-positioning-in-file-based-on-the-errors-from-make\]](#) in one of those "Perfect Programmer's Editor" threads and was delighted to discover that Vim already supports it. (Brendan Macmillan, Australia)

I just discovered Vim (5.0) and I'm telling everyone I know about it! I tell them Vim stands for Vi for the new (M)illenium. Thanks so much! (Matt F. Valentine)

I think from now on "vi" should be called "Vim Imitation", not the other way around. (Rungun Ramanathan)

The Law of Vim:

For each member b of the possible behaviour space B of program P , there exists a finite time t before which at least one user u in the total user space U of program P will request b becomes a member of the allowed behaviour space B' ($B' \leq B$).

In other words: Sooner or later everyone wants everything as an option. (Negri)

Whenever I move to a new computing platform, the first thing I do is to port Vim. Lately, I am simply stunned by its ease of compilation using the configure facility. (A.M. Sabuncu, Turkey)

The options are really excellent and very powerful. (Anish Maharaj)

The Spring user-interface designs are in, and word from the boutiques is that 80x24 text-only mode is back with a *vengeance! Vi editor clone Vim burst onto March desk-tops with a dazzling show of pastel syntax highlights for its 5.0 look. Strident and customizable, Vim raises eyebrows with its interpretation of the classic Vi single-key macro collection.

<http://www.ntk.net/index.cgi?back=archive98/now0327.txt&line=179#l>

I just wanted to take this opportunity to let you know that Vim 5 ROCKS! Syntax highlighting: how did I survive without it?! Thank you for creating mankind's best editor! (Mun Johl, USA)

Thanks again for Vim. I use it every day on Linux. (Eric Foster-Johnson, author of the book "UNIX Programming Tools")

The BEST EDITOR EVER (Stuart Woolford)

I have used most of Vim's fancy features at least once, many frequently, and I can honestly say that I couldn't live with anything less anymore. My productivity has easily doubled compared to what it was when I used vi. (Sitaram Chamarty)

I luv Vim. It is incredible. I'm naming my first-born Vimberly. (Jose Unpingco, USA)

Hint: "Vim" is "vi improved" - much better! (Sven Guckes, Germany)

I use Vim every day. I spend more time in Vim than in any other program... It's the best vi clone there is. I think it's great. (Craig Sanders, Australia)

I strongly advise using Vim--its infinite undo/redo saved me much grief. (Terry Brown)

Thanks very much for writing what in my opinion is the finest text editor on the planet. If I were to get another cat, I would name it "Vim". (Bob Sheehan, USA)

I typed :set all and the screen FILLED up with options. A whole screen of things to be set and unset. I saw some of my old friends like wrapmargin, modelines and showmode, but the screen was FILLED with new friends! I love them all! I love Vim! I'm so happy that I've found this editor! I feel like how I once felt when I started using vi after a couple of years of using ed. I never thought I'd forsake my beloved ed, but vi ... oh god, vi was great. And now, Vim. (Peter Jay Salzman, USA)

I am really happy with such a wonderful software package. Much better than almost any expensive, off the shelf program. (Jeff Walker)

Whenever I reread the Vim documentation I'm overcome with excitement at the power of the editor. (William Edward Webber, Australia)

Hurrah for Vim!! It is "at your fingertips" like vi, and has the extensions that vi sorely needs: highlighting for executing commands on blocks, an easily navigable and digestible help screen, and more. (Paul Pax)

The reason WHY I don't have this amazingly useful macro anymore, is that I now use Vim - and this is built in!! (Stephen Riehm, Germany)

I am a user of Vim and I love it. I use it to do all my programming, C, C++, HTML what ever. (Tim Allwine)

I discovered Vim after years of struggling with the original vi, and I just can't live without it anymore. (Emmanuel Mogenet, USA)

Emacs has not a bit of chance to survive so long as Vim is around. Besides, it also has the most detailed software documentation I have ever seen---much better than most commercial software! (Leiming Qian)

This version of Vim will just blow people apart when they discover just how fantastic it is! (Tony Nugent, Australia)

I took your advice & finally got Vim & I'm really impressed. Instant convert. (Patrick Killelea, USA)

Vim is by far my favorite piece of shareware and I have been particularly pleased with version 3.0. This is really a solid piece of work. (Robert Colon, USA)

Vim is a joy to use, it is so well thought and practical that I wonder why anybody would use visual development tools. Vim is powerful and elegant, it looks deceptively simple but is almost as complex as a 747 (especially when I look at my growing .vimrc), keep up that wonderful job, Vim is a centerpiece of the free software world. (Louis-David Mitterand, USA)

I cannot believe how great it is to use Vim. I think the guys at work are getting tired of hearing me bragging about it. Others eyes are lighting up. (Rick Croote)

Emacs takes way too much time to start up and run, it is too big and bulky for effective use and the interface is more confusing than it is of any help. Vim however is short, it is fast, it is powerful, it has a good interface and it

VIM REFERENCE MANUAL by Bram Moolenaar

TODO list for Vim todo

This is a veeeery long list of known bugs, current work and desired improvements. To make it a little bit accessible, the older items are grouped by subject. In the first column of the line a classification is used to be able to look for "the next thing to do":

Priority classification:

- 9 next point release
- 8 next release
- 7 as soon as possible
- 6 soon
- 5 should be included
- 4 nice to have
- 3 consider including
- 2 maybe not
- 1 probably not
- unclassified

votes-for-changes

See [develop.txt](#) for development plans. You can vote for which items should be worked on, but only if you sponsor Vim development. See [sponsor](#).

Issues can also be entered online: <https://github.com/vim/vim/issues>
 Only use this for bug reports, not for questions! Those belong on the maillist. Updates will be forwarded to the [vim_dev](#) maillist. Issues entered there will not be repeated below, unless there is extra information.

The #1234 numbers refer to an issue or pull request on github. To see it in a browser use: <https://github.com/vim/vim/issues/1234>
 (replace 1234 with the issue/pull number)

known-bugs

----- Known bugs and current work -----

FILETYPE_FILE is defined to the same value in multiple places.
 Only use the one in feature.h. Others too.

Mapping with modifier is not recognized after a partial mapping. Probably because the typeahead was simplified when looking for a matching mapping. Need to somehow undo the simplification. #12002

Windows scroll when using the autocmd window. #12085
 in restore_snapshot_rec() restore more values from the snapshot, instead of calling frame_new_height() and frame_new_width(), especially w_topline and w_skipcol.

Check places that source "path/*.vim" to not match other extensions, e.g. .vim9, on MS-Windows (short file name match, gets expanded to long file name).

E.g. for startup files, plugins, packs, etc.

When a help item can't be found, then open **'helpfile'**. Search for the tag in that file and give E149 only when not found. Helps for a tiny Vim installed without all the help files.

SpellCap highlight not updated - PR #12428

Virtual text problems:

- If **'list'** is on, **'below'** virtual text which includes 1 or 2 characters are gone (Issue #12028)
- Virtual text aligned "above": Wrong indentation when using tabs (Issue #12232)
- Virtual text to the right of the line that isn't visible doesn't cause the **'extends'** character to show in **'list'** mode. #12478
- Virtual text to the right of the line that fits exactly on screen causes that line and all lines below it not to be displayed (Issue #12213)
- Window screen gets wrong when a virtual text is placed on **'above'** or **'below'** on an empty line (Issue #11959)
- truncated Virtual text below an empty line causes display error #12493

When **'virtualedit'** is "all" and **'cursorcolumn'** is set, the wrong column may be highlighted. (van-de-bugger, 2018 Jan 23, #2576)

Improve profiling by caching matching functions: PR #12192

With **'smoothscroll'** set and "lastline" in **'display'**, moving the cursor to a wrapping line that makes the display scroll up may scroll much more than needed, thus jump-scrolling. (part of issue 12411)

Add filecopy() ? #12346

Implement foreach() PR #12166

Errors when running tests with valgrind:

- test_codestyle.vim: e.g.:
command line..script /home/mool/vim/vim90/src/testdir/runtest.vim[569]..function RunThe
command line..script /home/mool/vim/vim90/src/testdir/runtest.vim[569]..function RunThe
- test_gui.vim:
Found errors in Test_gui_mouse_event():

When every block in if/elseif ends in "throw" or "return" code following after "endif" should give an "unreachable code" error.

Upcoming larger works:

- Make spell checking work with recent .dic/.aff files, e.g. French. #4916
Make Vim understand the format somehow? Search for "spell" below.
Make sure suggestions are speedy, also with composed words (German).
- Make syntax highlighting faster and better.
Add a generic mechanism to test a syntax plugin: An input file for the filetype and a screendump of expected result. With a way to specify the setup (global variables) and another dump file from that.
Discuss alternatives for using other grammars (treesitter, TextMate).
 - Possibly conversion to Vim syntax rules.

- Other mechanism than group and cluster to nest syntax items, to be used for grammars.
- Possibly keeping the parsed syntax tree and incremental updates.
- tree-sitter doesn't handle incorrect syntax (while typing) properly.
- NeoVim uses treesitter, what can we learn from that?
- Vscode is asked to switch to treesitter:
 - <https://github.com/microsoft/vscode/issues/50140>
- Vscode uses TextMate. #9087 - Other people don't like it.
 - <https://github.com/icedman/vim-textmate>
- sublime grammar? Hugo mentions it's a moving target #9087
- Make clear how it relates to LSP.
- example plugin: <https://github.com/uga-rosa/dps-vsctm.vim>

Further Vim9 improvements:

- Classes and Interfaces. See [vim9-classes](#)
 - Cannot use class type of itself in the method (Issue #12369)
 - Getting member of variable with "any" type should be handled at runtime. Remove temporary solution from #12096 / patch 9.0.1375.
 - "obj.Method()" does not always work in a compiled function, assumes "obj" is a dictionary. #12196 Issue #12024 might be the same problem. Issue #11822: any.Func() can be a dict or an object call, need to handle this at runtime. Also see #12198 for an example. Possibly issue #11981 can be fixed at the same time (has two examples).
 - Forward declaration of a class? E.g. for Clone() function.
 - Email lifepillar 2023 Mar 26
 - When "Meta" is a class, is "const MetaAlias = Meta" allowed? It should either work or given an error. Possibly give an error now and implement it later (using a typedef). #12006
 - how about lock/unlock?
 - For chaining, allow using the class name as type for function return value.
 - Implement "specifies" interface
 - Implement generics
 - Add "assignable" (class or child)?
 - More efficient way for interface member index than iterating over list?
 - a variant of type() that returns a different type for each class?
 - list<number> and list<string> should also differ.
- implement :enum
- Promise class, could be used to wait on a popup close callback?
- class local to a function
- Use Vim9 for more runtime files.
- Inline call to map() and filter(), better type checking.
- When evaluating constants for script variables, some functions could work:
 - has(featureName), len(someString)
- Implement as part of an expression: ++expr, --expr, expr++, expr--.
- The use of the literal value "null" and the type specific "null_xxx" values is confusing (#13458, #11770).

Information missing in terminfo:

- Codes used for focus gained and lost termcodes are hard-coded in set_termname(), not named.
- t_fe enable focus-event tracking
- t_fd disable focus-event tracking

- Accept "hyper" and "meta" modifiers (16 and 32) from Kitty like Meta? 8 is actually "super".
- t_RV request terminal version string; xterm: "\033[>c" change in terminfo for "RV" uses the wrong escape sequence 7 - 14 Jan only
- Codes for <PasteStart> t_PS and <PasteEnd> t_PE; with bracketed paste: t_BE and t_BD.

Modifiers for various keys

- flag to indicate "xterm compatible modifiers" ?

Underline and similar:

- t_AU - Set underline color: like "AF" and "AB" entries.
- t_Ce undercurl and underline end
- t-Cs undercurl (curly underline) mode
- t_Us double underline mode
- t_ds dotted underline mode
- t_Ds dashed underline mode
- t_Te strikethrough end
- t_Ts strikethrough mode

Cursor codes: [termcap-cursor-shape](#)

- t_RC request terminal cursor blinking
- t_RS request terminal cursor style
- t_VS cursor normally visible (no blink)
- t_SI start insert mode (bar cursor shape)
- t_SR start replace mode (underline cursor shape)
- t_EI end insert or replace mode (block cursor shape)
- t_SC set cursor color start
- t_EC set cursor color end
- t_SH set cursor shape

State requests:

- t_RF request terminal foreground color
- t_RB request terminal background color

Less important (not needed for regular editing):

- t_IS set icon text start
- t_IE set icon text end
- t_ST save window title to stack
- t_RT restore window title from stack
- t_Si save icon text to stack
- t_Ri restore icon text from stack
- t_WP set window position (Y, X) in pixels
- t_GP get window position (Y, X) in pixels
- t_WS set window size (height, width in cells)

Vim internal, but there should be a terminfo entry for these:

- t_8f set foreground color (R, G, B) in printf() format
- t_8b set background color (R, G, B) in printf() format
- t_8u set underline color (R, G, B) in printf() format

Probably Vim internal, not in terminfo:

- t_TE end of "raw" mode
- t_TI put terminal into "raw" mode
- t_RK request terminal keyboard protocol state; sent after [t_TI](#)

Already working, not properly documented:

- t_u7 request cursor position

Popup windows:

- Add a function to redraw a specific popup window. Esp. to be used when editing the command line, when screen updating doesn't happen. (Shougo)

#10210 Example that shows the need on the issue.

Probably need to update all popup windows (they may overlap)

If the display is scrolled need to redraw everything later.

- Add a flag to make a popup window focusable?
 - CTRL-W** P cycle over any preview window or focusable popup, end up back in current window.
 - ? - switch between current window and all popup windows
 - Esc in popup window goes back to previous current window
- Add a termcap entry for changing the cursor when it goes under the popup and back. like t_SI and t_EI (t_SU and t_EU, where "U" means under?)
- With terminal in popup, allow for popup_hide() to temporarily hide it.?
- Make it possible to scroll horizontally when 'wrap' is off. Does this require an ascii scrollbar, like with the vertical one?
- Fire some autocommand event after a new popup window was created and positioned? PopupNew? Could be used to set some options or move it out of the way. (#5737)
However, it may also cause trouble, changing the popup of another plugin.
- Width is not computed correctly when minwidth and maxwidth are &columns and padding and a scrollbar are used. (#6676)
- Should popup_getoptions() also return the mask? #7774
- Add a way to use popup_menu() synchronously: instead of invoking the callback, return the choice. (Ben Jackson, #6534)
- When using a popup for the info of a completion menu, and there is not enough space, let the popup overlap with the menu. (#4544)
- Implement flip option.
- Make redrawing more efficient and avoid flicker:
 - put popup menu also in popup_mask?
- Match does not have right BG color if line length equals popup width. (#5658)
- Figure out the size and position better if wrapping inserts indent

'incsearch' with :s:

- :s/foo using **CTRL-G** moves to another line, should not happen, or use the correct line (it uses the last but one line) (Lifepillar, Aug 18, #3345)
- Also support range: :/foo/,/bar/delete
- Also support for user command, e.g. Cfilter
- :%s/foo should take the first match below the cursor line, unless there isn't one?
Then :%s?foo should take the first match above the cursor line.

Prompt buffer:

- Add a command line history, using up/down keys. issue #5010
- delay next prompt until plugin gives OK?
- add prompt_addtext({buf}, {expr}) none add text to a prompt buffer

Terminal debugger:

- Add a mechanism for configuration. Instead of several global variables use a dictionary. Use callbacks for some things, e.g. opening a debug window in a certain position.
- Make prompt-buffer variant work better.
- Add option to not open the program window. It's not used when attaching to an already running program. (M. Kelly)
- Use the optional token on requests, match the result with it. #10300
- When only gdb window exists, on "quit" edit another buffer.

- Termdebug does not work when Vim was built with mzscheme: gdb hangs just after "run". Everything else works, including communication channel. Not initializing mzscheme avoid the problem, thus it's not some #ifdef.
- Add support for lldb? issue #3565
- Could we do something similar for debugging Vim scripts? At least see the script while stepping through it. Simple version would use an extra window. More complete solution would actually run Vim in a Terminal and control it with another Vim instance.

Terminal emulator window:

- Can escape a terminal popup with win_gotoid(), should be an error
- No support for underline color, t_8u.
- When in terminal-Normal mode when the job finishes, the cursor jumps to the end but the window is not updated. This only happens when typing "a".
:term bash -c "for V in {0..5}; do echo \$V; sleep 1; done"
- GUI: cursor color is not updated when going to Terminal-Job mode. #12328
- GUI: cursor color is not updated when switching between tabs with a key. #12329
- GUI: cursor color is not updated when switching between tabs with a mouse click. #12330
- When started with ":terminal ++close" and the shell exits but there is a background process, the window remains open, because the channel still exists (and output still shows). Perhaps close the window when an explicit ++close was used? (#5931)
- Using "**CTRL-W** :confirm quite" and selecting "yes" should work like ":quit!". (Harm te Hennepe, #6129)
- When the job in the terminal doesn't use mouse events, let the scroll wheel scroll the scrollbar, like a terminal does at the shell prompt. #2490
A bit like using **CTRL-W** N first.
Jump back like with "a" when any key is typed.
And use modeless selection. #2962
- Use **CTRL-W** **CTRL-K** to enter a digraph? #5371
- When Vim runs in the terminal and changes the title, the statusline needs to be redrawn.
- GUI: When using ":set go+!=" a system() call causes the hit-enter prompt. (#3327)
- Allow for specifying the directory, with ++cwd={dir}.
- When pasting should call vterm_keyboard_start_paste(), e.g. when using K_MIDDLEMOUSE, calling insert_reg().
- Users expect parsing the :term argument like a shell does, also support single quotes. E.g. with: :term grep 'alice says "hello"' (#1999)
- Win32: Redirecting input does not work, half of Test_terminal_redir_file() is disabled.
- Win32: Redirecting output works but includes escape sequences.
- Win32: Make terminal used for :!cmd in the GUI work better. Allow for redirection.
- When the job only outputs lines, we could handle resizing the terminal better: store lines separated by line breaks, instead of screen lines, then when the window is resized redraw those lines.
- Redrawing is slow with Motif. (Ramel Eshed)
- For the GUI fill termios with default values, perhaps like pangoterm:
<http://bazaar.launchpad.net/~leoner/pangoterm/trunk/view/head:/main.c#L134>
- When '**encoding**' is not utf-8, or the job is using another encoding, setup conversions.

Patch adds showcmd() function #11708

Cursor is after the end of the line: #12137.

Crash when a variable is removed while listing variables (Issue #11435)

Autoconf: must use autoconf 2.69, later version generates lots of warnings
- try using autoconf 2.71 and fix all "obsolete" warnings #11322

Problem with Visual highlight when **'linebreak'** and **'showbreak'** are set.
#11272

'cindent': compound literal indented wrong. Check for " = " before "{"?
#12491

GUI Scroll test fails on FreeBSD when using Motif. See FIXME in
Test_scrollbars in src/test_gui.vim

Selected index returned by complete_info() does not match the index in the
list of items. #12230

Support dark mode for MS-Windows: #12282

Remote command escapes single quote with backslash, should be doubling the
single quote in vim_strsave_escaped_ext() #12202.

PR to add custom and customlist completion types. #12228

Can deref_func_name() and deref_function_name() be merged?

Using :global with a pattern containing \zs doesn't use the line where \zs
matches but the start of the pattern. #3695 If there is a useful application
for this, it can be made to work by changing the call to ml_setmarked():
ml_setmarked(lnum + regmatch.startpos[0].lnum);

After patch 8.2.4915 w_botline is computed much more often. Can this be
reduced?

When **'delcombine'** is set a put after "x" includes the base character and all
combining characters. (Ron Aaron, 2023 Apr 10)

Add BufDeletePost. #11041

Add winid arg to col() and charcol() #11466 (request #11461)

'switchbuf' set to "newtab" does not work for ":cfirst" when in the quickfix
window. #12436

When :argument has a non-number argument, use it like :buffer to find the
argument by name. #12272

Fold unexpectedly closed when inserting fold marker with **CTRL-V** (Issue #12320)

Can we make **'noendofline'** and **'endoffile'** visible? Should show by default, since it's an unusual situation.

- Show **'noendofline'** when it would be used for writing ('fileformat' "dos") with an upside down exclamation mark? NonText highlighting.
- Show **'endoffile'** when it would be used for writing ('fileformat' "dos") with **"CTRL-Z"**, NonText highlighting.
- Add **'fillchars'** items to change this, default behavior like:
 noeol:~
 eof:CTRL-Z

Test property disappears when using CR twice in a row. OK when some text was entered. (#11151)

Add **'keywordprg'** to various ftplugin files:

<https://github.com/vim/vim/pull/5566>

PR #11579 to add visualtext(), return Visually selected text.

PR #12032: Support Python 3 stable ABI.

PR #11860: Add more info to **'colorcolumn'**: display a character and highlight for each separate entry. Disadvantage: option value gets very complicated with multiple entries, e.g. every 8 columns.

Stray characters in the shell #11719, caused by requesting a response for:

- XT key sequences
- Whether modifyOtherKeys is active
- Whether kitty keyboard protocol is active

Can we not request XT key sequences, or reduce them drastically?

Issue #10512: Dynamic loading broken with Perl 5.36

Damien has a patch (2022 Dec 4)

Request #11965: Allow several "%=" items in **'statusline'**, makes it possible to have text in the center.

Add some kind of ":whathappend" command and functions to make visible what the last few typed keys and executed commands are. To be used when the user wonders what went wrong. Could also be used for statistics #12046.

- typed keys - Normal mode command - like what is recorded in a register and displayed by **'showcmd'**.
- register used - #12063
- executed command lines
- with more verbosity: what scripts/functions/autocommands were executed

NFA regexp does not handle composing characters well: #10286

```
[ḿ] matches both ḿ and ḿ̄  
\(ḿ|\ḿ̄\) matches ḿ and not ḿ̄
```

Is there a way to make **'autowriteall'** make a clean exit when the xterm is closed? (Dennis Nazic says files are preserved, okt 28). Perhaps handle TERM like HUP?

Better terminal emulator support:

- > Somehow request the terminfo entry from the terminal itself. The \$TERM value then is only relevant for whether this feature is supported or not. Replaces the xterm mechanism to request each entry separately. #6609 Multiplexers (screen, tmux) can request it to the underlying terminal, and pass it on with modifications.
How to get all the text quickly (also over ssh)? Can we use a side channel?
- > When xterm supports sending an Escape sequence for the Esc key, should have a way to request this state. That could be an XTGETTCAP entry, e.g. "k!". Add "esc_sends_sequence" flag.
If we know this state, then do not pretend going out of Insert mode in vgetorpeek(), where kitty_protocol_state is checked.
- > If a response ends up in a shell command, one way to avoid this is by sending t_RV last and delay starting a shell command until the response has been seen.
- > Can we use the req_more_codes_from_term() mechanism with more terminals? Should we repeat it after executing a shell command?
Can also add this to the 'keyprotocol' option: "mok2+tcap"
- > In the table of terminal names pointing to the list of termcap entries, add an optional additional one. So that "xterm-kitty" can first load "xterm" and then add "kitty" entries.

Using "A" and "o" in manually created fold (in empty buffer) does not behave consistently (James McCoy, #10698)

In a timer callback, when using ":echo" and then input() the message is overwritten. Could use ":echowin" and call redraw_cmd() in get_user_input(). #11299

Syntax include problem: #11277. Related to Patch 8.2.2761

To avoid flicker: add an option that when a screen clear is requested, instead of clearing it draws everything and uses "clear to end of line" for every line. Resetting 't_ut' already causes this?

Instead of prefixing "INTERNAL" to internal messages, add a message in iemsg() and siemsg() and translate it. Messages only given to them don't need translation.

When scheme can't be found by configure there is no clear "not found" message:
configure:5769: checking MzScheme install prefix
configure:5781: result:

Can "CSI nr X" be used instead of outputting spaces? Is it faster? #8002

Typed keys invisible after calling interrupt() from a timer. #10631

With a window height of 6 and 'scrolloff' set to 3, using "j" does not scroll evenly. (#10545) Need to handle this in scroll_cursor_bot().

Idea: when typing ":e /some/dir/" and "dir" does not exist, highlight in red.

":set shellpipe&" and ":set shellredir&" should use the logic from initialization to figure out the default value from 'shell'. Add a test for this.

Add a diff() function to use the built-in diff support in a script.
#12321 Is the returned value in the right form now?

Support translations for plugins: #11637 PR: #12447

- Need a tool like xgettext for Vim script, generates a .pot file.
Need the equivalent of _() and N_(), perhaps TR() and TRN().
- Instructions for how to create .po files and translate.
- Script or Makefile to generate .mo files.
- Instructions and perhaps a script to install the .mo files in the right place.
- Add variant of gettext() that takes a package name.

With concealed text mouse click doesn't put the cursor in the right position.
(Herb Sitz) Fix by Christian Brabandt, 2011 Jun 16. Doesn't work properly,
need to make the change in where RET_WIN_BUF_CHARTABSIZE() is called.
IDEA: when drawing the text, store the text byte index in ScreenLinesIdx[].
When converting screen column to text position use this.
The line number can be obtained from win->w_lines[].

Version of getchar() that does not move the cursor - #10603 Use a separate
argument for the new flag.

test_arglist func Test_all_not_allowed_from_cmdwin() hangs on MS-Windows.

Can we add highlighting to ":echowindow"?

Information for a specific terminal (e.g. gnome, tmux, konsole, alacritty) is
spread out. Make a section with copy/paste examples of script and pointers to
more information.

Problem that a previous silent ":throw" causes a following try/catch not to
work. (ZyX, 2013 Sep 28) With examples: (Malcolm Rowe, 2015 Dec 24)
Also see #8487 for an example.

Request to use "." for the cursor column in search pattern \%<.c and \%<.v.
(#8179)

":filter set termcap" only filters terminal codes, not keys. (#9297)

Add an option to restrict 'incsearch' to not scroll the view. (Tavis Ormandy)

Remove SPACE_IN_FILENAME ? It is only used for completion.

When 'term' starts with "foot" then default t_TI and t_TE to the values used
for the builtin xterm termcap.

Adding "10" to 'spellsuggest' causes spell suggestions to become very slow.
(#4087) Did patch 8.2.2379 help?

Also, z= in German on a long word can take a very long time, but CTRL-C to
interrupt does not work. Where to add ui_breakcheck()?
New English spell files also have very slow suggestions.

When 'spelloptions' is "camel" then zG doesn't work on some words.

(Gary Johnson, 17 Oct 2022)

SpellCap doesn't show below a closed fold. #12420

'**cdpath**' problems:

- Adding "~" to '**cdpath**' doesn't work for completion? (Davido, 2013 Aug 19)
- Problem with '**cdpath**' on MS-Windows when a directory is equal to \$HOME. (2006 Jul 26, Gary Johnson)
- Completion of ":cd" doesn't use '**cdpath**'. #374.

Make "g>" and "g<" in Visual mode move the text right or left.
Also for a block selection. #8558

When using dictionary insert completion with '**ignorecase**', the leading capital in a word should be preserved.

Add optional argument to virtcol() that specifies "start", "cursor" or "end" to tell which value from getvvcol() should be used. (#7964)
Value returned by virtcol() changes depending on how lines wrap. This is inconsistent with the documentation.

When '**wildignore**' has an entry ending in "/*" this means nothing matching the path before it will be added. When encountering a directory check this and if there is a match do not scan the directory (possibly speeds up :find a lot).
#12482

globpath() does not use '**wildignorecase**' at all? (related to #8350)

mksession uses :buffer instead of :edit in one place but not another. #10629

Add '**termguiattr**' option, use "gui=" attributes in the terminal? Would work with '**termguicolors**'. #1740

Patch for blockwise paste reporting changes: #6660. Asked for a PR.

Missing filetype test for bashrc, PKGBUILD, etc.

Add an option to not fetch terminal codes in xterm, to avoid flicker when t_Co changes.

Add ??= operator, "a ??= b" works like "a = a ?? b". #10343

When ":redir" is used while already active, the previous one is ended. But when redirecting to a local variable (function or script) storing the value won't work. At least give an error. Is there a way to make it work?
#10616

Completion for ":runtime" should show valid values, not what's in the current directory. (#11447)

Add a "description" property to mappings. #12205

Add an option to start_timer() to return from the input loop with K_IGNORE. This is useful e.g. when a popup was created that disables mappings, we need

to return from vgetc() to make this happen. #7011

Expanding <mods> should put the tab number from cmdmod.tab before "tab".
Any way to convert "\$" back by using a special value? (#6901)

Can we detect true color support? <https://gist.github.com/XVilka/8346728>
Try setting a color then request the current color, like using t_u7.

Add a v:register_used variable, which has the name of the register used for the last command, e.g. put. #12003

Make the jumplist behave like a tag stack. (#7738) Should there be a more time bound navigation, like with undo?

For testing, make a copy of ml_line_ptr instead of pointing it into the data block, so that valgrind can do out of bounds check. Set ML_LINE_DIRTY flag or add ML_LINE_ALLOCED.

Changing a capturing group to non-capturing changes the result: #7607

```
:echo matchstr('aaa bbb', '\(.{-1,}\>\)\|.*')  
aaa  
:echo matchstr('aaa bbb', '\%(.-{-1,}\>\)\|.*')
```

```
aaa bbb
```

Should also work without any group:

```
:echo matchstr('aaa bbb', '.{-1,}\>\|.*')  
aaa bbb (should be aaa)
```

Should add a match/str/list/pos method that also returns the test and position of submatches. #8355

Check out PR #543 (Roland Puntaier).

Patch for multibyte characters in langmap and applying a mapping on them.

(Christian Brabandt, 2015 Jun 12, update July 25)

Is this the right solution? Need to cleanup langmap behavior:

- in vgetorpeek() apply langmap to the typeahead buffer and put the result in a copy-buffer, only when langmap is appropriate for the current mode. Then check for mapping and let gotchars() work on the copy-buffer.
- Remove LANGMAP_ADJUST() in other parts of the code. Make sure the mode is covered by the above change.

So that replaying the register doesn't use keymap/langmap and still does the same thing.

Also see #737: langmap not applied to replaying recording.

Perhaps also related: #5147

Matchparen doesn't remove highlight after undo. (#7054)

Is OK when syntax HL is active.

Currently Del can be used to delete the last character of a typed count.

Can it also be used to delete an incomplete Normal mode command? (#7096)

After an operator: should work. After "a" or "i" for text objects: should work.

Using "au!" after "filetype on" is a bit slow. Can the matching of autocommands be made faster? (#7056)

Using a search pattern containing "%V" and 'hlsearch' set keeps old matches when the Visual area changes. #11694. Do a redraw when starting a Visual selection?

Append in Visual block mode inserts the wrong character.
Test_visual_block_mode() already has the proper check, which is commented out. (#8288)

Add the <=> (spaceship) operator and "cond ?< expr ?= expr ?> expr"
replace this:
 let left = GetLeftFunc()
 let right = GetRightFunc()
 let res = left < right ? lower : left == right ? equal : upper
by:
 let res = GetLeftFunc() <=> GetRightFunc() ?< lower ?= equal ?> upper
Patch to make :q work with local arglist. (Christian Brabandt, #6286)

Why does Test_invalid_sid() not work in the GUI?

":pedit" ignores the local working directory when 'pvp' is set (#7267)

Lua: updating wrong buffer when using newly created, unloaded buffer. (#6539)

File marks merging has duplicates since 7.4.1925. (Ingo Karkat, #5733)

A syntax plugin cannot use autocommands, it could be sourced from setting 'syntax' in a modeline. Add a function that indicates whether "secure" and/or "sandbox" are set.

Problem with auto-formatting - inserting space and putting cursor before added character. (#6154)

Auto-formatting comments joins a following non-comment line when the comment ends in a space. (Adam Levy, 18 Oct 2022)

When 'lazyredraw' is set sometimes the title is not updated. (Jason Franklin, 2020 Feb 3) Looks like a race condition.

With bash ":make" does not set v:shell_error. Possible solution: set 'shellpipe' to "2>&1| tee %s; exit \${PIPESTATUS[0]}" #5994

Using mode() when "/pat" is used in Visual mode returns "v" instead of "c", which is not useful. Return "c/v" instead. And "c/o" when using "d/pat". #6127

Add a cterm attribute for "dimmed" or "faint" text. (#8269)

When 'fileignorecase' is set ":e testfile.c" works to edit TestFile.c, but ":find testfile.c" does not ignore case. Might be related to #6088.

Error for reverse range when using :vimgrep in file "[id-01] file.txt". (#6919)

When changing the crypt key the buffer should be considered modified.
Like when changing **'fileformat'**. Save the old key in save_file_ff().
(Ninu-Ciprian Marginean)

Strange sequence of BufWipeout and BufNew events while doing omni-complete.
(Paul Jolly, #5656)

Get BufDelete without preceding BufNew. (Paul Jolly, #5694)

Later more requests for what to track.

Should we add new events that don't allow any buffer manipulation?

Really only for dealing with appearing and disappearing buffers, load and unload.

BufWinenter event not fired when saving unnamed buffer. (Paul Jolly, #5655)

Another spurious BufDelete. (Dani Dickstein, #5701)

Wrong error when using local arglist. (Harm te Hennepe, #6133)

Test loose_clipboard() by selecting text before suspending.

:unmap <c-n> gives error but does remove the mapping. (Antony Scriven, 2019 Dec 19)

Patch to add an option to enable/disable VTP. (Nobuhiro Takasaki, #5344)

Should have three values: empty, "off", "on". Name it **'winterm'**?

Patch to fix session file when using multiple tab pages. (Jason Franklin, 2019 May 20)

Also put :argadd commands at the start for all buffers, so that their order remains equal? Then %argdel to clean it up. Do try this with **'hidden'** set.

Also #5326: netrw buffers are not restored.

When **'backupdir'** has a path ending in double slash (meaning: use full path of the file) combined with **'patchmode'** the file name is wrong. (#5791)

Completion mixes results from the current buffer with tags and other files.

Happens when typing **CTRL-N** while still searching for results. E.g., type "b_" in terminal.c and then **CTRL-N** twice.

Should do current file first and not split it up when more results are found.

(Also #1890)

Help for ":argadd fname" says that if "fname" is already in the argument list that entry is used. But instead it's always added. (#6210)

Add flag AL_FIND_ADD, if there is one argument find it in the list.

Statusline highlighting error, off by one. (#5599)

":find" with **'path'** set to "data*" does not find files, while completion does find them. (Max Kukartsev, #6218)

Enable **'termbidi'** if \$VTE_VERSION >= 5703 ?

Python 3.8 doesn't work. (Antonios Hadjigeorgalis, #5509)

"--cleanF00" does not result in an error. (#5537)

Output from `assert_equalfile()` doesn't give a hint about what's different. Assuming the files are text, print the line with the difference.

Result of `synID()` sometimes wrong in help files. (#5252)

When a help file is opened that doesn't have `"ft=help"` in the modeline then the `FileType` is first set to `"text"` before it is set to `"help"`. (#8099)

Problem showing a line if the number column width changes when using `"o"`. (Mateusz Morusiewicz, #4245)

When using `:packadd` for a replacement language plugin, it is loaded after the default one. #4698

When using `:packadd` files under `"later"` are not used, which is inconsistent with packages under `"start"`. (xtal8, #1994)

Patch to add new motion `](` and `]{'`. (Yasuhiro Matsumoto, #5320)
Better: use the `"z"` prefix. or `]t` and `[t`.

Visual highlight not removed when `'display'` is `"lastline"` and line doesn't fit. (Kevin Lawler, #4457)

Current position in the changelist should be local to the buffer. (#2173)

Does not build with MinGW out of the box:

- `_stat64` is not defined, need to use `"struct stat"` in `vim.h`
- `WINVER` conflict, should use `0x0600` by default?
- `INT_MAX` not defined: need to include `<limits.h>` in `vim.h`

Display messed up with `matchparen`, wrapping and scrolling. (#5638)
Screen update bug related to `matchparen`. (Chris Heath, 2017 Mar 4, #1532)

When getting a focus event halfway a mapping this aborts the mapping. E.g. when `"qq"` is mapped and after the first `"q"` the mouse is moved outside of the `gvim` window (with focus follows mouse), then the `K_FOCUSLOST` key is put in the input buffer. (#5302)

`Check_external_diff()` is used too often. (Daniel Hahler, #4800)

Win32: after `"[I"` showing matches, scroll wheel messes up screen. (Tsakiridis, 2007 Feb 18)
Patch by Alex Dobrynin, 2007 Jun 3. Also fixes other scroll wheel problems.

Add a `WindowScrolled` event. Trigger around the same time as `CursorMoved`. Can be used to update highlighting. #3127 #5181

Incorrect formatting with `autoindent`. (Sebastian Gniazdowski, #4909)

Patch to add the `:bvimgrep` command. (Christian Brabandt, 2014 Nov 12)
Updated 2016 Jun 10, #858 Update 2017 Mar 28: use `<buffer>`.
Better use `":bufgrep"` ?

Improve fallback for menu translations, to avoid having to create lots of files that source the actual file. E.g. menu_da_de -> menu_da
Include part of #3242?

Patch for different behavior of text objects with quotes: #11976
Is this actually better?

Patch to have text objects defined by arbitrary single characters. (Daniel Thau, 2013 Nov 20, 2014 Jan 29, 2014 Jan 31)
Added tests (James McCoy, 2016 Aug 3, #958). Still needs more work.

Would be nice to set tab-local values for **'diffexpr'** and **'diffopt'**. Use `t:diffexpr_option` `t:diffopt_option`? (#4782)
Also make **'scrollopt'** tab-local, remove "hor" only for the current tab page.

Internal diff doesn't handle binary file like external diff does. (Mike Williams, 2018 Oct 30)

'[mark in wrong column after put. (#4776)

Problem with `:tmenu`: Detach item added with all modes? Issue #3563.

Add an argument to `expandcmd()` to expand like `":next"` does.

When both "a" and "l" is in **'formatoptions'** then auto-formatting also happens in a long line. #5189

The quoting of the `[command]` argument of `:terminal` is not clearly documented. Give a few examples. (#4288)

Opening a file with `--remote-tab-silent` that matches **'wildignore'** does not work, results in "E479: No match". (#4610)

7 Add an option to add one pixel column to the character width? Lucida Console italic is wider than the normal font ("d" overlaps with next char).
Opposite of **'linespace'**: **'columnspace'**.
Patch for this (Tristan Konolige, #1011, only added the option, no implem.)

Bug: script written with `"-W scriptout"` contains Key codes, while the script read with `"-s scriptin"` expects escape codes. Probably "scriptout" needs to be adjusted. (Daniel Steinberg, 2019 Feb 24, #4041)

Window size changes after closing a tab. (#4741)

Problem with colors in terminal window. (Jason Franklin, 2019 May 12)

Color schemes:

NOTE: modernizing the default colorschemes `_AND_` introducing new ones is now a project in its own right: <https://github.com/vim/colorschemes>. Feel free to reach out if you want to lend a hand.

- Lifepillar: Updated/cleaned up color schemes:

<https://github.com/lifepillar/vim8-colorschemes>.

- Include a few color schemes, based on popularity:

- http://www.vim.org/scripts/script_search_results.php?keywords=&script_type=color+schem

<http://vimawesome.com/?q=tag:color-scheme>

- Use names that indicate their appearance (Christian Brabandt, 2017 Aug 3)
 - monokai - Xia Crusoe (2017 Aug 4)
 - seoul256 - Christian Brabandt (2017 Aug 3)
 - gruvbox - Christian Brabandt (2017 Aug 3) (simplified version from Lifepillar, 2018 Jan 22, #2573)
 - janah - Marco Hinz (2017 Aug 4)
 - apprentice - Romain Lafourcade (2017 Aug 6) remarks about help file #1964
- Suggested by Hiroki Kokubun:
 - [Iceberg](<https://github.com/cocopon/iceberg.vim>) (my one)
 - [hybrid](<https://github.com/w0ng/vim-hybrid>)
- Include solarized color scheme?, it does not support termguicolors.
 - Sanitized version of pablo (Lifepillar, 2017 Nov 21)

Bug: "vipgw" does not put cursor back where it belongs. (Jason Franklin, 2019 Mar 5)

Some composing characters actually add a cell width to the character they are on top of, making the whole thing two characters wide. (#4526)

Should we include some part of pull request #4505, not increment changedtick in some cases? E.g. for ":write" when the changed flag was already off, the buffer didn't change at all.

When using a timer callback vgetc_busy is reset, allowing for using input(). But in a channel callback this does not happen. We need to do something similar to check_due_timer(). Also see #3809.

C syntax: {} inside () causes following {} to be highlighted as error. (Michalis Giannakidis, 2006 Jun 1)

Check: __attribute__((format(printf, on semsg() and siemsg()). Where was this added?

Add test for urxvt mouse codes. Also test that mouse coordinates can be negative. (see #4326)

'cmdheight' has a tab-local value, but it cannot be obtained with `:echo gettabwinvar(2, 1, '&cmdheight')` returns the value for the _current_ tab page. (Ingo Karkat, #4324)
:call settabwinvar(1, 1, '&cmdheight', 2) also doesn't work well.

When opening a file, allow for specifying the initial column position:

```
vim +12:5 file.txt line 12 column 5
:edit +12:5 file.txt
```

Should probably use the column as the character index.

This modeline throws unexpected errors: (#4165)

```
vim: syn=nosyntax
```

Make balloon_show() work outside of 'balloonexpr'? Users expect it to work: #2948. (related to #1512?)

Also see #2352, want better control over balloon, perhaps set the position. Should also be possible to add highlighting, like in the status line?

balloonexpr() on MS-Windows GUI doesn't handle accented chars? (nivaemail, 2018 Sep 14)

More warnings from static analysis:

<https://lgtm.com/projects/g/vim/vim/alerts/?mode=list>

Not existing directory in CDPATH leads to two shell calls. (#4525)

Use dict_iterate_start() / dict_iterate_next() instead of relying on the internals of the dict structure.

nvo-mode mapping works on Windows, not on Linux. (#3678)

Redo only remembers the last change. Could use "{count}g." to redo an older change. How does the user know which change? At least have a way to list them: ":repeats". Add to history, like search history and command line history.

When 'confirm' is set a "silent q" doesn't show the prompt. It should in this case. (Nate Peterson, 2019 Jan 31, #3892)

For "silent! q" it should not prompt and just fail.

Add <aevent>, which expands to the currently triggered autocommand event name. (Daniel Hahler, #4232) Or add it to v:event (easier to use but slightly more expensive).

Some xterm responses are not properly handled: (Markus Gömmel, 2019 Apr 1)

```
DCS 0 $ r Pt ST      should be ignored.  
DCS 0 + r/Pt/ ST    already ignored?
```

Using **CTRL-L** to add a character to the search string that contains \v, punctuation is repeated. (Smylers, 2018 Nov 17, #3621)

Using single wide base character with double wide composing character gives drawing errors. Fill up the base character? (Dominique, #4328)

When 'sidescrolloff' is set, using "zl" to go to the end of the line, suddenly scrolls back. Should allow for this scrolling, like 'scrolloff' does when using **CTRL-E**. (Yee Cheng Chin, #3721)

When splitting a window with few text lines, the relative cursor position is kept, which means part of the text isn't displayed. Better show all the text when possible. (Dylan Lloyd, #3973)

Make ":interactive !cmd" stop termcap mode, also when used in an autocommand. (#3692)

Add something like 'fillchars' local to window, but allow for specifying a highlight name. Esp. for the statusline.

And "extends" and "precedes" are also useful without 'list' set. Also in 'fillchars' or another option?

Sourceforge Vim pages still have content, make them empty, keep redirect. Check for PHP errors. (Wayne Davison, 2018 Oct 26)

Problem with Visual yank when **'linebreak'** and **'showbreak'** are set.
Patch with tests, but it's not clear how it is supposed to work. (tommm, 2018 Nov 17) Asked about this, Dec 22. Christian will have a look.

Update for xim-input-style help (Tony Mechelynck, 2019 Jan 10).
Feedback from someone who uses this?

Only output t_Cs when t_Ce is also set. do not use Cs and Ce termcap entries. (Daniel Hal
Add t_cS and t_cR for cursor color select and reset. Use Cs and Cr terminfo values.

Further xdiff changes:

- More options, e.g. different kind of whitespace diff.
- when editing text, update the surrounding diff blocks.
- omit diff.exe from distribution
- Can we make this show differences within a line?
- add option to use external diff above a certain size.

Difference between two regexp engines: #3373

When the last line wraps, selecting with the mouse below that line only includes the first screen line. (2018 Aug 23, #3368)

Refactored HTML indent file. (Michael Lee, #1821)
Asked to write a test.

Merge checking for **'cursorline'** and **'concealcursor'**, see neovim #9492.

Add a windowID argument to placing a sign, so that it only shows up in one window for the buffer.

Compiler warning (geeknik, 2017 Oct 26):
- undefined left shift in eval_string(), before hex2nr() (#2250)
Use unsigned for "nr".

Add Native language protocol server (LSP) support. (Yegappan Lakshmanan, 2018 Oct 28)

Patch to be able to use hex numbers with :digraph. (Lcd, 2015 Sep 6)
Update Sep 7. Update by Christian Brabandt, 2015 Sep 8, 2016 Feb 1.
Patch to be able to disable default digraphs (incomplete) (Eric Pruitt, 2018 Nov 22).

Patch to list user digraphs. (Christian Brabandt, 2012 Apr 14)

Setting **'columns'** in a BufEnter autocommand causes a second tab width to behave strangely, as if there is a gap and a vertical window separator. (Michael Soyka, 2018 Sep 23, #3477)

Add an option similar to **'lazyredraw'** to skip redrawing while executing a script or function.

Using a menu item while the "more" prompt is displayed doesn't work well. E.g. after using help->version. Have a key that ends the "more" prompt and

does nothing otherwise?

MS-Windows: write may fail if another program is reading the file.

If **'readonly'** is not set but the file appears to be readonly later, try again (wait a little while).

CreateFile() returns ERROR_SHARING_VIOLATION (Linwei, 2018 May 5)

Using --remote to open a file in which a # appears does not work on MS-Windows. Perhaps in \# the \ is seen as a path separator. (Axel Bender, 2017 Feb 9) Can we expand wildcards first and send the path literally to the receiving Vim? Or make an exception for #, it's not useful remotely.

Column number is wrong when using **'linebreak'** and **'wrap'**. (Keith Smiley, 2018 Jan 15, #2555)

Add Makefiles to the runtime/spell directory tree, since nobody uses Aap. Will have to explain the manual steps (downloading the .aff and .dic files, applying the diff, etc).

User dictionary ~/.vim/spell/lang.utf-8.add not used for spell checking until a word is re-added to it. (Matej Cepl, 2018 Feb 6)

Fold at end of the buffer behaves inconsistently. (James McCoy, 2017 Oct 9)

Implement option_save() and option_restore():

`option_restore({list})` option_restore()
Restore options previously saved by option_save().
When buffer-local options have been saved, this function must be called when the same buffer is the current buffer.
When window-local options have been saved, this function must be called when the same window is the current window.
When in the wrong buffer and/or window an error is given and the local options won't be restored.

`option_save({list})` option_save()
Saves the options named in {list}. The returned value can be passed to option_restore(). Example:

```
let s:saved_options = option_save([
  \ 'ignorecase',
  \ 'iskeyword',
  \ ])
au <buffer> BufLeave *
  \ call option_restore(s:saved_options)
```

The advantage over using ``:let`` is that global and local values are handled and the script ID is restored, so that ``:verbose set`` will show where the option was originally set, not where it was restored.

Alternatively: save and restore ALL options. Implementation needs to use copy-on-write. Return an ID from option_save(), when option_restore(ID) is called give an error if another option_save() was called in the meantime, they must be balanced.

"gvim --remote" from a directory with non-word characters changes the current directory (Paulo Marcel Coelho Arabic, 2017 Oct 30, #2266)

Also see #1689.

No profile information for function that executes ":quit". (Daniel Hahler, 2017 Dec 26, #2501)

A function on a dictionary is not profiled. (ZyX, 2010 Dec 25)

Add script number to profile? (#3330 breaks tests).

A function defined locally and lambda's are not easily recognized. Mention where they were defined somewhere.

ml_get errors with buggy script. (Dominique, 2017 Apr 30)

Error in emsg with buggy script. (Dominique, 2017 Apr 30)

Join truncates xml comment. (Dmitrii Tcyganok, 2017 Dec 24, #2494)
Requires '**formatoptions**' to include "j". (Gary Johnson, 2017 Dec 24)

Patch to support hunspell. (Matej Cepl, Jan 2018, #2500) Based on older patch in #846)

Doesn't work on Windows yet. Not ready to included, hard coded paths.

When a timer is running and typing **CTRL-R** on the command line, it is not redrawn properly. (xtal8, 2017 Oct 23, #2241)

In an optional package the "after" directory is not scanned?
(Renato Fabbri, 2018 Feb 22)

Patch for Neovim concerning restoring when closing help window. (glacambre neovim #7431)

Patch for improving detecting Ruby on Mac in configure. (Ilya Mikhaltsou, 2017 Nov 21)

When t_Co is changed from termresponse, the OptionSet autocommand event isn't triggered. Use the code from the end of set_num_option() in set_color_count().

When using command line window, CmdlineLeave is triggered without CmdlineEnter. (xtal8, 2017 Oct 30, #2263)

Add some way to get the nested state. Although CmdwinEnter is obviously always nested.

matchit hasn't been maintained for a long time. #955.

Problem with '**delcombine**'. (agguser, 2017 Nov 10, #2313)

'delcombine' does not work for the command line. (Tony Mechelynck, 2009 Jul 20)

MS-Windows: buffer completion doesn't work when using backslash (or slash) for a path separator. (xtal8, #2201)

Would be nice for Insert mode completion to highlight the text that was added (and may change when picking another completion).

Test more runtime files.

Window not closed when deleting buffer. (Harm te Hennepe, 2017 Aug 27, #2029)

Add options_default() / options_restore() to set several options to Vim defaults for a plugin. Comments from Zyx, 2017 May 10.
Perhaps use a vimcontext / endvimcontext command block.

After using :noautocmd CursorMoved may still trigger. (Andy Stewart, 2017 Sep 13, #2084). Set old position after the command.

When bracketed paste is used, pasting at the ":append" prompt does not get the line breaks. (Ken Takata, 2017 Aug 22)

Cannot copy modeless selection when cursor is inside it. (lkintact, #2300)

Test_writefile_fails_conversion failure on Solaris because of different iconv behavior. Skip when "uname" returns "SunOS"? (Pavel Heimlich, #1872)

'tagrelative' is broken in specific situation. (xaizek, 2017 Oct 19, #2221)

The ++ options for the :edit command are also useful on the Vim command line.

Overlong utf-8 sequence is displayed wrong. (Harm te Hennepe, 2017 Sep 14, #2089) Patch with possible solution by Björn Linse.

X11: Putting more than about 262040 characters of text on the clipboard and pasting it in another Vim doesn't work. (Dominique Pelle, 2008 Aug 21-23) clip_x11_request_selection_cb() is called with zero value and length.
Also: Get an error message from free() in the process that owns the selection. Seems to happen when the selection is requested the second time, but before clip_x11_convert_selection_cb() is invoked, thus in X library code.
Kazunobu Kuriyama is working on a proper fix. (2017 Jul 25)

Problem with three-piece comment. (Michael Lee, 2017 May 11, #1696)

Creating a partial with an autoload function is confused about the "self" attribute of the function. For an unknown function assume "self" and make that optional? (Bjorn Linse, 2017 Aug 5)

Cindent: returning a structure has more indent for the second item. (Sam Pagenkopf, 2017 Sep 14, #2090)

Patch from Christian Brabandt to preserve upper case marks when wiping out a buffer. (2013 Dec 9)
Also fixes #2166?

Profile of a dict function is lost when the dict is deleted. Would it be possible to collect this? (Daniel Hahler, #2350)

When checking if a bufref is valid, also check the buffer number, to catch the

case of :bwipe followed by :new.

Patch to skip writing a temp file for diffing if the buffer is equal to the existing file. (Akria Sheng, 2017 Jul 22)
Could also skip writing lines that are the same.

MS-Windows: Opening same file in a second gvim hangs. (Sven Bruggemann, 2017 Jul 4)

Setting '**clipboard**' to "unnamed" makes a global command very slow (Daniel Drucker, 2017 May 8).
This was supposed to be fixed, did it break again somehow?
Christian cannot reproduce it.

Using composing char in mapping does not work properly. maparg() shows the wrong thing. (Nikolai Pavlov, 2017 Jul 8, #1827)
Or is this not an actual problem?

Better TeX indent file. (Christian Brabandt, 2017 May 3)

Use gvimext.dll from the nightly build? (Issue #249)

'**synmaxcol**' works with bytes instead of screen cells. (Llondon, 2017 May 31, #1736)

Problem with using :cd when remotely editing a file. (Gerd Wachsmuth, 2017 May 8, #1690)

Memory leak in test97? The string is actually freed. Weird.

assert_fails() can only check for the first error. Make it possible to have it catch multiple errors and check all of them.

Add a toolbar in the terminal. Can be global, above all windows, or specific for one window.

Make maparg() also return the raw rhs, so that it doesn't depend on '**cpo**'. (Brett Stahlman, 2017 May 23)

Even better: add a way to disable a mapping temporarily and re-enable it later. This is for a sub-mode that is active for a short while (one buffer). Still need maplist() to find the mappings. What can we use to identify a mapping? Something unique would be better than the LHS.
Perhaps simpler: actually delete the mappings. Use maplist() to list matching mappings (with a lhs prefix, like maparg()), mapdelete() to delete, maprestore() to restore (using the output of maplist()).

Add an argument to :mkvimrc (or add another command) to skip mappings from plugins (source is a Vim script). No need to put these in a .vimrc, they will be defined when the plugin is loaded.

```
Use tb_set(winid, [{'text': 'stop', 'cb': callback, 'hi': 'Green'}])
   tb_highlight(winid, 'ToolBar')
   tb_get(winid)
```

json_encode(): should convert to utf-8. (Nikolai Pavlov, 2016 Jan 23)
What if there is an invalid character?

Json string with trailing \u should be an error. (Lcd)

import can't be used in define option when include matches too.
(Romain Lafourcade, 2017 Jun 18, #1519)

Wrong diff highlighting with three files. (2016 Oct 20, #1186)
Also get E749 on exit.
Another example in #1309

Suggestion to improve pt-br spell checking. (Marcelo D Montu, 2016 Dec 15,
#1330)

Error in test_startup_utf8 on Solaris. (Danek Duvall, 2016 Aug 17)

Rule to use "^" for statusline does not work if a space is defined with
highlighting for both stl and stlnc. Patch by Ken Hamada (itchyny, 2016 Dec 11)

Using **CTRL-G_U** in InsertCharPre causes trouble for redo. (Israel Chauca
Fuentes, 2017 Feb 12, #1470)

Add a "keytrans()" function, which turns the internal byte representation of a
key into a form that can be used for :map. E.g.

```
let xx = "\<C-Home>"  
echo keytrans(xx)  
<C-Home>
```

Check for errors E704 and E705 only does VAR_FUNC, should also do VAR_PARTIAL.
(Nikolai Pavlov, 2017 Mar 13, #1557)

Make a function to check for function-like type?

Implement named arguments for functions with optional arguments:

```
func Foo(start, count = 1, all = 1)  
call Foo(12, all = 0)
```

Add a command to take a range of lines, filter them and put the output
somewhere else. :{range}copy {dest} !cmd

The TermResponse event is not triggered when a plugin has set 'eventignore' to
"all". Netrw does this. (Gary Johnson, 2017 Jan 24)

Postpone the event until 'eventignore' is reset.

Expanding /**/ is slow. Idea by Luc Hermitte, 2017 Apr 14.

Once .exe with updated installer is available: Add remark to download page
about /S and /D options (Ken Takata, 2016 Apr 13)

Or point to nightly builds: <https://github.com/vim/vim-win32-installer/releases>

":sbr" docs state it respects 'switchbuf', but "vsplit" does not cause a
vertical split. (Haldean Brown, 2017 Mar 1)

Use ADDR_OTHER instead of ADDR_LINES for many more commands.

E.g. all the location list commands use a count.
Add tests for using number larger than number of lines in buffer.

Might be useful to have `isreadonly()`, like we have `islocked()`.
Avoids exceptions, e.g. when using the `b:` namespace as a dict.

Patch to make `v:shell_error` writable. (Christian Brabandt, 2016 Sep 27)
Useful to restore it. Is there another solution?

Patch for wrong cursor position on wrapped line, involving `breakindent`.
(Ozaki Kiichi, 2016 Nov 25)

Patch for `'cursorlinenr'` option. (Ozaki Kiichi, 2016 Nov 30)

Window resizing with `'winfixheight'`: With a vertical split the height changes anyway. (Tommy allen, 2017 Feb 21, #1502)

Invalid behavior with NULL list. (Nikolai Pavlov, #768)
E.g. `deepcopy(test_null_list())`

Patch to make it possible to extend a list with itself.
(Nikolai Pavlov, 2016 Sep 23)

Patch to add Zstandard compressed file support. (Nick Terrell, 2016 Oct 24)

On Windows buffer completion sees backslash as escape char instead of path separator. (Toffanim, 2016 Nov 24, #1274)

Should `:vmap` in `matchit.vim` be `:xmap`? (Tony Mechelynck)

Problem with whitespace in `errorformat`. (Gerd Wachsmuth, 2016 May 15, #807)

Support `sort(l, 'F')`, convert strings to float. (#7857)

`sort()` is not stable when using numeric/float sort (Nikolay Pavlov, 2016 Sep 4#1038)

`sort()` does not use `'smartcase'` for the skip pattern, even though `'ignorecase'` is used. (Filipe Brandenburger, #7322)

`+channel:`

- Add a `in_cb`, invoked when the write buffer has become empty. (Matteo Landi)
- Add `ch_readlines()`: for a channel in NL mode, reads as many lines as are available. Should be more efficient than looping over `ch_read()` with `ch_status()` to check for more.
- If buffer contents is changed in a callback, set `w_redr_status` so that it gets redrawn in `redraw_after_callback()`. #6120
- Add a separate timeout for opening a socket. Currently it's fixed at 50 msec, which is too small for a remote connection. (tverniquet, #2130)
- Writing raw mode to a buffer should still handle NL characters as line breaks. (Dmitry Zotikov, 2017 Aug 16)
- When `out_cb` executes `:sleep`, the `close_cb` may be invoked. (Daniel Hahler, 2016 Dec 11, #1320)
- Implement `job-term` ?

- Calling a function when receiving a "call" on a channel, using `feedkeys()` does not work. It does work from a timer. (Qiming Zhao, #3852)
- Channel test fails with Motif. Sometimes kills the X11 server.
- When a message in the queue but there is no callback, drop it after a while? Add timestamp to queued messages and callbacks with ID, remove after a minute. Option to set the droptime.
- Add an option to drop text of very long lines? Default to 1 Mbyte.
- Add remark about undo sync, is there a way to force it?
- When starting a job, have an option to open the server socket, so we know the port, and pass it to the command with `--socket-fd {nr}`. (Olaf Dabrunz, Feb 9) How to do this on MS-Windows?
- For connection to server, a "keep open" flag would be useful. Retry connecting in the main loop with zero timeout.
- `job_start()`: run job in a newly opened terminal (not a terminal window). With xterm could use `-S{pty}`. Although user could use `"xterm -e 'cmd arg'"`.

Regex problems:

- NFA engine can be slow for some patterns. Dominique found out that most time is spent in `addstate_here()` copying the threads. Instead of copying, let each thread point to the next one (by offset, the list is reallocated). (Dominique Pelle, 2019 Feb 18)
- Old engine: using `'incsearch'` `/\Zabc` does not highlight the "c" if it has a composing character. New engine is OK. (Tony Mechelynck, 2019 May 5)
- When search pattern has the base character both with and without combining character, search fails. E.g. "xxxx" in "xxxxxxxxxxxx". (agguser, #2312)
- `[:space:]` only matches ASCII spaces. Add `[:white:]` for all space-like characters, esp. including `0xa0`. Use character class zero.
- Since 7.4.704 the old regex engine fails to match `[:print:]` in `0xf6`. (Manuel Ortega, 2016 Apr 24)
Test fails on Mac. Avoid using `isalpha()`, `isalnum()`, etc? Depends on `LC_CTYPE`
- The old engine does not find a match for `"/\%#=1\(\)\{80}"`, the new engine matches everywhere.
- Using `win_linetabsize()` can still be slow. Cache the result, store `col` and `vcol`. Reset them when moving to another line.
- Very slow with a long line and Ruby highlighting. (John Whitley, 2014 Dec 4)
- Bug with pattern: `'\vblock (\d+)\. \.n.*\d+(\1)@<!\. $'` (Lech Lorens, 2014 Feb 3)
- Issue 164: freeze on regex search.
- `Ignorecase` not handled properly for multibyte characters. (Axel Bender, 2013 Dec 11)
- Using `@>` and `\?`. (Brett Stahlman, 2013 Dec 21) Remark from Marcin Szamotulski; Remark from Brett 2014 Jan 6 and 7.
- NFA regex doesn't handle `\%<v` correctly. (Ingo Karkat, 2014 May 12)
- Does not work with NFA regex engine:
`\%u`, `\%x`, `\%o`, `\%d` followed by a composing character
- Search for `\%d\+` may fail with E363. (Christian Brabandt, 2016 Oct 4)
- `\%'` does not work. `'%'` does work. (Masaaki Nakamura, 2016 Apr 4)
- Bug relating to back references. (Ingo Karkat, 2014 Jul 24)
- New RE does not give an error for empty group: `"\(\)\{2}"` (Dominique Pelle, 2015 Feb 7)
- Using back reference before the capturing group sometimes works with the old engine, can we do this with the new engine? E.g. with

"/\%(<\1>\)\@<=.*\%(<\|(\w\+)>)\@=" matching text inside HTML tags.
This problem is probably the same: "\%(^\1.*\$\n\)\@<=\\(\d\+).*\$".
(guotuofeng, 2015 Jun 22)

- Strange matching with "\(Hello\n\)\@<=A". (Anas Syed, 2015 Feb 12)
- Problem with \v(A)\@<=b+\1c. (Issue 334)
- Diff highlighting can be very slow. (Issue 309)
- Using %> for a virtual column has a check based on **'tabsize'**. Better would be to cache the result of win_linetabsize(col), storing both col and vcol, and use them to decide whether win_linetabsize() needs to be called. Reset col and vcol when moving to another line.
- this doesn't work: "syntax match ErrorMessage /.%9l\%>20c\&\%<28c/". Leaving out the \& works. Seems any column check after \& fails.
- Difference between two engines: ".*\zs\|@>\|" on text "///"
(Chris Paul, 2016 Nov 13) New engine not greedy enough?
Another one: echom matchstr(" sdfsfsf\n sdfsdfsdf",'^\n*')
(2017 May 15, #1252)

Idea from Sven: record sequence of keys. Useful to show others what they are doing (look over the shoulder), and also to see what happened.
Probably list of keystrokes, with some annotations for mode changes.
Could store in logfile to be able to analyse it with an external command.
E.g. to see when's the last time a plugin command was used.

cmap using execute() has side effects. (Killthemule, 2016 Aug 17, #983)

:map X may print invalid data. (Nikolay Pavlov, 2017 Jul 3, #1816)

Patch to order results from taglist(). (Duncan McDougall, 2016 Oct 25)

ml_get errors when reloading file. (Chris Desjardins, 2016 Apr 19)
Also with latest version.

Completion for input() does not expand environment variables. (chdiza, 2016 Jul 25, #948)

Patch to add **'systemencoding'**, convert between **'encoding'** and this for file names, shell commands and the like. (Kikuchan, 2010 Oct 14)
Assume the system converts between the actual encoding of the filesystem to the system encoding (usually utf-8).

MS-Windows: use WS_HIDE instead of SW_SHOWMINNOACTIVE in os_win32.c?
Otherwise task flickers in taskbar.

Second problem in #966: ins_compl_add_tv() uses get_dict_string() multiple times, overwrites the one buffer. (Nikolay Pavlov, 2016 Aug 5)

Patch to improve map documentation. Issue #799.

We can use '.' to go to the last change in the current buffer, but how about the last change in any buffer? Can we use ', (, is next to .)?

Ramel Eshed: system() is much slower than job_start(), why? (Aug 26)

When generating the Unicode tables with runtime/tools/unicode.vim the

emoji_width table has only one entry.

It's possible to add ",," to **'wildignore'**, an empty entry. Causes problems.
Reject the value? #710.

When doing "vi buf.md" a BufNew autocommand for *.md is not triggered.
Because of using the initial buffer? (Dun Peal, 2016 May 12)

Neovim patch for utfc_ptr2char_len() <https://github.com/neovim/neovim/pull/4574>
No test, needs some work to include.

Patch to improve indenting for C++ constructor with initializer list.
(Hirohito Higashi, 2016 Mar 31)

Zero-out crypt key information when no longer in use. (Ben Fritz, 2017 May 15)

Add stronger encryption. Could use libsodium (NaCl).
<https://github.com/jedisct1/libsodium/>
Possibly include the needed code so that it can be built everywhere.

Add a way to restart a timer. It's similar to timer_stop() and timer_start(),
but the reference remains valid.

Need to try out instructions in INSTALLpc.txt about how to install all
interfaces and how to build Vim with them.
Appveyor build with self-installing executable, includes getting most
interfaces: <https://github.com/k-takata/vim/tree/chrisbra-appveyor-build>
result: <https://ci.appveyor.com/project/k-takata/vim/history>

Problem using ":try" inside ":execute". (ZyX, 2013 Sep 15)

Patch to make tests pass with EBCDIC. (Owen Leibman, 2016 Apr 10)

Add ":read :command", to insert the output of an Ex command?
Can already do it with ":\$put =execute('command')".

exists(":tearoff") does not tell you if the command is implemented. (Tony
Mechelynck) Perhaps use exists("::tearoff") to check?

Use vim.vim syntax highlighting for help file examples, but without ":" in
'iskeyword' for syntax.

Installation of .desktop files does not work everywhere.
It's now fixed, but the target directory probably isn't right.
Add configure check?
Should use /usr/local/share/applications or /usr/share/applications.
Or use \$XDG_DATA_DIRS.
Also need to run update-desktop-database (Kuriyama Kazunobu, 2015 Nov 4)

Test object i{ and it do not behave the same. #1379
Do not include the linebreak at the start?

Feature request: add the "al" text object, to manipulate a screen line.
Especially useful when using **'linebreak'**

Patch to avoid redrawing tabline when the popup menu is visible.
(Christian Brabandt, 2016 Jan 28)

When the CursorMovedI event triggers, and **CTRL-X** was typed, a script cannot restore the mode properly. (Andrew Stewart, 2016 Apr 20)
Do not trigger the event?

Patch to make the behavior of "w" more straightforward, but not Vi compatible. With a '**cpo**' flag. (Christian Brabandt, 2016 Feb 8)

Patch to add optionproperties(). (Anton Lindqvist, 2016 Mar 27, update Apr 13)

Patch to add TagNotFound autocommand. (Anton Lindqvist, 2016 Feb 3)

Patch to add Error autocommand. (Anton Lindqvist, 2016 Feb 17)
Only remembers one error.

GVim: when both Tab and **CTRL-I** are mapped, use **CTRL-I** not for Tab.

Unexpected delay when using **CTRL-O** u. It's not timeoutlen.
(Gary Johnson, 2015 Aug 28)

Instead of separately uploading patches to the ftp site, we can get them from github with a URL like this:

<https://github.com/vim/vim/compare/v7.4.920%5E...v7.4.920.diff>

Diff for version.c contains more context, can't skip a patch.

Python: ":py raw_input('prompt')" doesn't work. (Manu Hack)

Comparing nested structures with "==" uses a different comparator than when comparing individual items.

Using uninitialized memory. (Dominique Pelle, 2015 Nov 4)

MS-Windows: When editing a file with a leading space, writing it uses the wrong name. (Aram, 2014 Nov 7) Vim 7.4.

Can't recognize the \$ProgramFiles(x86) environment variable. Recognize it specifically? First try with the parens, then without.

Patch to add :mapgroup, put mappings in a group like augroup.
(Yasuhiro Matsumoto, 2016 Feb 19)

Can we cache the syntax attributes, so that updates for '**relativenumber**' and '**cursorline**'/'**cursorcolumn**' are a lot faster? Thus store the attributes before combining them.

C highlighting: modern C allows: /* comment */ #ifdef
and also line continuation after #include.
I can't recommend it though.

Build with Python on Mac does not always use the right library.
(Kazunobu Kuriyama, 2015 Mar 28)

To support Thai (and other languages) word boundaries, include the ICU library: <http://userguide.icu-project.org/boundaryanalysis>

Patch to use two highlight groups for relative numbers. (Shaun Brady, 2016 Jan 30)

MS-Windows: Crash opening very long file name starting with "\\\". (Christian Brock, 2012 Jun 29)

The OptionSet autocommand event is not always triggered. (Rick Howe, 2015 Sep 24): :diffthis, :diffoff.

":set all&" still does not handle all side effects. Centralize handling side effects for when set by the user, on init and when reset to default.

":tag" does not jump to the right entry of a :tselect. (James Speros, 2015 Oct 9)

The argument for "-S" is not taken literally, the ":so" command expands wildcards. Add a ":nowild" command modifier? (ZyX, 2015 March 4)

Proposal to make options.txt easier to read. (Arnaud Decara, 2015 Aug 5)
Update Aug 14.

When using --remote-tab on MS-Windows '**encoding**' hasn't been initialized yet, the file name ends up encoded wrong. (Raul Coronado, 2015 Dec 21)

Example in editing.txt uses \$HOME with the expectation that it ends in a slash. For me it does, but perhaps not for everybody. Add a function that inserts a slash when needed? pathconcat(dir, path) (Thilo Six, 2015 Aug 12)

ml_updatechunk() is slow when retrying for another encoding. (John Little, 2014 Sep 11)

Patch to fix checking global option value when not using it. (Arnaud Decara, 2015 Jul 23)

When '**showbreak**' is set repeating a Visual operation counts the size of the '**showbreak**' text as part of the operation. (Axel Bender, 2015 Jul 20)

Patch to add grepfile(). (Scott Prager, 2015 May 26)
Work in progress.

Would be useful to have a treemap() or deepmap() function. Like map() but when an item is a list or dict would recurse into it.

Patch for global-local options consistency. (Arnaud Decara, 2015 Jul 22)
Is this right?

Patch to make getregtype() return the right size for non-linux systems. (Yasuhiro Matsumoto, 2014 Jul 8)
Breaks test_eval. Inefficient, can we only compute y_width when needed?

Patch to use different terminal mode settings for system(). (Hayaki Saito)
Does this work for everybody?

Patch for man.vim. (SungHyun Nam, 2015 May 20)
Doesn't work completely (Dominique Orban)

Patch to add a "literal" argument to bufnr(). (Olaf Dabrunz, 2015 Aug 4)

Extended file attributes lost on write (backupcopy=no). Issue 306.
Would require reading attributes from the original file with listxattr() and
getxattr() and adding them to the new file.

Patch to add :lockjumps. (Carlo Baldassi, 2015 May 25)
OK to not block marks?

Patch on Issue 72: '**autochdir**' causes problems for :vimgrep.

When two SIGWINCH arrive very quickly, the second one may be lost.
(Josh Triplett, 2015 Sep 17)

Make comments in the test Makefile silent. (Kartik Agaram, 2014 Sep 24)

Result of systemlist() does not show whether text ended in line break.
(Bjorn Linse, 2014 Nov 27)

When in '**comments**' "n:x" follows after three-part comment directly it repeats
any one-character from the previous line. (Kartik Agaram, 2014 Sep 19)

Patch: Let rare word highlighting overrule good word highlighting.
(Jakson A. Aquino, 2010 Jul 30, again 2011 Jul 2)

Patch to add digits argument to round(). (Yasuhiro Matsumoto, 2015 Apr 26)

Can assign to s:type when a function s:type has been defined.
Also the other way around: define a function while a variable with that name
was already defined.
(Yasuhiro Matsumoto, 2014 Nov 3)

Patch for ordered dict. (Ozaki Kiichi, 2015 May 7)

Patch for building a 32bit Vim with 64bit MingW compiler.
(Michael Soyka, 2014 Oct 15)

Patch: On MS-Windows shellescape() may have to triple double quotes.
(Ingo Karkat, 2015 Jan 16)

Patch for glob(), adding slash to normal files. (Ingo Karkat, 2014 Dec 22)

When entering and leaving the preview window autocommands are triggered, but
these may not work well. Perhaps set a flag to indicate that the preview
window is involved? (John Otter, 2015 Oct 27)

Using "." to repeat an Ex command puts that command in history. Probably
should not happen. If the command is the result of a mapping it's not put in

history either. (Jacob Niehus, 2014 Nov 2)
Patch from Jacob, Nov 2.

"hi link" does not respect groups with GUI settings only. (Mark Lodato, 2014 Jun 8)

Bug: Autocompleting ":tag/pat" replaces "/pat" with a match but does not insert a space. (Micha Mos, 2014 Nov 7)

No error for missing endwhile. (ZyX, 2014 Mar 20)

Patch to make extend() fail early when it might fail at some point.
(Olaf Dabrunz, 2015 May 2) Makes extend() slower, do we still want it?
Perhaps only the checks that can be done without looping over the dict or arguments.

Problem with transparent and matchgroup. Issue #475

Idea: For a window in the middle (has window above and below it), use right-mouse-drag on the status line to move a window up/down without changing its height? It's like dragging the status bar above it at the same time.

Patch to add a :domodeline command. (Christian Brabandt, 2014 Oct 21)

This does not give an error: (Andre Sihera, 2014 Mar 21)

```
vim -u NONE 1 2 3 -c 'bufdo if 1 | echo 1'
```

This neither: (ZyX)

```
vim -u NONE 1 2 3 -c 'bufdo while 1 | echo 1'
```

'viewdir' default on MS-Windows is not a good choice, it's a system directory. Change 'viewdir' to "\$HOME/vimfiles/view" and use 'viewdiralt' to also read from?

Include a plugin manager with Vim? vim-plug seems to be the best currently:

<https://github.com/junegunn/vim-plug>.

Also Vundle: <https://github.com/gmarik/vundle>

Or minpac: <https://github.com/k-takata/minpac>, since it leverages the builtin package feature.

Long message about this from ZyX, 2014 Mar 23. And following replies.

Also see <http://vim-wiki.mawercer.de/wiki/topic/vim%20plugin%20managment.html>

User view:

- Support multiple sources, basically any http:// URL. Or a central place that will work for everybody (github? redirects from vim.org?).
Be able to look into the files before deciding to install.
- Be able to try out a plugin and remove it again with (almost) no traces.
- Each plugin needs a "manifest" file that has the version, dependencies (including Vim version and features), conflicts, list of files, etc.
Updater uses that to decide what/how to update.
Dependencies can use a URL for specific versions, or short name for scripts on vim.org.
- Once a plugin is installed it remembers where it came from, updater checks there. Can manually update when really needed.
- Must be possible to install for one user. Also system wide?
- Can edit plugin config with Vim. Can temporarily disable a plugin.

- Run the update manually, find latest version and install.
- Be able to download without special tools, must work for 95% of users.

Implementation:

- Avoid the `'runtimepath'` getting long. Need some other way to keep each plugin separate.
- When installing or updating, first figure out what needs to be done. This may involve recursively fetching manifest files for dependencies. Then show the user what's going to change and ask for OK.
- Scripts on Vim.org must be able to consist of several files. Is zip format sufficient? Upload the manifest? Or refer to a site that has the manifest?
- Best is to fetch individual files or use a Vimball. Reduces dependency on tools that might be missing and allows inspection of the files before installing.

Out of scope:

- Overview of plugins, ratings, comments, etc. That's another world.
- Development work on plugins (although diff with distributed version would be useful).

When typing the first character of a command, e.g. "f", then using a menu, the menu item doesn't work. Clear typeahead when using a menu?

Editing an ascii file as ucs-2 or ucs-4 causes display errors.
(ZyX, 2014 Mar 30)

":Next 1 some-arg" does not complain about trailing argument. Also for various other commands. (ZyX, 2014 Mar 30)

Patch to skip sort if no line matches the expression.
(Christian Brabandt, 2014 Jun 25)

VMS: Select() doesn't work properly, typing ESC may hang Vim. Use sys\$qiow instead. (Samuel Ferencik, 2013 Sep 28)

Patch for XDG base directory support. (Jean François Bignolles, 2014 Mar 4)
Remark on the docs. Should not be a compile time feature. But then what?
Also see #2034.

Patch to define macros for hardcoded values. (Elias Diem, 2013 Dec 14)

Updated spec ftplugin. (Matěj Cepl, 2013 Oct 16)

Patch to handle integer overflow. (Aaron Burrow, 2013 Dec 12)

7 Windows XP: When using "ClearType" for text smoothing, a column of yellow pixels remains when typing spaces in front of a "D" ('guifont' set to "lucida_console:h8").

Patch by Thomas Tuegel, also for GTK, 2013 Nov 24

:help gives example for z?, but it does not work. m? and t? do work.

Discussion about canonicalization of Hebrew. (Ron Aaron, 2011 April 10)

Checking runtime scripts: Thilo Six, 2012 Jun 6.

When evaluating expression in backticks, autoload doesn't work.
(Andy Wokula, 2013 Dec 14)

Using `<nr>ifoo<esc>` can slow down Vim. Patch by Christian Brabandt, 2013 Dec 13. Only helps a bit, `10000ii<Esc>` is still too slow.

Javascript file where indent gets stuck on: GalaxyMaster, 2012 May 3.

The BufUnload event is triggered when re-using the empty buffer.
(Pokey Rule, 2013 Jul 22)
Patch by Marcin Szamotulski, 2013 Jul 22.

Patch to allow more types in `remote_expr()`. (Lech Lorens, 2014 Jan 5)
Doesn't work for string in list. Other way to pass all types of variables reliably?

Patch to add `{lhs}` to `:mapclear:` clear all maps starting with `{lhs}`.
(Christian Brabandt, 2013 Dec 9)

Exception caused by argument of return is not caught by try/catch.
(David Barnett, 2013 Nov 19)
Bug in try/catch: return with invalid compare throws error that isn't caught.
(ZyX, 2011 Jan 26)
try/catch not working for argument of return. (Matt Wozniski, 2008 Sep 15)
try/catch not working when inside a for loop. (ZyX, 2011 Jan 25)

Patch to fix that `'cedit'` is recognized after `:normal`. (Christian Brabandt, 2013 Mar 19, later message)

Patch to view coverage of the tests. (Nazri Ramliy, 2013 Feb 15)

Patch to add "Q" and "A" responses to interactive `:substitute`. They are carried over when using `:global`. (Christian Brabandt, 2013 Jun 19)

Bug with `'cursorline'` in diff mode. Line being scrolled into view gets highlighted as the cursor line. (Alessandro Ivaldi, 2013 Jun 4)

Two highlighting bugs. (ZyX, 2013 Aug 18)

Patch to support 'u' in interactive substitute. (Christian Brabandt, 2012 Sep 28) With tests: Oct 9.

Dialog is too big on Linux too. (David Fishburn, 2013 Sep 2)

- Add regex for `'paragraphs'` and `'sections'`: `'parare'` and `'sectre'`. Combine the two into a regex for searching. (Ned Konz)

Patch by Christian Brabandt, 2013 Apr 20, unfinished.

Bug: `findfile("any", "file:///tmp;")` does not work.

Patch to add `getsid()`. (Tyru, 2011 Oct 2) Do we want this? Update Oct 4. Or use `expand('<sid>')`?

Patch to make `confirm()` display colors. (Christian Brabandt, 2012 Nov 9)

Problem with refresh:always in completion. (Tyler Wade, 2013 Mar 17)

b:undo_ftplugin cannot call a script-local function. (Boris Danilov, 2013 Jan 7)

Win32: The Python interface only works with one version of Python, selected at compile time. Can this be made to work with version 2.1 and 2.2 dynamically?

Python: Be able to define a Python function that can be called directly from Vim script. Requires converting the arguments and return value, like with vim.bindeval().

Patch for :tabcloseleft, after closing a tab go to left tab. (William Bowers, 2012 Aug 4)

Patch to improve equivalence classes in regexp patterns. (Christian Brabandt, 2013 Jan 16, update Jan 17)

Patch with suggestions for starting.txt. (Tony Mechelynck, 2012 Oct 24)
But use Gnome instead of GTK?

Should be possible to enable/disable matchparen per window or buffer. Add a check for b:no_match_paren in Highlight_matching_Pair() (Marcin Szamotulski, 2012 Nov 8)

'**iminsert**' global value set when using ":setlocal iminsert"? (Wu, 2012 Jun 23)

Patch to append regexp to tag commands to make it possible to select one out of many matches. (Cody Cutler, 2013 Mar 28)

The input map for **CTRL-O** in mswin.vim causes problems after **CTRL-X CTRL-O**. Suggestion for another map. (Philip Mat, 2012 Jun 18)
But use "gi" instead of "a". Or use CTRL-\ **CTRL-O**.

When there are no command line arguments ":next" and ":argu" give E163, which is confusing. Should say "the argument list is empty".

URXVT:

- will get stuck if byte sequence does not contain the expected semicolon.
- Use urxvt mouse support also in xterm. Explanations:
<http://www.midnight-commander.org/ticket/2662>

Patch to add tests for if_xcmdsrv.c., Jul 8, need some more work. (Brian Burns)
New tests Jul 13. Update Jul 17. Discussion Jul 18.

Patch for input method status. (Hirohito Higashi, 2012 Apr 18)

Update Vim app icon (for Gnome). (Jakub Steiner, 2013 Dec 6)

Patch to use .png icons for the toolbar on MS-Windows. (Martin Giesekeing, 2013 Apr 18)

Patch for has('unnamedplus') docs. (Tony Mechelynck, 2011 Sep 27)

And one for gui_x11.txt.

":cd" doesn't work when current directory path contains "**".
finddir() has the same problem. (Yukihiro Nakadaira, 2012 Jan 10)
Requires a rewrite of the file_file_in_path code.

Should use has("browsefilter") in ftplugins. Requires patch 7.3.593.

Update for vim2html.pl. (Tyru, 2013 Feb 22)

Patch to sort functions starting with '<' after others. Omit dict functions,
they can't be called. (Yasuhiro Matsumoto, 2011 Oct 11)

Patch to pass list to or(), and() and xor(). (Yasuhiro Matsumoto, 2012 Feb 8)

Patch to improve "it" and "at" text object matching. (Christian Brabandt, 2011
Nov 20)

Patch to improve GUI find/replace dialog. (Christian Brabandt, 2012 May 26)
Update Jun 2.

`] moves to character after insert, instead of the last inserted character.
(Yukihiro Nakadaira, 2011 Dec 9)

Plugin for Modeleasy. (Massimiliano Tripoli, 2011 Nov 29)

BufWinLeave triggers too late when quitting last window in a tab page. (Lech
Lorens, 2012 Feb 21)

Patch for **'transparency'** option. (Sergiu Dotenco, 2011 Sep 17)
Only for MS-Windows. No documentation. Do we want this?

Patch to support cursor shape in Cygwin console. (Ben bgold, 2011 Dec 27)

On MS-Windows a temp dir with a & init causes system() to fail. (Ben Fritz,
2012 Jun 19)

'cursorline' is displayed too short when there are concealed characters and
'list' is set. (Dennis Preiser)

Patch 7.3.116 was the wrong solution.

Christian Brabandt has another incomplete patch. (2011 Jul 13)

Win32: Patch to use task dialogs when available. (Sergiu Dotenco, 2011 Sep 17)
New feature, requires testing. Made some remarks.

Win32: Patch for alpha-blended icons and toolbar height. (Sergiu Dotenco, 2011
Sep 17) Asked for feedback from others.

Win32: Cannot cd into a directory that starts with a space. (Andy Wokula, 2012
Jan 19)

Need to escape \$HOME on Windows for fnameescape()? (ZyX, 2011 Jul 21,
discussion 2013 Jul 4) Can't simply use a backslash, \ \$HOME has a different
meaning already. Would be possible to use \$\$HOME where \$HOME is to be used.

"2" in **'formatoptions'** not working in comments. (Christian Corneliussen, 2011 Oct 26)

Bug in repeating Visual "u". (Lawrence Kesteloot, 2010 Dec 20)

Windows keys not set properly on Windows 7? (cncyber, 2010 Aug 26)

When using a Vim server, a # in the path causes an error message. (Jeff Lanzarotta, 2011 Feb 17)

When there is a ">" in a line that "gq" wraps to the start of the next line, then the following line will pick it up as a leader. Should get the leader from the first line, not a wrapped line. (Matt Ackeret, 2012 Feb 27)

Using ":break" or something else that stops executing commands inside a ":finally" does not rethrow a previously uncaught exception. (ZyX, 2010 Oct 15)

Vim using lots of memory when joining lines. (John Little, 2010 Dec 3)

BT regexp engine: After trying a \@> match and failing, submatches are not cleared. See test64.

On 64 bit MS-Windows "long" is only 32 bits, but we sometimes need to store a 64 bits value. Change all number options to use nropt_T and define it to the right type.

string() can't parse back "inf" and "nan". Fix documentation or fix code? (ZyX, 2010 Aug 23)

When doing "redir => s:foo" in a script and then "redir END" somewhere else (e.g. in a function) it can't find s:foo.

When a script contains "redir => s:foo" but doesn't end redirection, a following "redir" command gives an error for not being able to access s:foo. (ZyX, 2011 Mar 27)

When setqflist() uses a filename that triggers a BufReadCmd autocommand Vim doesn't jump to the correct line with :cfirst. (ZyX, 2011 Sep 18)

Behavior of i" and a" text objects isn't logical. (Ben Fritz, 2013 Nov 19)

When setting a local option value from the global value, add a script ID that indicates this, so that ":verbose set" can give a hint. Check with options in the help file.

After patch 7.3.097 still get E15. (Yukihiro Nakadaira, 2011 Jan 18)
Also for another example (ZyX, 2011 Jan 24)

"0g@\$" puts ']' on last byte of multibyte. (ZyX, 2011 Jan 22)

Patch for :tabrecently. (Hirokazu Yoshida, 2012 Jan 30)

Problem with "syn sync grouphere". (Gustavo Niemeyer, 2011 Jan 27)

Loading autoload script even when usage is inside "if 0". (Christian Brabandt, 2010 Dec 18)

With a filler line in diff mode, it isn't displayed in the column with line number, but it is in the sign column. Doesn't look right. (ZyX 2011 Jun 5)
Patch by Christian Brabandt, 2011 Jun 5. Introduces new problems.

Add jump() function. (Marcin Szamotulski, 2013 Aug 29)
Is this needed? **CTRL-O** and **CTRL-I** do the same, just more difficult to use.

8 Add a command to jump to the next character highlighted with "Error".
Patch by Christian Brabandt, uses]e [e]t and [t. 2011 Aug 9.

Add event for when the text scrolls. A bit like CursorMoved. Also a similar one for insert mode. Use the event in matchparen to update the highlight if the match scrolls into view.

7 Use "++--", "+++--" for different levels instead of "+----" "+-----".
Patch by Christian Brabandt, 2011 Jul 27.
Update by Ben Fritz, with fix for TOhtml. (2011 Jul 30)

9 Add %F to '**errorformat**': file name without spaces. Useful on Unix to avoid matching something up to a time 11:22:33.
Patch by Christian Brabandt, 2011 Jul 27.

Patch to add up to 99 match groups. (Christian Brabandt, 2010 Dec 22)
Also add named groups: \#{name}(re) and \#{name}g

In the sandbox it's not allowed to do many things, but it's possible to change or set variables. Add a way to prevent variables from being changed in the sandbox? E.g.: ":protect g:restore_settings".

Win32: tear-off menu does not work when menu language is German. (Markus Bossler, 2011 Mar 2) Fixed by 7.3.095?

Wish for NetBeans commands:

- make it possible to have 'defineAnnoType' also handle terminal colors.

7.3.014 changed how backslash at end of line works, but still get a NUL when there is one backslash. (Ray Frush, 2010 Nov 18) What does the original ex do?

New esperanto spell file can't be processed. (Dominique Pelle, 2011 Jan 30)
- move compflags to separate growarray?
- instead of a regexp use a hashtable. Expand '?', '*', '+'. What would be the maximum repeat for * and +?

"L'Italie" noted as a spell error at start of the sentence. (Dominique Pelle, 2011 Feb 27)

Editing a file with a ^M with '**ff**' set to "mac", opening a help file, then the ^M is displayed as ^J sometimes. Getting '**ff**' value from wrong window/buffer?

When Vim is put in the background (SIGTSTP) and then gets a SIGHUP it doesn't exit. It exists as soon as back in the foreground. (Stephen Liang, 2011 Jan 9) Caused by vim_handle_signal(SIGNAL_BLOCK); in ui.c.

g` not working correctly when using :edit. It works OK when editing a file on the command line. (Ingo Karkat, 2011 Jan 25)

Since patch 7.2.46 Yankring plugin has become very slow, eventually make Vim crash? (Raiwil, 2010 Nov 17)

Regexp engine performance:

- Profiling:

```
./vim -u NONE -s ~/vim/test/ruby.vim
./vim -u NONE -s ~/vim/test/loop.vim
./vim -u NONE -s ~/vim/test/alsa.vim
./vim -s ~/vim/test/todo.vim
./vim -s ~/vim/test/xml.vim
```

Dominique Pelle: xmlSyncDT is particularly slow (Jun 7)

- More test files from the src/pkg/regexp/testdata directory in the Go repo.

- Performance tests:

- Using asciidoc syntax. (Marek Schimara, 2013 Jun 6)

- ~/vim/text/FeiqCfg.xml (file from Netjune)

- ~/vim/text/edl.svg (also XML)

- glts has five tests. (May 25)

- ~/vim/test/slowsearch

- ~/vim/test/rgb.vim

- search for a.*e*exn in the vim executable. Go to last line to use **'hlsearch'**.

- Slow combination of folding and PHP syntax highlighting. Script to reproduce it. Caused by "syntax sync fromstart" in combination with patch 7.2.274. (Christian Brabandt, 2010 May 27) Generally, folding with **'foldmethod'** set to "syntax" is slow. Do profiling to find out why.

Problem producing tags file when hebrew.frx is present. It has a BOM. Results in E670. (Tony Mechelynck, 2010 May 2)

'beval' option should be global-local.

Ruby: ":ruby print \$buffer.number" returns zero.

setpos() does not restore cursor position after :normal. (Tyru, 2010 Aug 11)

With "tw=55 fo+=a" typing space before) doesn't work well. (Scott Mcdermott, 2010 Oct 24)

Messages in message.txt are highlighted as examples.

When using cp850 the NBSP (0xff) is not drawn correctly. (Brett Stahlman, 2010 Oct 22) **'isprint'** is set to "@,161-255".

":echo "\x85" =~# '[\u0085]'" returns 1 instead of 0. (ZyX, 2010 Oct 3)

'cindent' not correct when **'list'** is set. (Zdravi Korusef, 2010 Apr 15)

C-indenting: A matching { in a comment is ignored, but intermediate { are not checked to be in a comment. Implement FM_SKIPCOMM flag of findmatchlimit(). Issue 46.

Using CompilerSet doesn't record where an option was set from. E.g., in the gcc compiler plugin. (Gary Johnson, 2010 Dec 13)

":helpgrep" does not put the cursor in the correct column when preceded by accented character. (Tony Mechelynck, 2010 Apr 15)

Don't call check_restricted() for histadd(), setbufvar(), settabvar(), setwinvar().

Patch for gVimExt to show an icon. (Dominik Riebeling, 2010 Nov 7)

When 'lines' is 25 and 'scrolloff' is 12, "j" scrolls zero or two lines instead of one. (Constantin Pan, 2010 Sep 10)

Gui menu edit/paste in block mode insert only inserts in one line (Bjorn Winckler, 2011 May 11)

Requires a map mode for Insert mode started from blockwise Visual mode.

Problem with cursor in the wrong column. (SungHyun Nam, 2010 Mar 11)
Additional info by Dominique Pelle. (also on 2010 Apr 10)

CreateFile and CreateFileW are used without sharing, filewritable() fails when the file was already open (e.g. script is being sourced). Add FILE_SHARE_READ|FILE_SHARE_WRITE in mch_access()? (Philippe Vaucher, 2010 Nov 2)

Is ~/bin (literally) in \$PATH supposed to work? (Paul, 2010 March 29)
Looks like only bash can do it. (Yakov Lerner)

Cscope "cs add" stopped working somewhat before 7.2.438. (Gary Johnson, 2010 Jun 29) Caused by 7.2.433?

Jumplist doesn't work properly in Insert mode? (Jean Johner, 2010 Mar 20)

Problem with transparent cmdline. Also: Terminal title is wrong with non-ASCII character. (Lily White, 2010 Mar 7)

iconv() doesn't fail on an illegal character, as documented. (Yongwei Wu, 2009 Nov 15, example Nov 26) Add argument to specify whether iconv() should fail or replace with a character and continue?

Add local time at start of --startuptime output.
Requires configure check for localtime().

Use format year-month-day hr:min:sec.

Patch to make ":hi link" also take arguments. (Nate Soares, 2012 Dec 4)

Shell not recognized properly if it ends in "csh -f". (James Vega, 2009 Nov 3)
Find tail? Might have a / in argument. Find space? Might have space in path.

Test 51 fails when language set to German. (Marco, 2011 Jan 9)
Dominique can't reproduce it.

'ambiwidth' should be global-local.

":function f(x) keepjumps" creates a function where every command is executed like it has ":keepjumps" before it.

Coverity: Check if there are new reported defects:
<https://scan.coverity.com/projects/241>

Problem with editing file in binary mode. (Ingo Krabbe, 2009 Oct 8)

Display error when 'tabline' that includes a file name with double-width characters. (2010 Aug 14, bootleg)

Problem with stop directory in findfile(). (Adam Simpkins, 2009 Aug 26)

Using ']' as the end of a range in a pattern requires double escaping:
/[[@-\]]] (Andy Wokula, 2011 Jun 28)

For running gvim on a USB stick: avoid the OLE registration. Use a command line argument -noregister.

When using an expression in 'statusline' leading white space sometimes goes missing (but not always). (ZyX, 2010 Nov 1)

When a mapping exists both for insert mode and lang-insert mode, the last one doesn't work. (Tyru, 2010 May 6) Or is this intended?

Still a problem with ":make" in the wrong directory. Caused by ":bufdo". (Ajit Thakkar, 2009 Jul 1) More information Jul 9, Jul 15.
Caused by "doautoall syntaxset BufEnter *" in syntax/nosyntax.vim ?
There also is a BufLeave/BufEnter aucmd to save/restore view.
Does the patch to save/restore globaldir work?

":bufdo normal gg" while 'hidden' is set leaves buffers without syntax highlighting. Don't disable Syntax autocommands then? Or add a flag/modifier to avoid changing 'eventignore'?

Patch for displaying 0x200c and 0x200d. (Ali Gholami Rudi, 2009 May 6)
Probably needs a bit of work.

List of encoding aliases. (Takao Fujiwara, 2009 Jul 18)
Are they all OK? Update Jul 22.

Win32: Improved Makefile for MSVC. (Leonardo Valeri Manera, 2010 Aug 18)

Win32: Expanding 'path' runs into a maximum size limit. (bgold12, 2009 Nov 15)

Win32: Patch for using .png files for icons. (Charles Peacech, 2012 Feb 5)

Putting a Visual block while 'visualedit' is "all" does not leave the cursor on the first character. (John Beckett, 2010 Aug 7)

Setting **'tags'** to "tagsdir/*" does not find "tagsdir/tags". (Steven K. Wong, 2009 Jul 18)

Patch to add "focusonly" to **'scrollopt'**, so that scrollbind also applies in window that doesn't have focus. (Jonathon Mah, 2009 Jan 12)
Needs more work.

Problem with `<script>` mappings (Andy Wokula, 2009 Mar 8)

When starting Vim with "gvim -f -u non_existent_file > foo.txt" there are a few control characters in the output. (Dale Wiles, 2009 May 28)

'cmdwinheight' is only used in last window when **'winheight'** is a large value. (Tony Mechelynck, 2009 Apr 15)

Status line containing winnr() isn't updated when splitting the window (Clark J. Wang, 2009 Mar 31)

When \$VIMRUNTIME is set in .vimrc, need to reload lang files. Already done for GTK, how about others? (Ron Aaron, 2010 Apr 10)

":tab split fname" doesn't set the alternate file in the original window, because win_valid() always returns FALSE. Below win_new_tabpage() in ex_docmd.c.

Space before comma in function definition not allowed: "function x(a , b)"
Give a more appropriate error message. Add a remark to the docs.

string_convert() should be able to convert between utf-8 and utf-16le. Used for GTK clipboard. Avoid requirement for iconv.

Now that colnr_T is int instead of unsigned, more type casts can be removed.

Don't load macmap.vim on startup, turn it into a plugin. (Ron Aaron, 2009 Apr 7) Reminder Apr 14.

Add "no_hlsearch" to winsaveview().

Bug: When reloading a buffer changed outside of Vim, BufRead autocommands are applied to the wrong buffer/window. (Ben Fritz, 2009 Apr 2, May 11)
Ignore window options when not in the right window?
Perhaps we need to use a hidden window for applying autocommands to a buffer that doesn't have a window.

When using "ab foo bar" and mapping `<Tab>` to `<Esc>`, pressing `<Tab>` after foo doesn't trigger the abbreviation like `<Esc>` would. (Ramana Kumar, 2009 Sep 6)

getbufvar() to get a window-local option value for a buffer that's not displayed in a window should return the value that's stored for that buffer.

":he ctrl_u" can be auto-corrected to ":he ctrl-u".

Diff mode out of sync. (Gary Johnson, 2010 Aug 4)

Win32: completion of file name ":e c:\!test" results in ":e c:\\!test", which does not work. (Nieko Maatjes, 2009 Jan 8, Ingo Karkat, 2009 Jan 22)

Using ~ works OK on 'a' with composing char, but not on 0x0418 with composing char 0x0301. (Tony Mechelynck, 2009 Mar 4)

This does not work yet: "a\(%C\)" (get composing characters into a submatch).

Inconsistent: starting with \$LANG set to es_ES.utf-8 gives Spanish messages, even though locale is not supported. But ":lang messages es_ES.utf-8" gives an error and doesn't switch messages. (Dominique Pelle, 2009 Jan 26)

When \$HOME contains special characters, such as a comma, escape them when used in an option. (Michael Hordijk, 2009 May 5)

Turn "esc" argument of expand_env_esc() into string of chars to be escaped.

Should make 'ignorecase' global-local, so that it makes sense setting it from a modeline.

Add cscope target to Makefile. (Tony Mechelynck, 2009 Jun 18, replies by Sergey Khorev)

Completion for ":buf" doesn't work properly on Win32 when 'shellslash' is off. (Henrik Ohman, 2009, Jan 29)

shellescape() depends on 'shellslash' for quoting. That doesn't work when 'shellslash' is set but using cmd.exe. (Ben Fritz)

Use a different option or let it depend on whether 'shell' looks like a unix-like shell?

Bug: in Ex mode (after "Q") backslash before line break, when yanked into a register and executed, results in <Nul>: instead of line break.

(Konrad Schwarz, 2010 Apr 16)

Have a look at patch for utf-8 line breaking. (Yongwei Wu, 2008 Mar 1, Mar 23)
Now at: <http://vimgadgets.sourceforge.net/liblinebreak/>

Greek sigma character should be lower cased depending on the context. Can we make this work? (Dominique Pelle, 2009 Sep 24)

When changing 'encoding' convert all the swap file names, so that we can still delete them. Also convert all buffer file names?

"ggip" in Insert mode has an off-by-one error, causing it to reflow text. (Raul Coronado, 2009 Nov 2)

MS-Windows: editing the first, empty buffer, 'ffs' set to "unix,dos", ":enew" doesn't set 'ff' to "unix". (Ben Fritz, 2008 Dec 5) Reusing the old buffer probably causes this.

'scrollbind' is not respected when deleting lines or undo. (Milan Vancura, 2009 Jan 16)

Having "Syntax" in **'eventignore'** for :bufdo may cause problems, e.g. for ":bufdo e" when buffers are open in windows. ex_listdo(eap) could set the option only for when jumping to another buffer, not when the command argument is executed.

":pedit %" with a BufReadPre autocommand causes the cursor to move to the first line. (Ingo Karkat, 2008 Jul 1) Ian Kelling is working on this. Similar problem with ":e". (Marc Montu, 2014 Apr 22)

Cursor line moves in other window when using **CTRL-W J** that doesn't change anything. (Dasn, 2009 Apr 7)

On Unix "glob('does not exist~')" returns the string. Without the "~" it doesn't. (John Little, 2008 Nov 9)
Shell expansion returns unexpanded string?
Don't use shell when "~" is not at the start?

When using ":e ++enc=foo file" and the file is already loaded with **'fileencoding'** set to "bar", then do_ecmd() uses that buffer, even though the fileencoding differs. Reload the buffer in this situation? Need to check for the buffer to be unmodified.
Unfinished patch by Ian Kelling, 2008 Jul 11. Followup Jul 14, need to have another look at it.

c.vim: XXX in a comment is colored yellow, but not when it's after "#if 0". (Ilya Dogolazky, 2009 Aug 7)

You can type ":w ++bad=x fname", but the ++bad argument is ignored. Give an error message? Or is this easy to implement? (Nathan Stratton Treadway, 2008 Aug 20) This is in ucs2bytes(), search for 0xBF. Using the ++bad argument is at the other match for 0xBF.

When adding "--complete=file" to a user command this also changes how the argument is processed for **<f-args>**. (Ivan Tishchenko, 2008 Aug 19)

Win32: associating a type with Vim doesn't take care of space after a backslash? (Robert Vibrant, 2008 Jun 5)

When **'rightleft'** is set, cursorcolumn isn't highlighted after the end of a line. It's also wrong in folds. (Dominique Pelle, 2010 Aug 21)

":help s/~" jumps to *s/~*, while ":help s/\~" doesn't find anything. (Tim Chase) Fix by Ian Kelling, 2008 Jul 14.

When mapping : to ; and ; to :, @; doesn't work like @: and @: doesn't work either. Matt Wozniski: nv_at() calls do_execreg() which uses put_in_typebuf(). Char mapped twice?

Despite adding save_subexpr() this still doesn't work properly:
Regexp: matchlist('12a4aaa', '^(\.{-})\(\%5c\@<=a\+\)\(.\+\)\?')
Returns ['12a4', **'aaa'**, '4aaa'], should be ['12a4', **'aaa'**, '']
Backreference not cleared when retrying after \@<= fails?
(Brett Stahlman, 2008 March 8)

Problem with `remote_send()`. (Charles Campbell, 2008 Aug 12)

`ftplugin` for help file should set `'isk'` to help file value.

Win32: remote editing fails when the current directory name contains "[".
(Ivan Tishchenko, Liu Yubao) Suggested patch by Chris Lubinski: Avoid escaping characters where the backslash is not removed later. Asked Chris for an alternate solution, also for `src/ex_getln.c`.

This also fails when the file or directory name contains "%". (Thoml, 2008 July 7)

Using `--remote-silent` while the current directory has a # in the name does not work, the # needs to be escaped. (Tramblay Bruno, 2012 Sep 15)

When using `remote-silent` the `-R` flag is not passed on. (Axel Bender, 2012 May 31)

Win32: A `--remote` command that has a directory name starting with a (doesn't work, the backslash is removed, assuming that it escapes the (. (Valery Kondakoff, 2009 May 13)

Win32: Using `"gvim --remote-tab-silent elšuti.txt"` doesn't work, the multibyte character isn't passed and edits `elsuti.txt`.
(Raúl Núñez de Arenas Coronado, 2015 Dec 18)

Problem with **CTRL-F**. (Charles Campbell, 2008 March 21)
Only happens with `"gvim -geometry "160x26+4+27" -u NONE -U NONE prop.c"`.
`'lines'` is 54. (2008 March 27)

Problem with pointer wrapping around in `getvcol()`. (Wolfgang Kroworsch, 2008 Oct 19) Check for `"col"` being `"MAXCOL"` separately?

Unexpectedly inserting a double quote. (Anton Woellert, 2008 Mar 23)
Works OK when `'cmdheight'` is 2.

8 Use a mechanism similar to omni completion to figure out the kind of tab for **CTRL-]** and jump to the appropriate matching tag (if there are several).

The utf class table is missing some entries:

- 0x2212, minus sign
- 0x2217, star
- 0x2500, bar
- 0x26ab, circle

Visual line mode doesn't highlight properly when `'showbreak'` is used and the line doesn't fit. (Dasn, 2008 May 1)

GUI: In Normal mode can't yank the modeless selection. Make `"gy"` do this?
Works like **CTRL-Y** in Command line mode.

C't: On utf-8 system, editing file with umlaut through Gnome results in URL with `%nn%nn`, which is taken as two characters instead of one.
Try to reproduce at work.

Patch for default choice in file changed dialog. (Bjorn Winckler, 2008 Oct 19)
Is there a way to list all the files first?

Fail to edit file after failed register access. Error flag remains set?
(Lech Lorens, 2010 Aug 30)

Problem with **'ts'** set to 9 and **'showbreak'** to ">>>". (Matthew Winn, 2007 Oct 1)

":tab help" always opens a new tab, while ":help" re-uses an existing window.
Would be more consistent when an existing tab is re-used. (Tony Mechelynck)

Using Aap to build Vim: add remarks about how to set personal preferences.
Example on <http://www.calmar.ws/tmp/aap.html>

When **'diffopt'** has "context:0" a single deleted line causes two folds to merge
and mess up syncing. (Austin Jennings, 2008 Jan 31)

Gnome improvements: Edward Catmur, 2007 Jan 7
Also use Save/Discard for other GUIs

New PHP syntax file, use it? (Peter Hodge)

":echoe" in catch block stops processing, while this doesn't happen outside of
a catch block. (ZyX, 2011 Jun 2)

Test 54 uses shell commands, that doesn't work on non-Unix systems. Use some
other way to test buffer-local autocommands.

The documentation mentions the priority for ":2match" and ":3match", but it
appears the last one wins. (John Beckett, 2008 Jul 22) Caused by adding
matchadd()? Suggested patch by John, 2008 Jul 24.

When **'encoding'** is utf-8 the command line is redrawn as a whole on every
character typed. (Tyler Spivey, 2008 Sep 3) Only redraw cmdline for
'arabicshape' when there is a character on the command line for which
(ARABIC_CHAR(u8c)) is TRUE.

Cheng Fang made javacomplete. (2007 Aug 11)
Asked about latest version: 0.77.1 is on www.vim.org.

More AmigaOS4 patches. (Peter Bengtsson, Nov 9)

Amiga patches with vbcc. (Adrien Destugues, 2010 Aug 30)
http://pulkomandy.ath.cx/drop/vim73_vbcc_amiga.diff

Problem with compound words? (Bert, 2008 May 6)
No warning for when flags are defined after they are used in an affix.

Screen redrawing when continuously updating the buffer and resizing the
terminal. (Yakov Lerner, 2006 Sept 7)

Add option settings to help ftplugin. (David Eggum, 2006 Dec 18)

Autoconf problem: when checking for iconv library we may add `-L/usr/local/lib`, but when compiling further tests `-liconv` is added without the `-L` argument, that may fail (e.g., `sizeof(int)`). (Blaine, 2007 Aug 21)

Problem with ".add" files when using two languages and restarting Vim. (Raul Coronado, 2008 Oct 30)

Popup menu redraw: Instead of first redrawing the text and then drawing the popup menu over it, first draw the new popup menu, remember its position and size and then redraw the text, skipping the characters under the popup menu. This should avoid flicker. Other solution by A.Politz, 2007 Aug 22.

When the file name has parenthesis, e.g., "foo (bar).txt", `!!:ls '%'` has the parenthesis escaped but not the space. That's inconsistent. Either escape neither or both. No escaping might be best, because it doesn't depend on particularities of the shell. (Zvi Har'El, 2007 Nov 10) (Teemu Likonen, 2008 Jun 3)

However, for backwards compatibility escaping might be necessary. Check if the user put quotes around the expanded item?

A throw in a function causes missing an `endif` below the call. (Spiros Bousbouras, 2011 May 16)

Error E324 can be given when a cron script has wiped out our temp directory. Give a clear error message about this (and tell them not to wipe out `/tmp`).

Color for `cUserLabel` should differ from case label, so that a mistake in a switch list is noticed:

```
switch (i)
{
case 1:
foobar:
}
```

Look at <http://www.gtk-server.org/> . It has a Vim script implementation.

Netbeans problem. Use `"nc -l 127.0.0.1 55555"` for the server, then run `gvim` with `"gvim -nb:localhost:55555:foo"`. From `nc` do: `'1:editFile!0 "foo"`. Then go to Insert mode and add a few lines. Then backspacing every other time moves the cursor instead of deleting. (Chris Kaiser, 2007 Sep 25)

Windows installer could add a "open in new tab of existing Vim" menu entry. GvimExt: patch to add "Edit with single Vim &tabbed" menu entry.

Just have two choices, always using one Vim and selecting between using an argument list or opening each file in a separate tab.

(Erik Falor, 2008 May 21, 2008 Jun 26)

Windows installer: licence text should not use indent, causes bad word wrap. (Benjamin Fritz, 2010 Aug 16)

Changes for Win32 makefile. (Mike Williams, 2007 Jan 22, Alexei Alexandrov, 2007 Feb 8)

Win32: Can't complete shell command names. Why is setting xp_context in set_one_cmd_context() inside #ifndef BACKSLASH_IN_FILENAME?

Win32: Patch for cscope external command. (Mike Williams, 2007 Aug 7)

Win32: XPM support only works with path without spaces. Patch by Mathias Michaelis, 2006 Jun 9. Another patch for more path names, 2006 May 31. New version: <http://members.tcnet.ch/michaelis/vim/patches.zip> (also for other patches by Mathias, see mail Feb 22)

Win32: compiling with normal features and OLE fails. Patch by Mathias Michaelis, 2006 Jun 4.

Win32: using CTRL-S in Insert mode doesn't remove the "+" from the tab pages label. (Tsakiridis, 2007 Feb 18) Patch from Ian Kelling, 2008 Aug 6.

Win32: using "gvim --remote-tab-silent fname" sometimes gives an empty screen with the more prompt. Caused by setting the guitablabel? (Thomas Michael Engelke, 2007 Dec 20 - 2008 Jan 17)

Win32: patch for fullscreen mode. (Liushaolin, 2008 April 17)

Win32: When 'shell' is bash shellescape() doesn't always do the right thing. Depends on 'shellslash', 'shellquote' and 'shellxquote', but shellescape() only takes 'shellslash' into account.

Menu item that does "xxd -r" doesn't work when 'fileencoding' is utf-16. Check for this and use iconv? (Edward L. Fox, 2007 Sep 12)
Does the conversion in the other direction work when 'fileencodings' is set properly?

Cursor displayed in the wrong position when using 'numberwidth'. (James Vega, 2007 Jun 21)

When \$VAR contains a backslash expand('\$VAR') removes it. (Teemu Likonen, 2008 Jun 18)

C++ indenting wrong with "=". (James Kanze, 2007 Jan 26)

":lockvar" should use copyID to avoid endless loop.

When using --remote-silent and the file name matches 'wildignore' get an E479 error. without --remote-silent it works fine. (Ben Fritz, 2008 Jun 20)

GVim: dialog for closing Vim should check if Vim is busy writing a file. Then use a different dialog: "busy saving, really quit? yes / no".

Check other interfaces for changing curbuf in a wrong way. Patch like for if_ruby.c.

":helpgrep" should use the directory from 'helpfile'.

The need_fileinfo flag is messy. Instead make the message right away and put it in keep_msg?

Editing a file remotely that matches **'wildignore'** results in a "no match" error. Should only happen when there are wildcards, not when giving the file name literally, and esp. if there is only one name.

Test 61 fails sometimes. This is a timing problem: "sleep 2" sometimes takes longer than 2 seconds.

Using `":au CursorMoved * cmd"` invokes `mch_FullName()`, which can be slow. Can this be avoided? (Thomas Waba, 2008 Aug 24)
Also for `":w"` without a file name.
The buffer has the full path in `ffname`, should pass this to the autocommand.

`"vim -C"` often has **'nocompatible'**, because it's set in some startup script. Set **'compatible'** after startup is done? Patch by James Vega, 2008 Feb 7.

VMS: while editing a file found in complex, Vim will save file into the first directory of the path and not to the original location of the file.
(Zoltan Arpadffy)

VMS: VFC files are in some cases truncated during reading (Zoltan Arpadffy)

`input()` completion should not insert a backslash to escape a space in a file name?

Ruby completion is insecure. Can this be fixed?

When **'backupskip'** is set from `$TEMP` special characters need to be escaped. (patch by Grembowietz, 2007 Feb 26, not quite right)
Another problem is that `file_pat_to_reg_pat()` doesn't recognize `"\"`, so `"\"` will be seen as a path separator plus `"\"`.

`gvim d:\path\path\(\FILE).xml` should not remove the `\` before the `(`. This also fails with `--remote`.

When doing `":quit"` the Netbeans "killed" event isn't sent. (Xavier de Gaye, 2008 Nov 10) call `netbeans_file_closed()` at the end of `buf_freeall()`, or in all places where `buf_freeall()` is called?

`aucmd_prebuf()` should also use a window in another tab page.

When unloading a buffer in a `BufHidden` autocommand the hidden flag is reset? (Bob Hiestand, 2008 Aug 26, Aug 27)

Substituting an area with a line break with almost the same area does change the Visual area. Can this be fixed? (James Vega, 2006 Sept 15)

GUI: When combining `fg en bg` make sure they are not equal.

Use different `pt_br` dictionary for spell checking. (Jackson A. Aquino, 2006 Jun 5)

Use different romanian dictionary for spell checking. (Andrei Popescu, Nov 2008) Use http://downloads.sourceforge.net/rospell/ro_R0.3.2.zip

Or the hunspell-ro.3.2.tar.gz file, it also has a iso-8859-2 list.

In a C file with spell checking, in "% integer" "nteger" is seen as an error, but "]"s" doesn't find it. "nteger" by itself is found. (Ralf Wildenhues, 2008 Jul 22)

There should be something about spell checking in the user manual.

Add an option to specify the character to use when a double-width character is moved to the next line. Default '>', set to a space to blank it out. Check that char is single width when it's set (compare with '**listchars**').

The generated vim.bat can avoid the loop for NT. (Carl Zmola, 2006 Sep 3)

When showing a diff between a non-existent file and an existing one, with the cursor in the empty buffer, the other buffer only shows the last line. Change the "insert" into a change from one line to many? (Yakov Lerner, 2008 May 27)

These two abbreviations don't give the same result:

```
let asdfasdf = "xyz\<Left>"
cabbr XXX <C-R>=asdfasdf<CR>
cabbr YYY xyz<Left>
```

Michael Dietrich: maximized gvim sometimes displays output of external command partly. (2006 Dec 7)

In FileChangedShell command it's no longer allowed to switch to another buffer. But the changed buffer may differ from the current buffer, how to reload it then?

For Aap: include a config.arg.example file with hints how to use config.arg.

Default for '**background**' is wrong when using xterm with 256 colors. Table with estimates from Matteo Cavalleri, 2014 Jan 10.

Setting '**background**' resets the Normal background color:

```
highlight Normal ctermbg=DarkGray
set background=dark
```

This is undesired, '**background**' is supposed to tell Vim what the background color is, not reset it.

Completion menu: For a wrapping line, completing a long file name, only the start of the path is shown in the menu. Should move the menu to the right to show more text of the completions. Shorten the items that don't fit in the middle?

Accessing file#var in a function should not need the g: prepended.

When exiting detects a modified buffer, instead of opening the buffer in the current tab, use an existing tab, if possible. Like finding a window where the buffer is displayed. (Antonios Tsakiridis)

When ":cn" moves to an error in the same line the message isn't shortened. Only skip shortening for ":cc"?

Problem with `:"call"` and dictionary function. Hari Krishna Dara, Charles Campbell 2006 Jul 06.

A custom completion function in a `:"command"` cannot be a Funcref. (Andy Wokula, 2007 Aug 25)

Problem with using `:redir` in user command completion function? (Hari Krishna Dara, 2006 June 21)

Another resizing problem when setting `'columns'` and `'lines'` to a very large number. (Tony Mechelynck, 2007 Feb 6)

After starting Vim, using `'0` to jump somewhere in a file, `:"sp"` doesn't center the cursor line. It works OK after some other commands.

Win32: Is it possible to have both postscript and Win32 printing?

Using UTF-8 character with `:"command"` does not work properly. (Matt Wozniski, 2008 Sep 29)

In the Netbeans interface add a `"vimeval"` function, so that the other side can check the result of `has("patch13")`.

Cursor line at bottom of window instead of halfway after saving view and restoring. Only with `'nowrap'`. (Robert Webb, 2008 Aug 25)

Netrw has trouble executing autocommands only for a directory. Add `<isdir>` and `<notisdir>` to autocommand patterns? Also `<isfile>`?

Add command modifier that skips wildcard expansion, so that you don't need to put backslashes before special chars, only for white space.

In `mswin.vim`: Instead of mapping `<C-V>` for Insert mode in a complicated way, can it be done like `:"imap <C-V> <MiddleMouse>` without negative side effects?

When right after `"vim file"`, `"M"` then **CTRL-W** v the windows are scrolled differently and unexpectedly. Caused by patch 7.2.398?

The magic clipboard format `"VimClipboard2"` appears in several places. Should be only one.

Win32, NTFS: When editing a specific infostream directly and `'backupcopy'` is `"auto"` should detect this situation and work like `'backupcopy'` is `"yes"`. File name is something like `"c:\path\foo.txt:bar"`, includes a colon. (Alex Jakushev, 2008 Feb 1)

Small problem displaying diff filler line when opening windows with a script. (David Luyer, 2007 Mar 1 ~/Mail/oldmail/mool/in.15872)

Is it allowed that `'backupext'` is empty? Problems when backup is in same dir as original file? If it's OK don't compare with `'patchmode'`. (Thierry Closen)

Patch for adding `:"lscscope"`. (Navdeep Parhar, 2007 Apr 26; update 2008 Apr

23)

":mkview" isn't called with the right buffer argument. Happens when using tabs and the autocommand "autocmd BufWinLeave * mkview". (James Vega, 2007 Jun 18)

When completing from another file that uses a different encoding completion text has the wrong encoding. E.g., when '**encoding**' is utf-8 and file is latin1. Example from Gombault Damien, 2007 Mar 24.

In gvim the backspace key produces a backspace character, but on Linux the VERASE key is Delete. Set VERASE to Backspace? (patch by Stephane Chazelas, 2007 Oct 16)

TermResponse autocommand isn't always triggered when using vimdiff. (Aron Griffis, 2007 Sep 19)

Create a gvimtutor.1 file and change Makefiles to install it.

When '**encoding**' is utf-8 typing text at the end of the line causes previously typed characters to be redrawn. Caused by patch 7.1.329. (Tyler Spivey, 2008 Sep 3, 11)

":vimgrep" does not recognize a recursive symlink. Is it possible to detect this, at least for Unix (using device/inode)?

When switching between windows the cursor is often put in the middle. Remember the relative position and restore that, just like lnum and col are restored. (Luc St-Louis)

Add an option for a minimal text length before inserting a line break for '**textwidth**'. Avoids very short lines when a very long word follows. (Kartik Agaram)

Better plugin support (not plugin manager, see elsewhere for that):

- Avoid use of feedkeys, add eval functions where needed:
 - manipulating the Visual selection?
- Add createmark(): add a mark like mM, but return a unique ID. Need some way to clean them up again... Use a name + the script ID.
Add createmark(, 'c') to track inserts/deletes before the column.
- Plugins need to make a lot of effort, lots of mappings, to know what happened before pressing the key that triggers a plugin action. How about keeping the last N pressed keys, so that they do not need to be mapped?
- equivalent of netbeans_beval_cb(). With an autocommand?
- Add something to enable debugging when a remote message is received.

More patches:

- Another patch for Javascript indenting. (Hari Kumar, 2010 Jul 11)
Needs a few tests.
- Add '**cscopeignorecase**' option. (Liang Wenzhi, 2006 Sept 3)
- Extra argument to strtrans() to translate special keys to their name (Eric Arnold, 2006 May 22)

- Mac: indicate whether a buffer was modified. (Nicolas Weber, 2006 Jun 30)
- Allow negative '**nrwidth**' for left aligning. (Nathan Laredo, 2006 Aug 16)
- ml_append_string(): efficiently append to an existing line. (Brad Beveridge, 2006 Aug 26) Use in some situations, e.g., when pasting a character at a time?
- recognize hex numbers better. (Mark Manning, 2006 Sep 13)
- Add <AbbrExpand> key, to expand an abbreviation in a mapping. (Kana Natsuno, 2008 Jul 17)
- Add '**wspara**' option, also accept blank lines like empty lines for "{" and "}". (Mark Lundquist, 2008 Jul 18)
- Patch to add **CTRL-T** to delete part of a path on cmdline. (Adek, 2008 Jul 21)
- Instead of creating a copy of the tutor in all the shell scripts, do it in vimtutor.vim. (Jan Minar, 2008 Jul 20)
- When fsync() fails there is no hint about what went wrong. Patch by Ben Schmidt, 2008 Jul 22.
- testdir/Make_dos_sh.mak for running tests with MingW. (Bill McCarthy, 2008 Sep 13)
- Replace ccomplete.vim by cppcomplete.vim from www.vim.org? script 1520 by Vissale Neang. (Martin Stubenschrott) Asked Vissale to make the scripts more friendly for the Vim distribution.
New version received 2008 Jan 6.
No maintenance in two years...
- Patch to open dropped files in new tabs. (Michael Trim, 2010 Aug 3)

Awaiting updated patches:

- 9 Mac unicode patch (Da Woon Jung, Eckehard Berns):
 - 8 Add patch from Muraoka Taro (Mar 16) to support input method on Mac?
New patch 2004 Jun 16
 - selecting proportional font breaks display
 - UTF-8 text causes display problems. Font replacement causes this.
 - Command-key mappings do not work. (Alan Schmitt)
 - With '**nopaste**' pasting is wrong, with '**paste**' Command-V doesn't work. (Alan Schmitt)
 - remove '**macatsui**' option when this has been fixed.
 - when '**macatsui**' is off should we always convert to "macroman" and ignore '**termencoding**'?
- 9 HTML indenting can be slow. Caused by using searchpair(). Can search() be used instead? A.Politz is looking into a solution.
- 8 Win32: Add minidump generation. (George Reilly, 2006 Apr 24)
- 7 Completion of network shares, patch by Yasuhiro Matsumoto.
Update 2004 Sep 6.
How does this work? Missing comments.
- 8 Add a few more command names to the menus. Patch from Jiri Brezina (28 feb 2002). Will mess the translations...
- 7 ATTENTION dialog choices are more logical when "Delete it" appears before "Quit". Patch by Robert Webb, 2004 May 3.
 - Include flipcase patch: ~/vim/patches/wall.flipcase2 ? Make it work for multibyte characters.
 - Win32: add options to print dialog. Patch from Vipin Aravind.
 - Patch to add highlighting for whitespace. (Tom Schumm, 2003 Jul 5) use the patch that keeps using HLF_8 if HLF_WS has not been given values.
Add section in help files for these highlight groups?

- 7 Add "DefaultFG" and "DefaultBG" for the colors of the menu. (Marcin Dalecki has a patch for Motif and Carbon)
- Add possibility to highlight specific columns (for Fortran). Or put a line in between columns (e.g., for **'textwidth'**).
Patch to add **'hlcolumn'** from Vit Stradal, 2004 May 20.
- confirm() add "flags" argument, with 'v' for vertical layout and 'c' for console dialog. (Haegg)
Flemming Madsen has a patch for the 'c' flag (2003 May 13)
- raisewin() raise gvim window (see HierAssist patch for Tcl implementation ~/vim/HierAssist/)
- taglist() add argument to specify maximum number of matches. useful for interactive things or completion.
- col('^') column of first non-white character.
Can use "len(substitute(getline('.'), '\S.*', '', '')) + 1", but that's ugly.
- 7 Add patch from Benoit Cerrina to integrate Vim and Perl functions better. Now also works for Ruby (2001 Nov 10)
- 7 When **'rightleft'** is set, the search pattern should be displayed right to left as well? See patch of Dec 26. (Nadim Shaikli)
- 8 Option to lock all used memory so that it doesn't get swapped to disk (unencrypted). Patch by Jason Holt, 2003 May 23. Uses mlock.
- 7 Add ! register, for shell commands. (patch from Grenie)
- 8 In the gzip plugin, also recognize *.gz.orig, *.gz.bak, etc. Like it's done for filetype detection. Patch from Walter Briscoe, 2003 Jul 1.
- 7 Add a "-@ filelist" argument: read file names from a file. (David Kotchan has a patch for it)
- 7 Add ":justify" command. Patch from Vit Stradal 2002 Nov 25.
- findmatch() should be adjusted for Lisp. See remark at get_lisp_indent(). Esp. \ (and \) should be skipped. (Dorai Sitaram, incomplete patch Mar 18)
- For GUI Find/Replace dialog support using a regexp. Patch for Motif and GTK by degreneur (nov 10 and nov 18).
- Patch for "paranoid mode" by Kevin Collins, March 7. Needs much more work.
- Patch for redo register. (Ben Schmidt, 2007 Oct 19)
Await response to question to make the register writable.

Better **'rightleft'** or BIDI support:

- Minimal Vi with bidi support: <https://github.com/aligrudi/neatvi>
By Ali Gholami Rudi, also worked on arabic.c

Spell checking:

- List of common misspellings in English:
https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines
- German:
https://de.wikipedia.org/wiki/Wikipedia:Liste_von_Tippfehlern/F%C3%BCr_Maschinen
- There are other languages.
- [s does not find missing capital at start of the line. #10838
Probably because the dot at the end of the previous line isn't seen.
- When **'cursorline'** is set and the first word should have SpellCap highlighting, redrawing the line removes it when moving the cursor away from the line. (#7085) Would need to inspect the end of the previous line

- and update "capcol_lnum" and "cap_col".
- Mixup of highlighting when there is a match and SpellBad. (ZyX, 2015 Jan 1)
- Spell files use a latin single quote. Unicode also has another single quote: 0x2019. (Ron Aaron, 2014 Apr 4)
New OpenOffice spell files support this with ICONV. But they are not compatible with Vim spell files. The old files can no longer be downloaded.
- Add a feature to only consider two spaces after a dot to start a new sentence. Don't give the capitalization error when there is one space.
- Add a way to specify punctuation characters. Add the superscript numbers by default: 0x2070, 0xb9, 0xb2, 0xb3, 0x2074 - 0x2079.
- In popup menu: If the only problem is the case of the first character, don't offer "ignore" and "add to word list".
- Spell menu: When using the Popup menu to select a replacement word, ":spellrepeat" doesn't work. SpellReplace() uses setline(). Can it use "z=" somehow? Or use a new function.

Quickfix/Location List:

- Window size is wrong when using quickfix window. (Lifepillar, 2018 Aug 24, #2999)
- When using **CTRL-W** CR in the quickfix window, the jumplist in the opened window is cleared, to avoid going back to the list of errors buffer (would have two windows with it). Can we just remove the jump list entries for the quickfix buffer?
- When an item in the quickfix list has a file name that does not exist, behave like the item was not a match for :cnext.
- When adding an item to a new quickfix list make ":cnext" jump to that item. Make a difference being at the first item and not having used :cnext at all. (Afanasiy Fet, 2017 Jan 3)
- When opening quickfix window, disable spell checking?
- Patch for supporting count before CR in quickfix window. (AOYAMA Shotaro, 2007 Jan 1)
- Patch from Herculano de Lima Einloft Neto for better formatting of the quickfix window (2004 dec 2)
- Amiga: When using quickfix with the Manx compiler we only get the first 25 errors. How do we get the rest?
- 8 The quickfix file is read without conversion, thus in 'encoding'. Add an option to specify the encoding of the errorfile and convert it. Also for ":grep" and ":helpgrep".
More generic solution: support a filter (e.g., by calling a function).
- 7 Add a command that goes back to the position from before jumping to the first quickfix location.

Vi incompatibility:

- Try new POSIX tests, made after my comments. (Geoff Clare, 2005 April 7)
Version 1.5 is in ~/src/posix/1.5. (Lynne Canal)
- 8 With undo/redo only marks in the changed lines should be changed. Other marks should be kept. Vi keeps each mark at the same text, even when it is deleted or restored. (Webb)
Also: A mark is lost after: make change, undo, redo and undo.
Example: "{d'" then "u" then "d'": deletes an extra line, because the ' position is one line down. (Veselinovic)
- 8 When stdin is not a tty, and Vim reads commands from it, an error should

- make Vim exit.
- 7 Unix Vim (not gvim): Typing **CTRL-C** in Ex mode should finish the line (currently you can continue typing, but it's truncated later anyway). Requires a way to make **CTRL-C** interrupt select() when in cooked input.
 - 8 When loading a file in the .exrc, Vi loads the argument anyway. Vim skips loading the argument if there is a file already. When no file argument given, Vi starts with an empty buffer, Vim keeps the loaded file. (Bearded)
 - 6 In Insert mode, when using <BS> or , don't wipe out the text, but only move back the cursor. Behaves like '\$' in 'coptions'. Use a flag in 'coptions' to switch this on/off.
 - 8 When editing a file which is a symbolic link, and then opening another symbolic link on the same file, Vim uses the name of the first one. Adjust the file name in the buffer to the last one used? Use several file names in one buffer???
 - Also: When first editing file "test", which is symlink to "test2", and then editing "test2", you end up editing buffer "test" again. It's not logical that the name that was first used sticks with the buffer.
 - 7 The ":undo" command works differently in Ex mode. Edit a file, make some changes, "Q", "undo" and _all_ changes are undone, like the ":visual" command was one command. On the other hand, an ":undo" command in an Ex script only undoes the last change (e.g., use two :append commands, then :undo).
 - 7 The ":map" command output overwrites the command. Perhaps it should keep the ":map" when it's used without arguments?
 - 7 **CTRL-L** is not the end of a section? It is for Posix! Make it an option.
 - 7 Add a way to send an email for a crashed edit session. Create a file when making changes (containing name of the swap file), delete it when writing the file. Supply a program that can check for crashed sessions (either all, for a system startup, or for one user, for in a .login file).
 - 7 Vi doesn't do autoindenting when input is not from a tty (in Ex mode).
 - 7 "z3<CR>" should still use the whole window, but only redisplay 3 lines.
 - 7 ":tag xx" should move the cursor to the first non-blank. Or should it go to the match with the tag? Option?
 - 7 Implement 'autoprint'/'ap' option.
 - 7 Add flag in 'coptions' that makes <BS> after a count work like (Sayre).
 - 7 Add flag in 'coptions' that makes operator (yank, filter) not move the cursor, at least when cancelled. (default Vi compatible).
 - 7 This Vi-trick doesn't work: "Q" to go to Ex mode, then "g/pattern/visual". In Vi you can edit in visual mode, and when doing "Q" you jump to the next match. Nvi can do it too.
 - 7 Support '\\' for line continuation in Ex mode for these commands: (Luebking)

g/./a\ line 1\ line 2\ .	g/pattern1/ s/pattern2/rep1\ line 2\ line 3\ line4/
-----------------------------------	--
 - 6 ":e /tmp/\$tty" doesn't work. ":e \$uid" does. Is \$tty not set because of the way the shell is started?
 - 6 Vi compatibility (optional): make "ia<CR><ESC>10." do the same strange thing. (only repeat insert for the first line).

GTK+ GUI:

- 9 Crash with X command server over ssh. (Ciaran McCreesh, 2006 Feb 6)

- 8 GTK 2: Combining UTF-8 characters not displayed properly in menus (Mikolaj Machowski) They are displayed as separate characters. Problem in creating a label?
- 8 GTK 2: Combining UTF-8 characters are sometimes not drawn properly. Depends on the font size, "monospace 13" has the problem. Vim seems to do everything right, must be a GTK bug. Is there a way to work around it?
- 9 Can't paste a Visual selection from GTK-gvim to vim in xterm or Motif gvim when it is longer than 4000 characters. Works OK from gvim to gvim and vim to vim. Pasting through xterm (using the shift key) also works. It starts working after GTK gvim loses the selection and gains it again.
- Gnome2: When moving the toolbar out of the dock, so that it becomes floating, it can no longer be moved. Therefore making it float has been blocked for now.
- Mapping with partial match not executed properly in GTK. (Ingo Karkat, #7082)
- Add more testing of the GTK GUI.
 - gtk_test_widget_click() can be used to simulate a click in a widget.
- Tests failing for "make testgui" with GTK:
 - Test_setbufvar_options()
 - Test_exit_callback_interval()
- When adding a timer from '**balloonexpr**' it won't fire, because g_main_context_iteration() doesn't return. Need to trigger an event when the timer expires.
- problem with 'L' in '**guioptions**' changing the window width. (Aaron Cornelius, 2012 Feb 6)
- Patch to invert characters differently in GTK. (Yukihiro Nakadaira, 2013 May 5)
- drawing a double-width combining character over single-width characters doesn't look right. (Dominique Pelle, 2010 Aug 8)
- tear-off menu does not work. (Kurt Sonnenmoser, 2010 Oct 25)
- Patch for GTK buttons X1Mouse and X2Mouse. (Christian J. Robinson, 2010 Aug 9)
- When setting '**columns**' in a startup script and doing ":vertical difflsplit" the window isn't redrawn properly, see two vertical bars.
- when the Tab pages bar appears or disappears while the window is maximized the window is no longer maximized. Patch that has some idea but doesn't work from Geoffrey Antos, 2008 May 5. Also: the window may no longer fit on the screen, thus the command line is not visible.
- Patch to support horizontal scroll wheel in GTK. Untested. (Bjorn Winckler, 2010 Jun 30)

Win32 GUI known bugs:

- Win32: tearoff menu window should have a scrollbar when it's taller than the screen.
- 8 The -P argument doesn't work very well with many MDI applications. The last argument of CreateWindowEx() should be used, see MSDN docs. Tutorial: <http://win32assembly.online.fr/tut32.html>
- 6 Win32 GUI: With "-u NONE -U NONE" and doing "**CTRL-W v**" "**CTRL-W o**", the ":" of ":only" is highlighted like the cursor. (Lipelis)
- 8 When '**encoding**' is "utf-8", should use '**guifont**' for both normal and wide characters to make Asian languages work. Win32 fonts contain both type of characters.
- 7 When font smoothing is enabled, redrawing can become very slow. The reason

- appears to be drawing with a transparent background. Would it be possible to use an opaque background in most places?
- 7 The cursor color indicating IME mode doesn't work properly. (Shizhu Pan, 2004 May 9)
- 8 Win32: When clicking on the gvim title bar, which gives it focus, produces a file-changed dialog, after clicking on a button in that dialog the gvim window follows the mouse. The button-up event is lost. Only with MS-Windows 98?
- Try this: ":set sw ts", get enter-prompt, then change the file in a console, go back to Vim and click "reload" in the dialog for the changed file: Window moves with the cursor!
- Put focus event in input buffer and let generic Vim code handle it?
- 8 When Vim is minimized and when maximizing it a file-changed dialog pops up, Vim isn't maximized. It should be done before the dialog, so that it appears in the right position. (Webb)
- 9 When selecting at the more-prompt or hit-enter-prompt, the right mouse button doesn't give popup menu.
- At the hit-enter prompt **CTRL-Y** doesn't work to copy the modeless selection.
- On the command line, don't get a popup menu for the right mouse button. Let the middle button paste selected text (not the clipboard but the non-Visual selection)? Otherwise **CTRL-Y** has to be used to copy the text.
- 8 When **'grepprg'** doesn't execute, the error only flashes by, the user can hardly see what is wrong. (Moore)
- Could use vimrun with an "-nowait" argument to only wait when an error occurs, but "command.com" doesn't return an error code.
- 8 When the **'shell'** cannot be executed, should give an appropriate error msg. Esp. for a filter command, currently it only complains the file could not be read.
- 7 At the hit-enter prompt scrolling now no longer works. Need to use the keyboard to get around this. Pretend **<CR>** was hit when the user tries to scroll?
- 7 Scrollbar width doesn't change when selecting other windows appearance. Also background color of Toolbar and rectangle below vert. scrollbar.
- 6 Drawing text transparently doesn't seem to work (when drawing part cursor).
- 8 CTRL key doesn't always work in combination with ALT key. It does work for function keys, not for alphabetic characters. Perhaps this is because **CTRL-ALT** is used by Windows as AltGr?
- 8 **CTRL--** doesn't work for AZERTY, because it's **CTRL-[** for QWERTY. How do we know which keyboard is being used?
- 7 When scrolling, and a background color is dithered, the dither pattern doesn't always join correctly between the scrolled area and the new drawn area (Koloseike).
- 8 When `gui_init_font()` is called with "*", `p_guifont` is freed while it might still be used somewhere. This is too tricky, do the font selection first, then set the new font by name (requires putting all logfont parameters in the font name).
- When running a fast timer, the cursor no longer blinks. Was reported: cursor blinks in terminal on widows with a timer. (xtal8, #2142)
 - Last message from startup doesn't show up when there is an echoerr command. (Cyril Slobin, 2009 Mar 13)

Motif:

- 6 New Motif toolbar button from Marcin Dalecki:
 - When the mouse pointer is over an Agide button the red becomes black. Something with the way colors are specified in the .xpm file.
 - The pixmap is two pixels smaller than it should be. The gap is filled with grey instead of the current toolbar background color.
- 9 Can configure be changed to disable netbeans if the Xpm library is required and it's missing?
- 8 When using the resource "Vim*borderwidth 2" the widgets are positioned wrong.
- 9 XIM is disabled by default for SGI/IRIX. Fix XIM so that 'imdisable' can be off by default.
- 9 XIM doesn't work properly for Motif. (Yasuhiro Matsumoto) For now, keep XIM active at all times when the input method has the preediting flag.
- 8 X11: A menu that contains an umlaut is truncated at that character. Happens when the locale is "C", which uses ASCII instead of ISO-8859-1. Is there a way to use latin1 by default? Gnome_init() seems to do this.
- 8 Perhaps use fontsets for everything?
- 6 When starting in English and switching the language to Japanese, setting the locale with ":lang", 'guifontset' and "hi menu font=", deleting all menus and setting them again, the menus don't use the new font. Most of the tooltips work though...
- 7 Motif: when using a file selection dialog, the specified file name is not always used (when specifying a filter or another directory).
- 8 When 'encoding' is different from the current locale (e.g., utf-8) the menu strings don't work. Requires conversion from 'encoding' to the current locale. Workaround: set 'langmenu'.

Motif GUI:

- gui_mch_browsedir() is missing, browsedir() doesn't work nicely.
- 7 Use XmStringCreateLocalized() instead of XmStringCreateSimple()? David Harrison says it's OK (it exists in Motif 1.2).
- 8 Lesstif: When deleting a menu that's torn off, the torn off menu becomes very small instead of disappearing. When closing it, Vim crashes. (Phillipps)

GUI:

- 9 On Solaris, creating the popup menu causes the right mouse button no longer to work for extending the selection. (Halevy)
- 9 When running an external program, it can't always be killed with **CTRL-C**. e.g., on Solaris 5.5, when using "K" (Keech). Other 'guipty' problems on Solaris 2.6. (Marley)
- 9 On Solaris: Using a "-geometry" argument, bigger than the window where Vim is started from, causes empty lines below the cmdline. (raf)
- 8 When setting 'langmenu', it should be effective immediately. Store both the English and the translated text in the menu structure. Re-generate the translation when 'langmenu' has changed.
- 8 Basic flaw in the GUI code: NextScreen is updated before calling gui_write(), but the GUI code relies on NextScreen to represent the state of where it is processing the output. Need better separation of Vim core and GUI code.
- 8 When fontset support is enabled, setting 'guifont' to a single font

doesn't work.

- 8 Menu priority for sub-menus for: Amiga.
- 8 When translating menus ignore the part after the Tab, the shortcut. So that the same menu item with a different shortcut (e.g., for the Mac) are still translated.
- 8 Add menu separators for Amiga.
- 8 Add way to specify the file filter for the browse dialog. At least for browse().
- 8 Add dialog for search/replace to other GUIs? Tk has something for this, use that code? Or use console dialog.
- 8 When selecting a font with the font dialog and the font is invalid, the error message disappears too quick.
- 7 More features in the find/replace dialog:
 - regexp on/off
 - search in selection/buffer/all buffers/directory
 - when all buffers/directory is used:
 - filter for file name
 - when directory is used:
 - subdirectory on/off
 - top directory browser
- 8 gui_check_colors() is not called at the right moment. Do it much later, to avoid problems.
- 8 gui_update_cursor() is called for a cursor shape change, even when there are mappings to be processed. Only do something when going to wait for input. Or maybe every 100 ms?
- 8 X11: When the window size is reduced to fit on screen, there are blank lines below the text and bottom scrollbar. "gvim -geometry 80x78+0+0". When the "+0+0" is omitted it works.
- 8 When starting an external command, and 'guipty' set, BS and DEL are mixed up. Set erase character somehow?
- 8 The compose key doesn't work properly (Cepas). Both for Win32 and X11.
- 7 The cursor in an inactive window should be hollow. Currently it's not visible.
- 7 GUI on Solaris 2.5.1, using /usr/dt/..: When gvim starts, cursor is hollow, after window lowered/raised it's OK. (Godfrey)
- 7 When starting GUI with ":gui", and window is made smaller because it doesn't fit on the screen, there is an extra redraw.
- 8 When setting font with .Xdefaults, there is an extra empty line at the bottom, which disappears when using ":set guifont=<Tab>". (Chadzelek)
- 8 When font shape changes, but not the size, doing ":set font=" does not redraw the screen with the new font. Also for Win32.
 - When the size changes, on Solaris 2.5 there isn't a redraw for the remaining part of the window (Phillipps).
- Flashes really badly in certain cases when running remotely from a Sun.
- 4 Re-write the code so that the highlighting isn't changed multiple times when doing a ":hi clear". The color changes happen three or more times currently. This is very obvious on a 66Mhz 486.

Win32 console:

- 8 Should \$USERPROFILE be preferred above \$HOMEDRIVE/\$HOMEPATH? No, but it's a good fallback, thus use:
 - \$HOME
 - \$HOMEDRIVE\$HOMEPATH

```

    SHGetSpecialFolderPath(NULL, lpzsPath, CSIDL_APPDATA, FALSE);
    $USERPROFILE
    SHGetSpecialFolderPath(NULL, lpzsPath, CSIDL_COMMON_APPDATA, FALSE);
    $ALLUSERSPROFILE
    $SYSTEMDRIVE\
    C:\

```

- 8 Win32 console: <M-Up> and <M-Down> don't work. (Geddes) We don't have special keys for these. Should use modifier + key.
- 8 Win32 console: caps-lock makes non-alpha keys work like with shift. Should work like in the GUI version.
- 8 Environment variables in DOS are not case sensitive. Make a define for STRCMP_ENV(), and use it when comparing environment var names.
- 8 Setting '**shellslash**' has no immediate effect. Change all file names when it is set/reset? Or only use it when actually executing a shell command?
- 8 When editing a file on a Samba server, case might matter. ":e file" followed by ":e FILE" will edit "file" again, even though "FILE" might be another one. Set last used name in buflist_new()? Fix do_ecmd(), etc.
- 8 When a buffer is editing a file like "**ftp://mach/file**", which is not going to be used like a normal file name, don't change the slashes to backslashes. (Ronald Hoellwarth)

Win32 console:

- 9 When editing a file by its short file name, it should be expanded into its long file name, to avoid problems like these: (Mccollister)
 - 1) Create a file called ".bashrc" using some other editor.
 - 2) Drag that file onto a shortcut or the actual executable.
 - 3) **Note** that the file name is something like BASHRC~1
 - 4) Go to File->Save As menu item and type ".bashrc" as the file name.
 - 5) Press "Yes" to indicate that I want to overwrite the file.
 - 6) **Note** that the message "File exists (add ! to override)" is displayed and the file is not saved.
- Use FindFirstFile() to expand a file name and directory in the path to its long name.
- 8 Also implement '**conskey**' option for the Win32 console version? Look at how Xvi does console I/O under Windows NT.
- 7 Re-install the use of \$TERM and support the use of different terminals, besides the console.
- 8 Use of <altgr> modifier doesn't work? 5.3 was OK. (Garcia-Suarez/Guckles)
- 9 Mapping <C-S-Tab> doesn't work correctly. How to see the difference with <C-S-i>?
- 9 tmpnam() uses file in root of file system: "\asdf". That doesn't work on a Netware network drive. Use same function as for Win32 GUI?
- 8 In os_win32.h, HAVE_STRICTMP and HAVE_STRNICMP are defined only if __GNUC__ is not defined. Shouldn't that be the other way around?

Amiga:

- 8 In mch_inchar() should use convert_input_safe() to handle incomplete byte sequences.
- 9 In mch_expandpath() a "*" is to be expanded, but "*" isn't. Remove backslashes in result.
- 8 Executing a shell, only one option for '**shell**' is separated. Should do all options, using white space separation.

Macintosh:

- GUI: `gui_mch_browsedir()` is missing.
- 7 Loading the Perl library only works on OS/X 10.2 or 10.3, never on both. Load the Perl library dynamically see Python sources file `dynload_mac` (Jack)
- dynamic linking: <http://developer.apple.com/technotes/tn2002/tn2064.html>
- 8 `inputdialog()` doesn't resize when giving more text lines. (David Fishburn, 2006 Sept 28)
- 8 Define `vim_mkdir()` for Macintosh.
- 8 Define `mch_writable()` for Macintosh.
- 9 When DiskLock is running, using a swap file causes a crash. Appears to be a problem with writing a file that starts with a dot. (Giacalone)
- 9 In `mac_expandpath()` check that handling of backslashes is done properly.
- Build problem with small features on Mac OS X 10.6. (Rainer, 2011 Jan 24)
- Version of `netbeans.c` for use with MacVim. (Kazuki Sakamoto, 2010 Nov 18)
- Mac with X11: clipboard doesn't work properly. (Raf, 2010 Aug 16)
- Move Carbon todo items to `os_mac.txt`. Note that this version is frozen, try the Cocoa version.
- After a `":vsplit"` the left scrollbar doesn't appear until `'columns'` is changed or the window is resized.
- Patch for `configure`: remove `arch` from ruby link args. (Knezevic, 2008 Mar 5) Alternative: Kazuki Sakamoto, Mar 7.
- trouble compiling with Motif, requires `--disable-darwin`. (Raf, 2008 Aug 1) Reply by Ben Schmidt.
- Using `gvim`: `netrw` window disappears. (Nick Lo, 2006 Jun 21)

"Small" problems:

- When using `e_secure` in `do_one_cmd()` mention the command being executed, otherwise it's not clear where it comes from.
- 9 For Turkish `vim_tolower()` and `vim_toupper()` also need to use `utf_` functions for characters below `0x80`. (Sertacyildiz)
- 9 When the last edited file is a help file, using `'0` in a new Vim doesn't edit the file as a help file. `'filetype'` is OK, but `'iskeyword'` isn't, file isn't readonly, etc.
- 8 When an `":edit"` is inside a `try` command and the ATTENTION prompt is used, the `:catch` commands are always executed, also when the file is edited normally. Should reset `did_emsg` and undo side effects. Also make sure the ATTENTION message shows up. Servatius Brandt works on this.
- 7 Vimtutor leaves escape sequence in terminal. This is the xterm response to requesting the version number. (Yasuhiro Matsumoto)
- 8 When redirecting and using `":silent"` the current column for displaying and redirection can be different. Use a separate variable to hold the column for redirection.
- 7 The messages for `"vim --help"` and `"vim --version"` don't use `'termencoding'`.
- Could the hit-enter prompt be avoided when a message only overlaps the `'showcmd'` area? Clear that area when the next cmd is typed.
- 8 When `'scrollbind'` is set, a window won't scroll horizontally if the cursor line is too short. Add a word in `'scrollopt'` to allow moving the cursor to longer line that is visible. A similar thing is done for the GUI when using the horizontal scrollbar.

- 8 When giving a `":bwipeout"` command a file-changed dialog may popup for this buffer, which is pointless. (Mike Williams)
- 8 On MS-Windows `":make"` doesn't show output while it is working. Use the `tee.exe` from <http://unxutils.sourceforge.net/> ? About 16 Kbyte in the `UnxUtils.zip` archive.
Is it better than what we have in `src/tee`?
- 8 When doing Insert mode completion a mapping cannot recursively call `edit()`, because the completion information is global. Put everything in an allocated structure?
- 7 `mb_off2cells()` doesn't work correctly on the tail byte of a double-byte character. (Yasuhiro Matsumoto) It should return 1 when used on a tail byte, like for utf-8. Store second byte of double-byte in `ScreenLines2[]` (like for `DBCS_JPNU`) and put a zero in the second byte (like for `UTF-8`).
- 7 Inside a function with `"perl <<EOF"` a line with `"$i++"` is recognized as an `":insert"` command, causing the following `"endfunction"` not to be found. Add skipping this perl construction inside function definitions.
- 7 When `'ttimeoutlen'` is 10 and `'timeoutlen'` is 1000, there is a keycode `"<Esc>a"` and a mapping `<Esc>x"`, when typing `"<Esc>a"` with half a second delay should not be interpreted as a keycode. (Hans Ginzel)
- 7 `":botright 1 new"` twice causes all window heights to be changed. Make the bottom window only bigger as much as needed.
- 7 The `Cygwin` and `MingW` makefiles define `"PC"`, but it's not used anywhere. Remove? (Dan Sharp)
- 9 User commands use the context of the script they were defined in. This causes a `"s:var"` argument to unexpectedly use a variable in the defining script, not the calling script. Add an argument to `":command"`: `"-keepcontext"`. Do replace `<SID>`, so that a function in the defining script can be called.
- 8 The Japanese message translations for MS-Windows are called `ja.sjis.po`, but they use encoding `cp932`. Rename the file and check that it still works.
- 8 A very long message in `confirm()` can't be quit. Make this possible with **CTRL-C**.
- 8 `"gf"` always excludes trailing punctuation characters. `file_name_in_line()` is currently fixed to use `".,;:!"`. Add an option to make this configurable?
- 8 `'hkmap'` should probably be global-local.
- 8 Using `":s"` in a function changes the previous replacement string. Save `"old_sub"` in `save_search_patterns()`?
- 8 Should allow multibyte characters for the delimiter: `":s+a+b"` where `"a"` is a multibyte character.
- 8 When appending to a file and `'patchmode'` isn't empty, a backup file is always written, even when the original file already exists.
- 9 When getting focus while writing a large file, could warn for this file being changed outside of Vim. Avoid checking this while the file is being written.
- 7 The message in `bt_dontwrite_msg()` could be clearer.
- 8 The script ID that is stored with an option and displayed with `":verbose set"` isn't reset when the option is set internally. For example when `'foldlevel'` is set from `'foldlevelstart'`.
- 8 Also store the line number with the script ID and use it for `":verbose"`, so that `"set nocompatible"` is found when it changes other option values. When an option is set indirectly mention the command? E.g. when `":diffsplit"` sets `'foldmethod'`.

- 8 In the fileformat dialog, "Cancel" isn't translated. Add a global variable for this. (Eduardo Fernandez)
- 9 When editing a file with **'readonly'** set, there is no check for an existing swap file. Then using `:"write"` (without making any changes) doesn't give a warning either. Should check for an existing swap file without creating one. Unfinished patch by Ian Kelling, 2008 July 14.
- 7 When **'showbreak'** is set, the amount of space a Tab occupies changes. Should work like **'showbreak'** is inserted without changing the Tabs.
- 7 When **'mousefocus'** is set and switching to another window with a typed command, the mouse pointer may be moved to a part of the window that's covered by another window and we lose focus. Only move in the y direction, not horizontally?
- 8 `:"hardcopy"`:
- Using the `cterm_color[]` table is wrong when `t_colors` is > 16 .
 - Need to handle unprintable characters.
 - Win32: On a B&W printer syntax highlighting isn't visible. Perform dithering to make grey text?
 - Add a flag in **'printoptions'** to add an empty page to make the total number even. `"addempty"`? (Mike Williams)
 - Respect **'linebreak'**. Perhaps also **'showbreak'**?
 - Should interpret **CTRL-L** as a page break.
 - Grey line numbers are not always readable. Add field in **'printoptions'**. Default to black when no syntax highlighting.
 - Be able to print a window in diff mode.
 - Be able to specify a colorscheme to use for printing. And a separate one for B&W printing (if that can be detected).
- 8 When **'virtualedit'** is "block,insert" and encoding is "utf-8", selecting a block of one double-wide character, then "d" deletes only half of it.
- 8 When **'virtualedit'** is set, should "I" in blockwise visual mode also insert in lines that don't extend into the block?
- 8 With **'virtualedit'** set, in Insert mode just after the end of line, **CTRL-O** yh does not yank the last character of the line. (Pavel Papushev)
Doing "hl" first appears to make it work.
- 8 With **'virtualedit'** set it's possible to move into the blank area from **'linebreak'**.
- 8 With **'virtualedit'** set and **'selection'** "exclusive", a Visual selection that ends in or after a tab, "d" doesn't delete (part of) the tab. (Helmut Stiegler)
- 9 When jumping to a tag, the search pattern is put in the history. When **'magic'** is on, the pattern may not work. Translate the pattern depending on `p_magic` when putting it in the history? Alternative: Store value of **'magic'** in history. (Margo)
- 9 `optwin.vim`: Restoring a mapping for `<Space>` or `<CR>` is not correct for `:"noremap"`. Add `"mapcmd({string}, {mode})`? Use code from `:"mkexrc"`.
- 9 `term_console` is used before it is set (msdos, Amiga).
- 9 Get out-of-memory for `:"g/^/, $s//@/"` on 1000 lines, this is not handled correctly. Get many error messages while redrawing the screen, which cause another redraw, etc.
- 8 `[<C-I>` doesn't work when `'*` is in **'iskeyword'**. `find_pattern_in_path()` must escape special characters in the pattern.
- 8 Vim can overwrite a read-only file with `:"w!"`. `:"w"` can't overwrite an existing file, `"w!"` can, but perhaps not a read-only file? Then use `:"w!!"` for that.
Or ask for permission to overwrite it (if file can be made writable) and

- restore file to readonly afterwards.
Overwriting a file for which a swap file exists is similar issue.
- 7 When compiled with "xterm_clipboard", startup can be slower and might get error message for invalid \$DISPLAY. Try connecting to the X server in the background (forked), so that Vim starts up quicker? Connect as soon as the clipboard is to be used (Visual select mode starts, paste from clipboard)
- 7 X11: Some people prefer to use CLIPBOARD instead of PRIMARY for the normal selection. Add an "xclipboard" argument to the '**clipboard**' option? (Mark Waggoner)
- 6 When the xterm reports the number of colors, a redraw occurs. This is annoying on a slow connection. Wait for the xterm to report the number of colors before drawing the screen. With a timeout.
- 8 When the builtin xterm termcap contains codes that are not wanted, need a way to avoid using the builtin termcap.
- 8 Xterm sends ^[[H for <Home> and ^[[F for <End> in some mode. Also recognize these keys? Mostly useful for xterm simulators, like gnometerm. See http://dickey.his.com/xterm/xterm.faq.html#xterm_pc_style.
- 8 '[' and ']' should be set to start/end of line when using a linewise operator (e.g., ":w").
- 8 **CTRL-A** can't handle big "long" numbers, they become negative. Check for "-" character, if not present, use unsigned long.
- 8 Add suspending with **CTRL-Z** at the "more" prompt, and when executing a long script in do_cmdline().
- 8 When using '**hidden**', many swap files will be open. When Vim runs into the maximum number of open files, error messages will appear. Detect that this problem is present, and close any hidden files that don't have changes.
- 8 Core dump within signal function: gdb doesn't show stack backtrace! Option to skip catch_signals()?
- 9 Repeating a "cw" with "." doesn't work if the text was pasted from the clipboard. (Thomas Jones) It's because the menu/toolbar item exits Insert mode and uses "gP". How to fix this without breaking inserting a block of text?
- 8 In Replace mode pasting from the clipboard (using menu or toolbar) inserts all the text. Add ":rmenu"?
- 8 Pasting with the mouse in Replace mode inserts the text, instead of overwriting, when it is more than one line. Same for using <C-R>.
- 9 **CTRL-E** and **CTRL-Y** don't work in small window when '**so**' is 4 and lines are wrapping (Acevedo/in.226). E.g., when using **CTRL-E**, window height 7, window might actually scroll down when last line of buffer is displayed. --> Remember if the previous command was "cursor follows screen" or "screen follow cursor" and use this in cursupdate().
- 7 tilde_replace() can only handle "~/", should also do "~user/".
Get the list of home directories (from /etc/passwd? Use getpwent()) and use some clever algorithm to match a path with that. Find common strings in the list?
- 8 When dragging status line with mouse, sometimes a jump when first clicking on the status line (caused by '**winheight**'). Select window on button up, instead of on button down.
- 8 Dragging the status line doesn't scroll but redraw.
- 8 When performing incremental search, should abort searching as soon as a character is typed.
- 8 How to set VIMRC_FILE to "\"something\" for configure? Why does this not

- work: CFLAGS='-DVIMRC_FILE=\"/mydir/myfile\"' ./configure
- 8 The temporary file is sometimes not writable. Check for this, and use an alternate name when it isn't. Or add the **'temptemplate'** option: template for the temp file name `":set temptemplate=/usr/tmp/?????.tmp"`. Also: Win32 version uses Windows temp directory, which might not work for cygwin bash.
 - 7 Get error `"*, \+ or \(\ operand could be empty"` for pattern `"\(\.\)\1\{3}"`. Remember flags for backreferences.
 - 7 When switching to Daylight Saving Time, Vim complains that a file has been changed since last read. Can we use a function that uses GMT?
 - 7 When completing an environment variable after a '\$', check for file names that contain a '\$' after all have been found.
 - 8 When "cm" termcap entry is missing, starting gvim shouldn't complain about it. (Lohner) Try out with "vt100" entry, cm replaced with cX.
 - 7 When an include file starts with `"../"`, the check for already visiting this file doesn't work. Need to simplify the file name.
 - 7 The names and comments for the arguments of `do_browse()` are confusing. "dflt" isn't the default file name when "initdir" is not NULL and "initdir" is the default path to be used.
 - 7 When **'scrolloff'** is exactly half the window height, "j" causes a scroll of two lines at a time. "k" doesn't do this. (Cory T. Echols)

I can't reproduce these (if you can, let me know how!):

- 9 NT 4.0 on NTFS file system: Editing `".bashrc"` (drag and drop), file disappears. Editing `".xyz"` is OK. Also, drag&drop only works for three files. (McCollister)

Problems that will (probably) not be solved:

- GTK: when using the popup menu with spelling suggestions and releasing the right mouse button before the menu appears selecting an item with the right mouse button has no effect. GTK does not produce an event for this.
- GTK 2: Cannot use the file selector. When using it many things become slow. This is caused by some code in GTK that writes `~/recently-used.xbel` every time an event is handled. It assumes the main loop is never quit, which is a wrong assumption. Also, it overwrites the file with different file permissions, which is a privacy issue. This needs to be fixed in GTK. A solution in Vim would be really complicated. (2008 Jul 31) This appears to be fixed in Vim 7.3.
- xterm title: The following scenario may occur (esp. when running the Vim test script): Vim 1 sets the title to "file1", then restores the title to "xterm" with an ESC sequence when exiting. Vim 2 obtains the old title with an X library call, this may result in "file1", because the window manager hasn't processed the "xterm" title yet. Can apparently only be worked around with a delay.
- In a terminal with **'mouse'** set such that the mouse is active when entering a command line, after executing a shell command that scrolls up the display and then pressing `":"`: Selecting text with the mouse works like the display wasn't scrolled. Vim doesn't know how much the external command scrolled up the display. Use Shift to select text.
- X windows: When `$DISPLAY` points to a X server where there is no access permission, trying to connect to the X server causes an error message. `XtOpenDisplay()` prints this directly, there is no way to avoid it.

- X windows: Setting '**guifontset**' to an illegal value sometimes crashes Vim. This is caused by a fault in a X library function, can't be solved in Vim.
- Win32 tcl: has("tcl") hangs when the tcl84.dll is from cygwin.
- Motif: When adding a menu item "Find this &Symbol", the "s" in "this" will be underlined, instead of in "Symbol". Motif doesn't let us specify which character gets the highlighting.
- Moving the cursor removes color in color-xterm. This is a color-xterm problem! color-xterm ver. 6.1 beta 3 and later work properly.
- In zsh, "gvim&" changes the terminal settings. This is a zsh problem. (Jennings)
- Problem with HPterm under X: old contents of window is lost (Cosentino).
- Amiga: The ":cq" command does not always abort the Manx compiler. Why?
- Linux: A file with protection r--rw-rw- is seen readonly for others. The access() function in GNU libc is probably wrong.
- When doing a **CTRL-Z** and typing a command for the shell, while Vim is busy (e.g. writing a file), the command for the shell is sometimes eaten by Vim, because the terminal mode is changed from RAW to CBREAK.
- An old version of GNU tgoto can't handle the terminfo code for "AF". The "%p1" is interpreted as "%p" and "1", causing color not to be working. Fix: Change the "%p1" in the "AF" and "AB" terminfo entries to "%p". (Benzinger).
- When running an external command from the GUI, typeahead is going to that program, not to Vim. It looks like the shell eats the characters, Vim can't get back what the external command didn't use.
- Win32 GUI: Error code from external command not returned in shell_error. It appears that cmd.exe and command.com don't return an error code.
- Win32 GUI: The Toolbar is a bit too high when the flat style is being used. We don't have control over the height of the Toolbar.
- Win32: All files created on the day of switching from winter to summer time cause "changed since editing started" messages. It goes away when the file is written again the next day, or the timezone is adjusted. DJGPP version is OK. (Zaimi) Looks like a problem with the Win32 library. Rebooting doesn't help. Time stamps look OK in directory. (Penn) Is this on FAT (stores wall clock time) or NTFS (stores UTS)?
- Win32, MS-Windows XP: \$HOME uses the wrong drive when the user profiles are not on the boot disk. This is caused by a wrong value of \$HOMEDRIVE. This is a bug in XP, see MSKB article 818134.
- Win32, MS-Windows: expanding plugin/**/*vim also picks up dir/ctags.vim,v. This is because the short file name is something like "ctags~1.vim" and that matches the pattern.
- SunOS 5.5.1 with Motif: The file open dialog does not have a horizontal scroll bar for the "files" selection. This is a problem in the Motif libraries, get a patch from Sun.
- Solaris 2.6 with GTK and Perl: gvim crashes when started. Problem with X input method called from GDK code. Without Perl it doesn't crash.
- VMS: Vimdiff doesn't work with the VMS diff, because the output looks different. This makes test 47 fail. Install a Unix-compatible diff.
- GTK with Gnome: Produces an error message when starting up:
 - Gdk-WARNING **: locale not supported by C library
 This is caused by the gnome library gnome_init() setting \$LC_CTYPE to "en_US". Not all systems support this locale name, thus causing the error. Hopefully a newer version of GTK/Gnome fixes this problem.
- GTK 2: With this mapping the hit-enter prompt is `_sometimes_` below the screen, at other times there is a grey area below the command line:

- `:nmap <F11> :if &guioptions=~'m' \|| set guioptions-=m \|| else \|| set guioptions+=m`
- GTK: When pasting a selection from Vim to xclipboard gvim crashes with a ABRT signal. Probably an error in the file gdkselection.c, the assert always fails when XmbTextListToTextProperty() fails. (Tom Allard)
- GTK 2: gives an assertion error for every non-builtin icon in the toolbar. This is a GTK 2.4.x bug, fixed in GTK 2.4.2. (Thomas de Grenier de Latour)
- When using an xterm that supports the termresponse feature, and the `'t_Co'` termcap option was wrong when Vim started, it will be corrected when the termresponse is received. Since the number of colors changes, the highlighting needs to be initialized again. This may cause colors defined in the vimrc file to be lost.
- On Windows NT 4.0 the number of files passed to Vim with drag&drop and "Edit with Vim" is limited. The maximum command line length is 255 chars.

----- extensions and improvements -----
extensions-improvements

Most interesting new features to be added when all bugs have been fixed:

- Using `":exe edit fname"` has escaping problems. Use `":edit ++(fname)"`. Thus use `"++="` to give arguments as expressions, comma-separated as if calling a function.
 With options: `":edit ++(['!', '++enc=abc'], ['+/pat'], fname)"`.
 Alternative: Make a function for Ex commands: `cmd_edit()`.
- Add COLUMN NUMBERS to `":"` commands `":line1,line2[col1,col2]cmd"`. Block can be selected with **CTRL-V**. Allow `'$'` (end of line) for col2. (issue #3292)
- ECLIPSE plugin. Problem is: the interface is very complicated. Need to implement part in Java and then connect to Vim. Some hints from Alexandru Roman, 2004 Dec 15. Should then also work with Oracle Jdeveloper, see JSR 198 standard <http://www.jcp.org/en/jsr/detail?id=198>.
 Eclim does it: <http://eclim.sourceforge.net/> (Eric Van Dewoestine)
 Plugin that uses a terminal emulator: <http://vimplugin.sf.net>
 And another one: <http://www.satokar.com/viplugin/index.php>
- STICKY CURSOR: Add a way of scrolling that leaves the cursor where it is. Especially when using the scrollbar. Typing a cursor-movement command scrolls back to where the cursor is.
- Scroll commands by screen line. g **CTRL-E** and g **CTRL-Y** ? Requires the first line to be able to start halfway.
- 8 Add a command to jump to a certain kind of tag. Allow the user to specify values for the optional fields. E.g., `":tag size type=m"`. Also allow specifying the file and command, so that the result of `taglist()` can be used.
- X11: Make it possible to run Vim inside a window of another program. This can be done with `XReparentWindow()`. But how exactly?

Documentation:

- 8 List of Vim runtime directories. `dotvim.txt` from Charles Campbell, 2007 Feb 20.
- 8 The GUI help should explain the Find and Find/Replace dialogs. Add a link to it from `":promptrepl"` and `":promptfind"`.
- 8 List of options should mention whether environment variables are expanded or not.
- 8 Extend `usr_27.txt` a bit. (Adam Seyfarth)

- 9 Make the Reference Manual more precise. For each command mention:
 - change to cursor position and curswant
 - if it can be undone (u/CTRL-R) and redone (.)
 - how it works for folded lines
 - how it works with multibyte characters
- 8 Spread the windows commands over the other files. For example, ":stag" should be with ":tag". Cross-link with tags to avoid too much double text.
- 8 Add tags for all features, e.g. "gui_running".
- 7 MS-Windows: When a wrong command is typed with an ALT key, give a hint to look at the help for '**winaltkeys**'.
- 7 Add a help.vim plugin that maps <Tab> to jump to the next tag in || and <C-Tab> (and <S-Tab>) to the previous tag.
Patch by Balazs Kezes, 2007 Dec 30. Remark from A. Politz.
- Check text editor compendium for vi and Vim remarks.

Help:

- First try using the ":help" argument literally, before using it as a pattern. And then match it as part of a tag.
- When a help item has multiple matches make it possible to use ":tn" to go to the other matches.
- Support a way to view (and edit) .info files.
- Implement a "sticky" help window, some help text lines that are always displayed in a window with fixed height. (Guckes) Use "~/.vimhelp" file, user can edit it to insert favorite commands, new account can contain a default contents.
- Make '**winminheight**' a local option, so that the user can set a minimal height for the help window (and other windows).
- ":help :s^I" should expand to ":help :substitute".
- Make the help key (<F1>) context sensitive?
- Learn mode: show short help while typing commands.

User Friendlier:

- 8 Windows install with install.exe: Use .exe instead of .bat files for links, so that command line arguments are passed on unmodified? (Walter Briscoe)
- 8 Windows install: Be able to associate Vim with a selection of file types?
- 8 Windows uninstall: Have uninstal.c delete the vimfiles directories that dosinst.c creates. List the contents of the directory (recursively) if the user asks for it. Requires an implementation of "rm -rf".
- 8 Remember the name of the vimrc file that was used (~/.vimrc, \$VIM/_vimrc, \$HOME/_vimrc, etc.) and add "edit vimrc" to the File menu.
- Add a way to save local settings and mappings into a new plugin file.
":mkplugin <file>"?
 - Add mappings local to a window: ":map <window> ..." #9339
- 9 Add buffer-local menu. Should offer a choice between removing the menu or disabling it. Be careful that tear-offs don't disappear (keep one empty item?).
Alternative: use BufEnter and BufLeave autocommands.
- 8 make a vimtutor script for Amiga and other systems.
- 7 When Vim detects a file is being edited elsewhere and it's a gvim session of the same user it should offer a "Raise" button, so that the other gvim

window can be displayed. (Eduard)

- 8 Support saving and restoring session for X windows? It should work to do ":mksession" and use "-S fname" for the restart command. The gui_x11_wm_protocol_handler() already takes care of the rest. global_event_filter() for GTK.

Tab pages:

- 9 GUI implementation for the tab pages line for other systems.
- 7 GUI: Control over the appearance of the text in the labels (bold, color, font, etc.)
- 8 Make GUI menu in tab pages line configurable. Like the popup menu.
- 8 balloons for the tab page labels that are shortened to show the full path.
- 7 :tabdup duplicate the tab with all its windows.
- 7 Option to put tab line at the left or right? Need an option to specify its width. It's like a separate window with ":tabs" output.
- 8 Add local options for each tab page? E.g., '**diffopt**' could differ between tab pages.
- 7 Add local highlighting for each tab page?

Spell checking:

- Support more regions? Caolan McNamara argues it's needed for es_XX.
https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=219777
- Unicode defines another quote character: 0x2019. Use it as an equivalent of a single quote, thus use it as a word character like a quote and match with words, replacing the curly quote with a single quote.
- Could filter ´ things for HTML before doing spell checking. Similarly for TeX.
- The Hungarian spell file uses four extra characters in the FOL/UPP/LOW items than other spell files with the ISO-8859-2 encoding, that causes problem when changing '**spelllang**'. There is no obvious way to fix this.
- Considering Hunspell 1.1.4:
What does MAXNGRAMSUGS do?
Is COMPLEXPREFIXES necessary when we have flags for affixes?
- There is no Finnish spell checking file. For openoffice Voikko is now used, which is based on Malaga: <http://home.arcor.de/bjoern-beutel/malaga/> (Teemu Likonen)
- 8 ":mkspell" still takes much too long in Hungarian dictionary from hunspell. Only solution appears to be to postpone secondary suffixes.
- 8 Handle postponed prefix with COMPOUNDPERMITFLAG or COMPOUNDFORBIDFLAG.
WFP_COMPPERMIT and WFP_COMPFORBID
- 8 implement use of <compoptions> in .spl file:
implement CHECKCOMPOUNDREP: when a compound word seems to be OK apply REP items and check if the result is a valid word.
implement CHECKCOMPOUNDDUP
implement CHECKCOMPOUNDTRIPLE
Add CHECKCOMPOUNDCASE: when compounding make leading capital lower case.
How is it supposed to work?
- Add a command the repeats]s and z=, showing the misspelled word in its context. Thus to spell-check a whole file.
- suggestion for "KG" to "kg" when it's keeppcase.
- For flags on affixes: Use a "AFFCOMPSET" flag; means the compound flags of the word are not used.

- Support breakpoint character ? 0xb7 and ignore it? Makes it possible to use same wordlist for hyphenation.
- Compound word is accepted if nr of words is <= COMPOUNDWORDMAX OR nr of syllables <= COMPOUNDSYLMAX. Specify using AND in the affix file?
- NEEDCOMPOUND also used for affix? Or is this called ONLYINCOMPOUND now? Or is ONLYINCOMPOUND only for inside a compound, not at start or end?
- Do we need a flag for the rule that when compounding is done the following word doesn't have a capital after a word character, even for Onecap words?
- New hunspell home page: <http://hunspell.sourceforge.net/>
 - Version 1.1.0 is out now, look into that.
 - Lots of code depends on LANG, that isn't right. Enable each mechanism in the affix file separately.
 - Example with compounding dash is bad, gets in the way of setting COMPOUNDMIN and COMPOUNDWORDMAX to a reasonable value.
 - PSEUDOROOT == NEEDAFFIX
 - COMPOUNDRROOT -> COMPOUNDED? For a word that already is a compound word Or use COMPOUNDED2, COMPOUNDED3, etc.
- CIRCUMFIX: when a word uses a prefix marked with the CIRCUMFIX flag, then the word must also have a suffix marked with the CIRCUMFIX flag. It's a bit primitive, since only one flag is used, which doesn't allow matching specific prefixes with suffixes.
Alternative:
PSFX {flag} {pchop} {padd} {pcond} {schop} {sadd}[/flags] {scond}
We might not need this at all, you can use the NEEDAFFIX flag and the affix which is required.
- When a suffix has more than one syllable, it may count as a word for COMPOUNDWORDMAX.
- Add flags to count extra syllables in a word. SYLLABLEADD1 SYLLABLEADD2, etc.? Or make it possible to specify the syllable count of a word directly, e.g., after another slash: /abc/3
- MORPHO item in affix file: ignore TAB and morphological field after word/flags and affix.
- Implement multiple flags for compound words and CMP item?
Await comments from other spell checking authors.
- Also see tklsPELL: <http://tkltrans.sourceforge.net/>
- 8 Charles Campbell asks for method to add "contained" groups to existing syntax items (to add @Spell).
Add ":syntax contains {pattern} add=@Spell" command? A bit like ":syn cluster" but change the contains list directly for matching syntax items.
- References: MySpell library (in OpenOffice.org).
<http://spellchecker.mozdev.org/source.html>
<http://whiteboard.openoffice.org/source/browse/whiteboard/lingucomponent/source/spell>
author: Kevin Hendricks <kevin.hendricks@sympatico.ca>
- 8 It is currently not possible to mark "can not" as rare, because "can" and "not" are good words. Find a way to let "rare" overrule "good"?
- 8 Make "en-rare" spell file? Ask Charles Campbell.
- 8 The English dictionaries for different regions are not consistent in their use of words with a dash.
- 7 Insert mode completion mechanism that uses the spell word lists.
- 8 Add hl groups to 'spelllang'?
:set spelllang=en_us,en-rare/SpellRare,en-math/SpellMath
More complicated: Regions with different languages? E.g., comments in English, strings in German (po file).

Diff mode:

- 9 When making small changes, e.g. deleting a character, update the diff. Possibly without running diff.
- 8 Also show difference with the file when editing started? Should show what can be undone. (Tom Popovich)

Folding:

- (commands still available: zI zJ zK zp zP zq zQ zV zy zY;
secondary: zB zS zT zZ, z=)
- Patch to make closed folds line up. (Charles Campbell, 2014 Sep 12)
Remark from Roland Eggner: does it cause crashes? (2014 Dec 12)
Updated patch by Roland Eggner, Dec 16
Updated patch from Charles, 2016 Jul 2
- Patch to make fold updates much faster. (Christian Brabandt, 2012 Dec)
- Patch to have the fold and sign column and at the last line of the buffer. (Marco Hinz, 2014 Sep 25)
Alternate suggestion: let all columns continue, also the number column.
- Patch to add FoldedLineNr highlighting: different highlighting for the line number of a closed fold. (eXerigumo Clanjor, 2013 Jul 15)
- Patch to use **'foldnestmax'** also for "marker" foldmethod. (Arnaud Lacombe, 2011 Jan 7)
- **'foldcolumn'** in modeline applied to wrong window when using a session. (Teemu Likonen, March 19)
- With foldmethod=syntax and nofoldenable comment highlighting isn't removed. (Marcin Szewczyk, 2017 Apr 26)
- The ":move" command does not honor closed folds. (Ryan Lue, #2351)
- When completion inserts the first match, it may trigger the line to be folded. Disable updating folds while completion is active? (Peter Odding, 2010 Jun 9)
- When **'foldmethod'** is "indent", adding an empty line below a fold and then indented text, creates a new fold instead of joining it with the previous one. (Evan Laforge, 2009 Oct 17)
- Add ":nofold". Range will apply without expanding to closed fold.
- 8 Vertical folds: looks like vertically split windows, but the cursor moves through the vertical separator, separator moves when scrolling.
- 8 Add "z/" and "z?" for searching in not folded text only. Or use a regexp item, so that it can be used in any pattern.
- 8 When a closed fold is displayed open because of **'foldminlines'**, the behavior of commands is still like the fold is closed. How to make the user aware of this?
- 8 Add an option **'foldskip'** with values like **'foldopen'** that specifies which commands skip over a closed fold.
- 8 "H" and "L" count buffer lines instead of window lines. (Servatius Brandt)
- 8 Add a way to add fold-plugins. Johannes Zellner has one for VB.
- 7 When using manual folding, the undo command should also restore folds.
- Allow completely hiding a closed fold. E.g., by setting **'foldtext'** to an empty string. Require showing a character in **'foldcolumn'** to avoid the missing line goes unnoticed.
How to implement this?
- When pressing the down arrow of a scrollbar, a closed fold doesn't scroll until after a long time. How to make scrolling with closed folds smoother?
- When creating a session, also store folds for buffers in the buffer list,

- using the wininfo in wi_folds.
- When currently editing the first file in the argument list the session file can contain:
 - args version.c main.c
 - edit version.c
- Can editing version.c twice be avoided?
- **'foldmethod'** "textobject": fold on sections and paragraph text objects.
- "zuf": undo change in manual fold. "zUf" redo change in manual fold. How to implement this?
- "zJ" command: add the line or fold below the fold in the fold under the cursor.
- **'foldmethod'** "syntax": "fold=3" argument: set fold level for a region or match.
- Apply a new foldlevel to a range of lines. (Steve Litt)

Multi-byte characters:

- When editing a file with both utf-8 and latin1 text Vim always falls back to latin1. Add a command to convert the latin1 characters to utf-8?
 - :unmix utf-8,latin1 filename
 Would only work when **'encoding'** is utf-8.
- 9 When the tail byte of a double-byte character is illegal (e.g., a CR), the display is messed up (Yasuhiro Matsumoto). Should check for illegal double-byte characters and display them differently (display each single byte).
- 9 **'fenc'** in modeline problem: add option to reload the file when **'fenc'** is set to a different value in a modeline? Option can be default on. Could it be done with an autocommand?
- 8 Add an item in **'fileencodings'** to check the first lines of a file for the encoding. See Python PEP: <http://www.python.org/peps/pep-0263.html>. To avoid getting a wrong encoding only accept something Emacs-like:
 - "-*- coding: enc-na_me.foo -*-" and " -*- coding= enc-na_me.foo -*-"
 Match with `"-*\-\\s*coding[:=]\\s*\\([:word:~_\\.\\+\\)\\s**\-"` and use first item.
- 8 Add an item in **'fileencodings'** to check the first line of an XML file for the encoding. `<?xml version="1.0" encoding="UTF-8"?>` Or "charset=UTF-8"? For HTML look for "charset=utf-8".
- 8 When a file was converted from **'fileencoding'** to **'encoding'**, a tag search should also do this on the search pattern. (Andrzej M. Ostruszka)
- 8 When filtering changes the encoding **'fileencoding'** may not work. E.g., when using xxd and **'fileencoding'** is "utf-16". Add an option to set a different fileencoding for filter output?
- 7 When converting a file fails, mention which byte could not be converted, so that the user can fix the problem.
- 8 Add configure option to be able to disable using the iconv library. (Udo Schweigert)
- 9 **'aleph'** should be set to 1488 for Unicode. (Zvi Har'El)
- 8 Should add test for using various commands with multibyte characters.
- 8 **'infercase'** doesn't work with multibyte characters.
- 8 toupper() function doesn't handle byte count changes.
- 7 Searching and composing characters:
 - When searching, should order of composing characters be ignored?
 - Add a special item to match with a composing character, so that composing characters can be manipulated.
- 8 Should implement **'delcombine'** for command line editing.

- 8 Detect overlong UTF-8 sequences and handle them like illegal bytes.
- 8 ":s/x/\u\1/" doesn't work, making uppercase isn't done for multibyte characters.
- 8 UTF-8: "r" in Visual mode doesn't take composing characters.
- 8 UTF-8: When there is a precomposed character in the font, use it instead of a character and a composing character. See xterm for an example.
- 7 When a character can't be displayed, display its digraph instead. **'display'** option to specify this.
- 7 Use ideas for nl_langinfo() from Markus Kuhn in enc_default(): (www.cl.cam.ac.uk/~mgk25/ucs/langinfo.c)
- GTK and Win32: Allow selecting fonts for **'guifontset'** with the fontselector somehow.
- GTK and Win32: make it possible to set the font for the menu to make it possible to have **'encoding'** different from the current locale.
- dbcs_class() only works for Japanese and Korean. Implement this for other encodings. The "euc-jp" and "euc-kr" choices might be wrong.
- Find some way to automatically select the right GUI font or fontset, depending on the default value of **'encoding'**. Irrelevant in the GTK+ 2 GUI so long as UTF-8 is used. For Windows, the charset_pairs[] table could be used. But how do we know if a font exists?
- Do keyboard conversion from **'termencoding'** to **'encoding'** with convert_input() for Mac GUI.
- Add mnemonics from RFC1345 longer than two characters. Support CTRL-K **_{mnemonic}_**
- Make **'breakat'** accept multibyte characters. Problem: can't use a lookup table anymore (breakat_flags[]). Simplistic solution: when **'formatoptions'** contains "m" also break a line at a multibyte character $\geq 0x100$. Making breakat support multibyte characters (Yasuhiro Matsumoto, #6598) Scroll doesn't work correctly, why?
- Add the possibility to enter mappings which are used whenever normal text could be entered. E.g., for "f" command. But not in Normal mode. Sort of opposite of **'langmap'**. Use ":amap" command?
- When breaking a line, take properties of multibyte characters into account. The "linebreak" program from Bruno Haible can do it: <ftp://ftp.ilog.fr/pub/Users/haible/gnu/linebreak-0.1.tar.gz> But it's very complicated...
- Problem with **'langmap'** being used on the rhs of a mapping. (Nikolai Weibull, 2008 May 14). Possibly related problem: Alexey Muranov, 2015 Apr 2

Printing:

- 7 Implement "undercurl" for printing.
- Add "page width" to wrap long lines.
- Win32: use a font dialog for setting **'printfont'**. Can reuse the code for the **'guifont'** dialog, put the common code in a separate function.
- Add the file timestamp to the page header (with an option). (George Reilly)
- Win32: when **'printfont'** is empty use **'guifont'**.
- Unix: Use some dialog box to do the obvious settings (paper size, printer name, portrait/landscape, etc).
- PostScript: Only works for text that can be converted to an 8-bit

- character set. How to support Unicode fully?
- Allow specifying the paper size, instead of using a standard size. Same units as for the margins.
- Support right-to-left text?
- 8 Make the foreground color darkening function preserve the hue of the color.

Syntax highlighting:

- 8 Make `":syn off"` use `'runtimepath'` instead of `$VIMRUNTIME`. (Gary Johnson) Should do the same for `":syn on"` and `":syn manual"`.
- 8 Support `"containedin"` argument for `":syn include"`, so that the defined cluster can be added to existing syntax items.
- 8 C syntax: Don't highlight `{ }` as errors inside `()` when used like this: `"({ something })"`, often used in GCC code.
- 7 Add a `"startgroup"` to a region. Used like `"nextgroup"` inside the region, preferred item at the start of the region. (Charles Campbell)
- 8 When editing a new file without a name and giving it a name (by writing it) and `'filetype'` is not set, detect the filetype. Avoid doing it for `":wq file"`.
- 7 For `"nextgroup"` we have `skipwhite`, `skipnl` and `skipempty`. It would be really nice to be able to skip with a pattern. Or skip with a syntax group. (Nikolai Weibull, 2007 Feb 27)
- 8 Make conversion to HTML faster (Write it in C or pre-compile the script).
- 9 There is still a redraw bug somewhere. Probably because a cached state is used in a wrong way. I can't reproduce it...
- 7 Be able to change only the background highlighting. Useful for Diff* and Search highlighting.
- 7 When `'number'` is set highlight the number of the current line. Must be enabled with an option, because it slows down display updating.
- 8 Allow the user to add items to the Syntax menu sorted, without having to change this for each release.
- 8 Add a `"matchcontains"` for regions: items contained in the start or end pattern, but not in the body.
- 8 Add a `"keepend-contained"` argument: Don't change the end of an item this one is contained in. Like `"keepend"` but specified on the contained item, instead of the containing item.
- 8 `cpp.vim`: In C++ it's allowed to use `{ }` inside `()`.
- 8 Some syntax files set `'iskeyword'`, they should use `"syn iskeyword"`. Also need a separate `'iskeyword'` for the command line, e.g., in a help window `":e /asdf/asdf/"` **CTRL-W** works different.
- 8 Add specific syntax item to match with parens/braces that don't have a `"%"` match. `:syntax nomatch cMatchError (,{,[,)},,] [contained]`
- 8 Highlight the text between two matching parens (e.g., with a grey background) when on one of the parens or in between them. Option for the `matchparen` plugin?
- 8 When using a `cterm`, and no `ctermfg` or `ctermbg` are defined, use start/stop sequences. Add remark in docs that `:if 'term' == "term-name"` should be used.
- 8 Add `@spell` cluster to String and Comment groups for many languages. Will allow spell checking. (Fleiner)
- 8 When listing syntax items, try to sort the keywords alphabetically. And re-insert the `[]` if possible.
- 8 Make it possible to use color of text for Visual highlight group (like for

- the Cursor).
- 8 It would be useful to make the highlight group name an expression. Then when there is a match, the expression would be evaluated to find out what highlight group to use. Could be used to check if the shell used in a password file appears in /etc/shells. (Nikolai Weibull)
 - syn match =s:checkShell(v:match) contained **'pattern'**
 - 8 Make it possible to only highlight a sub-expression of a match. Like using "\1" in a ":s" command.
 - 8 Support for deleting syntax items:
 - :syn keyword cTodo remove this
 - :syn match cTodo remove "pattern"
 - :syn region cString remove start="this" end="that"
 - 8 Add possibility to sync on something else, when the syncing in one way doesn't find match. For HTML: When no {script} is found, try looking for a '<'. (Fleiner)
 - 7 Replace the synchronizing method with a state machine specification? Should be able to start at any line in the file, search forwards or backwards, and use the result of matching a pattern.
 - 7 Use parsing like awk, so that e.g., a (without a matching) can be detected.
 - 8 Make it possible to use "inverted" highlighting, invert the original character. For Visual mode. (xterm-selection already does this).
 - 8 Highlight non-printable characters with "SpecialChar", linked to "Special". Display them with the digraph characters, if possible.
 - 8 Highlight the clipboard-selection with a highlight group.
 - 8 Be able to reset highlighting to its original (default) values.
 - 7 Be able to write current highlighting to a file as commands, similar to ":mkvimrc".
 - 8 Improve c.vim:
 - Add check for unterminated strings, with a variable to switch it on: "c_strict_ansi".
 - Detect unbalanced "#endif". Requires looking back a long way...
 - 8 Add an option to restrict the updating of syntax highlighting to the current line while in Insert mode.
 - 8 When guessing value of **'background'**, the syntax file has already been loaded (from the .gvimrc). After changing **'background'**, load it again?
 - 8 Add ":syn resync" command, to re-parse the whole file until the current display position.
 - 8 Should support "me" offset for a region start pattern. To be used to allow searching for the end pattern inside the match of the end pattern. Example: syn region pikeXX start="([^{]" end=")" should work on "()".
 - 8 When using a regexp for "contains=", should delay matching with it until redrawing happens. Set a flag when a group is added, check this flag when highlighting starts.
 - 7 It's possible for an item to be transparent, so that the colors of an item lower on the stack is used. Also do this with highlighting, so that the user can set transparent highlighting? E.g. a number in a C comment would get the color of a comment, a number in an assignment Normal. (Nikolai Weibull)
 - 7 Add "semitrans": Add highlighting. E.g., make the text bold, but keep the colors. And add colors, so that Green+Red becomes Yellow. E.g. for this html:
 - bold text <I> italic+bold text italic text </I>
 - 7 **CTRL-]** checks the highlight group for finding out what the tag is.

- 7 Add an explanation how a list of words can be used to highlight misspelled words.
- 7 Should find a better way to parse the :syntax and :highlight commands. Use tables or lists that can be shared by parsing for execution and completion?
- 8 Add ColorSchemePost autocommand event, so that scripts can set up their highlighting. (Salman Halim)
- 7 Add a few sets of colors (e.g. Borland Turbo C one). With a menu to select one of the sets.
- 8 Add offsets to sub-matches: "\(a*\) *"he=e1-1
'e' is end of match 'e1' is end of sub-match 1, 's2' is start of submatch 2, etc.
- 8 In Insert mode, when there are typeahead characters, postpone the highlighting (for "." command).
- 8 Syncing on comments isn't 100% correct when // lines mix with /* and */. For example: What about a line that starts with // and contains */?
- 8 Ignore /* and */ inside strings, when syncing.
- 7 Build a few more syntax files from the file "/usr/share/misc/vgrindefs": ISP, LDL, Icon, ratfor. And check "nedit/source/highlight.c".
- 6 Add possibility to have background color continue until the right edge of the window. Useful for comment blocks and function headings. (Rogall)
- Make it possible to add "contains" items for all items in a group. Useful when extending an already existing syntax file.
- Add line-continuation pattern for non-syncing items too?
- Add possibility to highlight the whole line, including the right margin (for comment blocks).
- Add '**hlmatch**' option: List of flags:
'c': highlight match for character under the cursor.
'b': highlight the previous (, and its match.
'a': highlight all text from the previous (until its match.
Also for {}, <>, etc.?
'e': highlight all braces without a match (slow?)
OR: add an argument "cursor" to the syntax command, which means that the region/match/keyword is only highlighted when the cursor is on it.
(Campbell)
Or do it like Elvis: define text objects and how to highlight them around the cursor. (Iain Truskett)
- 7 Make it possible to use all words in the tags files as Keyword. Can also be done with a script (but it's slow).
- 7 Make it possible to call a ":" command when a match is found. Should allow for adding keywords from the text (e.g. variables that are set). And allows for sections with different highlighting.
- 7 Add highlight group for commandline: "Commandline". Make sure it highlights the command line while typing a command, and any output from messages. And external commands?
- 8 Make a version that works like less, but with highlighting: read stdin for text, exit at end of file, don't allow editing, etc. moreim? lessim?
- 7 SpecialKey highlighting overrules syntax highlighting. Can't give an unprintable char another color. Would be useful for ^M at end of line.
- Syntax highlight for a region does not work with a "nextgroup" if the start match is empty. #8449
- The :syntax cchar value can only be a single character. It would be useful to support combining characters. (Charles Campbell) Also #4687
- Syntax highlighting slow (hangs) in SASS file. (Niek Bosch, 2013 Aug 21)

- Several syntax file match "`^\s*`" which may get underlined if that's in the highlight group. Add a "`\zs`" after it?
- patch to add "combine" flag to syntax commands. (so8res, 2012 Dec 6)
- Patch to add "combine" to `:syntax`, combines highlight attributes. (Nate Soares, 2012 Dec 3)
- Syntax update problem in one buffer opened in two windows, bottom window is not correctly updated. (Paul Harris, 2012 Feb 27)
- Syntax region with '`concealends`' and a '`cchar`' value, '`conceallevel`' set to 2, only one of the two ends gets the cchar displayed. (Brett Stahlman, 2010 Aug 21, Ben Fritz, 2010 Sep 14)
- Using "`syn sync breaklines=2`" works for when text is changed, but not when scrolling or redrawing. Should start search for syntax patterns above the first drawn line. (#8103)
- Syntax priority problem. (Charles Campbell, 2011 Sep 15)
- Syntax highlighting wrong for transparent region. (Doug Kearns, 2007 Feb 26)
- Bug in using a transparent syntax region. (Hanlen in vim-dev maillist, 2007 Jul 31)
- Syntax HL error caused by "containedin". (Peter Hodge, 2006 Oct 6)
- Open two windows on the same C code, delete a ")" in one window, resulting in highlighted "{" in that window, not in the other.
- When using "nextgroup" and the group has an empty match, there is no search at that position for another match. (Lukas Mai, 2008 April 11)

Vim script language:

- 8 Make the filename and line number available to script functions, so that they can give useful debugging info. The whole call stack would be ideal. At least use this for error messages.
- 7 Execute a function with standard option values. No need to save and restore option values. Especially useful for new options. Problem: how to avoid a performance penalty (esp. for string options)?
 - range for `:exec`, pass it on to the executed command. (Webb)
- 7 `:include` command: just like `:source` but doesn't start a new scriptID? Will be tricky for the list of script names.
- 8 Have a look at VSEL. Would it be useful to include? (Bigham)
- 8 Have a prefix for a function to make it unique. When using packages it can be the plugin name. Perhaps also have a way to remove everything that the package added? including autocommands.
- 7 Pre-parse or compile Vim scripts into a bytecode, like `:def` functions. Possibilities:
 1. Put the bytecode with the original script, with an `:if has('bytecode-1234')` around it, so that it's only used with a Vim that supports the version. Update the code with a command, can be used in an autocommand.
 2. Use a ".vic" file (like Python use .pyc). Create it when writing a .vim file. Problem: distribution, non-writable directory, etc.
 3. Use a cache directory for each user. Disadvantage: cache lookup may cost more time than bytecode wins.
- 7 Add argument to `winwidth()` to subtract the space taken by '`foldcolumn`', signs and/or '`number`'.
- 6 Add ++ and -- operators? They only work on variables (lvals), how to implement this?

- 8 Add functions:
- has(":command") Check if ":command" works. compare function with "ex_ni". E.g. for ":simalt".
 - escape() Add argument to specify what to escape with.
 - modestack() Instead of just the current mode return the stack of Insert / **CTRL-O** / :normal things.
 - realname() Get user name (first, last, full)
 - user_fullname() patch by Nikolai Weibull, Nov 3 2002
 - Only add this when also implemented for non-Unix systems, otherwise a shell cmd could be used.
 - get_user_name() gets login name.
 - menuprop({name}, {idx}, {what}) Get menu property of menu {name} item {idx}.
menuprop("", 1, "name") returns "File".
menuprop("File", 1, "n") returns "nmenu File.Open..." argument.
Patch by Ilya Sher, 2004 Apr 22
 - Return a list of menus and/or a dictionary with properties instead.
 - mapname({idx}, mode) return the name of the idx'th mapping.
Patch by Ilya Sher, 2004 Mar 4.
 - Return a list instead.
 - char2hex() convert char string to hex string.
 - crypt() encrypt string
 - decrypt() decrypt string
 - base64enc() base 64 encoding
 - base64dec() base 64 decoding
 - attributes() return file protection flags "drwxrwxrwx"
 - filecopy(from, to) Copy a file
 - shorten(fname) shorten a file name, like home_replace()
 - perl(cmd) call Perl and return string
 - inputrl() like input() but right-to-left
 - typed() return the characters typed and consumed (to find out what happened)
 - virtualmode() add argument to obtain whether "\$" was used in Visual block mode.
 - getacp() Win32: get codepage (Glenn Maynard)
 - libcall() Allow more than one argument.
 - libcallext() Like libcall(), but using a callback function to allow the library to execute a command or evaluate an expression.
- 7 Make bufname("'0") return the buffer name from mark '0. How to get the column and line number? col("'0") currently returns zero.
- 8 argc() returns 0 when using "vim -t tag". How to detect that no file was specified in any way? To be able to jump to the last edited file.
- 8 Pass the command line arguments to Vim scripts in some way. As v:args List? Or extra parameter to argv()?
- 8 Add command arguments with three dashes, passed on to Vim scripts.
- 6 User functions: Functions local to buffer "b:func()"
- 8 For Strings add ":let var[{expr}] = {expr}". When past the end of "var" just ignore.
- 8 The "=" register should be writable, if followed by the name of a variable,

- option or environment variable.
- 8 ":let &option" should list the value of the option.
- 8 ":let Func().foo = value" should work, also when "foo" doesn't exist. Also: ":let Func()[foo] = value" should work. Same for a List.
- 7 Add synIDlist(), making the whole list of syntax items on the syntax stack available as a List.
- 8 Add autocommand-event for when a variable is changed:
 - :au VarChanged {varname} {commands}
- 8 Add "has("gui_capable")", to check if the GUI can be started.
- 8 Add possibility to use variables like registers: characterwise (default), linewise (when ending in '\n'), blockwise (when ending in '\001'). reg0, rega, reg%, etc. Add functions linewise({expr}), blockwise({expr}) and charwise({expr}).
- 7 Make it possible to do any command on a string variable (make a buffer with one line, containing the string). Maybe add an (invisible) scratch buffer for this?
 - result = scratch(string, command)
 - result = apply(string, command)
 - result = execute(string, command)
 "command" would use <> notation.
 - Does scratch buffer have a number? Or re-use same number?
- 7 Add function to generate unique number (date in milliseconds).

Robustness:

- 6 Add file locking. Lock a file when starting to edit it with flock() or fcntl(). This patch has advisory file locking while reading/writing the file for Vim 5.4: ~/vim/patches/kahn_file_locking . The patch is incomplete (needs support for more systems, autoconf). Andy doesn't have time to work on it. Disadvantage: Need to find ways to gracefully handle failure to obtain a lock. When to release a lock: When buffer is unloaded?

Performance:

- 7 For string variables up to 3 bytes don't allocate memory, use v_list itself as a character array. Use VAR_SSTRING (short string).
- 7 Add '**lazysize**' option: Above this size Vim doesn't load everything before starting to edit a file. Things like '**fileencodings**' only work up to this size, modelines only work at the top. Useful for large log files where you only want to look at the first few pages. Use zero to disable it.
- 8 move_lines() copies every line into allocated memory, making reloading a buffer a lot slower than re-editing the file. Can the memline be locked so that we don't need to make a copy? Or avoid invoking ml_updatechunk(), that is taking a lot of time. (Ralf Wildenhues, 2008 Jul 7) With a patch, but does it work?
- 8 Turn b_syn_ic and b_syn_containedin into b_syn_flags.
- 9 Loading menu.vim still takes quite a bit of time. How to make it faster?
- 8 in_id_list() takes much time for syntax highlighting. Cache the result?
- 7 setpcmark() shifts the jumplist, this takes quite a bit of time when jumping around. Instead use an index for the start?
- 8 When displaying a space with only foreground highlighting, it's the same as a space without attributes. Avoid displaying spaces for the "~" lines when starting up in a color terminal.

- 8 Avoid alloc() for scratch buffer use, esp. in syntax.c. It's very slow on Win16.
- 8 Profiling shows that in_id_list() is used very often for C code. Can this function be improved?
- 8 For an existing file, the page size of the swap file is always the default, instead of using the block size of the device, because the swap file is created only after setting the block size in mf_open(). How can this be improved?
- 8 Set default for **'ttyscroll'** to half a screen height? Should speed up MS-DOS version. (Negri)
- 7 C syntax highlighting gets a lot slower after ":set foldmethod=syntax". (Charles Campbell) Inserting a "{" is very slow. (dman)
- 7 HTML syntax highlighting is slow for long lines. Try displaying <http://www.theregister.co.uk/content/4/22908.html>. (Andre Pang)
- 7 Check how performance of loading the wordlist can be improved (adding a lot of abbreviations).
- 7 Compile Ex commands to byte codes. Store byte codes in a vim script file at the end, after "compiled:". Make it look like a single comment line for old Vim versions. Insert first line "Vim script compiled <timestamp>". Only used compiled code when timestamp matches the file stat. Add command to compile a vim script and add it to the file in-place. Split Ex command executing into a parsing and executing phase. Use compiled code for functions, while loops, etc.
- 8 When defining autocommands (e.g., from \$VIMRUNTIME/filetype.vim), need to compare each pattern with all existing patterns. Use a hash code to avoid using strcmp() too often?
- 7 Include turbo_loader patches, speeding up reading a file? Speed up reading a file by reading it into a fixed-size buffer, creating the list of indexes in another buffer, and then copying the result into a memfile block with two copies. Then read the next block into another fixed-size buffer, create the second list of indexes and copy text from the two blocks to the memfile block.
- 7 do_cmdline(): Avoid that the command line is copied to allocated memory and freed again later all the time. For while loops, and for when called with an argument that can be messed with. Generic solution: Make a struct that contains a pointer and a flag that indicates if the pointer should be freed when replaced.
- 7 Check that the file size is not more than "sizeof(long)".
- Further improve finding mappings in maphash[] in vgetorpeek()
- 8 Syntax highlighting is slow when deleting lines. Try in \$VIMRUNTIME/filetype.vim.
- "out of memory" after deleting (1,\$d) and changing (:%s/^/> /) a lot of lines (27000) a few times. Memory fragmentation?
- Have a look at how pdksh does memory allocation (alloc.c). (Dalecki)
- Do profiling on:
 - :g/pat/normal cmd
 - deleting 10Mbyte worth of lines (netscape binary)
 - "[i" and "[d" (Yegappan Lakshmanan)
 - ":g/^/m0" on a 450Kbyte file. And the "u".
 - highlighting "~/vim/test/longline.tex", "~/vim/test/scwoop.tcl" and "~/vim/test/lockup.pl".
 - loading a syntax file to highlight all words not from a dictionary.
 - editing a Vim script with syntax highlighting on (loading vim.vim).
- 7 Screen updating can be further improved by only redrawing lines that were

changed (and lines after them, when syntax highlighting was used, and it changed).

- On each change, remember start and end of the change.
- When inserting/deleting lines, remember begin, end, and line count.
- Use macros/duarte/capicua for profiling. Nvi 1.71 is the fastest!
- When using a file with one long line (1Mbyte), then do "\$hhhh", is still very slow. Avoid calling getvcol() for each "h"?
- Executing a register, e.g. "1000000@" is slow, because ins_typebuf has to move the previous commands forward each time. Pass count from normal_cmd() down to do_execreg().
- Avoid calls to plines() for cursor line, use w_cline_height.
- After ":set nowrap" remove superfluous redraw with wrong hor. offset if cursor is right of the screen.
- 8 Make **CTRL-C** on Unix generate a signal, avoid using select() to check for a **CTRL-C** (it's slow).

Code size:

- 8 GUI: When NO_CONSOLE is defined, more code can be excluded.
- Put getline() and cookie in a struct, so only one argument has to be passed to do_cmdline() and other functions.
- 8 Make a GUI-only version for Unix?
- 8 In buf_write _() isn't needed when setting errmsg, do it once when using it.
- 7 When compiling with a GUI-only version, the code for cterm colors can be left out.
- 8 When compiled with a GUI-only version, the termcap entries for terminals can be removed.
- 8 Can the check for libelf in configure.ac be removed?

Messages:

- 8 When using ":q" in a changed file, the error says to "add !". Add the command so that beginners understand it: "use :q!".
- 8 For '**verbose**' level 12 prints commands from source'd files. How to skip lines that aren't executed? Perhaps move the echoing to do_cmdline()?
- 8 Use '**report**' for ":bdel"? (Krishna) To avoid these messages when using a script.
- Delete message after new command has been entered and have waited for key. Perhaps after ten seconds?
- Make message history available in "msg" variables: msg1, msg2, .. msg9.
- 9 Check handling of overwriting of messages and delays:
Very wrong: errors while redrawing cause endless loop.
When switching to another file and screen scrolls because of the long message and return must be typed, don't scroll the screen back before redrawing.
- 8 When address range is wrong you only get "Invalid range". Be a bit more specific: Negative, beyond last line, reverse range? Include the text.
- 8 Make it possible to ignore errors for a moment ('errorignore?'). Another option to switch off giving error messages ('errorquiet?'). Also an option not to give any messages ('quiet')? Or ":quiet on", ":quiet off".
Careful: For a severe error (out of memory), and when the user starts typing, error messages must be switched back on.
Also a flag to ignore error messages for shell commands (for mappings).

- Option to set time for `emsg()` sleep. Interrupt sleep when key is typed? Sleep before second message?
- 8 In Ex silent mode or when reading commands from a file, what exactly is not printed and what is? Check `":print"`, `":set all"`, `":args"`, `":vers"`, etc. At least there should be no prompt. (Smulders) And don't clear the screen when reading commands from `stdin`. (Kendall)
 - > Make a difference between informative messages, prompts, etc. and error messages, printing text, etc.
- 8 Window should be redrawn when resizing at the hit-enter prompt. Also at the `":tselect"` prompt. Find a generic solution for redrawing when a prompt is present (with a callback function?).

Screen updating:

- `screen_line()`:
 - insert/delete character stuff.
 - improve delete rest of line (spaces at end of line).
- When moving or resizing window, try to avoid a complete redraw (esp. when dragging the status line with the mouse).
- When `'lazyredraw'` set, don't echo `:ex` commands? Need a flag to redraw when waiting for a character.
- 8 Add a `":refresh [winnr]"` command, to force updating a window. Useful from an event handler where `":normal"` can't be used. Also useful when `'lazyredraw'` is set in a mapping.

Scrolling:

- 8 Add `"zy"` command: scroll horizontally to put the cursor in the middle.
- 6 Add option to set the overlap for **CTRL-F** and **CTRL-B**. (Garhi)
 - extend `'scrollbind'` option: `'scrollopt'` words "search", "relative", etc.. Also 'e'xecute some commands (search, vertical movements) in all bound windows.
- 7 Add `'scrollbind'` feature to make the offset of one window with the next one equal to the window height. When editing one file in both windows it looks like each window displays a page of the buffer.
 - Allow scrolling by dragging with the mouse (grab a character and move it up/down). Like the "hand" in Acrobat reader. Use Alt-LeftMouse for this? (Goldfarb)
 - Add command to execute some commands (search, vertical movements) in all bound windows.
 - Add `'search'` option to `'scrollopt'` to allow `'scrollbind'` windows to be bound by regexp searches
 - Add `"z>"` and `"z<"`: scroll sideways one screenful. (Campbell)
 - Add option to set the number of lines when not to scroll, instead of the fixed number used now (for terminals that scroll slow with a large number of lines but not with a single line).

Autoconf:

- 8 Should use `acconfig.h` to define prototypes that are used by `autoheader`.
- 8 Some compilers don't give an error for `"-OPT:Olimit"` but a warning. (Webb) Add a check for the warning, so that `"Olimit"` can be added automatically?
- Autoconf: Use `@datadir@` for the system independent files. Make sure the system dependent and system independent files are separated. (Leitner).
- Add autoconf check for `waitpid()/wait4()`.

- Remove `fcntl()` from `autoconf`, all systems have it?
- Set default for `'dictionary'`, add search for dictionary to `autoconf`.

Perl interface:

- 8 Rename `typemap` file to something else?
- 7 Make buffers accessed as Perl arrays. (Clark)
- 7 Make it possible to compile with non-ANSI C?
- 6 Tcl/Tk has the `"load"` command: load a shared library (`.so` or `.dll`).

Shared libraries:

- 8 `libcall()` can keep the library around instead of always calling `dlclose()`. (Jason Felice, 2018 Mar 20)
- 6 Add support for loading shared libraries, and calling functions in it.
 - `:libload internal-name libname`
 - `:libunload internal-name`
 - `:liblist`
 - `:libcall internal-name function(arg1, arg2, ...)`
 - `:libcall function(arg1, arg2, ...)`

`libcall()` can have only one integer or String argument at the moment.
- 6 Have a look on how Perl handles loading dynamic libraries.

Tags:

- 9 With `":set tags=./tags,..../tags"` and a tag appears in both tags files it is added twice. Requires figuring out the actual file name for each found match. Remove `tag_fname` from the match and combine it with the `fname` in the match (without expanding or other things that take time). When `'tagrelative'` is off `tag_fname` isn't needed at all.
- 8 For `'tags'` wildcard in the file name is not supported, only in the path. This is due to it using `file-searching`. Suboptimal solution would be to make the filename or the whole option use `wildcards` globbing, better would be to merge the 2 kinds of globbing. originally (Erik Falor, 2008 April 18), updated (Ian Kelling, 2008 July 4)
- 7 Can `CTRL-]` (jump to tag) include a following `."` and `"->"` to restrict the number of possible matches? Check tags file for an item that has members. (Flemming Madsen)
- 8 Scope arguments for `":tag"`, e.g.: `":tag class:cPage open"`, like Elvis.
- 8 When output of `":tselect"` is long, getting the more-prompt, should be able to type the tag number directly.
- 7 Add the possibility to use the `"-t {tag}"` argument multiple times. Open a window for each tag.
- 7 Make output of `":tselect"` a bit nicer. Use highlighting?
- 7 Highlight the `"tag 1 of >2"` message. New highlight group, or same as `"hit bottom"` search message.
- 7 When using `":tag"` at the top of the tag stack, should add another entry, so `CTRL-T` can bring you back to where you are now AND to where you were before the previous `":tag"` command. (Webb)
 - When doing `"[^I"` or `"[^D"` add position to tag stack.
 - Add command to put current position to tag stack: `":tpush"`.
 - Add functions to save and restore the tag stack? Or a command to switch to another tag stack? So that you can do something else and come back to what you were working on.

- 7 When using **CTRL-]** on `someClass::someMethod`, separate class from method and use `":ta class:someClass someMethod"`.
Include C++ tags changes (Bertin). Change `"class::func"` tag into `"func"` with `"class=class"`? Docs in `oldmail/bertin/in.xxx`.
- 7 Add `":tagargs"`, to set values for fields:

```

:tagargs class:someclass file:version.c
:tagargs clear

```

These are then the default values (changes the order of priority in tag matching).
- 7 Support for `"gtags"` and `"global"`? With `":rtag"` command?
There is an example for how to do this in `Nvi`.
Or do it like Elvis: `'tagprg'` and `'tagprgonce'` options. (Yamaguchi)
The Elvis method is far more flexible, do it that way.
- 7 Support `"col:99"` extra field, to position the cursor in that column. With a flag in `'cptions'` to switch it off again.
- 7 Better support for jumping to where a function or variable is used. Use the `id-utils`, with a connection to `"gid"` (Emacs can do it too). Add `":idselect"`, which uses an `"ID"` database (made by `"mkid"`) like `"tselect"`.

Win32 GUI:

- 8 Make debug mode work while starting up (`vim -D`). Open console window for the message and input?
- 8 When using "Edit with Vim" for one file it changes directory, when several files are selected and using "Edit with single Vim" the directory isn't changed. At least change directory when the path is the same for all files. Perhaps just use the path of the first file or use the longest common part of the path.
- 8 Add font argument to set the `lfCharSet`. (Bobcik)
- 8 Somehow automatically detect the system language and set `$LANG`, so that `gettext` and menus work.
- 8 Could keep console open to run multiple commands, to avoid the need to hit return in every console.
Also: Look at how Emacs does run external commands:
<http://www.cs.washington.edu/homes/voelker/nemacs.html>.
- 8 Need a separate PopUp menu for modeless selection. Need two new commands: Copy selection to clipboard, Paste selection (as typed text).
- 8 Support copy/paste for other file formats. At least HTML, perhaps RTF. Add `"copy special"` and `"paste special"` commands?
- 7 Use different default colors, to match the current Windows color scheme. `Sys_WindowText`, `Sys_Window`, etc. (Lionel Schaffhauser)
- 7 Use `<C-Tab>` to cycle through open windows (e.g., the find dialog).
- 7 `<Esc>` should close a dialog.
- 7 Keep the console for external commands open. Don't wait for a key to be hit. Re-open it when the user has closed it anyway. Or use a prepended command: `":nowait {cmd}"`, or `":quiet"`, which executes `{cmd}` without any prompts.
- 7 Should be able to set an option so that when you double click a file that is associated with Vim, you can either get a new instance of Vim, or have the file added into an already running Vim.
- 7 The `"-P"` argument only works for the current codepage. Use wide functions to find the window title.

GUI:

- 7 Implement ":popup" for other systems than Windows.
- 8 Implement ":tearoff" for other systems than Win32 GUI.
- 6 Implement ":untearoff": hide a torn-off menu.
- 8 When using the scrollbar to scroll, don't move the cursor position. When moving the cursor: scroll to the cursor position.
- 9 Make <S-Insert> paste from the clipboard by default. (Kunze)
- 7 Menu local to a buffer, like mappings. Or local to a filetype?
- 8 In Buffers menu, add a choice whether selecting a buffer opens it in the current window, splits the window or uses ":hide".
- 8 Dragging the mouse pointer outside of a Vim Window should make the text scroll. Return a value from gui_send_mouse_event() to the machine specific code to indicate the time in which the event should be repeated.
- 8 Make it possible to ignore a mouse click when it's used to give Vim (gvim) window focus. Also when a mouse click is used to bring a window to front.
- 8 Make the split into system independent code and system specific code more explicit. There are too many #ifdefs in gui.c.
If possible, separate the Vim code completely from the GUI code, to allow running them in separate processes.
- 7 X11: Support cursorColor resource and "-cr" argument.
- 8 X11 (and others): CTRL-; is not different from !;'. Set the modifier mask to include CTRL for keys where CTRL produces the same ASCII code.
- 7 Add some code to handle proportional fonts on more systems? Need to draw each character separately (like xterm). Also for when a double-width font is not exactly double-width. (Maeda)
- 8 Should take font from xterm where gvim was started (if no other default).
- 8 Selecting font names in X11 is difficult, make a script or something to select one.
- 8 Visual highlighting should keep the same font (bold, italic, etc.).
- 8 Add flag to 'guioptions' to not put anything in the clipboard at all?
- 8 Should support a way to use keys that we don't recognize yet. Add a command that adds entries to special_keys somehow. How do we make this portable (X11, Win32, ..)?
- 7 Add a flag to 'guioptions' that tells not to remove inactive menu items. For systems where greying-out or removing menu items is very slow. The menu items would remain visibly normally, but not do anything.
- 7 Add ":minimize" and ":maximize", which iconize the window and back. Useful when using gvim to run a script (e.g. 2html.vim).
- 7 X11: Is it possible to free allocated colors, so that other programs can use them again? Otherwise, allow disabling allocating the default colors. Or allocate an own colormap (check UAE). With an option to use it. For the commandline, "--install" is mostly used for X11 programs.
- 7 Should support multi-column menus.
- Should add option for where to put the "Help" menu: like Motif at the far right, or with the other menus (but still at the right).
- Add menu item to "Keep Insert mode".
- 8 ":mkgvimrc" command, that includes menus.
- 6 Big change: Move GUI to separate program "vimgui", to make startup of vim a lot faster, but still be able to do "vim -g" or ":gui".
- 7 More explicit mouse button binding instead of 'mousemodel'?
- 7 Add option to set the position of the window on the screen. 'windowpos', which has a value of "123,456": <x>, <y>.
Or add a command, like ":winsize"?
- 7 Add toolbar for more GUIs.

- 8 Make it possible to use "amenu icon=BuiltIn##", so that the toolbar item name can be chosen free.
- 7 Make it possible to put the toolbar on top, left, right and/or bottom of the window? Allows for softkey-like use.
- 6 Separate the part of Vim that does the editing from the part that runs the GUI. Communicate through a pseudo-tty. Vim starts up, creates a pty that is connected to the terminal. When the GUI starts, the pty is reconnected to the GUI process. When the GUI stops, it is connected to the terminal again. Also use the pty for external processes, it looks like a vt100 terminal to them. Vim uses extra commands to communicate GUI things.
- 7 Motif: For a confirm() dialog <Enter> should be ignored when no default button selected, <Esc> should close the dialog.
- 7 When using a pseudo-tty Vim should behave like some terminal (vt52 looks simple enough). Terminal codes to/from shell should be translated.
- Would it be useful to be able to quit the GUI and go back to the terminal where it was started from?
- 7 Support "-visual <type>" command line argument.

Autocommands:

- 9 When triggering WinNew provide the window ID somehow. #10633
- 9 Rework the code from FEAT_OSFILETYPE for autocmd-osfiletypes to use 'filetype'. Only for when the current buffer is known.
- Put autocommand event names in a hashtable for faster lookup?
- 8 When the SwapExists event is triggered, provide information about the swap file, e.g., whether the process is running, file was modified, etc. Must be possible to check the situation that it's probably OK to delete the swap file. (Marc Merlin)
- 8 When all the patterns for an event are "*" there is no need to expand buffer names to a full path. This can be slow for NFS.
- 7 For autocommand events that trigger multiple times per buffer (e.g., CursorHold), go through the list once and cache the result for a specific buffer. Invalidate the cache when adding/deleting autocommands or changing the buffer name.
- 7 Add TagJump event: do something after jumping to a tag.
- 8 Add "TagJumpFile" autocommand: When jumping to another file for a tag. Can be used to open "main.gz" when "main.c" isn't found.
- 8 Use another option than 'updatetime' for the CursorHold event. The two things are unrelated for the user (but the implementation is more difficult).
- 7 Add autocommand event for when a buffer cannot be abandoned. So that the user can define the action taking (autowrite, dialog, fail) based on the kind of file. (Yakov Lerner) Or is BufLeave sufficient?
- 8 Autocommand for when modified files have been found, when getting input focus again (e.g., FileChangedFocus).
Check when: getting focus, jumping to another buffer, ...
- 8 Autocommands should not change registers. And marks? And the jumplist? And anything else? Add a command to save and restore these things.
- 8 Add autocommands, user functions and user commands to ":mkvimrc".
- 6 Add KeymapChanged event, so that the effects of a different keymap can be handled (e.g., other font) (Ron Aaron)
- 7 When trying to open a directory, trigger an OpenDirectory event.
- 7 Add file type in front of file pattern: <d> for directory, <l> for link,

- <x> for executable, etc. With commas to separate alternatives. The autocommand is only executed when both the file type AND the file pattern match. (Leonard)
- 5 Add option that specifies extensions which are to be discarded from the file name. E.g. **'ausuffix'**, with ".gz,.orig". Such that file.c.gz will trigger the "*.c" autocommands. (Belabas)
 - 7 Add something to break the autocommands for the current event, and for what follows. Useful for a "BufWritePre" that wants to avoid writing the file.
 - 8 When editing "tt.gz", which is in DOS format, **'fileformat'** stays at "unix", thus writing the file changes it. Somehow detect that the read command used dos fileformat. Same for **'fileencoding'**.
- Add events to autocommands:
 - Error - When an error happens
 - VimLeaveCheck - Before Vim decides to exit, so that it can be cancelled when exiting isn't a good idea.
 - CursorHoldC - CursorHold while command-line editing
 - WinMoved - when windows have been moved around, e.g, ":wincmd J"
 - SearchPost - After doing a search command (e.g. to do "M")
 - ShutDown - when the system is about to shut down
 - InsertCharPost - user typed a character in Insert mode, after inserting the char.
 - BufModified - When a buffer becomes modified, or unmodified (for putting a [+] in the window title or checking out the file from CVS).
 - BufFirstChange - When making a change, when **'modified'** is set. Can be used to do a :preserve for remote files.
 - BufChange - after a change was made. Set some variables to indicate the position and number of inserted/deleted lines, so that marks can be updated. HierAssist has patch to add BufChangePre, BufChangePost and RevertBuf. (Shah)
 - ViewChanged - triggered when the text scrolls and when the window size changes.
 - QuickfixList - when any entry in the current list changes or another list is selected
 - QuickfixPosition - when selecting another entry in the current quickfix list
 - Write the file now and then ('autosave'):
 - 'autosave'** **'as'** **'noautosave'** **'noas'**
 - 'autosave'** **'as'** number (default 0)
 - Automatically write the current buffer to file N seconds after the last change has been made and when **'modified'** is still set.
 - Default: 0 = do not autosave the buffer.
 - Alternative: have **'autosave'** use **'updatetime'** and **'updatecount'** but make them save the file itself besides the swapfile.
 - Buffer autocommands are a bit inconsistent. Add a separate set of autocommands for the buffer lifecycle:
 - BufIsCreated (after buffer ID exists)
 - BufIsLoaded (after buffer ID has content)
 - BufIsUnloaded (after buffer ID no longer has)
 - BufIsWiped (after buffer ID was wiped)
 - BufIsRenamed (after buffer ID gets another name)
- The buffer list and windows are locked, no changes possible

Omni completion:

- Add a flag to **'complete'** to be able to do omni completion with **CTRL-N** (and mix it with other kinds of completion).
- Ideas from the Vim 7 BOF at SANE:
 - For interpreted languages, use the interpreter to obtain information. Should work for Java (Eclipse does this), Python, Tcl, etc. Richard Emberson mentioned working on an interface to Java.
 - Check Readline for its completion interface.
- Ideas from others:
 - <http://www.wholetomato.com/>
 - http://www.vim.org/scripts/script.php?script_id=747
 - <http://sourceforge.net/projects/insenvim>
or <http://insenvim.sourceforge.net>
 - Java, XML, HTML, C++, JSP, SQL, C#
 - MS-Windows only, lots of dependencies (e.g. Perl, Internet explorer), uses .dll shared libraries.
 - For C++ uses \$INCLUDE environment var.
 - Uses Perl for C++.
 - Uses ctags to find the info:
`ctags -f $allTagsFile --fields=+aiKmnsSz --language-force=C++ --C++-kinds=`
 - www.vim.org script 1213 (Java Development Environment) (Fuchuan Wang)
 - IComplete: http://www.vim.org/scripts/script.php?script_id=1265
and <http://stud4.tuwien.ac.at/~e0125672/icomplete/>
 - <http://cedet.sourceforge.net/intellisense.shtml> (for Emacs)
 - Ivan Villanueva has something for Java.
 - Emacs: http://www.xref-tech.com/xrefactory/more_c_completion.html
 - Completion in .NET framework SharpDevelop: <http://www.icsharpcode.net>
- Pre-expand abbreviations, show which abbrevs would match?

Insert mode completion/expansion:

- Is it possible to keep the complete menu open when calling complete()? (Prabir Shrestha, 2017 May 19, #1713)
- When **'completeopt'** has "noselect" does not insert a newline. (Lifepillar, 2017 Apr 23, #1653)
- Can **'completeopt'** be made buffer-local? (#5487)
- When complete() first argument is before where insert started and **'backspace'** is Vi compatible, the completion fails. (Hirohito Higashi, 2015 Feb 19)
- The CompleteDone autocommand needs some info passed to it:
 - The word that was selected (empty if abandoned complete)
 - Type of completion: tag, omnifunc, user func.
- When a:base in **'completofunc'** starts with a number it's passed as a number, not a string. (Sean Ma) Need to add flag to call_func_relist() to force a string value.
- When editing the text and pressing **CTRL-N** again goes back to originally completed text, edited text is gone. (Peng Yu, 2008 Jul 24)
Suggestion by Ben Schmidt, 2008 Aug 6.
- GUI implementation of the popup menu.
- 7 When searching in other files the name flash by, too fast to read. Only display a name every second or so, like with ":vimgrep".
- 7 When expanding file names with an environment variable, add the match with

- the unexpanded var. So \$HOME/tmp expands to "/home/guy/tmp" and "\$HOME/tmp"
- 8 When there is no word before the cursor but something like "sys." complete with "sys.". Works well for C and similar languages.
 - 9 ^X^L completion doesn't repeat correctly. It uses the first match with the last added line, instead of continuing where the last match ended. (Webb)
 - 8 Add option to set different behavior for Insert mode completion:
 - ignore/match case
 - different characters than **'iskeyword'**
 - 8 Add option **'isexpand'**, containing characters when doing expansion (so that "." and "\" can be included, without changing **'iskeyword'**). (Goldfarb)
Also: **'istagword'**: characters used for **CTRL-]**.
When **'isexpand'** or **'istagword'** are empty, use **'iskeyword'**.
Alternative: Use a pattern so that start and end of a keyword can be defined, only allow dash in the middle, etc.
 - 8 Add a command to undo the completion, go back to the original text.
 - 7 Completion of an abbreviation: Can leave letters out, like what Instant text does: www.textware.com
 - 8 Use the class information in the tags file to do context-sensitive completion. After "foo." complete all member functions/variables of "foo". Need to search backwards for the class definition of foo. Should work for C++ and Java.
Even more context would be nice: "import java.^N" -> "io", "lang", etc.
 - 7 When expanding \$HOME/dir with ^X^F keep the \$HOME (with an option?).
 - 7 Add **CTRL-X** command in Insert mode like **CTRL-X CTRL-N**, that completes WORDS instead of words.
 - 8 Add **CTRL-X CTRL-R**: complete words from register contents.
 - 8 Add completion of previously inserted texts (like what **CTRL-A** does). Requires remembering a number of insertions.
 - 8 Add 'f' flag to **'complete'**: Expand file names.
Also apply **'complete'** to whole line completion.
 - Add a flag to **'complete'** to only scan local header files, not system header files. (Andri Moell)
 - Make it possible to search include files in several places. Use the **'path'** option? Can this be done with the dictionary completion (use wildcards in the file name)?
 - Make **CTRL-X CTRL-K** do a binary search in the dictionary (if it's sorted).
 - Speed up **CTRL-X CTRL-K** dictionary searching (don't use a regexp?).
 - Set a mark at the position where the match was found (file mark, could be in another file).
 - Add **CTRL-A** command in **CTRL-X** mode: show all matches.
 - Make **CTRL-X CTRL-L** use the **'complete'** option?
 - Add command in **CTRL-X** mode to add following words to the completed string (e.g. to complete "Pointer->element" with **CTRL-X CTRL-P CTRL-W CTRL-W**)
 - **CTRL-X CTRL-F**: Use **'path'** to find completions.
 - **CTRL-X CTRL-F**: Option to use forward slashes on MS-Windows?
 - **CTRL-X CTRL-F**: Don't replace "\$VIM" with the actual value. (Kelly)
 - Allow listing all matches in some way (and picking one from the list).

Command line editing:

- 7 Add commands (keys) to delete from the cursor to the end of the command line.

- 8 Custom completion of user commands can't use the standard completion functions. Add a hook to invoke a user function that returns the type of completion to be done: "file", "tag", "custom", etc.
- Add flags to **'whichwrap'** for command line editing (cursor right at end of lines wraps to start of line).
- Make editing the command line work like Insert mode in a single-line view on a buffer that contains the command line history. But this has many disadvantages, only implement it when these can be solved. Elvis has run into these, see remarks from Steve (~/Mail/oldmail/kirkendall/in.00012).
 - Going back in history and editing a line there would change the history. Would still need to keep a copy of the history elsewhere. Like the cmdwin does now already.
 - Use **CTRL-O** to execute one Normal mode command. How to switch to normal mode for more commands? <Esc> should cancel the command line. **CTRL-T**?
 - To allow "/" and "=" need to recursively call getcmdline(), overwrite the cmdline. But then we are editing a command-line again. How to avoid that the user gets confused by the stack of command lines?
 - Use edit() for normal cmdline editing? Would have to integrate getcmdline() into edit(). Need to solve conflicts between Insert mode and Command-line mode commands. Make it work like Korn shell and tcsh.
- Problems:
 - Insert: completion with **'wildchar'**
 - Insert: use cmdline abbreviations
 - Insert: **CTRL-D** deletes indent instead of listing matches
 - Normal: no **CTRL-W** commands
 - Normal: no ":" commands?
 - Normal: allow Visual mode only within one line.
- where to show insert/normal mode message? Change highlighting of character in first column?
- Implementation ideas:
 - Set "curwin" and "curbuf" to the command line window and buffer.
 - curwin->w_topline is always equal to curwin->w_cursor.lnum.
 - never set **'number'**, no folding, etc. No status line.
 - sync undo after entering a command line?
 - use NV_NOCL flag for commands that are not allowed in Command-line Mode.

Command line completion:

- Feature request: Complete members of a dictionary. (Luc Hermitte, 2017 Jan 4, #1350)
- Completion of ":e" is "earlier", should be "edit". Complete to the matching command instead of doing this alphabetically. (Mikel Jorgensen)
- 8 Change expand_interactively into a flag that is passed as an argument.
- 8 With command line completion after '%' and '#', expand current/alternate file name, so it can be edited. Also with modifiers, such as "%:h".
- 8 When completing command names, either sort them on the long name, or list them with the optional part inside [].
- 8 Add an option to ignore case when doing interactive completion. So that ":e file<Tab>" also lists "Filelist" (sorted after matching case matches).
- 7 Completion of ":map x ": fill in the current mapping, so that it can be edited. (Sven Guckes)
- For **'wildmenu'**: Simplify "../bar" when possible.
- When using <Up> in wildmenu mode for a submenu, should go back to the

- current menu, not the first one. E.g., ":emenu File.Save<Up>".
- 8 When using backtick expansion, the external command may write a greeting message. Add an option or commands to remove lines that match a regexp?
 - 7 When listing matches of files, display the common path separately from the file names, if this makes the listing shorter. (Webb)
 - Add command line completion for ":ilist" and friends, show matching identifiers (Webb).
 - 8 Add command line completion for "old value" of a command. ":args <key>" would result in the current list of arguments, which you can then edit.
 - 7 Add command line completion with **CTRL-X**, just like Insert mode completion. Useful for ":s/word/xx/".
 - Add command to go back to the text as it was before completion started. Also to be used for <Up> in the command line.
 - Add '**wildlongest**' option: Key to use to find longest common match for command line completion (default **CTRL-L**), like '**wildchar**'. (Cregut)
 - Also: when there are several matches, show them line a **CTRL-D**.
 - With '**wildmode**' set to "longest:full,full" and pressing Tab once the first entry in wildmenu is highlighted, that shouldn't happen. (Yuki Watanabe, 2011 Feb 12)
 - After using <Tab> for command line completion after ":ta blah" and getting E33 (no tags file), further editing the command to e.g., ":echo '**blah**'", the command is not executed. Fix by Ian Kelling?
 - Command line completion: Scanning for tags doesn't check for typed key now and then? Hangs for about 5 seconds. Appears to be caused by finding include files with "foo/**" in '**path**'. (Kalisiak, 2006 July 15)
 - Additional info: When using the **wildcards** ** globing, vim hangs indefinitely on lots of directories. The **file-searching** globing, like in ":set path=/**" does not hang as often as with globing with **wildcards**, like in ":!find /**/file". This is for files that unix "find" can find very quickly. Merging the 2 kinds of globing might make this an easier fix. (Ian Kelling, 2008 July 4)
 - Command line completion when '**cmdheight**' is maximum and '**wildmenu**' is set, only one buffer line displayed, causes display errors.
 - Completing with '**wildmenu**' and using <Up> and <Down> to move through directory tree stops unexpectedly when using ":cd " and entering a directory that doesn't contain other directories.
 - 8 Command line completion: buffers "foo.txt" and "../b/foo.txt", completing ":buf foo<Tab>" doesn't find the second one. (George V. Reilly)
 - 8 Add more command line completion for :syntax.
 - 8 Add more command line completion for :highlight.
 - Wildmenu not deleted: "gvim -u NONE", ":set nocp wildmenu cmdheight=3 laststatus=2", **CTRL-D CTRL-H CTRL-H CTRL-H**. (A.Politz, 2008 April 1) Works OK with Vim in an xterm.
 - If the variable "g:x#y#z" exists completion after ":echo g:x#" doesn't work.

Command line history:

- Add "KeyWasTyped" flag: It's reset before each command and set when a character from the keyboard is consumed. Value is used to decide to put a command line in history or not. Put line in history if it didn't completely result from one mapping.
- When using ":browse", also put the resulting edit command in the history, so that it can be repeated. (Demirel)

Insert mode:

- 9 When **'autoindent'** is set, hitting **<CR>** twice, while there is text after the cursor, doesn't delete the autoindent in the resulting blank line. (Rich Wales) This is Vi compatible, but it looks like a bug.
- 8 When using **CTRL-O** in Insert mode, then executing an insert command "a" or "i", should we return to Insert mode after **<Esc>**? (Eggink) Perhaps it can be allowed a single time, to be able to do "**<C-O>**10xyz**<Esc>**". Nesting this further is confusing. "**:map <F2> 5aabc<Esc>**" works only once from Insert mode.
- 8 When using **CTRL-G CTRL-O** do like **CTRL-\ CTRL-O**, but when returning with the cursor in the same position and the text didn't change continue the same change, so that "." repeats the whole insert.
- 7 Use **CTRL-G <count>** to repeat what follows. Useful for inserting a character multiple times or repeating **CTRL-Y**.
 - Make **'revins'** work in Replace mode.
- 7 Use **'matchpairs'** for **'showmatch'**: When inserting a character check if it appears in the rhs of **'matchpairs'**.
 - In Insert mode (and command line editing?): Allow undo of the last typed character. This is useful for **CTRL-U**, **CTRL-W**, delete and backspace, and also for characters that wrap to the next line. Also: be able to undo **CTRL-R** (insert register). Possibly use **'backspace'="whole"** for a mode where at least a **<CR>** that inserts autoindent is undone by a single **<BS>**.
 - Use **CTRL-G** in Insert mode for an extra range of commands, like "g" in Normal mode.
 - Make **'paste'** work without resetting other options, but override their value. Avoids problems when changing files and modelines or autocommands are used.
 - When typing **CTRL-V** and a digit higher than 2, only expect two digits.
 - Insert binary numbers with **CTRL-V b**.
 - Make it possible to undo **<BS>**, **<C-W>** and **<C-U>**. Bash uses **CTRL-Y**.

'cindent', **'smartindent'**:

- 9 Wrapping a variable initialization should have extra indent:

```
char * veryLongName =
    "very long string"
```

Also check if "cino=+10" is used correctly.
- 8 Lisp indenting: "\\\" confuses the indenter. (Dorai Sitaram, 2006 May 17)
- 8 Why are continuation lines outside of a {} block not indented? E.g.:

```
long_type foo =
value;
```
- 8 Java: Inside an anonymous class, after an "else" or "try" the indent is too small. (Vincent Bergbauer)
Problem of using {} inside (), **'cindent'** doesn't work then.
- 8 In C++ it's possible to have {} inside (): (Kirshna)

```
func(
    new String[] {
        "asdf",
        "asdf"
    }
);
```

8 In C++ a function isn't recognized inside a namespace:
(Chow Loong Jin)

```
namespace {
    int
        func(int arg) {
    }
}
```

6 Add **'cino'** flag for this function argument layout: (Spencer Collyer)

```
func( arg1
      , arg2
      , arg3
    );
```

7 Add separate "(0" option into inside/outside a function (Zellner):

```
func(
    int x)          // indent like "(4"
{
    if (a
        && b)      // indent like "(0"
```

9 Using "{" in a comment: (Helmut Stiegler)

```
if (a)
{
    if (b)
    {
        // {
    }
} <-- this is indented incorrect
```

Problem is that find_start_brace() checks for the matching brace to be in a comment, but not braces in between. Requires adding a comment check to findmatchlimit().

- Make smartindenting configurable. Add **'sioptions'**, e.g. '#' setting the indent to 0 should be switched on/off.
- 7 Support ANSI style function header, with each argument on its own line.
- "[p" and "]"p" should use **'cindent'** code if it's on (only for the first line).
- Add option to **'cindent'** to set indent for comments outside of {}?
- Make a command to line up a comment after a code line with a previous comment after a code line. Can **'cindent'** do this automatically?
- When **'cindent'**ing a '}', showmatch is done before fixing the indent. It looks better when the indent is fixed before the showmatch. (Webb)
- Add option to make indenting work in comments too (for commented-out code), unless the line starts with "*".
- Don't use **'cindent'** when doing formatting with "gq"?
- When formatting a comment after some text, insert the '*' for the new line (indent is correct if **'cindent'** is set, but '*' doesn't get inserted).
- 8 When **'comments'** has both "s1:/*,mb:*,ex:*/" and "s1:(*,mb:*,ex:*)", the 'x' flag always uses the first match. Need to continue looking for more matches of "*" and remember all characters that could end the comment.
- For smartindent: When typing **'else'** line it up with matching **'if'**.
- **'smartindent'**: allow patterns in **'cinwords'**, for e.g. TeX files, where lines start with "\item".
- Support this style of comments (with an option): (Brown)
/* here is a comment that
is just autoindented, and
nothing else */

- Add words to **'cinwords'** to reduce the indent, e.g., "end" or "fi".
 - 7 Use Tabs for the indent of starting lines, pad with spaces for continuation lines. Allows changing **'tabstop'** without messing up the indents.
- Patch by Lech Lorens, 2010 Mar. Update by James McCoy, 2014 Mar 15.

Java:

- 8 Can have {} constructs inside parens. Include changes from Steve Odendahl?
- 8 Recognize "import java.util.Vector" and use \$CLASSPATH to find files for "[i" commands and friends.
- For files found with **'include'**: handle "*" in included name, for Java. (Jason)
- How to make a "package java.util" cause all classes in the package to be searched? Also for "import java.util.*". (Mark Brophy)

'comments':

- 8 When formatting C comments that are after code, the "*" isn't repeated like it's done when there is no code. And there is no automatic wrapping. Recognize comments that come after code. Should insert the comment leader when it's "#" or "//".
- Other way around: when a C command starts with "* 4" the "*" is repeated while it should not. Use syntax HL comment recognition?
- 7 When using "comments=fg:--", Vim inserts three spaces for a new line. When hitting a TAB, these spaces could be removed.
- 7 The 'n'esting flag doesn't do the indenting of the last (rightmost) item.
- 6 Make strings in **'comments'** option a RE, to be able to match more complicated things. (Phillipps) Use a special flag to indicate that a regexp is used.
- 8 Make the **'comments'** option with "/* * */" lines only repeat the "*" line when there is a "/" before it? Or include this in **'cindent'**?

Virtual edit:

- 8 Make the horizontal scrollbar work to move the text further left.
- 7 Allow specifying it separately for Tabs and beyond end-of-line?

Text objects:

- 8 Add text object for fold, so that it can be yanked when it's open.
- 8 Add text script for text object commands "aw", "iW", etc.
- 8 Add text object for part of a CamelHumpedWord and under_scored_word. (Scott Graham) "ac" and "au"?
- 8 Add a text object for any kind of quoting, also with multibyte characters. Option to specify what quotes are recognized (default: all) use "aq" and "iq". Use **'quotepairs'** to define pairs of quotes, like **'matchpairs'**?
- 8 Add text object for any kind of parens, also multibyte ones.
- 8 Add a way to make an ":omap" for a user-defined text object. Requires changing the starting position in oap->start.
- 8 Add "gp" and "gP" commands: insert text and make sure there is a single space before it, unless at the start of the line, and after it, unless at

- the end of the line or before a ".".
- 7 Add objects with backwards extension? Use "I" and "A". Thus "2dAs" deletes the current and previous sentence. (Jens Paulus)
 - 7 Add "g{" and "g}" to move to the first/last character of a paragraph (instead of the line just before/after a paragraph as with "{" and "}").
 - 6 Ignore comment leaders for objects. Make "das" work in reply-email.
 - 5 Make it possible to use syntax group matches as a text object. For example, define a "ccItem" group, then do "da<ccItem>" to delete one. Or, maybe just define "dai", delete-an-item, to delete the syntax item the cursor is on.

Select mode:

- 8 In blockwise mode, typed characters are inserted in front of the block, backspace deletes a column before the block. (Steve Hall)
- 7 Alt-leftmouse starts block mode selection in MS Word.
See http://vim.wikia.com/wiki/Use_Alt-Mouse_to_select_blockwise.
- 7 Add Cmdline-select mode. Like Select mode, but used on the command line.
 - Change gui_send_mouse_event() to pass on mouse events when 'mouse' contains 'C' or 'A'.
 - Catch mouse events in ex_getln.c. Also shift-cursor, etc., like in normal_cmd().
 - remember start and end of selection in cmdline_info.
 - Typing text replaces the selection.

Visual mode:

- 8 Support using "." in Visual mode. Use the operator applied to the Visual selection, if possible.
 - When dragging the Visual selection with the mouse and 'scrolloff' is zero, behave like 'scrolloff' is one, so that the text scrolls when the pointer is in the top line.
 - Displaying size of Visual area: use 24-33 column display. When selecting multiple lines, up to about a screenful, also count the characters.
- 8 When using "I" or "A" in Visual block mode, short lines do not get the new text. Make it possible to add the text to short lines too, with padding where needed.
- 7 With a Visual block selected, "2x" deletes a block of double the width, "3y" yanks a block of triple width, etc.
- 7 When selecting linewise, using "itext" should insert "text" at the start of each selected line.
- 8 What is "R" supposed to do in Visual mode?
- 8 Make Visual mode local to the buffer. Allow changing to another buffer. When starting a new Visual selection, remove the Visual selection in any other buffer. (Ron Aaron)
- 8 Support dragging the Visual area to drop it somewhere else. (Ron Aaron, Ben Godfrey)
- 7 Support dragging the Visual area to drop it in another program, and receive dropped text from another program. (Ben Godfrey)
- 7 With blockwise Visual mode and "c", "C", "I", "A", etc., allow the use of a <CR>. The entered lines are repeated over the Visual area.
- 7 Filtering a block should only apply to the block, not to the whole lines. When the number of lines is increased, add lines. When decreased, pad with

- spaces or delete? Use ":'<,'>" on the command line.
- 8 After filtering the Visual area, make "gv" select the filtered text? Currently "gv" only selects a single line, not useful.
 - 7 Don't move the cursor when scrolling? Needed when the selection should stay the same. Scroll to the cursor at any movement command. With an option!
 - 7 In Visual block mode, need to be able to define a corner on a position that doesn't have text? Also: when using the mouse, be able to select part of a TAB. Even more: Add a mode where the cursor can be on a screen position where there is no text. When typing, add spaces to fill the gap. Other solution: Always use curswant, so that you can move the cursor to the right column, and then use up/down movements to select the line, without changing the column.
 - 6 ":left" and ":right" should work in Visual block mode.
 - 7 **CTRL-I** and **CTRL-O** should work in Visual mode, but only jump to marks in the current buffer.
 - 6 In non-Block mode, "I" should insert the same text in front of each line, before the first non-blank, "gI" in column 1.
 - 6 In non-Block mode, "A" should append the same text after each line.
 - 6 When in blockwise visual selection (CTRL-V), allow cursor to be placed right of the line. Could also allow cursor to be placed anywhere on a TAB or other special character.
 - 6 Add commands to move selected text, without deselecting.

More advanced repeating commands:

- Add "." command for visual mode: redo last visual command (e.g. ":fmt").
 - Add command to repeat last movement. Including count.
 - Add "." command after operator: repeat last command of same operator. E.g. "c." will repeat last change, also when "x" used since then (Webb). "y." will repeat last yank. "c2." will repeat the last but one change?
- Also: keep history of Normal mode commands, add command to list the history and/or pick an older command.
- History stack for . command? Use "g." command.

Mappings and Abbreviations:

- 8 When "0" is mapped (it is a movement command) this mapping should not be used after typing another number, e.g. "20l". (Charles Campbell)
Is this possible without disabling the mapping of the following command?
- 8 Should mapping <C-A> and <C-S-A> both work?
- 7 ":abbr b byte", append "b " to an existing word still expands to "byte". This is Vi compatible, but can we avoid it anyway?
- 8 To make a mapping work with a prepended "x to select a register, store the last _typed_ register name and access it with "&".
- 8 Add ":amap", like ":amenu".
- 7 Add a mapping that works always, for remapping the keyboard.
- 8 Add ":cab!", abbreviations that only apply to Command-line mode and not to entering search strings.
- 8 Add a flag to ":abbrev" to eat the character that triggers the abbreviation. Thus "abb ab xxx" and typing "ab<Space>" inserts "xxx" and not the <Space>.
- 8 Give a warning when using **CTRL-C** in the lhs of a mapping. It will never

- (?) work.
- 7 Add `<0x8f>` (hex), `<0o33>` (octal) and `<123>` (decimal) to `<>` notation?
- 7 When someone tries to unmap with a trailing space, and it fails, try unmapping without the trailing space. Helps for `":unmap xx | unmap yy"`.
- 6 Context-sensitive abbreviations: Specify syntax group(s) in which the abbreviations are to be used.
- Add mappings that take arguments. Could work like the `":s"` command. For example, for a mouse escape sequence:


```
:mapexp <Esc>{\([0-9]*\),\([0-9]*\); H\1j\2l
```
- Add optional `<Number>` argument for mappings:


```
:map <Number>q          ^W^W<Number>G
:map <Number>q<Number>t ^W^W<Number1-1>G<Number2>l
:map q<Char>           :s/<Char>/\u\0/g
```
- Or implicit:


```
:map q                  <Register>d<Number>$
```
- Add command to repeat a whole mapping (`":"` only repeats the last change in a mapping). Also: Repeat a whole insert command, including any mappings that it included. Sort-of automatic recording?
- Include an option (or flag to `'cptions'`) that makes errors in mappings not flush the rest of the mapping (like `nvi` does).
- Use context sensitiveness of completion to switch abbreviations and mappings off for `:unab` and `:unmap`.
- 6 When using mappings in Insert mode, insert characters for incomplete mappings first, then remove them again when a mapping matches. Avoids that characters that are the start of some mapping are not shown until you hit another character.
- Add mappings for replace mode: `":rmap"`. How do we then enter mappings for non-replace Insert mode?
- Add separate mappings for Visual-character/block/line mode?
- Add `'mapstop'` command, to stop recursive mappings.
- List mappings that have a raw escape sequence both with the name of the key for that escape sequence (if there is one) and the sequence itself.
- List mappings: Once with special keys listed as `<>`, once with meta chars as `<M-a>`, once with the byte values (octal?). Sort of "spell mapping" command?
- When entering mappings: Add the possibility to enter meta keys like they are displayed, within `<>`: `<M-a>`, `<~@>` or `<|a>`.
- Allow multiple arguments to `:unmap`.
- Command to show keys that are not used and available for mapping `":freekeys"`.
- Allow any character except white space in abbreviations lhs (Riehm).

Incsearch:

- Wrong scrolling when using `incsearch`. Patch by Christian Brabandt, 2014 Dec 4. Is this a good solution?
- Temporarily open folds to show where the search ends up. Restore the folds when going to another line. Patch to open folds for `'incsearch'`. (Christian Brabandt, 2015 Jan 6)
- Bug with `'incsearch'` going to wrong line. (Wolfram Kresse, 2009 Aug 17) Only with `"vim -u NONE"`.
- When no match is found and the user types more, the screen is redrawn anyway. Could skip that. Esp. if the line wraps and the text is scrolled up every time.
- When `incsearch` used and hitting return, no need to search again in many

- cases, saves a lot of time in big files. (Slootman wants to work on this?)
 When not using special characters, can continue search from the last match
 (or not at all, when there was no match). See oldmail/webb/in.872.
- 9 incsearch is incorrect for "/that/<Return>/this;/;" (last search pattern
 isn't updated).

Searching:

- 9 Should have an option for :vimgrep to find lines without a match.
- 8 Add "g/" and "gb" to search for a pattern in the Visually selected text?
 "g?" is already used for rot13.
 The vis.vim script has a ":S" command that does something like this.
 Can use "g/" in Normal mode, uses the '<' to '>' area.
 Use "&/" for searching the text in the Visual area?
- 9 Add "v" offset: "/pat/v": search for pattern and start Visual mode on the
 matching text.
- 8 Add a modifier to interpret a space like "_s\+" to make it much easier to
 search for a phrase.
- 8 Add a mechanism for recursiveness: "\@([[^()]*\@g[^()]*\))". \@g stands
 for "go recursive here" and \@(\) marks the recursive part.
 Perl does it this way:
 \$paren = qr/ \(([^()] | (??{ \$paren })) * \) /x;
 Here \$paren is evaluated when it's encountered. This is like a regexp
 inside a regexp. In the above terms it would be:
 \@(\(\([^\(\)]*\|@g\)\)*\)
- 8 Show the progress every second. Could use the code that checks for **CTRL-C**
 to find out how much time has passed. Or use SIGALRM. Where to show the
 number?
- 7 Support for approximate-regexps to find similar words (agrep
<http://www.tgries.de/agrep/> tre: <http://laurikari.net/tre/index.html>).
- 8 Add an item for a big character range, so that one can search for a
 chinese character: \z[234-1234] or \z[XX-YY] or \z[0x23-0x234].
- 7 Add an item stack to allow matching (). One side is "push X on
 the stack if previous atom matched". Other side is "match with top of
 stack, pop it when it matches". Use "\@pX" and "\@m"?
 Example: \((\@p).\{-}\@m\)*
- 7 Add a flag to "/pat/" to discard an error. Useful to continue a mapping
 when a search fails. Could be "/pat/E" (e is already used for end offset).
- 7 Add pattern item to use properties of Unicode characters. In Perl it's
 "\p{L}" for a letter. See Regular Expression Pocket Reference.
- 8 Would it be possible to allow ":23,45/pat/flags" to search for "pat" in
 lines 23 to 45? Or does this conflict with Ex range syntax?
- 8 Allow identical pairs in 'matchpairs'. Restrict the search to the current
 line.
- 7 Allow longer pairs in 'matchpairs'. Use matchit.vim as an
 example.
- 8 Make it possible to define the character that "%" checks for in
 #if/#endif. For nmake it's !if/!endif.
- For "%" command: set hierarchy for which things include other things that
 should be ignored (like "*/" or "#endif" inside /* */).
 Also: use "%" to jump from start to end of syntax region and back.
 Alternative: use matchit.vim
- 8 A pattern like "\([^\a]\+\)" takes an awful long time. Recognize that
 the recursive "\+" is meaningless and optimize for it.

- This one is also very slow on `"/ * some comment */": "^\\/*\\(.*[^/])*$"`.
- 7 Recognize `"[a-z]"`, `"[0-9]"`, etc. and replace them with the faster `"\l"` and `"\d"`.
- 7 Add a way to specify characters in `<C-M>` or `<Key>` form. Could be `\%<C-M>`.
- 8 Add an argument after `":s/pat/str/"` for a range of matches. For example, `":s/pat/str/#3-4"` to replace only the third and fourth "pat" in a line.
- 8 When `'iskeyword'` is changed the matches from `'hlsearch'` may change. (Benji Fisher) redraw if some options are set while `'hlsearch'` is set?
- 8 Add an option not to use `'hlsearch'` highlighting for `":s"` and `":g"` commands. (Kahn) It would work like `":noh"` is used after that command. Also: An extra flag to do this once, and a flag to keep the existing search pattern.
- Make `'hlsearch'` a local/global option, so that it can be disabled in some of the windows.
 - Add `\%h{group-name};` to search for a specific highlight group.
 - Add `\%s{syntax-group};` to search for a specific syntax group.
 - Support Perl regexp. Use PCRE (Perl Compatible RE) package. (Shade) Or translate the pattern to a Vim one. Don't switch on with an option for typed commands/mappings/functions, it's too confusing. Use `"\@@"` in the pattern, to avoid incompatibilities.
- 8 Add a way to access the last substitute text, what is used for `":s//~/"`. Can't use the `~` register, it's already used for drag & drop.
- Remember flags for backreferenced items, so that `"*"` can be used after it. Check with `"\(\S\)\1\{3}"`. (Hemmerling)
- 8 Flags that apply to the whole pattern. This works for all places where a regexp is used. Add `"\q"` to not store this pattern as the last search pattern?
- Add flags to search command (also for `":s"?):`
 - i ignore case
 - I use case
 - p use Perl regexp syntax (or POSIX?)
 - v use Vi regexp syntax
 - f forget pattern, don't keep it for "n" command
 - F remember pattern, keep it for "n" command
- Perl uses these too:
- e evaluate the right side as an expression (Perl only)
 - m multiple line expression (we don't need it)
 - o compile only once (Perl only)
 - s single line expression (we don't need it)
 - x extended regexp (we don't need it)
- When used after `":g"` command, backslash needed to avoid confusion with the following command.
- Add `'searchflags'` for default flags (replaces `'gdefault'`).
- Add command to display the last used substitute pattern and last used pattern. (Margo) Maybe make it accessible through a register (like `"` for search string)?
- 7 Use T-search algorithm, to speed up searching for strings without special characters. See C't article, August 1997.
- Add `'fuzzycase'` option, so that case doesn't matter, and `'-'` and `'_'` are equivalent (for Unix filenames).
 - Add `'v'` flag to search command: enter Visual mode, with the matching text as Visual area. (variation on idea from Bertin)
 - Searching: `"/this//that/"` should find "that" after "this".

- Add global search commands: Instead of wrapping at the end of the buffer, they continue in another buffer. Use flag after search pattern:
 - a for the next file in the argument list
 - f for file in the buffer list
 - w for file edited in a window.
 e.g. "/pat/f". Then "n" and "N" work through files too. "f" flag also for ":s/pat/foo/f"??? Then when 'autowrite' and 'hidden' are both not set, ask before saving files: "Save modified buffer "/path/file"? (Yes/Hide/No Save-all/hide-All/Quit) ".
- ":s/pat/foo/3": find 3rd match of "pat", like sed. (Thomas Koehler)
- 7 When searching with 'n' give message when getting back where the search first started. Remember start of search in '/' mark.
- 7 Add option that scrolls screen to put cursor in middle of screen after search always/when off-screen/never. And after a ":tag" command. Maybe specify how many lines below the screen causes a redraw with the cursor in the middle (default would be half a screen, zero means always).
- 6 Support multiple search buffers, so macros can be made without side effects.
- 7 From xvim: Allow a newline in search patterns (also for :s, can delete newline). Add BOW, EOW, NEWL, NLORANY, NLBUTANY, magic 'n' and 'r', etc. [not in xvim:] Add option to switch on matches crossing ONE line boundary.
- 7 Add ":iselect", a combination of ":ilist" and ":tselect". (Aaron) (Zellner) Also ":dselect".
- Searching for \%> does not find anything when using line Visual selection. Probably because it's using MAXCOL. #8238
- Regexp to search for duplicate lines does not work correctly: /\(^.*\n\)\1 (Chris Morgan, #6239)
- Problem with upwards search on Windows (works OK on Linux). (Brett Stahlman, 2014 Jun 8)
- Searching mixed with Visual mode doesn't redraw properly. (James Vega, 2010 Nov 22)
- Searching for composing char works, but not when inside []. (ZyX, Benjamin R. Haskell, 2010 Aug 24)
- When 'smartcase' is set and using CTRL-L to add to the search pattern it may result in no matches. Convert chars to lower case? (Erik Wognsen, 2009 Apr 16)

Undo:

- 9 ":gundo" command: global undo. Undoes changes spread over multiple files in the order they were made. Also ":gredo". Both with a count. Useful when tests fail after making changes and you forgot in which files.
- 9 After undo/redo, in the message show whether the buffer is modified or not.
- 8 Search for pattern in undo tree, showing when it happened and the text state, so that you can jump to it.
- 8 Undo tree: visually show the tree somehow (Damian Conway)
 - Show only the leaves, indicating how many changed from the branch and the timestamp?
 - Put branch with most recent change on the left, older changes get more indent?
- Make it possible to undo all the commands from a mapping, including a trailing unfinished command, e.g. for ":map K ix^[r".
- When accidentally hitting "R" instead of Ctrl-R, further Ctrl-R is not

- possible, even when typing `<Esc>` immediately. (Grahn) Also for "i", "a", etc. Postpone saving for undo until something is really inserted?
- 8 When Inserting a lot of text, it can only be undone as a whole. Make undo sync points at every line or word. Could recognize the start of a new word (white space and then non-white space) and backspacing. Can already use **CTRL-G** u, but that requires remapping a lot of things.
- 8 Make undo more memory-efficient: Compare text before and after change, only remember the lines that really changed.
- 7 Add undo for a range of lines. Can change these back to a previous version without changing the rest of the file. Stop doing this when a change includes only some of these lines and changes the line count. Need to store these undo actions as a separate change that can be undone.
- For `u_save()` include the column number. This can be used to set '[' and ']'. And in the future the undo can be made more efficient (Webb).
 - In out-of-memory situations: Free allocated space in undo, and reduce the number of undo levels (with confirmation).
 - Instead of [+], give the number of changes since the last write: [+123]. When undoing to before the last write, change this to a negative number: [-99].
 - With undo with simple line delete/insert: optimize screen updating.
 - When executing macro's: Save each line for undo only once.
 - When doing a global substitute, causing almost all lines to be changed, undo info becomes very big. Put undo info in swap file??
 - MS-Windows: when writing undo file the infostreams are copied in `mch_copy_file_attribute()`, that seems unnecessary. (#7925) Add a flag to only copy attributes?
 - undo result wrong: Masato Nishihata, #4798
 - After recovering from a swap file the undofile should not be used, it causes corruption. (#6631)
 - When the computer crashes while writing the undofile, the contents may be lost. Write to a temp file, `fsync` and rename. (#8879)
 - Undo puts cursor in wrong line after "cG<Esc>" undo.
 - Undo history wrong when ":next file" re-uses a buffer. (#5426) `ex_next()` should pass flag to `do_argfile()`, then to `do_ecmd()`. Is there a test for this?
 - Undo problem: "g-" doesn't go back, gets stuck. (Björn Linse, 2016 Jul 18)
 - Undo message is not always properly displayed. Patch by Ken Takata, 2013 oct 3. Doesn't work properly according to Yukihiro Nakadaira. Also see #1635.
 - When using ":diffput" through a mapping, undo in the target buffer isn't synced. (Ryan Carney, 2016 Sep 14)
 - The undo file name can get too long. (Issue #346)
For the path use a hash instead of `dir%dir%dir%name hash%name`.
Check both for some time for backwards compatibility.
Alternatively: create the directory structure under '**undodir**'.
 - Patch to add ":undorecover", get as much text out of the undo file as possible. (Christian Brabandt, 2014 Mar 12, update Aug 22)
 - Patch to support :undo absolute jump to file save number. (Christian Brabandt, 2010 Nov 5)
 - There should be a way after an abbreviation has expanded to go back to what was typed. **CTRL-G** h ? Would also undo last word or line break inserted perhaps. And undo **CTRL-W**. **CTRL-G** l would redo.
 - When using '**cryptmethod**' `xchaha20` the undo file is not encrypted. Need to handle extra bytes.

Buffer list:

- 7 Command to execute a command in another buffer: `":inbuf {bufname} {cmd}"`. Also for other windows: `":inwin {winnr} {cmd}"`. How to make sure that this works properly for all commands, and still be able to return to the current buffer/window? E.g.: `":inbuf xxx only"`.
- 8 Add File.{recent-files} menu entries: Recently edited files. Ron Aaron has a plugin for this: `mru.vim`.
- 8 Unix: Check all uses of `fnamecmp()` and `fnamencmp()` if they should check inode too.
- 7 Add another number for a buffer, which is visible for the user. When creating a new buffer, use the lowest number not in use (or the highest number in use plus one?).
- 7 Offer some buffer selection from the command line? Like using `":ls"` and asking for a buffer number. (Zachmann)
 - When starting to edit a file that is already in the buffer list, use the file name argument for the new short file name. (Webb)
 - Add an option to make `":bnext"` and `":bprev"` wrap around the end of the buffer list. Also for `":next"` and `":prev"`?
- 7 Add argument to `":ls"` which is a pattern for buffers to list. E.g. `":ls *.c"`. (Thompson)
- 7 Add expansion of buffer names, so that `*.c` is expanded to all buffer names. Needed for `":bdel *.c"`, `":bunload *.c"`, etc.
- 8 Support for `<afile>` where a buffer name is expected.
- 7 Add an option to mostly use slashes in file names. Separately for internal use and for when executing an external program?
- 8 Some file systems are case-sensitive, some are not. Besides **'wildignorecase'** there might be more parts inside `CASE_INSENSITIVE_FILENAME` that are useful on Unix.
 - When using `":bwipe!"` also get rid of references to be buffer, e.g. in the jumplist and alternate file.
 - `":bnext"` in a help buffer is supposed to go to the next help buffer, but it goes to any buffer, and then `:bnext` skips help buffers, since they are unlisted. (#4478)
 - `:buffer` completion does not escape `"` properly and results in a regexp error. (#5467)
 - Problem with two buffers with the same name a/b, if it didn't exist before and is created outside of Vim. (dskloetg, 2018 Jul 16, #3219)
 - Add an option with file patterns, to be used when unloading a buffer: If there is a match, remove entries for the buffer from marks, jumplist, etc. To be used for git temp files.

Swap (.swp) files:

- 8 If writing to the swap file fails, should try to open one in another directory from **'dir'**. Useful in case the file system is full and when there are short file name problems.
- 8 Also use the code to try using a short file name for the backup and swap file for the Win32 and Dos 32 bit versions.
- 8 When a file is edited by root, add `$LOGNAME` to know who did su.
- 8 When the edited file is a symlink, try to put the swap file in the same dir as the actual file. Adjust `FullName()`. Avoids editing the same file twice (e.g. when using quickfix). Also try to make the name of the backup

- file the same as the actual file?
 Use the code for resolve()?
- 7 When using 64 bit inode numbers, also store the top 32 bits. Add another field for this, using part of bo_fname[], to keep it compatible.
 - 7 When editing a file on removable media, should put swap file somewhere else. Use something like 'r' flag in 'viminfo'. 'diravoid'?
 Also: Be able to specify minimum disk space, skip directory when not enough room.
 - 7 Add a configure check for which directory should be used: /tmp, /var/tmp or /var/preserve.
 - Add an option to create a swap file only when making the first change to the buffer. (Liang) Or only when the buffer is not read-only.
 - Add option to set "umask" for backup files and swap files (Antwerpen). 'backupumask' and 'swapumask'? Or 'umaskbackup' and 'umaskswap'?
 - When editing a readonly file, don't use a swap file but read parts from the original file. Also do this when the file is huge (>'maxmem'). We do need to load the file once to count the number of lines? Perhaps keep a cached list of which line is where.
 - When editing a file with ":edit" the output of :swapname is relative, while editing it with "vim file" it is absolute. (#355) Which one should it be?
 - When recovering a file, put the swap file name in b:recovered_swapfile. Then a command can delete it.
 - In the ATTENTION message about an existing swap file, mention the name of the process that is running. It might actually be some other program, e.g. after a reboot.
 - When running Vim in silent ex mode, an existing swapfile causes Vim to wait for a user action without a prompt. (Maarten Billemont, 2012 Feb 3) Do give the prompt? Quit with an error?
 - After doing "su" \$HOME can be the old user's home, thus ~root/file is not correct. Don't use it in the swap file.
 - In the swapfile dialog, add a H(elp) option that gives more info about what each choice does. Similar to ":help swap-exists-choices"

Viminfo:

- 7 Can probably remove the code that checks for a writable viminfo file, because we now do the chown() for root, and others can't overwrite someone else's viminfo file.
- 8 When there is no .viminfo file and someone does "su", runs Vim, a root-owned .viminfo file is created. Is there a good way to avoid this? Perhaps check the owner of the directory. Only when root?
- 8 Add argument to keep the list of buffers when Vim is started with a file name. (Schild)
- 8 Keep the last used directory of the file browser (File/Open menu).
- 8 Remember the last used register for "@@".
- 8 Remember the redo buffer, so that "." works after restarting.
- 8 Remember a list of last accessed files. To be used in the "File.Open Recent" menu. Default is to remember 10 files or so.
 Also remember which files have been read and written. How to display this?
- 7 Also store the "." register (last inserted text).
- 7 Make it possible to store buffer names in viminfo file relative to some directory, to make them portable over a network. (Aaron)

- 6 Store a snapshot of the currently opened windows. So that when quitting Vim, and then starting again (without a file name argument), you see the same files in the windows. Use `":mksession"` code?
- Make marks present in `.viminfo` usable as file marks: Display a list of "last visited files" and select one to jump to.
- Add the debug command line history to `viminfo`.
- Using `"wviminfo /tmp/viminfo"` does not store file marks that Vim knows about, it only works when merging with an existing file. (Shougo, 2017 Jun 19, #1781)
- Writing nested List and Dict in `viminfo` gives error message and can't be read back. (Yukihiro Nakadaira, 2010 Nov 13)
- When a register contains illegal bytes, writing `viminfo` in utf-8 and reading it back doesn't result in utf-8. (Devin Bayer)
- 8 With `'viminfo'` set such that the `.viminfo` file is written on a FAT filesystem, an illegal file name may be created: `.vim"`.
- 8 For each buffer that is opened, the `viminfo` file is opened and read to check for file marks. This can be slow.
- 8 When `write_viminfo()` is used while there are many orphaned `viminfo` tempfiles writing the `viminfo` file fails. Give a clear error message so that the user knows the files have to be deleted.

Modelines:

- 8 Before trying to execute a modeline, check that it looks like one (valid option names). If it's very wrong, silently ignore it. Ignore a line that starts with `"Subject: "`.
- Add an option to whitelist options that are allowed in a modeline. This would allow careful users to use modelines, e.g., only allowing `'shiftwidth'`.
- Add an option to let modelines only set local options, not global ones such as `'encoding'`.
- When an option value is coming from a modeline, do not carry it over to another edited file? Would need to remember the value from before the modeline setting.
- Allow setting a variable from a modeline? Only allow using fixed strings, no function calls, to avoid a security problem.
- Allow `":doauto BufRead x.cpp"` in modelines, to execute autocommands for `.cpp` files.
- Support the `"abbreviate"` command in modelines (Kearns). Careful for characters after `<Esc>`, that is a security leak.
- Add an option setting to ask the user if the modelines are to be executed or not. Same for `.exrc` in local dir.

Sessions:

- Session file contains absolute paths when `"curdir"` is removed from `'sessionoptions'`, making it impossible to have a session with a relative path. (#4450)
- Session file only contains local option values for buffers that are in a window, not other buffers. (#7532)
- Script generated by `:mksession` does not work well if there are windows with modified buffers:
 - change `"silent only"` into `"silent only!"`
 - change `"edit fname"` of first buffer to `"hide edit fname"`

- skip "badd fname" if "fname" is already in the buffer list
- remove remark about unloading buffers from documentation
- When session file has name in argument list but the buffer was deleted, the buffer is not deleted when using the session file. (#1393) Should add the buffer in hidden state.
- When a session file is created and there are "nofile" buffers, these are not filled. Need to trigger BufReadCmd autocommands. Also handle deleting the initial empty buffer better. (ZyX, 2015 March 8)
- Setting the spell file in a session only reads the local additions, not the normal spell file. (Enno Nagel, 2014 Mar 29)
- Directory wrong in session file, caused by ":lcd" in BufEnter autocommand. (Felix Kater, 2009 Mar 3)
- Something wrong with session that has "cd" commands and "badd", in such a way that Vim doesn't find the edited file in the buffer list, causing the ATTENTION message? (Tony Mechelynck, 2008 Dec 1) Also: swap files are in ~/tmp/ One has relative file name ".mozilla/...".
- Session file creation: '**autochdir**' causes trouble. Keep it off until after loading all files.
- Session file generates error upon loading, cause by --remote-silent-tab. (7tommm (ytommm) 2010 Nov 24)
- 8 DOS/Windows: ":mksession" generates a "cd" command where "aa\#bb" means directory "#bb" in "aa", but it's used as "aa#bb". (Ronald Hoellwarth)
- 7 When there is a "help.txt" window in a session file, restoring that session will not get the "LOCAL ADDITIONS" back.
- 8 With ":mksession" always store the '**sessionoptions**' option, even when "options" isn't in it. (St-Amant)
- 8 When using ":mksession", also store a command to reset all options to their default value, before setting the options that are not at their default value.
- 7 With ":mksession" also store the tag stack and jump history. (Michal Malecki)

Options:

- 7 ":with option=value | command": temporarily set an option value and restore it after the command has executed.
- 8 Make "old" number options that really give a number of effects into string options that are a comma-separated list. The old number values should also be supported.
- 8 Add commands to save and restore an option, which also preserves the flag that marks if the option was set. Useful to keep the effect of setting '**compatible**' after ":syntax on" has been used.
- 7 There is '**titleold**', why is there no '**iconold**'? (Chazelas)

External commands:

- 8 When filtering text, redirect stderr so that it can't mess up the screen and Vim doesn't need to redraw it. Also for ":r !cmd".
- 4 Set separate shell for ":sh", piping "range!filter", reading text "r !ls" and writing text "w !wc". (Deutsche) Allow arguments for fast start (e.g. -f).
- 4 Allow direct execution, without using a shell.
- 4 Run an external command in the background. But how about I/O in the GUI? Careful: don't turn Vim into a shell!

- 4 Add feature to disable using a shell or external commands.

Multiple Windows:

- 7 "vim -o0 file ..." use both horizontal and vertical splits.
- 8 Add **CTRL-W T**: go to the top window in the column of the current window.
And **CTRL-W B**: go to bottom window.
- 7 Use **CTRL-W <Tab>**, like alt-tab, to switch between buffers. Repeat **<Tab>** to select another buffer (only loaded ones?), **<BS>** to go back, **<Enter>** to select buffer, **<Esc>** to go back to original buffer.
- 7 Make it possible to edit a new buffer in the preview window. A script can then fill it with something. ":popen"?
- 7 Add a '**tool**' window: behaves like a preview window but there can be several. Don't count it in only_one_window(). (Alexei Alexandrov)
- 6 Add an option to resize the shell when splitting and/or closing a window. ":vsp" would make the shell wider by as many columns as needed for the new window. Specify a maximum size (or use the screen size). ":close" would shrink the shell by as many columns as come available. (Demirel)
- 7 When starting Vim several times, instantiate a Vim server, that allows communication between the different Vims. Feels like one Vim running with multiple top-level windows. Esp. useful when Vim is started from an IDE too. Requires some form of inter process communication.
- Support a connection to an external viewer. Could call the viewer automatically after some seconds of non-activity, or with a command. Allow some way of reporting scrolling and cursor positioning in the viewer to Vim, so that the link between the viewed and edited text can be made.

Marks:

- 8 Add ten marks for last changed files: ':0, ':1, etc. One mark per file.
- 8 When cursor is first moved because of scrolling, set a mark at this position. (Rimon Barr) Use '-.
- 8 Add a command to jump to a mark and make the motion inclusive. g'm and g`m?
- 8 The '"' mark is set to the first line, even when doing ":next" a few times. Only set the '"' mark when the cursor was really moved in a file.
- 8 Make `` and ', which would position the new cursor position in the middle of the window, restore the old topline (or relative position) from when the mark was set.
- 7 Make a list of file marks in a separate window. For listing all buffers, matching tags, errors, etc. Normal commands to move around. Add commands to jump to the mark (in current window or new window). Start it with ":browse marks"?
- 6 Add a menu that lists the Marks like ":marks". (Amerige)
- 7 For ":jumps", ":tags" and ":marks", for not loaded buffers, remember the text at the mark. Highlight the column with the mark.
- 7 Highlight each mark in some way (With "Mark" highlight group). Or display marks in a separate column, like '**number**' does.
- 7 Use d"m to delete rectangular area from cursor to mark m (like Vile's \m command).
- 7 Try to keep marks in the same position when:
 - replacing with a line break, like in ":s/pat/^M/", move marks after the line break column to the next line. (Acevedo)
 - inserting/deleting characters in a line.
- 5 Include marks for start/end of the current word and section. Useful in

mappings.

- 6 Add "unnamed mark" feature: Like marks for the ":g" command, but place and unplace them with commands before doing something with the lines. Highlight the marked lines somehow.

Digraphs:

- 7 Make "ga" show the keymap for a character, if it exists. Also show the code of the character after conversion to **'fileencoding'**.
- Use digraph table to tell Vim about the collating sequence of special characters?
- 8 Add command to remove one or more (all) digraphs. (Brown)
- 7 Support different sets of digraphs (depending on the character set?). At least Latin1/Unicode, Latin-2, MS-DOS (esp. for Win32).

Writing files:

- In vim_rename(), should lock "from" file when deleting "to" file for systems other than Amiga. Avoids problems with unexpected longname to shortname conversion.
- 8 write mch_isdevice() for Amiga, Mac, VMS, etc.
- 8 When appending to a file, Vim should also make a backup and a **'patchmode'** file.
- 8 **'backupskip'** doesn't write a backup file at all, a bit dangerous for some applications. Add **'backupelsewhere'** to write a backup file in another directory? Or add a flag to **'backupdir'**?
- 6 Add an option to write a new, numbered, backup file each time. Like **'patchmode'**, e.g., **'backupmode'**.
- 6 Make it possible to write **'patchmode'** files to a different directory. E.g., ":set patchmode=~/backups/*.orig". (Thomas)
- 6 Add an option to prepend something to the backup file name. E.g., "#". Or maybe allow a function to modify the backup file name?
- 8 Only make a backup when overwriting a file for the first time. Avoids losing the original when writing twice. (Slootman)
- 7 On non-Unix machines, also overwrite the original file in some situations (file system full, it's a link on an NFS partition).
- 7 When editing a file, check that it has been changed outside of Vim more often, not only when writing over it. E.g., at the time the swap file is flushed. Or every ten seconds or so (use the time of day, check it before waiting for a character to be typed).
- 8 When a file was changed since editing started, show this in the status line of the window, like "[time]".
Make it easier to reload all outdated files that don't have changes. Automatic and/or with a command.

Substitute:

- 8 Substitute with hex/unicode number "%xff" and "%uabcd". Just like "%uabcd" in search pattern.
- 8 Make it easier to replace in all files in the argument list. E.g.: ":argsub/oldword/newword/". Works like ":argdo %s/oldword/newword/g|w".
- :s///p prints the line after a substitution.
- With :s///c replace &, ~, etc. when showing the replacement pattern.
- 8 With :s///c allow scrolling horizontally when **'nowrap'** is effective.

- Also allow a count before the scrolling keys.
- Add number option to ":s//2": replace second occurrence of string? Or: :s///N substitutes N times.
- Add answers to ":substitute" with 'c' flag, used in a ":global", e.g.: ":g/pat1/s/pat2/pat3/cg": 'A' do all remaining replacements, 'Q' don't do any replacements, 'u' undo last substitution.
- 7 Substitute in a block of text. Use `{line}.{column}` notation in an Ex range, e.g.: ":1.3,\$.5s" means to substitute from line 1 column 3 to the last line column 5.
- 5 Add commands to bookmark lines, display bookmarks, remove bookmarks, operate on lines with bookmarks, etc. Like ":global" but with the possibility to keep the bookmarks and use them with several commands. (Stanislav Sitar)

Mouse support:

- 8 Add 'o' flag to 'mouse'?
- 7 Be able to set a 'moushape' for the popup menu.
- 8 Add 'mouse' flag, which sets a behavior like Visual mode, but automatic yanking at the button-up event. Or like Select mode, but typing gets you out of Select mode, instead of replacing the text. (Bhaskar)
- Implement mouse support for the Amiga console.
- Using right mouse button to extend a blockwise selection should attach to the nearest corner of the rectangle (four possible corners).
- Precede mouse click by a number to simulate double clicks?!?
- When mouse click after 'r' command, get character that was pointed to.

Argument list:

- 6 Add command to put all filenames from the tag files in the argument list. When given an argument, only use the files where that argument matches (like `grep -l ident`) and jump to the first match.
- 6 Add command to form an args list from all the buffers?

Registers:

- 8 Don't display empty registers with ":display". (Etienne)
- 8 Add put command that overwrites existing text. Should also work for blocks. Useful to move text around in a table. Works like using "R ^R r" for every line.
- When appending to a register, also report the total resulting number of lines. Or just say "99 more lines yanked", add the "more".
- When inserting a register in Insert mode with **CTRL-R**, don't insert comment leader when line wraps?
- The ":@" commands should take a range and execute the register for each line in the range.
- Add "P" command to insert contents of unnamed register, move selected text to position of previous deleted (to swap foo and bar in " + foo")
- 8 Should be able to yank and delete into the "/" register. How to take care of the flags (offset, magic)?
- In Select mode the deleted text always goes into the unnamed register. Use **CTRL-R** to specify the register to use. (#9531)
- When "+" register is set then "" points to it. If another Vim grabs the "+" register, then "" doesn't contain anything. Make it still follow "+".

- (#6435)
- Pasting a register in Visual mode cannot be repeated. (Mahmoud Al-Qudsi, 2018 Apr 26, #2849)
 - Patch to add option that tells whether small deletes go into the numbered registers. (Aryeh Leib Taurog, 2013 Nov 18)
 - With "unamedplus" in '**clipboard**' pasting in Visual mode causes error for empty register. (Michael Seiwald, 2011 Jun 28) I can't reproduce it.
 - Consider making YankRing or something else that keeps a list of yanked text part of standard Vim. The "1 to "9 registers are not sufficient.
 - 6 When yanking into the unnamed registers several times, somehow make the previous contents also available (like it's done for deleting). What register names to use? g"1, g"2, etc.?
Also do this for the small delete register "-.

Debug mode:

- 8 Add breakpoints for setting an option
- 8 Add breakpoints for assigning to a variable.
- 7 Store the history from debug mode in viminfo.
- 7 Make the debug mode history available with histget() et al.
- In debug mode, using **CTRL-R** = to evaluate a function causes stepping through the function. (Hari Krishna Dara, 2006 Jun 28)

Various improvements:

- 7 Add plugins for formatting? Should be able to make a choice depending on the language of a file (English/Korean/Japanese/etc.).
Setting the '**langformat**' option to "chinese" would load the "format/chinese.vim" plugin.
The plugin would set '**formatexpr**' and define the function being called.
Edward L. Fox explains how it should be done for most Asian languages. (2005 Nov 24)
Alternative: patch for utf-8 line breaking. (Yongwei Wu, 2008 Feb 23)
- 7 [t to move to previous xml/html tag (like "vatov"),]t to move to next ("vatv").
- 7 [< to move to previous xml/html tag, e.g., previous .]< to move to next ,]< to next , [< to previous .
- 8 Add ":rename" command: rename the file of the current buffer and rename the buffer. Buffer may be modified.
- 7 Instead of filtering errors with a shell script it should be possible to do this with Vim script. A function that filters the raw text that comes from the '**makeprg**'?
- 7 Allow a window not to have a statusline. Makes it possible to use a window as a buffer-tab selection.
- 8 Allow non-active windows to have a different statusline. (Yakov Lerner)
- 7 Add an invisible buffer which can be edited. For use in scripts that want to manipulate text without changing the window layout.
- 8 Add a command to revert to the saved version of file; undo or redo until all changes are gone.
- 6 "vim -q -" should read the list of errors from stdin. (Gautam Mudunuri)
- 8 Add "--remote-fail": When contacting the server fails, exit Vim.
Add "--remote-self": When contacting the server fails, do it in this Vim.
Overrules the default of "--remote-send" to fail and "--remote" to do it in this Vim.

- 8 When Vim was started without a server, make it possible to start one, as if the "--servername" argument was given. `":startserver <name>"?`
- 8 No address range can be used before the command modifiers. This makes them difficult to use in a menu for Visual mode. Accept the range and have it apply to the following command.
- 8 Add the possibility to set `'fileformats'` to force a format and strip other CR characters. For example, for "dos" files remove CR characters at the end of the line, so that a file with mixed line endings is cleaned up. To just not display the CR characters: Add a flag to `'display'`?
- 7 Some compilers give error messages in which the file name does not have a path. Be able to specify that `'path'` is used for these files.
- 7 Xterm sends `<Esc>O3F` for `<M-End>`. Similarly for other `<M-Home>`, `<M-Left>`, etc. Combinations of Alt, Ctrl and Shift are also possible. Recognize these to avoid inserting the raw byte sequence, handle like the key without modifier (unless mapped).
- 6 Add `"gG"`: like what `"gj"` is to `"j"`: go to the N'th window line.
- 8 Add command like `":normal"` that accepts `<Key>` notation like `":map"`.
- 9 Support ACLs on more systems.
- 7 Add `ModeMsgVisual`, `ModeMsgInsert`, etc. so that each mode message can be highlighted differently.
- 7 Add a message area for the user. Set some option to reserve space (above the command line?). Use an `":echouser"` command to display the message (truncated to fit in the space).
- 7 Add `%s` to `'keywordprg'`: replace with word under the cursor. (Zellner)
- 8 Support printing on Unix. Can use `"lpansi.c"` as an example. (Bookout)
- 8 Add `put` command that replaces the text under it. Esp. for blockwise Visual mode.
- 7 Enhance `termresponse` stuff: Add `t_CV(?)`: pattern of term response, use regexp: `"\e\[[>?][0-9;]*c"`, but only check just after sending `t_RV`.
- 7 Add `"g|"` command: move to N'th column from the left margin (after wrapping and applying `'leftcol'`). Works as `"|"` like what `"g0"` is to `"0"`.
- 7 Support setting `'equalprg'` to a user function name.
- 7 Highlight the characters after the end-of-line differently.
- 7 When `'whichwrap'` contains `"l"`, `"$dl"` should join lines?
- 8 Add an argument to `configure` to use `$CFLAGS` and not modify it? (Mooney)
- 8 Enabling features is a mix of `configure` arguments and defines in `feature.h`. How to make this consistent? `Feature.h` is required for non-unix systems. Perhaps let `configure` define `CONF_XXX`, and use `#ifdef CONF_XXX` in `feature.h`? Then what should `min-features` and `max-features` do?
- 8 Add `"g^E"` and `"g^Y"`, to scroll a screen-full line up and down.
- 8 Add `":confirm"` handling in `open_exfile()`, for when file already exists.
- 8 When quitting with changed files, make the dialog list the changed file and allow `"write all"`, `"discard all"`, `"write some"`. The last one would then ask `"write"` or `"discard"` for each changed file. Patch in `HierAssist` does something like this. (Shah)
- 7 Use `growarray` for `replace` stack.
- 7 Have a look at `viH` (Hellenic or Greek version of Vim). But a solution outside of Vim might be satisfactory (Haritsis).
- 3 Make `"2d%"` work like `"d%d%"` instead of `"d2%"`?
- 7 `"g CTRL-O"` jumps back to last used buffer. Skip `CTRL-O` jumps in the same buffer. Make `jumplist` remember the last ten accessed buffers?
- 7 Make it possible to set the size of the `jumplist` (also to a smaller number than the default). (Nikolai Weibull)
- Add code to disable the CAPS key when going from Insert to Normal mode.

- Set date/protection/etc. of the patchfile the same as the original file.
- Use growarray for termcodes[] in term.c
- Add <window-99>, like <cword> but use filename of 99'th window.
- 7 Add a way to change an operator to always work characterwise-inclusive (like "v" makes the operator characterwise-exclusive). "x" could be used.
- Make a set of operations on list of names: expand wildcards, replace home dir, append a string, delete a string, etc.
- Remove using mktemp() and use tmpname() only? Ctags does this.
- When replacing environment variables, and there is one that is not set, turn it into an empty string? Only when expanding options? (Hiebert)
- Option to set command to be executed instead of producing a beep (e.g. to call "play newbeep.au").
- Add option to show the current function name in the status line. More or less what you find with "[[k", like how 'cindent' recognizes a function. (Bhatt).
- "[r" and "]r": like "p" and "P", but replace instead of insert (esp. for blockwise registers).
- Add 'timecheck' option, on by default. Makes it possible to switch off the timestamp warning and question. (Dodt).
- Add an option to set the time after which Vim should check the timestamps of the files. Only check when an event occurs (e.g., character typed, mouse moved). Useful for non-GUI versions where keyboard focus isn't noticeable.
- Make 'smartcase' work even though 'ic' isn't set (Webb).
- 7 When formatting text, allow to break the line at a number of characters. Use an option for this: 'breakchars'? Useful for formatting Fortran code.
- Add flag to 'formatoptions' to be able to format book-style paragraphs (first line of paragraph has larger indent, no empty lines between paragraphs). Complements the '2' flag. Use '>' flag when larger indent starts a new paragraph, use '<' flag when smaller indent starts a new paragraph. Both start a new paragraph on any indent change.
- 8 The 'a' flag in 'formatoptions' is too dangerous. In some way only do auto-formatting in specific regions, e.g. defined by syntax highlighting.
- 8 Allow using a trailing space to signal a paragraph that continues on the next line (MIME text/plain; format=flowed, RFC 2646). Can be used for continuous formatting. Could use 'autoformat' option, which specifies a regexp which triggers auto-formatting (for one line).
":set autoformat=\\s\$".
- Be able to redefine where a sentence stops. Use a regexp pattern?
- Support multibyte characters for sentences. Example from Ben Peterson.
- 7 Add command "g)" to go to the end of a sentence, "g(" to go back to the end of a sentence. (Servatius Brandt)
- Be able to redefine where a paragraph starts. For "[[" where the '{' is not in column 1.
- 6 Add ":cdprev": go back to the previous directory. Need to remember a stack of previous directories. We also need ":cdnext".
- 7 Should ":cd" for MS-DOS go to \$HOME, when it's defined?
- Make "gq<CR>" work on the last line in the file. Maybe for every operator?
- Add more redirecting of Ex commands:
:redir #> bufname
:redir #>> bufname (append)
- Give error message when starting :redir: twice or using END when no redirection was active.
- Setting of options, specifically for a buffer or window, with

- ":set window.option" or ":set buffer.option=val". Or use ":buffer.set".
Also: "buffer.map <F1> quit".
- 6 Would it be possible to change the color of the cursor in the Win32 console? (Klaus Hast)
- Add :delcr command:
 - :delcr**
 - :[range]delcr[!] Check [range] lines (default: whole buffer) for lines ending in <CR>. If all lines end in <CR>, or [!] is used, remove the <CR> at the end of lines in [range]. A **CTRL-Z** at the end of the file is removed. If [range] is omitted, or it is the whole file, and all lines end in <CR> **'textmode'** is set.
 - Should integrate addstar() and file_pat_to_reg_pat().
 - When working over a serial line with 7 bit characters, remove meta characters from **'isprint'**.
 - Use fchdir() in init_homedir(), like in FullName().
 - In win_update(), when the GUI is active, always use the scrolling area. Avoid that the last status line is deleted and needs to be redrawn.
 - That "cTx" fails when the cursor is just after 'x' is Vi compatible, but may not be what you expect. Add a flag in **'cptions'** for this? More general: Add an option to allow "c" to work with a null motion.
 - Give better error messages by using errno (strerror()).
 - Give "Usage:" error message when command used with wrong arguments (like Nvi).
 - Make **'restorescreen'** option also work for xterm (and others), replaces the SAVE_XTERM_SCREEN define.
- 7 Support for ":winpos" In xterm: report the current window position.
- Give warning message when using ":set t_xx=asdf" for a termcap code that Vim doesn't know about. Add flag in **'shortmess'**?
- 6 Add ":che <file>", list all the include paths which lead to this file.
- For a commandline that has several commands (:s, :d, etc.) summarize the changes all together instead of for each command (e.g. for the rot13 macro).
 - Add command like "[I" that also shows the tree of included files.
 - ":set sm^L" results in ":set s", because short names of options are also expanded. Is there a better way to do this?
 - Add ":@!" command, to "@!" like what ":source!" is to ":source".
- 8 Add ":@!": repeat last command with forceit set.
- Add 't_normal': Used whenever t_me, t_se, t_ue or t_Zr is empty.
 - ":cab map test ^V| je", ":cunab map" doesn't work. This is vi compatible!
 - **CTRL-W CTRL-E** and **CTRL-W CTRL-Y** should move the current window up or down if it is not the first or last window.
 - Include-file-search commands should look in the loaded buffer of a file (if there is one) instead of the file itself.
- 7 Change **'nrformats'** to include the leader for each format. Example:
nrformats=hex:\$,binary:b,octal:0
- Add setting of **'nrformats'** to syntax files.
- **'path'** can become very long, don't use NameBuff for expansion.
 - When unhiding a hidden buffer, put the same line at top of the window as the one before hiding it. Or: keep the same relative cursor position (so many percent down the windows).
 - Make it possible for the **'showbreak'** to be displayed at the end of the line. Use a comma to separate the part at the end and the start of the line? #754 Highlight the linebreak characters, add flag in **'highlight'**.

- Make **'showbreak'** local to a window.
- Some string options should be expanded if they have wildcards, e.g. **'dictionary'** when it is `"*.h"`.
- Use a specific type for number and boolean options, making it possible to change it for specific machines (e.g. when a long is 64 bit).
- Add option for `<Insert>` in replace mode going to normal mode. (Nugent)
- Add a next/previous possibility to `"[^I"` and friends.
- Add possibility to change the HOME directory. Use the directory from the passwd file? (Antwerpen)
- 8 Add commands to push and pop all or individual options. `":setpush tw"`, `":setpop tw"`, `":setpush all"`. Maybe pushing/popping all options is sufficient. `":setflush"` resets the option stack?
How to handle an aborted mapping? Remember position in tag stack when mapping starts, restore it when an error aborts the mapping?
- Change `":fixdel"` into option **'fixdel'**, `t_del` will be adjusted each time `t_bs` is set? (Webb)
- `"gc"`: goto character, move absolute character positions forward, also counting newlines. `"gC"` goes backwards (Weigert).
- When doing **CTRL-^**, redraw buffer with the same topline. (Demirel) Store cursor row and window height to redraw cursor at same percentage of window (Webb).
- Besides remembering the last used line number of a file, also remember the column. Use it with **CTRL-^** et. al.
- Check for non-digits when setting a number option (careful when entering hex codes like `0xff`).
- Add option to make `."` redo the `"@r"` command, instead of the last command executed by it. Also to make `."` redo the whole mapping. Basically: redo the last TYPED command.
- Support URL links for `^X^F` in Insert mode, like for `"gf"`.
- Support `%name%` expansion for `"gf"` on Windows.
- Make `"gf"` work on `"file://c:/path/name"`. `"file:/c:/"` and `"file:///c:/"` should also work?
- Add **'urlpath'**, used like **'path'** for when `"gf"` used on a URL?
- 8 When using `"gf"` on an absolute file name, while editing a remote file (starts with `scp://` or `http://`) should prepend the method and machine name.
- When finding a URL or file name, and it doesn't exist, try removing a trailing `'.'`.
- Add `":path"` command modifier. Should work for every command that takes a file name argument, to search for the file name in **'path'**. Use `find_file_in_path()`.
- Highlight control characters on the screen: Shows the difference between **CTRL-X** and `"^"` followed by `"X"` (Colon).
- Integrate parsing of cmdline command and parsing for expansion.
- Create a program that can translate a `.swp` file from any machine into a form usable by Vim on the current machine.
- Add `":noro"` command: Reset **'ro'** flag for all buffers, except ones that have a readonly file. `":noro!"` will reset all **'ro'** flags.
- Add a variant of **CTRL-V** that stops interpretation of more than one character. For entering mappings on the command line where a key contains several special characters, e.g. a trailing newline.
- Make `'2'` option in **'formatoptions'** also work inside comments.
- Add `'s'` flag to **'formatoptions'**: Do not break when inside a string. (Dodt)
- When window size changed (with the mouse) and made too small, set it back

- to the minimal size.
- Add "]">" and "[<", shift comment at end of line (command; /* comment */).
- Should not call cursorcmd() for each vgetc() in getcmdline().
- ":split file1 file2" adds two more windows (Webb).
- Don't give message "Incomplete last line" when editing binary file.
- Add ":a", ":i" for preloading of named buffers.
- When entering text, keep other windows on same buffer updated (when a line entered)?
- Check out how screen does output optimizing. Apparently this is possible as an output filter.
- In dosub() regexexec is called twice for the same line. Try to avoid this.
- Window updating from memline.c: insert/delete/replace line.
- Optimize ml_append() for speed, esp. for reading a file.
- V..c should keep indent when 'ai' is set, just like [count]cc.
- Updatescript() can be done faster with a string instead of a char.
- Screen updating is inefficient with **CTRL-F** and **CTRL-B** when there are long lines.
- Uppercase characters in Ex commands can be made lowercase?
- 8 Add option to show characters in text not as "|A" but as decimal ("^129"), hex ("\x81") or octal ("\201") or meta (M-x). Nvi has the 'octal' option to switch from hex to octal. Vile can show unprintable characters in hex or in octal.
- 7 Tighter integration with xxd to edit binary files. Make it more easy/obvious to use. Command line argument?
- How does vi detect whether a filter has messed up the screen? Check source. After ":w !command" a wait_return?
- Improve screen updating code for doput() (use s_ins()).
- With 'p' command on last line: scroll screen up (also for terminals without insert line command).
- Use insert/delete char when terminal supports it.
- Optimize screen redraw for slow terminals.
- Optimize "dw" for long row of spaces (say, 30000).
- Add "-d null" for editing from a script file without displaying.
- In Insert mode: Remember the characters that were removed with backspace and re-insert them one at a time with <key1>, all together with <key2>.
- Amiga: Add possibility to set a keymap. The code in amiga.c does not work yet.
- Implement 'redraw' option.
- Add special code to 'sections' option to define something else but '{' or '}' as the start of a section (e.g. one shiftwidth to the right).
- 7 Allow using Vim in a pipe: "ls | vim -u xxx.vim - | yyy". Only needs implementing ":w" to stdout in the buffer that was read from stdin. Perhaps writing to stdout will work, since stderr is used for the terminal I/O.
- 8 Allow opening an unnamed buffer with ":e !cmd" and ":sp !cmd". Vile can do it.
- Add commands like]] and [[that do not include the line jumped to.
- When :unab without matching "from" part and several matching "to" parts, delete the entry that was used last, instead of the first in the list.
- Add text justification option.
- Set boolean options on/off with ":set paste=off", ":set paste=on".
- After "inv"ing an option show the value: ":set invpaste" gives "paste is off".
- Check handling of **CTRL-V** and '\ ' for ":" commands that do not have TRLBAR.

- When a file cannot be opened but does exist, give error message.
- Amiga: When 'r' protection bit is not set, file can still be opened but gives read errors. Check protection before opening.
- When writing check for file exists but no permission, "Permission denied".
- If file does not exist, check if directory exists.
- Settings edit mode: make file with ":set opt=xx", edit it, parse it as ex commands.
- ":set -w all": list one option per line.
- Amiga: test for 'w' flag when reading a file.
- :table command (Webb)
- Add new operator: clear, make area white (replace with spaces): "g ".
- Add command to ":read" a file at a certain column (blockwise read?).
- Add sort of replace mode where case is taken from the old text (Goldfarb).
- Allow multiple arguments for ":read", read all the files.
- Support for tabs in specific columns: ":set tabcol=8,20,34,56" (Demirel).
- Add '**searchdir**' option: Directories to search for file name being edited (Demirel).
- Modifier for the put command: Change to linewise, charwise, blockwise, etc.
- Add commands for saving and restoring options ":set save" "set restore", for use in macro's and the like.
- Keep output from listings in a window, so you can have a look at it while working in another window. Put cmdline in a separate window?
- Add possibility to put output of Ex commands in a buffer or file, e.g. for ":set all". ":r :set all"?
- When the '**equalalways**' option is set, creating a new window should not result in windows to become bigger. Deleting a window should not result in a window to become smaller (Webb).
- When resizing the whole Vim window, the windows inside should be resized proportionally (Webb).
- Include options directly in option table, no indirect pointers. Use mkofttab to make option table?
- When doing ":w dir", where "dir" is a directory name, write the current file into that directory, with the current file name (without the path)?
- Support for '**dictionary**'s that are sorted, makes access a lot faster (Haritsis).
- Add "^Vrx" on the command line, replace with contents of register x. Used instead of **CTRL-R** to make repeating possible. (Marinichev)
- Add "^Vb" on the command line, replace with word before or under the cursor?
- Support mapping for replace mode and "r" command (Vi doesn't do this)?
- 8 Sorting of filenames for completion is wrong on systems that ignore case of filenames. Add '**ignorefn case**' option. When set, case in filenames is ignored for sorting them. Patch by Mike Williams: ~/vim/patches/ignorefn case. Also change what matches? Or use another option name.
- 8 Should be able to compile Vim in another directory, with \$(srcdir) set to where the sources are. Add \$(srcdir) in the Makefile in a lot of places. (Netherton)
- 6 Make it configurable when "J" inserts a space or not. Should not add a space after "(", for example.
- 5 When inserting spaces after the end-of-line for '**virtualedit**', use tabs when the user wants this (e.g., add a "tab" field to '**virtualedit**'). (Servatius Brandt)

From Elvis:

- Use "instman.sh" to install manpages?
- Add ":alias" command.
- Search patterns:
 - \@ match word under cursor.
- but do:
 - \@w match the word under the cursor?
 - \@W match the WORD under the cursor?
- 8 ":window" command:
 - :win + next window (up)
 - :win ++ idem, wrapping
 - :win - previous window (down)
 - :win -- idem, wrapping
 - :win nr to window number "nr"
 - :win name to window editing buffer "name"
- 7 ":cc" compiles a single file (default: current one). 'ccprg' option is program to use with ":cc". Use ":compile" instead of ":cc"?

From xvi:

- CTRL-_: swap 8th bit of character.
- Add egrep-like regex type, like xvi (Ned Konz) or Perl (Emmanuel Mogenet)

From vile:

- When horizontal scrolling, use '>' for lines continuing right of a window.
- Support putting .swp files in /tmp: Command in rc.local to move .swp files from /tmp to some directory before deleting files.

Far future and "big" extensions:

- Instead of using a Makefile and autoconf, use a simple shell script to find the C compiler and do everything with C code. Translate something like an Aap recipe and configure.ac to C. Avoids depending on Python, thus will work everywhere. With batch file to find the C compiler it would also work on MS-Windows.
- Make it easy to setup Vim for groups of users: novice vi users, novice Vim users, C programmers, xterm users, GUI users,...
- Change layout of blocks in swap file: Text at the start, with '\n' in between lines (just load the file without changes, except for Mac). Indexes for lines are from the end of the block backwards. It's the current layout mirrored.
- Make it possible to edit a register, in a window, like a buffer.
- Add stuff to syntax highlighting to change the text (upper-case keywords, set indent, define other highlighting, etc.).
- Mode to keep C-code formatted all the time (sort of on-line indent).
- Several top-level windows in one Vim session. Be able to use a different font in each top-level window.
- Allow editing above start and below end of buffer (flag in 'virtualedit').
- Smart cut/paste: recognize words and adjust spaces before/after them.
- Add open mode, use it when terminal has no cursor positioning.
- Special "drawing mode": a line is drawn where the cursor is moved to. Backspace deletes along the line (from jvim).

- Support for underlining (underscore-BS-char), bold (char-BS-char) and other standout modes switched on/off with , **'overstrike'** option (Reiter).
- Add vertical mode (Paul Jury, Demirel): "5vbw" deletes a word in five lines, "3vitextESC" will insert "text" in three lines, etc..
- 4 Recognize l, #, p as **'flags'** to EX commands:
 - :g/RE/#l shall print lines with line numbers and in list format.
 - :g/RE/dp shall print lines that are deleted.
 POSIX: Commands where flags shall apply to all lines written: list, number, open, print, substitute, visual, &, z. For other commands, flags shall apply to the current line after the command completes. Examples:
 - :7,10j #l Join the lines 7-10 and print the result in list
- Allow two or more users to edit the same file at the same time. Changes are reflected in each Vim immediately. Could work with local files but also over the internet. See <http://www.codingmonkeys.de/subethaedit/>.

vim:tw=78:sw=4:sts=4:ts=8:noet:ft=help:norl:

Development of Vim. development

This text is important for those who want to be involved in further developing Vim.

1. Design goals design-goals
2. Coding style coding-style
3. Design decisions design-decisions
4. Assumptions design-assumptions

See the file README.txt in the "src" directory for an overview of the source code.

Vim is open source software. Everybody is encouraged to contribute to help improving Vim. For sending patches a unified diff "diff -u" is preferred. You can create a pull request on github, but it's not required. Also see http://vim.wikia.com/wiki/How_to_make_and_submit_a_patch.

=====

1. Design goals design-goals

Most important things come first (roughly).

Note that quite a few items are contradicting. This is intentional. A balance must be found between them.

VIM IS... VI COMPATIBLE design-compatible

First of all, it should be possible to use Vim as a drop-in replacement for Vi. When the user wants to, Vim can be used in compatible mode and hardly any differences with the original Vi will be noticed.

Exceptions:

- We don't reproduce obvious Vi bugs in Vim.
- There are different versions of Vi. I am using Version 3.7 (6/7/85) as a reference. But support for other versions is also included when possible. The Vi part of POSIX is not considered a definitive source.
- Vim adds new commands, you cannot rely on some command to fail because it didn't exist in Vi.
- Vim will have a lot of features that Vi doesn't have. Going back from Vim to Vi will be a problem, this cannot be avoided.
- Some things are hardly ever used (open mode, sending an e-mail when crashing, etc.). Those will only be included when someone has a good reason why it should be included and it's not too much work.
- For some items it is debatable whether Vi compatibility should be maintained. There will be an option flag for these.

VIM IS... IMPROVED

design-improved

The Improved bits of Vim should make it a better Vi, without becoming a completely different editor. Extensions are done with a "Vi spirit".

- Use the keyboard as much as feasible. The mouse requires a third hand, which we don't have. Many terminals don't have a mouse.
- When the mouse is used anyway, avoid the need to switch back to the keyboard. Avoid mixing mouse and keyboard handling.
- Add commands and options in a consistent way. Otherwise people will have a hard time finding and remembering them. Keep in mind that more commands and options will be added later.
- A feature that people do not know about is a useless feature. Don't add obscure features, or at least add hints in documentation that they exist.
- Minimize using CTRL and other modifiers, they are more difficult to type.
- There are many first-time and inexperienced Vim users. Make it easy for them to start using Vim and learn more over time.
- There is no limit to the features that can be added. Selecting new features is one based on (1) what users ask for, (2) how much effort it takes to implement and (3) someone actually implementing it.

VIM IS... MULTI PLATFORM

design-multi-platform

Vim tries to help as many users on as many platforms as possible.

- Support many kinds of terminals. The minimal demands are cursor positioning and clear-screen. Commands should only use key strokes that most keyboards have. Support all the keys on the keyboard for mapping.
- Support many platforms. A condition is that there is someone willing to do Vim development on that platform, and it doesn't mean messing up the code.
- Support many compilers and libraries. Not everybody is able or allowed to install another compiler or GUI library.
- People switch from one platform to another, and from GUI to terminal version. Features should be present in all versions, or at least in as many as possible with a reasonable effort. Try to avoid that users must switch between platforms to accomplish their work efficiently.
- That a feature is not possible on some platforms, or only possible on one platform, does not mean it cannot be implemented. [This intentionally contradicts the previous item, these two must be balanced.]

VIM IS... WELL DOCUMENTED

design-documented

- A feature that isn't documented is a useless feature. A patch for a new feature must include the documentation.
- Documentation should be comprehensive and understandable. Using examples is recommended.
- Don't make the text unnecessarily long. Less documentation means that an item is easier to find.

VIM IS... HIGH SPEED AND SMALL IN SIZE

design-speed-size

Using Vim must not be a big attack on system resources. Keep it small and

fast.

- Computers are becoming faster and bigger each year. Vim can grow too, but no faster than computers are growing. Keep Vim usable on older systems.
- Many users start Vim from a shell very often. Startup time must be short.
- Commands must work efficiently. The time they consume must be as small as possible. Useful commands may take longer.
- Don't forget that some people use Vim over a slow connection. Minimize the communication overhead.
- Items that add considerably to the size and are not used by many people should be a feature that can be disabled.
- Vim is a component among other components. Don't turn it into a massive application, but have it work well together with other programs.

VIM IS... MAINTAINABLE

design-maintain

- The source code should not become a mess. It should be reliable code.
- Use the same layout in all files to make it easy to read [coding-style](#).
- Use comments in a useful way! Quoting the function name and argument names is NOT useful. Do explain what they are for.
- Porting to another platform should be made easy, without having to change too much platform-independent code.
- Use the object-oriented spirit: Put data and code together. Minimize the knowledge spread to other parts of the code.

VIM IS... FLEXIBLE

design-flexible

Vim should make it easy for users to work in their preferred styles rather than coercing its users into particular patterns of work. This can be for items with a large impact (e.g., the '[compatible](#)' option) or for details. The defaults are carefully chosen such that most users will enjoy using Vim as it is. Commands and options can be used to adjust Vim to the desire of the user and its environment.

VIM IS... NOT

design-not

- Vim is not a shell or an Operating System. It does provide a terminal window, in which you can run a shell or debugger. E.g. to be able to do this over an ssh connection. But if you don't need a text editor with that it is out of scope (use something like screen or tmux instead).
A satirical way to say this: "Unlike Emacs, Vim does not attempt to include everything but the kitchen sink, but some people say that you can clean one with it. ;-)"
To use Vim with gdb see [terminal-debugger](#). Other (older) tools can be found at <http://www.agide.org> and <http://clewn.sf.net>.
- Vim is not a fancy GUI editor that tries to look nice at the cost of being less consistent over all platforms. But functional GUI features are welcomed.

=====
2. Coding style

coding-style

These are the rules to use when making changes to the Vim source code. Please stick to these rules, to keep the sources readable and maintainable.

This list is not complete. Look in the source code for more examples.

MAKING CHANGES

style-changes

The basic steps to make changes to the code:

1. Get the code from github. That makes it easier to keep your changed version in sync with the main code base (it may be a while before your changes will be included). You do need to spend some time learning git, it's not the most user friendly tool.
2. Adjust the documentation. Doing this first gives you an impression of how your changes affect the user.
3. Make the source code changes.
4. Check `../doc/todo.txt` if the change affects any listed item.
5. Make a patch with "git diff". You can also create a pull request on github, but it's the diff that matters.
6. Make a [note](#) about what changed, preferably mentioning the problem and the solution. Send an email to the [vim-dev](#) maillist with an explanation and include the diff. Or create a pull request on github.

C COMPILER

style-compiler ANSI-C C89 C99

The minimal C compiler version supported is C89, also known as ANSI C. Later standards, such as C99, are not widely supported, or at least not 100% supported. Therefore we use only some of the C99 features and explicitly disallow some (this will gradually be adjusted over time).

Please don't make changes everywhere to use the C99 features, it causes merge problems for existing patches. Only use them for new and changed code.

Comments

Traditionally Vim uses `/* comments */`. We intend to keep it that way for file and function headers and larger blocks of code, E.g.:

```
/*
 * The "foo" argument does something useful.
 * Return OK or FAIL.
 */
```

For new code or lines of code that change, it is preferred to use `//` comments. Especially when it comes after code:

```
int some_var; // single line comment useful here
```

Enums

The last item in an enum may have a trailing comma. C89 didn't allow this.

Types

"long long" is allowed and can be expected to be 64 bits. Use `%lld` in printf formats. Also "long long unsigned" with `%llu`.

Declarations

Now that the minimal supported compiler is MSVC 2015 declarations do not need to be at the start of a block. However, it is often a good idea to do this anyway.

Declaration of the for loop variable inside the loop is recommended:

```
for (int i = 0; i < len; ++i)
```

Since this is clearly an advantage we'll use this more often.

Not to be used

These C99 features are not to be used, because not enough compilers support them:

- Variable length arrays (even in C11 this is an optional feature).
- `_Bool` and `_Complex` types.
- "inline" (it's hardly ever needed, let the optimizer do its work)
- flexible array members: Not supported by HP-UX C compiler (John Marriott)

USE OF COMMON FUNCTIONS

style-functions

Some functions that are common to use, have a special Vim version. Always consider using the Vim version, because they were introduced with a reason.

NORMAL NAME	VIM NAME	DIFFERENCE OF VIM VERSION
<code>free()</code>	<code>vim_free()</code>	Checks for freeing NULL
<code>malloc()</code>	<code>alloc()</code>	Checks for out of memory situation
<code>malloc()</code>	<code>lalloc()</code>	Like <code>alloc()</code> , but has long argument
<code>strcpy()</code>	<code>STRCPY()</code>	Includes cast to <code>(char *)</code> , for <code>char_u * args</code>
<code>strchr()</code>	<code>vim_strchr()</code>	Accepts special characters
<code>strrchr()</code>	<code>vim_strrchr()</code>	Accepts special characters
<code>isspace()</code>	<code>vim_isspace()</code>	Can handle characters > 128
<code>iswhite()</code>	<code>vim_iswhite()</code>	Only TRUE for tab and space
<code>memcpy()</code>	<code>mch_memmove()</code>	Handles overlapped copies
<code>bcopy()</code>	<code>mch_memmove()</code>	Handles overlapped copies
<code>memset()</code>	<code>vim_memset()</code>	Uniform for all systems

NAMES

style-names

Function names can not be more than 31 characters long (because of VMS).

Don't use "delete" or "this" as a variable name, C++ doesn't like it.

Because of the requirement that Vim runs on as many systems as possible, we need to avoid using names that are already defined by the system. This is a list of names that are known to cause trouble. The name is given as a regexp pattern.

<code>is.*()</code>	POSIX, <code>ctype.h</code>
<code>to.*()</code>	POSIX, <code>ctype.h</code>

d_.*	POSIX, dirent.h
l_.*	POSIX, fcntl.h
gr_.*	POSIX, grp.h
pw_.*	POSIX, pwd.h
sa_.*	POSIX, signal.h
mem.*	POSIX, string.h
str.*	POSIX, string.h
wcs.*	POSIX, string.h
st_.*	POSIX, stat.h
tms_.*	POSIX, times.h
tm_.*	POSIX, time.h
c_.*	POSIX, termios.h
MAX.*	POSIX, limits.h
__.*	POSIX, system
_[A-Z].*	POSIX, system
E[A-Z0-9]*	POSIX, errno.h
.*_t	POSIX, for typedefs. Use *_T instead.
wait	don't use as argument to a function, conflicts with types.h
index	shadows global declaration
time	shadows global declaration
new	C++ reserved keyword
clear	Mac curses.h
echo	Mac curses.h
instr	Mac curses.h
meta	Mac curses.h
newwin	Mac curses.h
nl	Mac curses.h
overwrite	Mac curses.h
refresh	Mac curses.h
scroll	Mac curses.h
typeahead	Mac curses.h
basename()	GNU string function
dirname()	GNU string function
get_env_value()	Linux system function

VARIOUS

style-various

Typedef'ed names should end in "_T":

```
typedef int some_T;
```

Define'ed names should be uppercase:

```
#define SOME_THING
```

Features always start with "FEAT_":

```
#define FEAT_F00
```

Don't use '\', some compilers can't handle it. '' works fine.

Don't use:

```
#if HAVE_SOME
```

Some compilers can't handle that and complain that "HAVE_SOME" is not defined.
Use

```
#ifdef HAVE_SOME
or
#if defined(HAVE_SOME)
```

STYLE

[style-examples](#)

General rule: One statement per line.

Wrong: if (cond) a = 1;

OK: if (cond)
 a = 1;

Wrong: while (cond);

OK: while (cond)
 ;

Wrong: do a = 1; while (cond);

OK: do
 a = 1;
 while (cond);

Wrong: if (cond) {
 cmd;
 cmd;
 } else {
 cmd;
 cmd;
 }

OK: if (cond)
 {
 cmd;
 cmd;
 }
 else
 {
 cmd;
 cmd;
 }

When a block has one line the braces can be left out. When an if/else has braces on one block, it usually looks better when the other block also has braces:

OK: if (cond)
 cmd;
 else
 cmd;

```

OK:      if (cond)
          {
            cmd;
          }
          else
          {
            cmd;
            cmd;
          }

```

Use ANSI (new style) function declarations with the return type on a separate indented line.

Wrong: `int function_name(int arg1, int arg2)`

```

OK:      /*
          * Explanation of what this function is used for.
          *
          * Return value explanation.
          */
          int
function_name(
          int      arg1,          // short comment about arg1
          int      arg2)        // short comment about arg2
{
          int      local;        // comment about local

          local = arg1 * arg2;

```

SPACES AND PUNCTUATION

style-spaces

No space between a function name and the bracket:

```

Wrong:  func (arg);
OK:     func(arg);

```

Do use a space after `if`, `while`, `switch`, etc.

```

Wrong:  if(arg)      for(;;)
OK:     if (arg)    for (;;)

```

Use a space after a comma and semicolon:

```

Wrong:  func(arg1,arg2);      for (i = 0;i < 2;++i)
OK:     func(arg1, arg2);     for (i = 0; i < 2; ++i)

```

Use a space before and after `'='`, `'+'`, `'/'`, etc.

```

Wrong:  var=a*5;
OK:     var = a * 5;

```

In general: Use empty lines to group lines of code together. Put a comment

just above the group of lines. This makes it easier to quickly see what is being done.

```
OK:    /* Prepare for building the table. */
        get_first_item();
        table_idx = 0;

        /* Build the table */
        while (has_item())
            table[table_idx++] = next_item();

        /* Finish up. */
        cleanup_items();
        generate_hash(table);
```

3. Design decisions

design-decisions

Folding

Several forms of folding should be possible for the same buffer. For example, have one window that shows the text with function bodies folded, another window that shows a function body.

Folding is a way to display the text. It should not change the text itself. Therefore the folding has been implemented as a filter between the text stored in a buffer (buffer lines) and the text displayed in a window (logical lines).

Naming the window

The word "window" is commonly used for several things: A window on the screen, the xterm window, a window inside Vim to view a buffer.

To avoid confusion, other items that are sometimes called window have been given another name. Here is an overview of the related items:

screen	The whole display. For the GUI it's something like 1024x768 pixels. The Vim shell can use the whole screen or part of it.
shell	The Vim application. This can cover the whole screen (e.g., when running in a console) or part of it (xterm or GUI).
window	View on a buffer. There can be several windows in Vim, together with the command line, menubar, toolbar, etc. they fit in the shell.

Spell checking

develop-spell

When spell checking was going to be added to Vim a survey was done over the available spell checking libraries and programs. Unfortunately, the result was that none of them provided sufficient capabilities to be used as the spell checking engine in Vim, for various reasons:

- Missing support for multibyte encodings. At least UTF-8 must be supported, so that more than one language can be used in the same file.

Doing on-the-fly conversion is not always possible (would require iconv support).

- For the programs and libraries: Using them as-is would require installing them separately from Vim. That's mostly not impossible, but a drawback.
- Performance: A few tests showed that it's possible to check spelling on the fly (while redrawing), just like syntax highlighting. But the mechanisms used by other code are much slower. Myspell uses a hashtable, for example. The affix compression that most spell checkers use makes it slower too.
- For using an external program like aspell a communication mechanism would have to be setup. That's complicated to do in a portable way (Unix-only would be relatively simple, but that's not good enough). And performance will become a problem (lots of process switching involved).
- Missing support for words with non-word characters, such as "Etten-Leur" and "et al.", would require marking the pieces of them OK, lowering the reliability.
- Missing support for regions or dialects. Makes it difficult to accept all English words and highlight non-Canadian words differently.
- Missing support for rare words. Many words are correct but hardly ever used and could be a misspelled often-used word.
- For making suggestions the speed is less important and requiring to install another program or library would be acceptable. But the word lists probably differ, the suggestions may be wrong words.

Spelling suggestions

develop-spell-suggestions

For making suggestions there are two basic mechanisms:

1. Try changing the bad word a little bit and check for a match with a good word. Or go through the list of good words, change them a little bit and check for a match with the bad word. The changes are deleting a character, inserting a character, swapping two characters, etc.
2. Perform soundfolding on both the bad word and the good words and then find matches, possibly with a few changes like with the first mechanism.

The first is good for finding typing mistakes. After experimenting with hashtables and looking at solutions from other spell checkers the conclusion was that a trie (a kind of tree structure) is ideal for this. Both for reducing memory use and being able to try sensible changes. For example, when inserting a character only characters that lead to good words need to be tried. Other mechanisms (with hashtables) need to try all possible letters at every position in the word. Also, a hashtable has the requirement that word boundaries are identified separately, while a trie does not require this. That makes the mechanism a lot simpler.

Soundfolding is useful when someone knows how the words sounds but doesn't know how it is spelled. For example, the word "dictionary" might be written as "daktonerie". The number of changes that the first method would need to try is very big, it's hard to find the good word that way. After soundfolding the words become "tktnr" and "tkxnry", these differ by only two letters.

To find words by their soundfolded equivalent (soundalike word) we need a list of all soundfolded words. A few experiments have been done to find out what the best method is. Alternatives:

1. Do the sound folding on the fly when looking for suggestions. This means

walking through the trie of good words, soundfolding each word and checking how different it is from the bad word. This is very efficient for memory use, but takes a long time. On a fast PC it takes a couple of seconds for English, which can be acceptable for interactive use. But for some languages it takes more than ten seconds (e.g., German, Catalan), which is unacceptably slow. For batch processing (automatic corrections) it's too slow for all languages.

2. Use a trie for the soundfolded words, so that searching can be done just like how it works without soundfolding. This requires remembering a list of good words for each soundfolded word. This makes finding matches very fast but requires quite a lot of memory, in the order of 1 to 10 Mbyte. For some languages more than the original word list.
3. Like the second alternative, but reduce the amount of memory by using affix compression and store only the soundfolded basic word. This is what Aspell does. Disadvantage is that affixes need to be stripped from the bad word before soundfolding it, which means that mistakes at the start and/or end of the word will cause the mechanism to fail. Also, this becomes slow when the bad word is quite different from the good word.

The choice made is to use the second mechanism and use a separate file. This way a user with sufficient memory can get very good suggestions while a user who is short of memory or just wants the spell checking and no suggestions doesn't use so much memory.

Word frequency

For sorting suggestions it helps to know which words are common. In theory we could store a word frequency with the word in the dictionary. However, this requires storing a count per word. That degrades word tree compression a lot. And maintaining the word frequency for all languages will be a heavy task. Also, it would be nice to prefer words that are already in the text. This way the words that appear in the specific text are preferred for suggestions.

What has been implemented is to count words that have been seen during displaying. A hashtable is used to quickly find the word count. The count is initialized from words listed in COMMON items in the affix file, so that it also works when starting a new file.

This isn't ideal, because the longer Vim is running the higher the counts become. But in practice it is a noticeable improvement over not using the word count.

=====

4. Assumptions

design-assumptions

Size of variables:

char	8 bit signed
char_u	8 bit unsigned
int	32 or 64 bit signed (16 might be possible with limited features)
unsigned	32 or 64 bit unsigned (16 as with ints)
long	32 or 64 bit signed, can hold a pointer

Note that some compilers cannot handle long lines or strings. The C89

standard specifies a limit of 509 characters.

Debugging Vim

debug-vim

This is for debugging Vim itself, when it doesn't work properly. For debugging Vim scripts, functions, etc. see [debug-scripts](#)

- | | |
|---|-----------------------------|
| 1. Location of a crash, using gcc and gdb | debug-gcc |
| 2. Locating memory leaks | debug-leaks |
| 3. Windows Bug Reporting | debug-win32 |

-
- | | | |
|---|---------------------------|---------------------|
| 1. Location of a crash, using gcc and gdb | debug-gcc | gdb |
|---|---------------------------|---------------------|

When Vim crashes in one of the test files, and you are using gcc for compilation, here is what you can do to find out exactly where Vim crashes. This also applies when using the MingW tools.

1. Compile Vim with the "-g" option (there is a line in the src/Makefile for this, which you can uncomment). Also make sure "strip" is disabled (do not install it, or use the line "STRIP = /bin/true").
2. Execute these commands (replace "11" with the test that fails):


```
cd testdir
gdb ../vim
run -u unix.vim -U NONE -s dotest.in test11.in
```
3. Check where Vim crashes, gdb should give a message for this.
4. Get a stack trace from gdb with this command:


```
where
```

 You can check out different places in the stack trace with:


```
frame 3
```

 Replace "3" with one of the numbers in the stack trace.

-
- | | | |
|--------------------------|-----------------------------|--------------------------|
| 2. Locating memory leaks | debug-leaks | valgrind |
|--------------------------|-----------------------------|--------------------------|

If you suspect Vim is leaking memory and you are using Linux, the valgrind tool is very useful to pinpoint memory leaks.

First of all, build Vim with EXITFREE defined. Search for this in MAKEFILE and uncomment the line.

Use this command to start Vim:

```
valgrind --log-file=valgrind.log --leak-check=full ./vim
```

Note: Vim will run much slower. If your `.vimrc` is big or you have several plugins you need to be patient for startup, or run with the `"--clean"` argument.

There are often a few leaks from libraries, such as `getpwuid()` and `XtVaAppCreateShell()`. Those are unavoidable. The number of bytes should be very small a Kbyte or less.

3. Windows Bug Reporting

[debug-win32](#)

If the Windows version of Vim crashes in a reproducible manner, you can take some steps to provide a useful bug report.

3.1 GENERIC

You must obtain the debugger symbols (PDB) file for your executable: `gvim.pdb` for `gvim.exe`, or `vim.pdb` for `vim.exe`. The PDB should be available from the same place that you obtained the executable. Be sure to use the PDB that matches the EXE (same date).

If you built the executable yourself with the Microsoft Visual C++ compiler, then the PDB was built with the EXE.

If you have Visual Studio, use that instead of the VC Toolkit and WinDbg.

For other compilers, you should always use the corresponding debugger: `gdb` (see above [debug-gcc](#)) for the Cygwin and MinGW compilers.

[debug-vs2005](#)

3.2 Debugging Vim crashes with Visual Studio 2005/Visual C++ 2005 Express

First launch `vim.exe` or `gvim.exe` and then launch Visual Studio. (If you don't have Visual Studio, follow the instructions at [get-ms-debuggers](#) to obtain a free copy of Visual C++ 2005 Express Edition.)

On the Tools menu, click Attach to Process. Choose the Vim process.

In Vim, reproduce the crash. A dialog will appear in Visual Studio, telling you about the unhandled exception in the Vim process. Click Break to break into the process.

Visual Studio will pop up another dialog, telling you that no symbols are loaded and that the source code cannot be displayed. Click OK.

Several windows will open. Right-click in the Call Stack window. Choose Load Symbols. The Find Symbols dialog will open, looking for `(g)vim.pdb`. Navigate to the directory where you have the PDB file and click Open.

At this point, you should have a full call stack with vim function names and line numbers. Double-click one of the lines and the Find Source dialog will

appear. Navigate to the directory where the Vim source is (if you have it.)

If you don't know how to debug this any further, follow the instructions at `":help bug-reports"`. Paste the call stack into the bug report.

If you have a non-free version of Visual Studio, you can save a minidump via the Debug menu and send it with the bug report. A minidump is a small file (<100KB), which contains information about the state of your process. Visual C++ 2005 Express Edition cannot save minidumps and it cannot be installed as a just-in-time debugger. Use WinDbg, [debug-windbg](#), if you need to save minidumps or you want a just-in-time (postmortem) debugger.

[debug-windbg](#)

3.3 Debugging Vim crashes with WinDbg

See [get-ms-debuggers](#) to obtain a copy of WinDbg.

As with the Visual Studio IDE, you can attach WinDbg to a running Vim process. You can also have your system automatically invoke WinDbg as a postmortem debugger. To set WinDbg as your postmortem debugger, run `"windbg -I"`.

To attach WinDbg to a running Vim process, launch WinDbg. On the File menu, choose Attach to a Process. Select the Vim process and click OK.

At this point, choose Symbol File Path on the File menu, and add the folder containing your Vim PDB to the sympath. If you have Vim source available, use Source File Path on the File menu. You can now open source files in WinDbg and set breakpoints, if you like. Reproduce your crash. WinDbg should open the source file at the point of the crash. Using the View menu, you can examine the call stack, local variables, watch windows, and so on.

If WinDbg is your postmortem debugger, you do not need to attach WinDbg to your Vim process. Simply reproduce the crash and WinDbg will launch automatically. As above, set the Symbol File Path and the Source File Path.

To save a minidump, type the following at the WinDbg command line:

```
.dump vim.dmp
```

[debug-minidump](#)

3.4 Opening a Minidump

If you have a minidump file, you can open it in Visual Studio or in WinDbg.

In Visual Studio 2005: on the File menu, choose Open, then Project/Solution. Navigate to the .dmp file and open it. Now press F5 to invoke the debugger. Follow the instructions in [debug-vs2005](#) to set the Symbol File Path.

In WinDbg: choose Open Crash Dump on the File menu. Follow the instructions in [debug-windbg](#) to set the Symbol File Path.

[get-ms-debuggers](#)

3.5 Obtaining Microsoft Debugging Tools

The Debugging Tools for Windows (including WinDbg) can be downloaded from

<http://www.microsoft.com/whdc/devtools/debugging/default.msp>
This includes the WinDbg debugger.

Visual C++ 2005 Express Edition can be downloaded for free from:
<http://msdn.microsoft.com/vstudio/express/visualC/default.aspx>



uganda Uganda copying copyright license

SUMMARY

iccf ICCF

Vim is Charityware. You can use and copy it as much as you like, but you are encouraged to make a donation for needy children in Uganda. Please see [kcc](#) below or visit the ICCF web site, available at these URLs:

<https://iccf-holland.org/>
<https://www.vim.org/iccf/>
<https://www.iccf.nl/>

You can also sponsor the development of Vim. Vim sponsors can vote for features. See [sponsor](#) . The money goes to Uganda anyway.

The Open Publication License applies to the Vim documentation, see [manual-copyright](#) .

=== begin of license ===

VIM LICENSE

- I) There are no restrictions on distributing unmodified copies of Vim except that they must include this license text. You can also distribute unmodified parts of Vim, likewise unrestricted except that they must include this license text. You are also allowed to include executables that you made from the unmodified Vim sources, plus your own usage examples and Vim scripts.
- II) It is allowed to distribute a modified (or extended) version of Vim, including executables and/or source code, when the following four conditions are met:
 - 1) This license text must be included unmodified.
 - 2) The modified Vim must be distributed in one of the following five ways:
 - a) If you make changes to Vim yourself, you must clearly describe in the distribution how to contact you. When the maintainer asks you (in any way) for a copy of the modified Vim you distributed, you must make your changes, including source code, available to the maintainer without fee. The maintainer reserves the right to include your changes in the official version of Vim. What the maintainer will do with your changes and under what license they will be distributed is negotiable. If there has been no negotiation then this license, or a later version, also applies to your changes. The current maintainers are listed here: <https://github.com/orgs/vim/people>. If this changes it will be announced in appropriate places (most likely vim.sf.net, www.vim.org and/or comp.editors). When it is completely impossible to contact the maintainer, the obligation to send him your changes ceases. Once the maintainer has confirmed that he has received your changes they will not have to be sent again.

- b) If you have received a modified Vim that was distributed as mentioned under a) you are allowed to further distribute it unmodified, as mentioned at I). If you make additional changes the text under a) applies to those changes.
 - c) Provide all the changes, including source code, with every copy of the modified Vim you distribute. This may be done in the form of a context diff. You can choose what license to use for new code you add. The changes and their license must not restrict others from making their own changes to the official version of Vim.
 - d) When you have a modified Vim which includes changes as mentioned under c), you can distribute it without the source code for the changes if the following three conditions are met:
 - The license that applies to the changes permits you to distribute the changes to the Vim maintainer without fee or restriction, and permits the Vim maintainer to include the changes in the official version of Vim without fee or restriction.
 - You keep the changes for at least three years after last distributing the corresponding modified Vim. When the maintainer or someone who you distributed the modified Vim to asks you (in any way) for the changes within this period, you must make them available to him.
 - You clearly describe in the distribution how to contact you. This contact information must remain valid for at least three years after last distributing the corresponding modified Vim, or as long as possible.
 - e) When the GNU General Public License (GPL) applies to the changes, you can distribute the modified Vim under the GNU GPL version 2 or any later version.
- 3) A message must be added, at least in the output of the ":version" command and in the intro screen, such that the user of the modified Vim is able to see that it was modified. When distributing as mentioned under 2)e) adding the message is only required for as far as this does not conflict with the license used for the changes.
- 4) The contact information as required under 2)a) and 2)d) must not be removed or changed, except that the person himself can make corrections.

III) If you distribute a modified version of Vim, you are encouraged to use the Vim license for your changes and make them available to the maintainer, including the source code. The preferred way to do this is by e-mail or by uploading the files to a server and e-mailing the URL. If the number of changes is small (e.g., a modified Makefile) e-mailing a context diff will do. The e-mail address to be used is maintainer@vim.org

IV) It is not allowed to remove this license from the distribution of the Vim sources, parts of it or from a modified version. You may use this license for previous Vim releases instead of the license that they came with, at your option.

=== end of license ===

Note:

- If you are happy with Vim, please express that by reading the rest of this file and consider helping needy children in Uganda.
- If you want to support further Vim development consider becoming a [sponsor](#) . The money goes to Uganda anyway.
- According to Richard Stallman the Vim license is GNU GPL compatible. A few minor changes have been made since he checked it, but that should not make a difference.
- If you link Vim with a library that goes under the GNU GPL, this limits further distribution to the GNU GPL. Also when you didn't actually change anything in Vim.
- Once a change is included that goes under the GNU GPL, this forces all further changes to also be made under the GNU GPL or a compatible license.
- If you distribute a modified version of Vim, you can include your name and contact information with the "--with-modified-by" configure argument or the MODIFIED_BY define.

=====

Kibaale Children's Centre

[kcc](#) [Kibaale](#) [charity](#)

Kibaale Children's Centre (KCC) is located in Kibaale, a small town in the south of Uganda, near Tanzania, in East Africa. The area is known as Rakai District. The population is mostly farmers. Although people are poor, there usually is enough food. But this district is suffering from AIDS more than any other part of the world. Some say that it started there. Estimations are that in the past 10 to 30% of the Ugandans are infected with HIV. Because parents die, there are many orphans. In this district about 60,000 children have lost one or both parents, out of a population of 350,000. Although AIDS is now mostly under control, the problems are still continuing.

The children need a lot of help. The KCC is working hard to provide the needy with food, medical care and education. Food and medical care to keep them healthy now, and education so that they can take care of themselves in the future. KCC works on a Christian base, but help is given to children of any religion.

The key to solving the problems in this area is education. This has been neglected in the past years with president Idi Amin and the following civil wars. Now that the government is stable again, the children and parents have to learn how to take care of themselves and how to avoid infections. There is also help for people who are ill and hungry, but the primary goal is to prevent people from getting ill and to teach them how to grow healthy food.

Most of the orphans are living in an extended family. An uncle or older sister is taking care of them. Because these families are big and the income (if any) is low, a child is lucky if it gets healthy food. Clothes, medical care and schooling is beyond its reach. To help these needy children, a sponsorship program was put into place. A child can be financially adopted. For a few dollars a month KCC sees to it that the child gets indispensable items, is healthy, goes to school and KCC takes care of anything else that

needs to be done for the child and the family that supports it.

Besides helping the child directly, the environment where the child grows up needs to be improved. KCC helps schools to improve their teaching methods. There is a demonstration school at the centre and teacher trainings are given. Health workers are being trained, hygiene education is carried out and households are stimulated to build a proper latrine. I helped setting up a production site for cement slabs. These are used to build a good latrine. They are sold below cost price.

There is a clinic at the project, which provides children and their family medical help. Since 2020 a maternity ward was added and 24/7 service is available. When needed, transport to a hospital is offered. Immunization programs are carried out and help is provided when an epidemic is breaking out (measles and cholera have been a problem).

[donate](#)

Summer 1994 to summer 1995 I spent a whole year at the centre, working as a volunteer. I have helped to expand the centre and worked in the area of water and sanitation. I learned that the help that the KCC provides really helps. When I came back to Holland, I wanted to continue supporting KCC. To do this I'm raising funds and organizing the sponsorship program. Please consider one of these possibilities:

1. Sponsor a child in primary school: 17 euro a month (or more).
2. Sponsor a child in secondary school: 25 euro a month (or more).
3. Sponsor the clinic: Any amount a month or quarter
4. A one-time donation

Compared with other organizations that do child sponsorship the amounts are very low. This is because the money goes directly to the centre. Less than 5% is used for administration. This is possible because this is a small organization that works with volunteers. If you would like to sponsor a child, you should have the intention to do this for at least one year.

How do you know that the money will be spent right? First of all you have my personal guarantee as the author of Vim. I trust the people that are working at the centre, I know them personally. Furthermore, the centre has been co-sponsored and inspected by World Vision, Save the Children Fund and is now under the supervision of Pacific Academy Outreach Society. The centre is visited about once a year to check the progress (at our own cost). I have visited the centre myself many times, starting in 1993. The visit reports are on the ICCF web site.

If you have any further questions, send me e-mail: Bram@vim.org.

The address of the centre is:

Kibaale Children's Centre
p.o. box 1658
Masaka, Uganda, East Africa

Sending money:

[iccf-donations](#)

Check the ICCF web site for the latest information! See [iccf](#) for the URL.

- USA: The methods mentioned below can be used.
If you must send a check send it to our Canadian partner:
<https://www.kuwasha.net/>
- Canada: Contact Kuwasha in Surrey, Canada. They take care of the Canadian sponsors for the children in Kibaale. Kuwasha forwards 100% of the money to the project in Uganda. You can send them a one time donation directly.
Please send me a [note](#) so that I know what has been donated because of Vim. Look on their site for information about sponsorship: <https://www.kuwasha.net/>
If you make a donation to Kuwasha you will receive a tax receipt which can be submitted with your tax return.
- Holland: Transfer to the account of "Stichting ICCF Holland" in Amersfoort. This will allow for tax deduction if you live in Holland. ING bank, IBAN: NL95 INGB 0004 5487 74
- Germany: It is possible to make donations that allow for a tax return. Check the ICCF web site for the latest information:
<https://iccf-holland.org/germany.html>
- Europe: Use a bank transfer if possible. See "Others" below for the swift code and IBAN number.
Any other method should work. Ask for information about sponsorship.
- Credit Card: You can use PayPal to send money with a Credit card. This is the most widely used Internet based payment system. It's really simple to use. Use this link to find more info:
https://www.paypal.com/en_US/mrb/pal=XAC62PML3GF8Q
The e-mail address for sending the money to is:
Bram@iccf-holland.org
- Others: Transfer to this account if possible:
ING bank: IBAN: NL95 INGB 0004 5487 74
Swift code: INGBNL2A
under the name "stichting ICCF Holland", Amersfoort
Checks are not accepted.

Starting Vim

starting

- 1. Vim arguments [vim-arguments](#)
- 2. Vim on the Amiga [starting-amiga](#)
- 3. Running eVim [evim-keys](#)
- 4. Initialization [initialization](#)
- 5. \$VIM and \$VIMRUNTIME [\\$VIM](#)
- 6. Suspending [suspend](#)
- 7. Exiting [exiting](#)
- 8. Saving settings [save-settings](#)
- 9. Views and Sessions [views-sessions](#)
- 10. The viminfo file [viminfo-file](#)

1. Vim arguments

[vim-arguments](#)

Most often, Vim is started to edit a single file with the command

```
vim filename -vim
```

More generally, Vim is started with:

```
vim [option | filename] ..
```

Option arguments and file name arguments can be mixed, and any number of them can be given. However, watch out for options that take an argument.

For compatibility with various Vi versions, see [cmdline-arguments](#) .

Exactly one out of the following five items may be used to choose how to start editing:

[-file](#) [---](#)

filename One or more file names. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. To avoid a file name starting with a '-' being interpreted as an option, precede the arglist with "--", e.g.:

```
vim -- -filename
```

All arguments after the "--" will be interpreted as file names, no other options or "+command" argument can follow. For behavior of quotes on MS-Windows, see [win32-quotes](#) .

[--](#)

- This argument can mean two things, depending on whether Ex mode is to be used.

Starting in Normal mode:

```
vim -
ex -v -
```

Start editing a new buffer, which is filled with text that is read from stdin. The commands that would normally be read from stdin will now be read from stderr. Example:

```
find . -name "*.c" -print | vim -
```

The buffer will be marked as modified, so that you are reminded to save the text when trying to exit. If you don't like that, put these lines in your vimrc:

```
" Don't set 'modified' when reading from stdin
au StdinReadPost * set nomodified
```

Starting in Ex mode:

```
ex -
vim -e -
exim -
vim -E
```

Start editing in silent mode. See `-s-ex`.

- `-t {tag}` -t -tag
A tag. "tag" is looked up in the tags file, the associated file becomes the current file, and the associated command is executed. Mostly this is used for C programs, in which case "tag" often is a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function (see `tags`).
- `-q [errorfile]` -q -qf
QuickFix mode. The file with the name `[errorfile]` is read and the first error is displayed. See `quickfix`. If `[errorfile]` is not given, the `'errorfile'` option is used for the file name. See `'errorfile'` for the default value.
- (nothing) Without one of the four items above, Vim will start editing a new buffer. It's empty and doesn't have a file name.

The startup mode can be changed by using another name instead of "vim", which is equal to giving options:

ex	vim -e	Start in Ex mode (see <code>Ex-mode</code>).	ex
exim	vim -E	Start in improved Ex mode (see <code>Ex-mode</code>). (normally not installed)	exim
view	vim -R	Start in read-only mode (see <code>-R</code>).	view
gvim	vim -g	Start the GUI (see <code>gui</code>).	gvim
gex	vim -eg	Start the GUI in Ex mode.	gex
gview	vim -Rg	Start the GUI in read-only mode.	gview
rvim	vim -Z	Like "vim", but in restricted mode (see <code>-Z</code>).	rvim
rview	vim -RZ	Like "view", but in restricted mode.	rview
rgvim	vim -gZ	Like "gvim", but in restricted mode.	rgvim
rgview	vim -RgZ	Like "gview", but in restricted mode.	rgview
evim	vim -y	Easy Vim: set <code>'insertmode'</code> (see <code>-y</code>).	evim
eview	vim -yR	Like "evim" in read-only mode.	eview

```
vimdiff vim -d      Start in diff mode  diff-mode
gvimdiff vim -gd    Start in diff mode  diff-mode
```

Additional characters may follow, they are ignored. For example, you can have "gvim-8" to start the GUI. You must have an executable by that name then, of course.

On Unix, you would normally have one executable called Vim, and links from the different startup-names to that executable. If your system does not support links and you do not want to have several copies of the executable, you could use an alias instead. For example:

```
alias view vim -R
alias gvim vim -g
```

startup-options

The option arguments may be given in any order. Single-letter options can be combined after one dash. There can be no option arguments after the "--" argument.

On VMS all option arguments are assumed to be lowercase, unless preceded with a slash. Thus "-R" means recovery and "-/R" readonly.

```
--help          -h  --help  -?
```

```
-?
```

```
-h              Give usage (help) message and exit.
                See info-message about capturing the text.
```

--version

```
--version      Print version information and exit. Same output as for
                :version command.
                See info-message about capturing the text.
```

--noplugin

```
--noplugin     Skip loading plugins. Resets the 'loadplugins' option.
```

Note that the `-u` argument may also disable loading plugins:

argument	load: vimrc	files	plugins	defaults.vim
(nothing)	yes	yes	yes	yes
-u NONE	no	no	no	no
-u DEFAULTS	no	no	no	yes
-u NORC	no	yes	no	no
--noplugin	yes	no	no	yes

```
--startuptime {fname}  --startuptime
```

During startup write timing messages to the file {fname}. This can be used to find out where time is spent while loading your .vimrc, plugins and opening the first file. When {fname} already exists new messages are appended. {only available when compiled with the `+startuptime` feature}

--literal

```
--literal      Take file names literally, don't expand wildcards. Not needed
                for Unix, because Vim always takes file names literally (the
```

shell expands wildcards).

Applies to all the names, also the ones that come before this argument.

- `+{num}` -+
The cursor will be positioned on line "num" for the first file being edited. If "num" is missing, the cursor will be positioned on the last line.
- `+/{pat}` -+/
The cursor will be positioned on the first line containing "pat" in the first file being edited (see [pattern](#) for the available search patterns). The search starts at the cursor position, which can be the first line or the cursor position last used from [vminfo](#) . To force a search from the first line use "+1 +/pat".
- `+{command}` -+c
`-c {command}` -c
{command} will be executed after the first file has been read (and after autocommands and modelines for that file have been processed). "command" is interpreted as an Ex command. If the "command" contains spaces, it must be enclosed in double quotes (this depends on the shell that is used).
Example:

```
vim "+set si" main.c
vim "+find stdio.h"
vim -c "set ff=dos" -c wq mine.mak
```
- Note:** You can use up to 10 "+" or "-c" arguments in a Vim command. They are executed in the order given. A "-S" argument counts as a "-c" argument as well.
- `--cmd {command}` --cmd
{command} will be executed before processing any vimrc file. Otherwise it acts like `-c {command}`. You can use up to 10 of these commands, independently from "-c" commands.
- `-S {file}` -S
The {file} will be sourced after the first file has been read. This is an easy way to do the equivalent of:

```
-c "source {file}"
```

It can be mixed with "-c" arguments and repeated like "-c". The limit of 10 "-c" arguments applies here as well. {file} cannot start with a "-".

Do not use this for running a script to do some work and exit Vim, you won't see error messages. Use `-u` instead.
- `-S`
Works like "-S Session.vim". Only when used as the last argument or when another "-" option follows.
- `-r` -r
Recovery mode. Without a file name argument, a list of existing swap files is given. With a file name, a swap file

is read to recover a crashed editing session. See [crash-recovery](#) .

- L -L
Same as -r.
- R -R
Readonly mode. The **'readonly'** option will be set for all the files being edited. You can still edit the buffer, but will be prevented from accidentally overwriting a file. If you forgot that you are in View mode and did make some changes, you can overwrite a file by adding an exclamation mark to the Ex command, as in ":w!". The **'readonly'** option can be reset with ":set noro" (see the options chapter, [options](#)). Subsequent edits will not be done in readonly mode. Calling the executable "view" has the same effect as the -R argument. The **'updatecount'** option will be set to 10000, meaning that the swap file will not be updated automatically very often. See [-M](#) for disallowing modifications.
- m -m
Modifications not allowed to be written. The **'write'** option will be reset, so that writing files is disabled. However, the **'write'** option can be set to enable writing again.
- M -M
Modifications not allowed. The **'modifiable'** option will be reset, so that changes are not allowed. The **'write'** option will be reset, so that writing files is disabled. However, the **'modifiable'** and **'write'** options can be set to enable changes and writing.
- Z -Z restricted-mode E145 E981
Restricted mode. All commands that make use of an external shell are disabled. This includes suspending with **CTRL-Z**, ":sh", filtering, the [system\(\)](#) function, backtick expansion and libcall(). Also disallowed are [delete\(\)](#) , [rename\(\)](#) , [mkdir\(\)](#) , [job_start\(\)](#) , [setenv\(\)](#) etc. Interfaces, such as Python, Ruby and Lua, are also disabled, since they could be used to execute shell commands. Perl uses the Safe module. For Unix restricted mode is used when the last part of \$SHELL is "nologin" or "false". **Note** that the user may still find a loophole to execute a shell command, it has only been made difficult.
- g -g
Start Vim in GUI mode. See [gui](#) . For the opposite see [-v](#) .
- v -v
Start Ex in Vi mode. Only makes a difference when the executable is called "ex" or "gvim". For gvim the GUI is not started if possible.

- e** -e
Start Vim in Ex mode, see [Ex-mode](#) . Only makes a difference when the executable is not called "ex".
- E** -E
Start Vim in improved Ex mode [gQ](#) . Only makes a difference when the executable is not called "exim".
- s** -s-ex
Silent or batch mode. Only when Vim was started as "ex" or when preceded with the "-e" argument. Otherwise see [-s](#) , which does take an argument while this use of "-s" doesn't. To be used when Vim is used to execute Ex commands from a file instead of a terminal. Switches off most prompts and informative messages. Also warnings and error messages. The output of these commands is displayed (to stdout):

```

:print
:list
:number
:set          to display option values.

```

When **'verbose'** is non-zero messages are printed (for debugging, to stderr).
'term' and \$TERM are not used.
If Vim appears to be stuck try typing "qa!<Enter>". You don't get a prompt thus you can't see Vim is waiting for you to type something.
Initializations are skipped (except the ones given with the "-u" argument).
Example:

```
vim -e -s <thefilter thefile
```

For the opposite, to see errors from the script, execute the file with the [-u](#) flag:

```
vim -u thefilter thefile
```
- b** -b
Binary mode. File I/O will only recognize <NL> to separate lines. The **'expandtab'** option will be reset. The **'textwidth'** option is set to 0. **'modeline'** is reset. The **'binary'** option is set. This is done after reading the vimrc/exrc files but before reading any file in the arglist. See also [edit-binary](#) .
- l** -l
Lisp mode. Sets the **'lisp'** and **'showmatch'** options on.
- A** -A
Arabic mode. Sets the **'arabic'** option on. {only when compiled with the [+arabic](#) features (which include [+rightleft](#)), otherwise Vim gives an error message and exits}
- F** -F
This was used for Farsi mode, which has been removed.

See [farsi.txt](#) .

- H** -H
Hebrew mode. Sets the **'hkmap'** and **'rightleft'** options on.
{only when compiled with the [+rightleft](#) feature, otherwise
Vim gives an error message and exits}
- V[N]** -V verbose
Verbose. Sets the **'verbose'** option to [N] (default: 10).
Messages will be given for each file that is ":source"d and
for reading or writing a viminfo file. Can be used to find
out what is happening upon startup and exit.
Example:
`vim -V8 foobar`
- V[N]{filename}**
Like -V and set **'verbosefile'** to {filename}. The result is
that messages are not displayed but written to the file
{filename}. {filename} must not start with a digit.
Example:
`vim -V20vimlog foobar`
- log {filename}** --log
Start logging and write entries to {filename}.
This works like calling ``ch_logfile({filename}, 'ao')`` very
early during startup.
{only available with the [|+eval|](#) and [|+channel|](#) feature}
- D** -D
Debugging. Go to debugging mode when executing the first
command from a script. [debug-mode](#)
{not available when compiled without the [|+eval|](#) feature}
- C** -C
Compatible mode. Sets the **'compatible'** option. You can use
this to get **'compatible'**, even though a .vimrc file exists.
Keep in mind that the command `":set nocompatible"` in some
plugin or startup script overrules this, so you may end up
with **'nocompatible'** anyway. To find out, use:
`:verbose set compatible?`
Several plugins won't work with **'compatible'** set. You may
want to set it after startup this way:
`vim "+set cp" filename`
Also see [compatible-default](#) .
- N** -N
Not compatible mode. Resets the **'compatible'** option. You can
use this to get **'nocompatible'**, when there is no .vimrc file
or when using `"-u NONE"`.
Also see [compatible-default](#) .
- y** -y easy
Easy mode. Implied for [evim](#) and [eview](#) . Starts with
'insertmode' set and behaves like a click-and-type editor.

This sources the script `$VIMRUNTIME/evim.vim`. Mappings are set up to work like most click-and-type editors, see [evim-keys](#) . The GUI is started when available.

- n** -n
- No swap file will be used. Recovery after a crash will be impossible. Handy if you want to view or edit a file on a very slow medium (e.g., a floppy). Can also be done with `":set updatecount=0"`. You can switch it on again by setting the **'updatecount'** option to some value, e.g., `":set uc=100"`.
NOTE: Don't combine `-n` with `-b`, making `-nb`, because that has a different meaning: `-nb` .
'updatecount' is set to 0 AFTER executing commands from a vimrc file, but before the GUI initializations. Thus it overrides a setting for **'updatecount'** in a vimrc file, but not in a gvimrc file. See [startup](#) .
When you want to reduce accesses to the disk (e.g., for a laptop), don't use `-n`, but set **'updatetime'** and **'updatecount'** to very big numbers, and type `":preserve"` when you want to save your work. This way you keep the possibility for crash recovery.
- o[N]** -o
- Open N windows, split horizontally. If [N] is not given, one window is opened for every file given as argument. If there is not enough room, only the first few files get a window. If there are more windows than arguments, the last few windows will be editing an empty file.
- O[N]** -O
- Open N windows, split vertically. Otherwise it's like `-o`. If both the `-o` and the `-O` option are given, the last one on the command line determines how the windows will be split.
- p[N]** -p
- Open N tab pages. If [N] is not given, one tab page is opened for every file given as argument. The maximum is set with **'tabpagemax'** pages (default 10). If there are more tab pages than arguments, the last few tab pages will be editing an empty file. Also see [tabpage](#) .
- T {terminal}** -T
- Set the terminal type to "terminal". This influences the codes that Vim will send to your terminal. This is normally not needed, because Vim will be able to find out what type of terminal you are using. (See [terminal-info](#) .)
- not-a-term** --not-a-term
- Tells Vim that the user knows that the input and/or output is not connected to a terminal. This will avoid the warning and the two second delay that would happen. Also avoids the "Reading from stdin..." message. Also avoids the "N files to edit" message.

`--gui-dialog-file {name}` `--gui-dialog-file`
 When using the GUI, instead of showing a dialog, write the title and message of the dialog to file `{name}`. The file is created or appended to. Only useful for testing, to avoid that the test gets stuck on a dialog that can't be seen. Without the GUI the argument is ignored.

`--ttyfail` `--ttyfail`
 When the stdin or stdout is not a terminal (tty) then exit right away.

`-d` `-d`
 Start in diff mode, like `vimdiff`.
{not available when compiled without the |+diff| feature}

`-d {device}` `+diff`
 Only on the Amiga and when not compiled with the `+diff` feature. Works like `"-dev"`.

`-dev {device}` `-dev`
 Only on the Amiga: The `{device}` is opened to be used for editing.
 Normally you would use this to set the window position and size: `"-d con:x/y/width/height"`, e.g., `"-d con:30/10/600/150"`. But you can also use it to start editing on another device, e.g., `AUX:.`

`-f` `-f`
 GUI: Do not disconnect from the program that started Vim. 'f' stands for "foreground". If omitted, the GUI forks a new process and exits the current one. `"-f"` should be used when `gvim` is started by a program that will wait for the edit session to finish (e.g., mail or readnews). If you want `gvim` never to fork, include 'f' in `'guioptions'` in your `gvimrc`. Careful: You can use `"-gf"` to start the GUI in the foreground, but `"-fg"` is used to specify the foreground color. gui-fork

Amiga: Do not restart Vim to open a new window. This option should be used when Vim is started by a program that will wait for the edit session to finish (e.g., mail or readnews). See [amiga-window](#).

MS-Windows: This option is not supported. However, when running Vim with an installed `vim.bat` or `gvim.bat` file it works.

`--nofork` `--nofork`
 GUI: Do not fork. Same as `-f`.

`-u {vimrc}` `-u E282`
 The file `{vimrc}` is read for initializations. Most other initializations are skipped; see [initialization](#).

This can be used to start Vim in a special mode, with special mappings and settings. A shell alias can be used to make this easy to use. For example:

```
alias vimc vim -u ~/.c_vimrc !*
```

Also consider using autocommands; see [autocommand](#) .

When `{vimrc}` is equal to "NONE" (all uppercase), all initializations from files and environment variables are skipped, including reading the `gvimrc` file when the GUI starts. Loading plugins is also skipped.

When `{vimrc}` is equal to "NORC" (all uppercase), this has the same effect as "NONE", but loading plugins is not skipped.

When `{vimrc}` is equal to "DEFAULTS" (all uppercase), this has the same effect as "NONE", but the `defaults.vim` script is loaded, which will also set `'nocompatible'`. Also see `--clean` .

Using the "-u" argument with another argument than DEFAULTS has the side effect that the `'compatible'` option will be on by default. This can have unexpected effects. See `'compatible'` .

`-U` E230

`-U {gvimrc}` The file `{gvimrc}` is read for initializations when the GUI starts. Other GUI initializations are skipped. When `{gvimrc}` is equal to "NONE", no file is read for GUI initializations at all. [gui-init](#)
Exception: Reading the system-wide menu file is always done.

`-i`

`-i {viminfo}` The file "viminfo" is used instead of the default viminfo file. If the name "NONE" is used (all uppercase), no viminfo file is read or written, even if `'viminfo'` is set or when `":rv"` or `":wv"` are used. See also [viminfo-file](#) .

`--clean`

`--clean` Similar to `"-u DEFAULTS -U NONE -i NONE"`:
- initializations from files and environment variables is skipped
- `'runtimepath'` and `'packpath'` are set to exclude home directory entries (does not happen with `-u DEFAULTS`).
- the `defaults.vim` script is loaded, which implies `'nocompatible'`: use Vim defaults
- no `gvimrc` script is loaded
- no viminfo file is read or written
Note that a following "-u" argument overrules the effect of `"-u DEFAULTS"`.

`-x`

`-x` Use encryption to read/write files. Will prompt for a key, which is then stored in the `'key'` option. All writes will then use this key to encrypt the text. The `'-x'` argument is not needed when reading a file, because there is a check if the file that is being read has been encrypted, and Vim asks for a key automatically. [encryption](#)

- X
- X Do not try connecting to the X server to get the current window title and copy/paste using the X clipboard. This avoids a long startup time when running Vim in a terminal emulator and the connection to the X server is slow. See `--startuptime` to find out if affects you. Only makes a difference on Unix or VMS, when compiled with the `+X11` feature. Otherwise it's ignored. To disable the connection only for specific terminals, see the `'clipboard'` option. When the X11 Session Management Protocol (XSMP) handler has been built in, the `-X` option also disables that connection as it, too, may have undesirable delays. When the connection is desired later anyway (e.g., for client-server messages), call the `serverlist()` function. This does not enable the XSMP handler though.
- s
- s {scriptin} The script file "scriptin" is read. The characters in the file are interpreted as if you had typed them. The same can be done with the command `:"source! {scriptin}"`. If the end of the file is reached before the editor exits, further characters are read from the keyboard. Only works when not started in Ex mode, see `-s-ex` . See also `complex-repeat` .
- w_nr
- w {number}
-w{number} Set the `'window'` option to {number}.
- w
- w {scriptout} All the characters that you type are recorded in the file "scriptout", until you exit Vim. This is useful if you want to create a script file to be used with "vim -s" or `:"source!"`. When the "scriptout" file already exists, new characters are appended. See also `complex-repeat` . {scriptout} cannot start with a digit. If you want to record what is typed in a human readable for you can use `ch_logfile()` , It adds "raw key input" lines. Also see `--log` .
- W
- W {scriptout} Like -w, but do not append, overwrite an existing file.
- remote [+{cmd}] {file} ...
Open the {file} in another Vim that functions as a server. Any non-file arguments must come before this. See `--remote` .
- remote-silent [+{cmd}] {file} ...
Like --remote, but don't complain if there is no server. See `--remote-silent` .
- remote-wait [+{cmd}] {file} ...

Like `--remote`, but wait for the server to finish editing the file(s).
See `--remote-wait` .

`--remote-wait-silent` [`+{cmd}`] `{file}` ...
Like `--remote-wait`, but don't complain if there is no server.
See `--remote-wait-silent` .

`--servername` `{name}`
Specify the name of the Vim server to send to or to become.
See `--servername` .

`--remote-send` `{keys}`
Send `{keys}` to a Vim server and exit.
See `--remote-send` .

`--remote-expr` `{expr}`
Evaluate `{expr}` in another Vim that functions as a server.
The result is printed on stdout.
See `--remote-expr` .

`--serverlist` Output a list of Vim server names and exit. See `--serverlist` .

`--socketid` `{id}` `--socketid`
GTK+ GUI Vim only. Make gvim try to use GtkPlug mechanism, so that it runs inside another window. See `gui-gtk-socketid` for details.

`--windowid` `{id}` `--windowid`
Win32 GUI Vim only. Make gvim try to use the window `{id}` as a parent, so that it runs inside that window. See `gui-w32-windowid` for details.

`--echo-wid` `--echo-wid`
GTK+ GUI Vim only. Make gvim echo the Window ID on stdout, which can be used to run gvim in a kpart widget. The format of the output is:
`WID: 12345\n`

`--role` `{role}` `--role`
GTK+ 2 GUI only. Set the role of the main window to `{role}`. The window role can be used by a window manager to uniquely identify a window, in order to restore window placement and such. The `--role` argument is passed automatically when restoring the session on login. See `gui-gnome-session`

`-P` `{parent-title}` `-P` MDI E671 E672
Win32 only: Specify the title of the parent application. When possible, Vim will run in an MDI window inside the application.
`{parent-title}` must appear in the window title of the parent application. Make sure that it is specific enough.
Note that the implementation is still primitive. It won't

work with all applications and the menu doesn't work.

-nb

-nb

-nb={fname}

-nb:{hostname}:{addr}:{password}

Attempt connecting to Netbeans and become an editor server for it. The second form specifies a file to read connection info from. The third form specifies the hostname, address and password for connecting to Netbeans. [netbeans-run](#) {only available when compiled with the [+netbeans_intg](#) feature; if not then -nb will make Vim exit}

If the executable is called "view", Vim will start in Readonly mode. This is useful if you can make a hard or symbolic link from "view" to "vim". Starting in Readonly mode can also be done with "vim -R".

If the executable is called "ex", Vim will start in "Ex" mode. This means it will accept only ":" commands. But when the "-v" argument is given, Vim will start in Normal mode anyway.

Additional arguments are available on unix like systems when compiled with X11 GUI support. See [gui-resources](#) .

2. Vim on the Amiga

[starting-amiga](#)

Starting Vim from the Workbench

[workbench](#)

Vim can be started from the Workbench by clicking on its icon twice. It will then start with an empty buffer.

Vim can be started to edit one or more files by using a "Project" icon. The "Default Tool" of the icon must be the full pathname of the Vim executable. The name of the ".info" file must be the same as the name of the text file. By clicking on this icon twice, Vim will be started with the file name as current file name, which will be read into the buffer (if it exists). You can edit multiple files by pressing the shift key while clicking on icons, and clicking twice on the last one. The "Default Tool" for all these icons must be the same.

It is not possible to give arguments to Vim, other than file names, from the workbench.

Vim window

[amiga-window](#)

Vim will run in the CLI window where it was started. If Vim was started with the "run" or "runback" command, or if Vim was started from the workbench, it will open a window of its own.

Technical detail:

To open the new window a little trick is used. As soon as Vim recognizes that it does not run in a normal CLI window, it will

create a script file in "t:". This script file contains the same command as the one Vim was started with, and an "endcli" command. This script file is then executed with a "newcli" command (the "c:run" and "c:newcli" commands are required for this to work). The script file will hang around until reboot, or until you delete it. This method is required to get the ":sh" and "!!" commands to work correctly. But when Vim was started with the -f option (foreground mode), this method is not used. The reason for this is that when a program starts Vim with the -f option it will wait for Vim to exit. With the script trick, the calling program does not know when Vim exits. The -f option can be used when Vim is started by a mail program which also waits for the edit session to finish. As a consequence, the ":sh" and "!!" commands are not available when the -f option is used.

Vim will automatically recognize the window size and react to window resizing. Under Amiga DOS 1.3, it is advised to use the fastfonts program, "FF", to speed up display redrawing.

3. Running eVim

evim-keys

EVim runs Vim as click-and-type editor. This is very unlike the original Vi idea. But it helps for people that don't use Vim often enough to learn the commands. Hopefully they will find out that learning to use Normal mode commands will make their editing much more effective.

In Evim these options are changed from their default value:

:set nocompatible	Use Vim improvements
:set insertmode	Remain in Insert mode most of the time
:set hidden	Keep invisible buffers loaded
:set backup	Keep backup files (not for VMS)
:set backspace=2	Backspace over everything
:set autoindent	auto-indent new lines
:set history=50	keep 50 lines of Ex commands
:set ruler	show the cursor position
:set incsearch	show matches halfway typing a pattern
:set mouse=a	use the mouse in all modes
:set hlsearch	highlight all matches for a search pattern
:set whichwrap+=<, >, [,]	<Left> and <Right> wrap around line breaks
:set guioptions-=a	non-Unix only: don't do auto-select

Key mappings:

<CTRL-Q>	quit, using `:confirm` prompt if there are changes
<Down>	moves by screen lines rather than file lines
<Up>	idem
Q	does "gq", formatting, instead of Ex mode
<BS>	in Visual mode: deletes the selection
CTRL-X	in Visual mode: Cut to clipboard
<S-Del>	idem
CTRL-C	in Visual mode: Copy to clipboard
<C-Insert>	idem
CTRL-V	Pastes from the clipboard (in any mode)

<S-Insert>	idem
CTRL-Z	undo
CTRL-Y	redo
<M-Space>	system menu
CTRL-A	select all
<C-Tab>	next window, CTRL-W w
<C-F4>	close window, CTRL-W c

Additionally:

- ":behave mswin" is used `:behave`
- syntax highlighting is enabled
- filetype detection is enabled, filetype plugins and indenting is enabled
- in a text file `'textwidth'` is set to 78

One hint: If you want to go to Normal mode to be able to type a sequence of commands, use **CTRL-L**. `i_CTRL-L`

There is no way to stop "easy mode", you need to exit Vim.

4. Initialization initialization startup

This section is about the non-GUI version of Vim. See [gui-fork](#) for additional initialization when starting the GUI.

At startup, Vim checks environment variables and files and sets values accordingly. Vim proceeds in this order:

1. Set the `'shell'` and `'term'` option SHELL COMSPEC TERM
 The environment variable SHELL, if it exists, is used to set the `'shell'` option. On Win32, the COMSPEC variable is used if SHELL is not set.
 The environment variable TERM, if it exists, is used to set the `'term'` option. However, `'term'` will change later when starting the GUI (step 8 below).
2. Process the arguments
 The options and file names from the command that start Vim are inspected.
 The `-V` argument can be used to display or log what happens next, useful for debugging the initializations.
 The `--cmd` arguments are executed.
 Buffers are created for all files (but not loaded yet).
3. Execute Ex commands, from environment variables and/or files vimrc excrc
 An environment variable is read as one Ex command line, where multiple commands must be separated with `'|'` or `"<NL>"`.
 A file that contains initialization commands is called a "vimrc" file. Each line in a vimrc file is executed as an Ex command line. It is sometimes also referred to as "exrc" file. They are the same type of file, but "exrc" is what Vi always used, "vimrc" is a Vim specific name. Also see [vimrc-intro](#).

Places for your personal initializations:

Unix	\$HOME/.vimrc or \$HOME/.vim/vimrc
MS-Windows	\$HOME/_vimrc, \$HOME/vimfiles/vimrc or \$VIM/_vimrc
Amiga	s:.vimrc, home:.vimrc, home:vimfiles:vimrc or \$VIM/.vimrc
Haiku	\$HOME/config/settings/vim/vimrc

The files are searched in the order specified above and only the first one that is found is read.

RECOMMENDATION: Put all your Vim configuration stuff in the \$HOME/.vim/ directory (\$HOME/vimfiles/ for MS-Windows). That makes it easy to copy it to another system.

If Vim was started with "-u filename", the file "filename" is used. All following initializations until 4. are skipped. \$MYVIMRC is not set.

"vim -u NORC" can be used to skip these initializations without reading a file. "vim -u NONE" also skips loading plugins. -u

If Vim was started in Ex mode with the "-s" argument, all following initializations until 4. are skipped. Only the "-u" option is interpreted.

evim.vim

- a. If vim was started as `evim` or `eview` or with the `-y` argument, the script \$VIMRUNTIME/evim.vim will be loaded.

system-vimrc

- b. For Unix, MS-Windows, VMS, Macintosh and Amiga the system vimrc file is read for initializations. The path of this file is shown with the ":version" command. Mostly it's "\$VIM/vimrc". Note that this file is ALWAYS read in 'compatible' mode, since the automatic resetting of 'compatible' is only done later. Add a ":set nocp" command if you like. For the Macintosh the \$VIMRUNTIME/macmap.vim is read.

VIMINIT .vimrc _vimrc EXINIT .exrc _exrc \$MYVIMRC

- c. Five places are searched for initializations. The first that exists is used, the others are ignored. The \$MYVIMRC environment variable is set to the file that was first found, unless \$MYVIMRC was already set and when using VIMINIT.

I The environment variable VIMINIT (see also `compatible-default`) (*)
The value of \$VIMINIT is used as an Ex command line.

II The user vimrc file(s):

"\$HOME/.vimrc"	(for Unix) (*)
"\$HOME/.vim/vimrc"	(for Unix) (*)
"s:.vimrc"	(for Amiga) (*)
"home:.vimrc"	(for Amiga) (*)
"home:vimfiles:vimrc"	(for Amiga) (*)
"\$VIM/.vimrc"	(for Amiga) (*)
"\$HOME/_vimrc"	(for Win32) (*)
"\$HOME/vimfiles/vimrc"	(for Win32) (*)
"\$VIM/_vimrc"	(for Win32) (*)
"\$HOME/config/settings/vim/vimrc"	(for Haiku) (*)

Note: For Unix and Amiga, when ".vimrc" does not exist, "_vimrc" is also tried, in case an MS-DOS compatible file system is used. For MS-Windows ".vimrc" is checked after "_vimrc", in case long file names are used.

Note: For Win32, "\$HOME" is checked first. If no "_vimrc" or ".vimrc" is found there, "\$VIM" is tried. See `$VIM` for when \$VIM is not set.

III The environment variable EXINIT.

The value of \$EXINIT is used as an Ex command line.

IV The user exrc file(s). Same as for the user vimrc file, but with "vimrc" replaced by "exrc". But only one of ".exrc" and "_exrc" is used, depending on the system. And without the (*)!

V The default vimrc file, \$VIMRUNTIME/defaults.vim. This sets up options values and has "syntax on" and "filetype on" commands, which is what most new users will want. See `defaults.vim`.

d. If the '**exrc**' option is on (which is NOT the default), the current directory is searched for three files. The first that exists is used, the others are ignored.

- The file ".vimrc" (for Unix, Amiga) (*)
 "_vimrc" (for Win32) (*)
- The file "_vimrc" (for Unix, Amiga) (*)
 ".vimrc" (for Win32) (*)
- The file ".exrc" (for Unix, Amiga)
 "_exrc" (for Win32)

(*) Using this file or environment variable will cause '**compatible**' to be off by default. See `compatible-default`.

Note: When using the `mzscheme` interface, it is initialized after loading the vimrc file. Changing '**mzschemedll**' later has no effect.

4. Load the plugin scripts.

`load-plugins`

This does the same as the command:

```
:runtime! plugin/**/*.vim
```

The result is that all directories in the '**runtimepath**' option will be searched for the "plugin" sub-directory and all files ending in ".vim" will be sourced (in alphabetical order per directory), also in subdirectories.

However, directories in '**runtimepath**' ending in "after" are skipped here and only loaded after packages, see below.

Loading plugins won't be done when:

- The '**loadplugins**' option was reset in a vimrc file.
- The `--noplugin` command line argument is used.
- The `--clean` command line argument is used.
- The "-u NONE" command line argument is used `-u .`
- When Vim was compiled without the `+eval` feature.

Note that using "-c 'set noloadplugins'" doesn't work, because the commands from the command line have not been executed yet. You can use "--cmd 'set noloadplugins'" or "--cmd 'set loadplugins'" `--cmd`.

Packages are loaded. These are plugins, as above, but found in the "start" directory of each entry in '**packpath**'. Every plugin directory found is added in '**runtimepath**' and then the plugins are sourced. See

[packages](#) .

The plugins scripts are loaded, as above, but now only the directories ending in "after" are used. [Note](#) that `'runtimepath'` will have changed if packages have been found, but that should not add a directory ending in "after".

5. Set `'shellpipe'` and `'shellredir'`
The `'shellpipe'` and `'shellredir'` options are set according to the value of the `'shell'` option, unless they have been set before. This means that Vim will figure out the values of `'shellpipe'` and `'shellredir'` for you, unless you have set them yourself.
6. Set `'updatecount'` to zero, if "-n" command argument used
7. Set binary options
If the "-b" flag was given to Vim, the options for binary editing will be set now. See [-b](#) .
8. Perform GUI initializations
Only when starting "gvim", the GUI initializations will be done. See [gui-init](#) .
9. Read the viminfo file
If the `'viminfo'` option is not empty, the viminfo file is read. See [viminfo-file](#) .
10. Read the quickfix file
If the "-q" flag was given to Vim, the quickfix file is read. If this fails, Vim exits.
11. Open all windows
When the `-o` flag was given, windows will be opened (but not displayed yet).
When the `-p` flag was given, tab pages will be created (but not displayed yet).
When switching screens, it happens now. Redrawing starts.
If the "-q" flag was given to Vim, the first error is jumped to.
Buffers for all windows will be loaded, without triggering `BufAdd` autocommands.
12. Execute startup commands
If a "-t" flag was given to Vim, the tag is jumped to.
The commands given with the `-c` and `+cmd` arguments are executed.
If the `'insertmode'` option is set, Insert mode is entered.
The starting flag is reset, `has("vim_starting")` will now return zero.
The `v:vim_did_enter` variable is set to 1.
The `VimEnter` autocommands are executed.

The `$MYVIMRC` or `$MYGVIMRC` file will be set to the first found vimrc and/or gvimrc file.

[Some hints on using initializations](#)

Standard setup:

Create a vimrc file to set the default settings and mappings for all your edit sessions. Put it in a place so that it will be found by 3b:

```
~/vimrc          (Unix)
s:vimrc          (Amiga)
$VIM\_vimrc      (Win32)
~/config/settings/vim/vimrc (Haiku)
```

Note that creating a vimrc file will cause the **'compatible'** option to be off by default. See [compatible-default](#) .

Local setup:

Put all commands that you need for editing a specific directory only into a vimrc file and place it in that directory under the name ".vimrc" ("_vimrc" for Win32). **NOTE:** To make Vim look for these special files you have to turn on the option **'exrc'**. See [trojan-horse](#) too.

System setup:

This only applies if you are managing a Unix system with several users and want to set the defaults for all users. Create a vimrc file with commands for default settings and mappings and put it in the place that is given with the ":version" command.

Saving the current state of Vim to a file

Whenever you have changed values of options or when you have created a mapping, then you may want to save them in a vimrc file for later use. See [save-settings](#) about saving the current state of settings to a file.

Avoiding setup problems for Vi users

Vi uses the variable EXINIT and the file "~/exrc". So if you do not want to interfere with Vi, then use the variable VIMINIT and the file "vimrc" instead.

Amiga environment variables

On the Amiga, two types of environment variables exist. The ones set with the DOS 1.3 (or later) setenv command are recognized. See the AmigaDos 1.3 manual. The environment variables set with the old Manx Set command (before version 5.0) are not recognized.

MS-Windows line separators

On MS-Windows, Vim assumes that all the vimrc files have <CR><NL> pairs as line separators. This will give problems if you have a file with only <NL>s and have a line like ":map xx yy^M". The trailing ^M will be ignored.

Vi compatible default value

compatible-default

When Vim starts, the **'compatible'** option is on. This will be used when Vim starts its initializations. But as soon as:

- a user vimrc file is found, or
- a vimrc file in the current directory is found, or
- the "VIMINIT" environment variable is set, or
- the "-N" command line argument is given, or
- the "--clean" command line argument is given, or
- the `defaults.vim` script is loaded, or
- a gvimrc file was found,

then the option will be set to **'nocompatible'**.

Note that this does NOT happen when a system-wide vimrc file was found.

This has the side effect of setting or resetting other options (see **'compatible'**). But only the options that have not been set or reset will be changed. This has the same effect like the value of **'compatible'** had this value when starting Vim.

'compatible' is NOT reset, and `defaults.vim` is not loaded:

- when Vim was started with the `-u` command line argument, especially with `"-u NONE"`, or
- when started with the `-C` command line argument, or
- when the name of the executable ends in "ex". (This has been done to make Vim behave like "ex", when it is started as "ex")

But there is a side effect of setting or resetting **'compatible'** at the moment a `.vimrc` file is found: Mappings are interpreted the moment they are encountered. This makes a difference when using things like "`<CR>`". If the mappings depend on a certain value of **'compatible'**, set or reset it before giving the mapping.

Defaults without a .vimrc file

defaults.vim E1187

If Vim is started normally and no user vimrc file is found, the `$VIMRUNTIME/defaults.vim` script is loaded. This will set **'compatible'** off, switch on syntax highlighting and a few more things. See the script for details. **NOTE:** this is done since Vim 8.0, not in Vim 7.4. (it was added in patch 7.4.2111 to be exact).

This should work well for new Vim users. If you create your own `.vimrc`, it is recommended to add these lines somewhere near the top:

```
unlet! skip_defaults_vim
source $VIMRUNTIME/defaults.vim
```

Then Vim works like before you had a `.vimrc`. Copying `$VIMRUNTIME/vimrc_example` is way to do this. Alternatively, you can copy `defaults.vim` to your `.vimrc` and modify it (but then you won't get updates when it changes).

If you don't like some of the defaults, you can still source `defaults.vim` and revert individual settings. See the `defaults.vim` file for hints on how to revert each item.

skip_defaults_vim

If you use a system-wide vimrc and don't want `defaults.vim` to change settings,

set the "skip_defaults_vim" variable. If this was set and you want to load defaults.vim from your .vimrc, first unlet skip_defaults_vim, as in the example above.

Avoiding trojan horses

trojan-horse

While reading the "vimrc" or the "exrc" file in the current directory, some commands can be disabled for security reasons by setting the **'secure'** option. This is always done when executing the command from a tags file. Otherwise it would be possible that you accidentally use a vimrc or tags file that somebody else created and contains nasty commands. The disabled commands are the ones that start a shell, the ones that write to a file, and ":autocmd". The ":map" commands are echoed, so you can see which keys are being mapped.

If you want Vim to execute all commands in a local vimrc file, you can reset the **'secure'** option in the EXINIT or VIMINIT environment variable or in the global "exrc" or "vimrc" file. This is not possible in "vimrc" or "exrc" in the current directory, for obvious reasons.

On Unix systems, this only happens if you are not the owner of the vimrc file. Warning: If you unpack an archive that contains a vimrc or exrc file, it will be owned by you. You won't have the security protection. Check the vimrc file before you start Vim in that directory, or reset the **'exrc'** option. Some Unix systems allow a user to do "chown" on a file. This makes it possible for another user to create a nasty vimrc and make you the owner. Be careful!

When using tag search commands, executing the search command (the last part of the line in the tags file) is always done in secure mode. This works just like executing a command from a vimrc/exrc in the current directory.

If Vim startup is slow

slow-start

If Vim takes a long time to start up, use the `--startuptime` argument to find out what happens. There are a few common causes:

- If the Unix version was compiled with the GUI and/or X11 (check the output of ":version" for "+GUI" and "+X11"), it may need to load shared libraries and connect to the X11 server. Try compiling a version with GUI and X11 disabled. This also should make the executable smaller. Use the `-X` command line argument to avoid connecting to the X server when running in a terminal.
- If you have "viminfo" enabled, the loading of the viminfo file may take a while. You can find out if this is the problem by disabling viminfo for a moment (use the Vim argument `--i NONE`, `-i`). Try reducing the number of lines stored in a register with `:set viminfo='20,<50,s10'`. [viminfo-file](#) .

Intro message

:intro

When Vim starts without a file name, an introductory message is displayed (for those who don't know what Vim is). It is removed as soon as the display is redrawn in any way. To see the message again, use the ":intro" command (if there is not enough room, you will see only part of it).

To avoid the intro message on startup, add the 'I' flag to **'shortmess'**.

info-message

The `--help` and `--version` arguments cause Vim to print a message and then exit. Normally the message is sent to stdout, thus can be redirected to a file with:

```
vim --help >file
```

From inside Vim:

```
:read !vim --help
```

When using `gvim`, it detects that it might have been started from the desktop, without a terminal to show messages on. This is detected when both stdout and stderr are not a tty. This breaks the `:read` command, as used in the example above. To make it work again, set `'shellredir'` to `">"` instead of the default `">&":`

```
:set shellredir=>
:read !gvim --help
```

This still won't work for systems where `gvim` does not use stdout at all though.

5. \$VIM and \$VIMRUNTIME

\$VIM

The environment variable `"$VIM"` is used to locate various user files for Vim, such as the user startup script `".vimrc"`. This depends on the system, see [startup](#) .

To avoid the need for every user to set the `$VIM` environment variable, Vim will try to get the value for `$VIM` in this order:

1. The value defined by the `$VIM` environment variable. You can use this to make Vim look in a specific directory for its support files. Example:

```
setenv VIM /home/paul/vim
```
2. The path from `'helpfile'` is used, unless it contains some environment variable too (the default is `"$VIMRUNTIME/doc/help.txt"`: chicken-egg problem). The file name (`"help.txt"` or any other) is removed. Then trailing directory names are removed, in this order: `"doc"`, `"runtime"` and `"vim{version}"` (e.g., `"vim82"`).
3. For Win32 Vim tries to use the directory name of the executable. If it ends in `"/src"`, this is removed. This is useful if you unpacked the `.zip` file in some directory, and adjusted the search path to find the vim executable. Trailing directory names are removed, in this order: `"runtime"` and `"vim{version}"` (e.g., `"vim82"`).
4. For Unix the compile-time defined installation directory is used (see the output of `:version`).

Once Vim has done this once, it will set the `$VIM` environment variable. To change it later, use a `:let` command like this:

```
:let $VIM = "/home/paul/vim/"
```

\$VIMRUNTIME

The environment variable `"$VIMRUNTIME"` is used to locate various support

files, such as the on-line documentation and files used for syntax highlighting. For example, the main help file is normally "\$VIMRUNTIME/doc/help.txt".

You don't normally set \$VIMRUNTIME yourself, but let Vim figure it out. This is the order used to find the value of \$VIMRUNTIME:

1. If the environment variable \$VIMRUNTIME is set, it is used. You can use this when the runtime files are in an unusual location.
2. If "\$VIM/vim{version}" exists, it is used. {version} is the version number of Vim, without any '-' or '.'. For example: "\$VIM/vim82". This is the normal value for \$VIMRUNTIME.
3. If "\$VIM/runtime" exists, it is used.
4. The value of \$VIM is used. This is for backwards compatibility with older versions.
5. When the 'helpfile' option is set and doesn't contain a '\$', its value is used, with "doc/help.txt" removed from the end.

For Unix, when there is a compiled-in default for \$VIMRUNTIME (check the output of ":version"), steps 2, 3 and 4 are skipped, and the compiled-in default is used after step 5. This means that the compiled-in default overrules the value of \$VIM. This is useful if \$VIM is "/etc" and the runtime files are in "/usr/share/vim/vim82".

Once Vim has done this once, it will set the \$VIMRUNTIME environment variable. To change it later, use a ":let" command like this:

```
:let $VIMRUNTIME = "/home/piet/vim/vim82"
```

In case you need the value of \$VIMRUNTIME in a shell (e.g., for a script that greps in the help files) you might be able to use this:

```
VIMRUNTIME=`vim -e -T dumb --cmd 'exe "set t_cm=<C-M>"|echo $VIMRUNTIME|quit' | tr
```

Don't set \$VIMRUNTIME to an empty value, some things may stop working.

=====

6. Suspending

suspend

iconize iconise CTRL-Z v_CTRL-Z

CTRL-Z

Suspend Vim, like ":stop".

Works in Normal and in Visual mode. In Insert and Command-line mode, the **CTRL-Z** is inserted as a normal character. In Visual mode Vim goes back to Normal mode.

Note: if **CTRL-Z** undoes a change see [mswin.vim](#) .

```
:sus[pend][!] or
```

```
:sus :suspend :st :stop
```

```
:st[op][!]
```

Suspend Vim.

If the '!' is not given and 'autowrite' is set, every buffer with changes and a file name is written out.

If the '!' is given or 'autowrite' is not set, changed buffers are not written, don't forget to bring Vim back to the foreground later!

In the GUI, suspending is implemented as iconising gvim. In MS-Windows, gvim

is minimized.

On many Unix systems, it is possible to suspend Vim with **CTRL-Z**. This is only possible in Normal and Visual mode (see next chapter, [vim-modes](#)). Vim will continue if you make it the foreground job again. On other systems, **CTRL-Z** will start a new shell. This is the same as the `:"sh"` command. Vim will continue if you exit from the shell.

In X-windows the selection is disowned when Vim suspends. this means you can't paste it in another application (since Vim is going to sleep an attempt to get the selection would make the program hang).

7. Exiting

exiting

There are several ways to exit Vim:

- Close the last window with `:"quit"`. Only when there are no changes.
- Close the last window with `:"quit!"`. Also when there are changes.
- Close all windows with `:"qall"`. Only when there are no changes.
- Close all windows with `:"qall!"`. Also when there are changes.
- Use `:"cquit"`. Also when there are changes.

When using `:"cquit"` or when there was an error message Vim exits with exit code 1. Errors can be avoided by using `:"silent!"` or with `:"catch"`.

8. Saving settings

save-settings

Mostly you will edit your vimrc files manually. This gives you the greatest flexibility. There are a few commands to generate a vimrc file automatically. You can use these files as they are, or copy/paste lines to include in another vimrc file.

`:"mk[exrc] [file]` `:"mk` `:"mkexrc`
Write current key mappings and changed options to `[file]` (default `".exrc"` in the current directory), unless it already exists.

`:"mk[exrc]! [file]` Always write current key mappings and changed options to `[file]` (default `".exrc"` in the current directory).

`:"mkv[imrc][!] [file]` `:"mkv` `:"mkvi` `:"mkvimrc`
Like `:"mkexrc"`, but the default is `".vimrc"` in the current directory. The `:"version"` command is also written to the file.

These commands will write `:"map"` and `:"set"` commands to a file, in such a way that when these commands are executed, the current key mappings and options will be set to the same values. The options `'columns'`, `'endofline'`, `'fileformat'`, `'key'`, `'lines'`, `'modified'`, `'scroll'`, `'term'`, `'textmode'`, `'ttyfast'` and `'ttymouse'` are not included, because these are terminal or file dependent. Note that the options `'binary'`, `'paste'` and `'readonly'` are included, this might not always be what you want.

When special keys are used in mappings, The **'cptions'** option will be temporarily set to its Vim default, to avoid the mappings to be misinterpreted. This makes the file incompatible with Vi, but makes sure it can be used with different terminals.

Only global mappings are stored, not mappings local to a buffer.

A common method is to use a default ".vimrc" file, make some modifications with ":map" and ":set" commands and write the modified file. First read the default ".vimrc" in with a command like ":source ~piet/.vimrc.Cprogs", change the settings and then save them in the current directory with ":mkvimrc!". If you want to make this file your default .vimrc, move it to your home directory (on Unix), s: (Amiga) or \$VIM directory (MS-Windows). You could also use autocommands **autocommand** and/or modelines **modeline** .

vimrc-option-example

If you only want to add a single option setting to your vimrc, you can use these steps:

1. Edit your vimrc file with Vim.
2. Play with the option until it's right. E.g., try out different values for **'guifont'**.
3. Append a line to set the value of the option, using the expression register '=' to enter the value. E.g., for the **'guifont'** option:

```
o:set guifont=<C-R>=&guifont<CR><Esc>
```

[<C-R> is a **CTRL-R**, <CR> is a return, <Esc> is the escape key]

You need to escape special characters, esp. spaces.

Note that when you create a .vimrc file, this can influence the **'compatible'** option, which has several side effects. See **'compatible'** .

":mkvimrc", ":mkexrc" and ":mksession" write the command to set or reset the **'compatible'** option to the output file first, because of these side effects.

9. Views and Sessions

views-sessions

This is introduced in sections [21.4](#) and [21.5](#) of the user manual.

View view-file

A View is a collection of settings that apply to one window. You can save a View and when you restore it later, the text is displayed in the same way. The options and mappings in this window will also be restored, so that you can continue editing like when the View was saved.

Session session-file

A Session keeps the Views for all windows, plus the global settings. You can save a Session and when you restore it later the window layout looks the same. You can use a Session to quickly switch between different projects, automatically loading the files you were last working on in that project.

Views and Sessions are a nice addition to viminfo-files, which are used to remember information for all Views and Sessions together **viminfo-file** .

You can quickly start editing with a previously saved View or Session with the

-S argument:
vim -S Session.vim

All this is {not available when compiled without the |+mksession| feature}.

```
                                :mks    :mksession
:mks[ession][!] [file] Write a Vim script that restores the current editing
                        session.
                        When [!] is included an existing file is overwritten.
                        When [file] is omitted "Session.vim" is used.
```

The output of ":mksession" is like ":mkvimrc", but additional commands are added to the file. Which ones depends on the 'sessionoptions' option. The resulting file, when executed with a ":source" command:

1. Restores global mappings and options, if 'sessionoptions' contains "options". Script-local mappings will not be written.
2. Restores global variables that start with an uppercase letter and contain at least one lowercase letter, if 'sessionoptions' contains "globals".
3. Closes all windows in the current tab page, except the current one; closes all tab pages except the current one (this results in currently loaded buffers to be unloaded, some may become hidden if 'hidden' is set or otherwise specified); wipes out the current buffer, if it is empty and unnamed.
4. Restores the current directory if 'sessionoptions' contains "curdir", or sets the current directory to where the Session file is if 'sessionoptions' contains "sesdir".
5. Restores GUI Vim window position, if 'sessionoptions' contains "winpos".
6. Restores screen size, if 'sessionoptions' contains "resize".
7. Reloads the buffer list, with the last cursor positions. If 'sessionoptions' contains "buffers" then all buffers are restored, including hidden and unloaded buffers. Otherwise only buffers in windows are restored.
8. Restores all windows with the same layout. If 'sessionoptions' contains "help", help windows are restored. If 'sessionoptions' contains "blank", windows editing a buffer without a name will be restored. If 'sessionoptions' contains "winsize" and no (help/blank) windows were left out, the window sizes are restored (relative to the screen size). Otherwise, the windows are just given sensible sizes.
9. Restores the Views for all the windows, as with :mkview . But 'sessionoptions' is used instead of 'viewoptions'.
10. If a file exists with the same name as the Session file, but ending in "x.vim" (for eXtra), executes that as well. You can use *x.vim files to specify additional settings and actions associated with a given Session, such as creating menu items in the GUI version.

After restoring the Session, the full filename of your current Session is available in the internal variable "v:this_session" [this_session-variable](#) . An example mapping:

```
:nmap <F2> :wa<Bar>exe "mksession! " .. v:this_session<CR>:so ~/sessions/
```

This saves the current Session, and starts off the command to load another.

A session includes all tab pages, unless "tabpages" was removed from 'sessionoptions'. [tab-page](#)

The `SessionLoadPost` autocmd event is triggered after a session file is loaded/sourced.

SessionLoad-variable

While the session file is loading the `SessionLoad` global variable is set to 1. Plugins can use this to postpone some work until the `SessionLoadPost` event is triggered.

```
:mkvie[w][!] [file]          :mkvie  :mkview
```

Write a Vim script that restores the contents of the current window. When `[!]` is included an existing file is overwritten. When `[file]` is omitted or is a number from 1 to 9, a name is generated and `'viewdir'` prepended. When the last path part of `'viewdir'` does not exist, this directory is created. E.g., when `'viewdir'` is `"$VIM/vimfiles/view"` then `"view"` is created in `"$VIM/vimfiles"`. An existing file is always overwritten then. Use `:loadview` to load this view again. When `[file]` is the name of a file (`'viewdir'` is not used), a command to edit the file is added to the generated file.

The output of `":mkview"` contains these items:

1. The argument list used in the window. When the global argument list is used it is reset to the global list. The index in the argument list is also restored.
2. The file being edited in the window. If there is no file, the window is made empty.
3. Restore mappings, abbreviations and options local to the window if `'viewoptions'` contains `"options"` or `"localoptions"`. For the options it restores only values that are local to the current buffer and values local to the window. When storing the view as part of a session and `"options"` is in `'sessionoptions'`, global values for local options will be stored too.
4. Restore folds when using manual folding and `'viewoptions'` contains `"folds"`. Restore manually opened and closed folds.
5. The scroll position and the cursor position in the file. Doesn't work very well when there are closed folds.
6. The local current directory, if it is different from the global current directory and `'viewoptions'` contains `"curdir"`.

Note that Views and Sessions are not perfect:

- They don't restore everything. For example, defined functions, autocommands and `":syntax on"` are not included. Things like register contents and command line history are in `viminfo`, not in Sessions or Views.
- Global option values are only set when they differ from the default value. When the current value is not the default value, loading a Session will not set it back to the default value. Local options will be set back to the default value though.
- Existing mappings will be overwritten without warning. An existing mapping may cause an error for ambiguity.
- When storing manual folds and when storing manually opened/closed folds, changes in the file between saving and loading the view will mess it up.

- The Vim script is not very efficient. But still faster than typing the commands yourself!

```

:lo[adview] [nr]          :lo  :loadview
                          Load the view for the current file. When [nr] is
                          omitted, the view stored with ":mkview" is loaded.
                          When [nr] is specified, the view stored with ":mkview
                          [nr]" is loaded.

```

The combination of ":mkview" and ":loadview" can be used to store up to ten different views of a file. These are remembered in the directory specified with the '**viewdir**' option. The views are stored using the file name. If a file is renamed or accessed through a (symbolic) link the view will not be found.

You might want to clean up your '**viewdir**' directory now and then.

To automatically save and restore views for *.c files:

```

au BufWinLeave *.c mkview
au BufWinEnter *.c silent loadview

```

10. The viminfo file

```

viminfo viminfo-file E136
E575    E576    E577

```

If you exit Vim and later start it again, you would normally lose a lot of information. The viminfo file can be used to remember that information, which enables you to continue where you left off.

This is introduced in section [21.3](#) of the user manual.

The viminfo file is used to store:

- The command line history.
- The search string history.
- The input-line history.
- Contents of non-empty registers.
- Marks for several files.
- File marks, pointing to locations in files.
- Last search/substitute pattern (for 'n' and '&').
- The buffer list.
- Global variables.

The viminfo file is not supported when the `+viminfo` feature has been disabled at compile time.

You could also use a Session file. The difference is that the viminfo file does not depend on what you are working on. There normally is only one viminfo file. Session files are used to save the state of a specific editing Session. You could have several Session files, one for each project you are working on. Viminfo and Session files together can be used to effectively enter Vim and directly start working in your desired setup. [session-file](#)

```
viminfo-read
```

When Vim is started and the '**viminfo**' option is non-empty, the contents of the viminfo file are read and the info can be used in the appropriate places.

The `v:oldfiles` variable is filled. The marks are not read in at startup (but file marks are). See `initialization` for how to set the `'viminfo'` option upon startup.

`viminfo-write`

When Vim exits and `'viminfo'` is non-empty, the info is stored in the viminfo file (it's actually merged with the existing one, if one exists). The `'viminfo'` option is a string containing information about what info should be stored, and contains limits on how much should be stored (see `'viminfo'`).

Merging happens in two ways. Most items that have been changed or set in the current Vim session are stored, and what was not changed is filled from what is currently in the viminfo file. For example:

- Vim session A reads the viminfo, which contains variable START.
- Vim session B does the same
- Vim session A sets the variables AAA and BOTH and exits
- Vim session B sets the variables BBB and BOTH and exits

Now the viminfo will have:

```
START - it was in the viminfo and wasn't changed in session A or B
AAA   - value from session A, session B kept it
BBB   - value from session B
BOTH  - value from session B, value from session A is lost
```

`viminfo-timestamp`

For some items a timestamp is used to keep the last changed version. Here it doesn't matter in which sequence Vim sessions exit, the newest item(s) are always kept. This is used for:

- The command line history.
- The search string history.
- The input-line history.
- Contents of non-empty registers.
- The jump list
- File marks

The timestamp feature was added before Vim 8.0. Older versions of Vim, starting with 7.4.1131, will keep the items with timestamp, but not use them. Thus when using both an older and a newer version of Vim the most recent data will be kept.

Notes for Unix:

- The file protection for the viminfo file will be set to prevent other users from being able to read it, because it may contain any text or commands that you have worked with.
- If you want to share the viminfo file with other users (e.g. when you "su" to another user), you can make the file writable for the group or everybody. Vim will preserve this when replacing the viminfo file. Be careful, don't allow just anybody to read and write your viminfo file!
- Vim will not overwrite a viminfo file that is not writable by the current "real" user. This helps for when you did "su" to become root, but your \$HOME is still set to a normal user's home directory. Otherwise Vim would create a viminfo file owned by root that nobody else can read.
- The viminfo file cannot be a symbolic link. This is to avoid security issues.

Marks are stored for each file separately. When a file is read and `'viminfo'`

is non-empty, the marks for that file are read from the viminfo file. **NOTE:** The marks are only written when exiting Vim, which is fine because marks are remembered for all the files you have opened in the current editing session, unless `":bdel"` is used. If you want to save the marks for a file that you are about to abandon with `":bdel"`, use `":wv"`. The '[' and ']' marks are not stored, but the '' mark is. The '' mark is very useful for jumping to the cursor position when the file was last exited. No marks are saved for files that start with any string given with the "r" flag in **'viminfo'**. This can be used to avoid saving marks for files on removable media (for MS-Windows you would use `"ra:,rb:"`, for Amiga `"rdf0:,rdf1:,rdf2:"`). The `v:oldfiles` variable is filled with the file names that the viminfo file has marks for.

viminfo-file-marks

Uppercase marks ('A to 'Z) are stored when writing the viminfo file. The numbered marks ('0 to '9) are a bit special. When the viminfo file is written (when exiting or with the `":wviminfo"` command), '0 is set to the current cursor position and file. The old '0 is moved to '1, '1 to '2, etc. This resembles what happens with the "1 to "9 delete registers. If the current cursor position is already present in '0 to '9, it is moved to '0, to avoid having the same position twice. The result is that with "'0", you can jump back to the file and line where you exited Vim. To do that right away, try using this command:

```
vim -c "normal '0"
```

In a csh compatible shell you could make an alias for it:

```
alias lvim vim -c "'normal "'0'"'
```

For a bash-like shell:

```
alias lvim='vim -c "normal \'0'"'
```

Use the "r" flag in **'viminfo'** to specify for which files no marks should be remembered.

VIMINFO FILE NAME

viminfo-file-name

- The default name of the viminfo file is `"$HOME/.viminfo"` for Unix, `"s:.viminfo"` for Amiga, `"$HOME_viminfo"` for Win32. For Win32, when `$HOME` is not set, `"$VIM_viminfo"` is used. When `$VIM` is also not set, `"c:_viminfo"` is used.
- The 'n' flag in the **'viminfo'** option can be used to specify another viminfo file name **'viminfo'**.
- The "-i" Vim argument can be used to set another file name, `-i`. When the file name given is "NONE" (all uppercase), no viminfo file is ever read or written. Also not for the commands below!
- The **'viminfofile'** option can be used like the "-i" argument. In fact, the value from the "-i" argument is stored in the **'viminfofile'** option.
- For the commands below, another file name can be given, overriding the default and the name given with **'viminfo'** or "-i" (unless it's NONE).

CHARACTER ENCODING

[viminfo-encoding](#)

The text in the viminfo file is encoded as specified with the **'encoding'** option. Normally you will always work with the same **'encoding'** value, and this works just fine. However, if you read the viminfo file with another value for **'encoding'** than what it was written with, some of the text (non-ASCII characters) may be invalid. If this is unacceptable, add the **'c'** flag to the **'viminfo'** option:

```
:set viminfo+=c
```

Vim will then attempt to convert the text in the viminfo file from the **'encoding'** value it was written with to the current **'encoding'** value. This requires Vim to be compiled with the **+iconv** feature. Filenames are not converted.

MANUALLY READING AND WRITING

[viminfo-read-write](#)

Two commands can be used to read and write the viminfo file manually. This can be used to exchange registers between two running Vim programs: First type **":wv"** in one and then **":rv"** in the other. **Note** that if the register already contained something, then **":rv!"** would be required. Also **note** however that this means everything will be overwritten with information from the first Vim, including the command line history, etc.

The viminfo file itself can be edited by hand too, although we suggest you start with an existing one to get the format right. It is reasonably self-explanatory once you're in there. This can be useful in order to create a second file, say **"~/my_viminfo"** which could contain certain settings that you always want when you first start Vim. For example, you can preload registers with particular data, or put certain commands in the command line history. A line in your **.vimrc** file like

```
:rviminfo! ~/.my_viminfo
```

can be used to load this information. You could even have different viminfos for different types of files (e.g., C code) and load them based on the file name, using the **":autocmd"** command (see [:autocmd](#)).

[viminfo-errors](#)

When Vim detects an error while reading a viminfo file, it will not overwrite that file. If there are more than 10 errors, Vim stops reading the viminfo file. This was done to avoid accidentally destroying a file when the file name of the viminfo file is wrong. This could happen when accidentally typing **"vim -i file"** when you wanted **"vim -R file"** (yes, somebody accidentally did that!). If you want to overwrite a viminfo file with an error in it, you will either have to fix the error, or delete the file (while Vim is running, so most of the information will be restored).

```
:rv[iminfo][!] [file] :rv :rviminfo E195  
Read from viminfo file [file] (default: see  
viminfo-file-name above).  
If [!] is given, then any information that is  
already set (registers, marks, v:oldfiles , etc.)  
will be overwritten
```

`:wv[iminfo][!] [file]` `:wv` `:wviminfo` E137 E138 E574 E886 E929
Write to viminfo file [file] (default: see `viminfo-file-name` above).
This command has no effect when `'viminfofile'` has been set to "NONE".
The information in the file is first read in to make a merge between old and new info. When [!] is used, the old information is not read first, only the internal info is written. If `'viminfo'` is empty, marks for up to 100 files will be written.
When you get error "E929: Too many viminfo temp files" check that no old temp files were left behind (e.g. `~/viminfo*`) and that you can write in the directory of the `.viminfo` file.

`:ol[dfiles]` `:ol` `:oldfiles`
List the files that have marks stored in the viminfo file. This list is read on startup and only changes afterwards with `:rviminfo!`. Also see `v:oldfiles`.
The number can be used with `c_#<`.
The output can be filtered with `:filter`, e.g.:
`filter /\.vim/ oldfiles`
The filtering happens on the file name.
{only when compiled with the |+eval| feature}

`:bro[wse] ol[dfiles][!]`
List file names as with `:oldfiles`, and then prompt for a number. When the number is valid that file from the list is edited.
If you get the `press-enter` prompt you can press "q" and still get the prompt to enter a file number.
Use ! to abandon a modified buffer. `abandon`
{not when compiled with tiny features}

Editing files

edit-files

- 1. Introduction edit-intro
- 2. Editing a file edit-a-file
- 3. The argument list argument-list
- 4. Writing writing
- 5. Writing and quitting write-quit
- 6. Dialogs edit-dialogs
- 7. The current directory current-directory
- 8. Editing binary files edit-binary
- 9. Encryption encryption
- 10. Timestamps timestamps
- 11. File Searching file-searching

=====
1. Introduction

edit-intro

Editing a file with Vim means:

- 1. reading the file into a buffer
- 2. changing the buffer with editor commands
- 3. writing the buffer into a file

current-file

As long as you don't write the buffer, the original file remains unchanged. If you start editing a file (read a file into the buffer), the file name is remembered as the "current file name". This is also known as the name of the current buffer. It can be used with "%" on the command line `:%`.

alternate-file

If there already was a current file name, then that one becomes the alternate file name. It can be used with "#" on the command line `:_#` and you can use the `CTRL-^` command to toggle between the current and the alternate file. However, the alternate file name is not changed when `:keepalt` is used. An alternate file name is remembered for each window.

:keepalt :keepa

`:keepalt {cmd}` Execute `{cmd}` while keeping the current alternate file name. **Note** that commands invoked indirectly (e.g., with a function) may still set the alternate file name.

All file names are remembered in the buffer list. When you enter a file name, for editing (e.g., with `":e filename"`) or writing (e.g., with `":w filename"`), the file name is added to the list. You can use the buffer list to remember which files you edited and to quickly switch from one file to another (e.g., to copy text) with the `CTRL-^` command. First type the number of the file and then hit **CTRL-^**.

CTRL-G or **CTRL-G :f :fi :file**
:f[ile] Prints the current file name (as typed, unless ":cd" was used), the cursor position (unless the 'ruler' option is set), and the file status (readonly, modified, read errors, new file). See the 'shortmess' option about how to make this message shorter.

:f[ile]! like **:file** , but don't truncate the name even when 'shortmess' indicates this.

{count}CTRL-G Like **CTRL-G**, but prints the current file name with full path. If the count is higher than 1 the current buffer number is also given.

g CTRL-G **g_CTRL-G word-count byte-count**
Prints the current position of the cursor in five ways: Column, Line, Word, Character and Byte. If the number of Characters and Bytes is the same then the Character position is omitted.

If there are characters in the line that take more than one position on the screen (<Tab> or special character), or characters using more than one byte per column (characters above 0x7F when 'encoding' is utf-8), both the byte column and the screen column are shown, separated by a dash.

Also see the 'ruler' option and the `wordcount()` function.

{Visual}g CTRL-G **v_g_CTRL-G**
Similar to "g CTRL-G", but Word, Character, Line, and Byte counts for the visually selected region are displayed. In Blockwise mode, Column count is also shown. (For {Visual} see `Visual-mode` .)

:f[ile][!] {name} **:file_f**
Sets the current file name to {name}. The optional ! avoids truncating the message, as with **:file** . If the buffer did have a name, that name becomes the `alternate-file` name. An unlisted buffer is created to hold the old name.

:0f[ile][!] **:0file**
Remove the name of the current buffer. The optional ! avoids truncating the message, as with **:file** .

:buffers
:files
:ls List all the currently known file names. See `windows.txt` **:files** **:buffers** **:ls** .

Vim will remember the full path name of a file name that you enter. In most cases when the file name is displayed only the name you typed is shown, but the full path name is being used if you used the `:cd` command `:cd` .

home-replace

If the environment variable \$HOME is set, and the file name starts with that string, it is often displayed with HOME replaced with "~". This was done to keep file names short. When reading or writing files the full name is still used, the "~" is only used when displaying file names. When replacing the file name would result in just "~", "~/ " is used instead (to avoid confusion between options set to \$HOME with **'backupext'** set to "~").

When writing the buffer, the default is to use the current file name. Thus when you give the `ZZ` or `:wq` command, the original file will be overwritten. If you do not want this, the buffer can be written into another file by giving a file name argument to the `:write` command. For example:

```
vim testfile
[change the buffer with editor commands]
:w newfile
:q
```

This will create a file "newfile", that is a modified copy of "testfile". The file "testfile" will remain unchanged. Anyway, if the **'backup'** option is set, Vim renames or copies the original file before it will be overwritten. You can use this file if you discover that you need the original file. See also the **'patchmode'** option. The name of the backup file is normally the same as the original file with **'backupext'** appended. The default "~" is a bit strange to avoid accidentally overwriting existing files. If you prefer ".bak" change the **'backupext'** option. Extra dots are replaced with '_' on MS-Windows machines, when Vim has detected that an MS-DOS-like filesystem is being used (e.g., messydos or crossdos) or when the **'shortname'** option is on. The backup file can be placed in another directory by setting **'backupdir'**.

auto-shortname

Technical: On the Amiga you can use 30 characters for a file name. But on an MS-DOS-compatible filesystem only 8 plus 3 characters are available. Vim tries to detect the type of filesystem when it is creating the .swp file. If an MS-DOS-like filesystem is suspected, a flag is set that has the same effect as setting the **'shortname'** option. This flag will be reset as soon as you start editing a new file. The flag will be used when making the file name for the ".swp" and ".~" files for the current file. But when you are editing a file in a normal filesystem and write to an MS-DOS-like filesystem the flag will not have been set. In that case the creation of the ".~" file may fail and you will get an error message. Use the **'shortname'** option in this case.

When you started editing without giving a file name, "No File" is displayed in messages. If the `:write` command is used with a file name argument, the file name for the current file is set to that file name. This only happens when the 'F' flag is included in **'coptions'** (by default it is included) `cpo-F` . This is useful when entering text in an empty buffer and then writing it to a file. If **'coptions'** contains the 'f' flag (by default it is NOT included)

`cpo-f` the file name is set for the `":read file"` command. This is useful when starting Vim without an argument and then doing `":read file"` to start editing a file.

When the file name was set and `'filetype'` is empty the filetype detection autocommands will be triggered.

`not-edited`

Because the file name was set without really starting to edit that file, you are protected from overwriting that file. This is done by setting the `"notedited"` flag. You can see if this flag is set with the `CTRL-G` or `":file"` command. It will include `"[Not edited]"` when the `"notedited"` flag is set. When writing the buffer to the current file name (with `":w!"`), the `"notedited"` flag is reset.

`abandon`

Vim remembers whether you have changed the buffer. You are protected from losing the changes you made. If you try to quit without writing, or want to start editing another file, Vim will refuse this. In order to overrule this protection, add a `':'` to the command. The changes will then be lost. For example: `":q"` will not work if the buffer was changed, but `":q!"` will. To see whether the buffer was changed use the `"CTRL-G"` command. The message includes the string `"[Modified]"` if the buffer has been changed, or `"+"` if the `'m'` flag is in `'shortmess'`.

If you want to automatically save the changes without asking, switch on the `'autowriteall'` option. `'autowrite'` is the associated Vi-compatible option that does not work for all commands.

If you want to keep the changed buffer without saving it, switch on the `'hidden'` option. See `hidden-buffer`. Some commands work like this even when `'hidden'` is not set, check the help for the command.

=====

2. Editing a file

`edit-a-file`

`:e :edit reload`

`:e[dit] [++opt] [+cmd]` Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim. This fails when changes have been made to the current buffer and `'autowriteall'` isn't set or the file can't be written.

Also see `++opt` and `+cmd`.

`:edit! discard`

`:e[dit]! [++opt] [+cmd]`

Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.

Also see `++opt` and `+cmd`.

`:edit_f`

`:e[dit] [++opt] [+cmd] {file}`

Edit `{file}`.

This fails when changes have been made to the current buffer, unless `'hidden'` is set or `'autowriteall'` is

set and the file can be written.
Also see `++opt` and `+cmd` .

`:e[dit]!` `[++opt]` `[+cmd]` `{file}` `:edit!_f`
Edit `{file}` always. Discard any changes to the current buffer.
Also see `++opt` and `+cmd` .

`:e[dit]` `[++opt]` `[+cmd]` `#[count]` `:edit_#` `:e#`
Edit the `[count]`th buffer (as shown by `:files`).
This command does the same as `[count]` `CTRL-^`. But `":e #"` doesn't work if the alternate buffer doesn't have a file name, while `CTRL-^` still works then.
Also see `++opt` and `+cmd` .

`:ene[w]` `:ene` `:enew`
Edit a new, unnamed buffer. This fails when changes have been made to the current buffer, unless `'hidden'` is set or `'autowriteall'` is set and the file can be written.
If `'fileformats'` is not empty, the first format given will be used for the new buffer. If `'fileformats'` is empty, the `'fileformat'` of the current buffer is used.

`:ene[w]!` `:ene!` `:enew!`
Edit a new, unnamed buffer. Discard any changes to the current buffer.
Set `'fileformat'` like `:enew` .

`:fin[d][!]` `[++opt]` `[+cmd]` `{file}` `:fin` `:find`
Find `{file}` in `'path'` and then `:edit` it.

`:{count}fin[d][!]` `[++opt]` `[+cmd]` `{file}`
Just like `":find"`, but use the `{count}` match in `'path'`. Thus `":2find file"` will find the second "file" found in `'path'`. When there are fewer matches for the file in `'path'` than asked for, you get an error message.

`:ex` `[++opt]` `[+cmd]` `[file]` `:ex`
Same as `:edit` .

`:vi[sual][!]` `[++opt]` `[+cmd]` `[file]` `:vi` `:visual`
When used in Ex mode: Leave `Ex-mode` , go back to Normal mode. Otherwise same as `:edit` .

`:vie[w][!]` `[++opt]` `[+cmd]` `file` `:vie` `:view`
When used in Ex mode: Leave `Ex-mode` , go back to Normal mode. Otherwise same as `:edit` , but set

'readonly' option for this buffer.

CTRL-^

CTRL-^ CTRL-6

Edit the alternate file. Mostly the alternate file is the previously edited file. This is a quick way to toggle between two files. It is equivalent to ":e #", except that it also works when there is no file name.

If the 'autowrite' or 'autowriteall' option is on and the buffer was changed, write it.

Mostly the ^ character is positioned on the 6 key, pressing CTRL and 6 then gets you what we call **CTRL-^**. But on some non-US keyboards **CTRL-^** is produced in another way.

{count}CTRL-^

Edit [count]th file in the buffer list (equivalent to ":e #[count]"). This is a quick way to switch between files.

See CTRL-^ above for further details.

[count]]f
[count][f

]f [f

Same as "gf". Deprecated.

[count]gf

gf E446 E447

Edit the file whose name is under or after the cursor. Mnemonic: "goto file".

Uses the 'isfname' option to find out which characters are supposed to be in a file name. Trailing punctuation characters ".,;:!" are ignored. Escaped spaces "\ " are reduced to a single space.

Uses the 'path' option as a list of directory names to look for the file. See the 'path' option for details about relative directories and wildcards.

Uses the 'suffixesadd' option to check for file names with a suffix added.

If the file can't be found, 'includeexpr' is used to modify the name and another attempt is done.

If a [count] is given, the count'th file that is found in the 'path' is edited.

This command fails if Vim refuses to abandon the current file.

If you want to edit the file in a new window use CTRL-W_CTRL-F .

If you do want to edit a new file, use:

:e <file>

To make gf always work like that:

:map gf :e <file><CR>

If the name is a hypertext link, that looks like

"type://machine/path", you need the netrw plugin.

For Unix the '~' character is expanded, like in

"~user/file". Environment variables are expanded too expand-env .

v_gf

`{Visual}[count]gf` Same as "gf", but the highlighted text is used as the name of the file to edit. **'isfname'** is ignored. Leading blanks are skipped, otherwise all blanks and special characters are included in the file name. (For `{Visual}` see [Visual-mode](#) .)

`[count]gF` Same as "gf", except if a number follows the file name, then the cursor is positioned on that line in the file. **gF**
The file name and the number must be separated by a non-filename (see **'isfname'**) and non-numeric character. " line " is also recognized, like it is used in the output of ``:verbose command UserCmd``. White space between the filename, the separator and the number are ignored.
Examples:

```
eval.c:10
eval.c @ 20
eval.c (30)
eval.c 40
```

`{Visual}[count]gF` Same as "v_gf". **v_gF**

These commands are used to start editing a single file. This means that the file is read into the buffer and the current file name is set. The file that is opened depends on the current directory, see `:cd` .

See [read-messages](#) for an explanation of the message that is given after the file has been read.

You can use the `":e!"` command if you messed up the buffer and want to start all over again. The `":e"` command is only useful if you have changed the current file name.

Besides the things mentioned here, more special items for where a filename is expected are mentioned at [cmdline-special](#) . **:filename {file}**

Note for systems other than Unix: When using a command that accepts a single file name (like `":edit file"`) spaces in the file name are allowed, but trailing spaces are ignored. This is useful on systems that regularly embed spaces in file names (like MS-Windows and the Amiga). Example: The command `":e Long File Name "` will edit the file "Long File Name". When using a command that accepts more than one file name (like `":next file1 file2"`) embedded spaces must be escaped with a backslash.

Wildcards in `{file}` are expanded, but as with file completion, **'wildignore'** and **'suffixes'** apply. Which wildcards are supported depends on the system. These are the common ones:

```
?      matches one character
*      matches anything, including nothing
```

`**` matches anything, including nothing, recurses into directories
`[abc]` match 'a', 'b' or 'c'

To avoid the special meaning of the wildcards prepend a backslash. However, on MS-Windows the backslash is a path separator and "path\[abc]" is still seen as a wildcard when "[" is in the **'isfname'** option. A simple way to avoid this is to use "path\[[]abc]", this matches the file "path\[abc]".

starstar-wildcard

Expanding "**" is possible on Unix, Win32, macOS and a few other systems (but it may depend on your **'shell'** setting on Unix and macOS. It's known to work correctly for zsh; for bash this requires at least bash version >= 4.X).

This allows searching a directory tree. This goes up to 100 directories deep.

Note there are some commands where this works slightly differently, see [file-searching](#) .

Example:

```
:n **/*.txt
```

Finds files:

```
aaa.txt  
subdir/bbb.txt  
a/b/c/d/ccc.txt
```

When non-wildcard characters are used right before or after "**" these are only matched in the top directory. They are not used for directories further down in the tree. For example:

```
:n /usr/inc**/types.h
```

Finds files:

```
/usr/include/types.h  
/usr/include/sys/types.h  
/usr/inc/old/types.h
```

Note that the path with "/sys" is included because it does not need to match "/inc". Thus it's like matching "/usr/inc*/*/*...", not "/usr/inc*/inc*/inc*".

backtick-expansion `-expansion

On Unix and a few other systems you can also use backticks for the file name argument, for example:

```
:next `find . -name ver\\*.c -print`  
:view `ls -t *.patch | head -n1`
```

Vim will run the command in backticks using the **'shell'** and use the standard output as argument for the given Vim command (error messages from the shell command will be discarded).

To see what shell command Vim is running, set the **'verbose'** option to 4. When the shell command returns a non-zero exit code, an error message will be displayed and the Vim command will be aborted. To avoid this make the shell always return zero like so:

```
:next `find . -name ver\\*.c -print ||| true`
```

The backslashes before the star are required to prevent the shell from expanding "ver*.c" prior to execution of the find program. The backslash before the shell pipe symbol "|" prevents Vim from parsing it as command termination.

This also works for most other systems, with the restriction that the backticks must be around the whole item. It is not possible to have text directly before the first or just after the last backtick.

``=` E1083

You can have the backticks expanded as a Vim expression, instead of as an external command, by putting an equal sign right after the first backtick, e.g.:

```
:e `=tempname()`
```

The expression can contain just about anything, thus this can also be used to avoid the special meaning of `'"`, `'|'`, `'%'` and `'#'`. However, `'wildignore'` does apply like to other wildcards.

Environment variables in the expression are expanded when evaluating the expression, thus this works:

```
:e `=$HOME .. '/.vimrc`
```

This uses `$HOME` inside a string and it will be used literally, most likely not what you intended:

```
:e `='$HOME' .. '/.vimrc`
```

If the expression returns a string then names are to be separated with line breaks. When the result is a `List` then each item is used as a name. Line breaks also separate names.

Note that such expressions are only supported in places where a filename is expected as an argument to an Ex-command.

`++opt` `[++opt]`

The `[++opt]` argument can be used to force the value of `'fileformat'`, `'fileencoding'` or `'binary'` to a value for one command, and to specify the behavior for bad characters. The form is:

```
++{optname}
```

Or:

```
++{optname}={value}
```

Where `{optname}` is one of:

		<code>++ff</code>	<code>++enc</code>	<code>++bin</code>	<code>++nobin</code>	<code>++edit</code>
<code>ff</code>	or fileformat	overrides	<code>'fileformat'</code>			
<code>enc</code>	or encoding	overrides	<code>'fileencoding'</code>			
<code>bin</code>	or binary	sets	<code>'binary'</code>			
<code>nobin</code>	or nobinary	resets	<code>'binary'</code>			
<code>bad</code>		specifies behavior for bad characters				
<code>edit</code>		for <code>:read</code> only: keep option values as if editing a file				

`{value}` cannot contain white space. It can be any valid value for these options. Examples:

```
:e ++ff=unix
```

This edits the same file again with `'fileformat'` set to "unix".

```
:w ++enc=latin1 newfile
```

This writes the current buffer to "newfile" in latin1 format.

The message given when writing a file will show "[converted]" when `'fileencoding'` or the value specified with `++enc` differs from `'encoding'`.

There may be several `++opt` arguments, separated by white space. They must all appear before any `+cmd` argument.

++bad

The argument of "++bad=" specifies what happens with characters that can't be converted and illegal bytes. It can be one of three things:

- ++bad=X A single-byte character that replaces each bad character.
- ++bad=keep Keep bad characters without conversion. **Note** that this may result in illegal bytes in your text!
- ++bad=drop Remove the bad characters.

The default is like "++bad=?": Replace each bad character with a question mark. In some places an inverted question mark is used (0xBF).

Note that not all commands use the ++bad argument, even though they do not give an error when you add it. E.g. `:write` .

Note that when reading, the '**fileformat**' and '**fileencoding**' options will be set to the used format. When writing this doesn't happen, thus a next write will use the old value of the option. Same for the '**binary**' option.

+cmd [+cmd]

The `[+cmd]` argument can be used to position the cursor in the newly opened file, or execute any other command:

- + Start at the last line.
- +{num} Start at line {num}.
- +/{pat} Start at first line containing {pat}.
- +{command} Execute {command} after opening the new file.
 {command} is any Ex command.

To include a white space in the {pat} or {command}, precede it with a backslash. Double the number of backslashes.

```
:edit +/The\ book            file
:edit +/dir\ dirname\        file
:edit +set\ dir=c:\\\\temp    file
```

Note that in the last example the number of backslashes is halved twice: Once for the "+cmd" argument and once for the ":set" command.

file-formats

The '**fileformat**' option sets the <EOL> style for a file:

'fileformat'	characters	name	
"dos"	<CR><NL> or <NL>	DOS format	DOS-format
"unix"	<NL>	Unix format	Unix-format
"mac"	<CR>	Mac format	Mac-format

Previously '**textmode**' was used. It is obsolete now.

When reading a file, the mentioned characters are interpreted as the <EOL>. In DOS format (default for Win32), <CR><NL> and <NL> are both interpreted as the <EOL>. **Note** that when writing the file in DOS format, <CR> characters will be added for each single <NL>. Also see `file-read` .

When writing a file, the mentioned characters are used for <EOL>. For DOS format <CR><NL> is used. Also see `DOS-format-write` .

You can read a file in DOS format and write it in Unix format. This will replace all <CR><NL> pairs by <NL> (assuming '**fileformats**' includes "dos"):

```
:e file
```

```
:set fileformat=unix
:w
```

If you read a file in Unix format and write with DOS format, all `<NL>` characters will be replaced with `<CR><NL>` (assuming `'fileformats'` includes "unix"):

```
:e file
:set fileformat=dos
:w
```

If you start editing a new file and the `'fileformats'` option is not empty (which is the default), Vim will try to detect whether the lines in the file are separated by the specified formats. When set to "unix,dos", Vim will check for lines with a single `<NL>` (as used on Unix and Amiga) or by a `<CR><NL>` pair (MS-Windows). Only when ALL lines end in `<CR><NL>`, `'fileformat'` is set to "dos", otherwise it is set to "unix". When `'fileformats'` includes "mac", and no `<NL>` characters are found in the file, `'fileformat'` is set to "mac".

If the `'fileformat'` option is set to "dos" on non-MS-Windows systems the message "[dos format]" is shown to remind you that something unusual is happening. On MS-Windows systems you get the message "[unix format]" if `'fileformat'` is set to "unix". On all systems but the Macintosh you get the message "[mac format]" if `'fileformat'` is set to "mac".

If the `'fileformats'` option is empty and DOS format is used, but while reading a file some lines did not end in `<CR><NL>`, "[CR missing]" will be included in the file message.

If the `'fileformats'` option is empty and Mac format is used, but while reading a file a `<NL>` was found, "[NL missing]" will be included in the file message.

If the new file does not exist, the `'fileformat'` of the current buffer is used when `'fileformats'` is empty. Otherwise the first format from `'fileformats'` is used for the new file.

Before editing binary, executable or Vim script files you should set the `'binary'` option. A simple way to do this is by starting Vim with the "-b" option. This will avoid the use of `'fileformat'`. Without this you risk that single `<NL>` characters are unexpectedly replaced with `<CR><NL>`.

You can encrypt files that are written by setting the `'key'` option. This provides some security against others reading your files. [encryption](#)

END OF LINE AND END OF FILE

[eol-and-eof](#)

Vim has several options to control the file format:

<code>'fileformat'</code>	the <code><EOL></code> style: Unix, DOS, Mac
<code>'endofline'</code>	whether the last line ends with a <code><EOL></code>
<code>'endoffile'</code>	whether the file ends with a CTRL-Z
<code>'fixendofline'</code>	whether to fix eol and eof

The first three values are normally detected automatically when reading the file and are used when writing the text to a file. While editing the buffer it looks like every line has a line ending and the **CTRL-Z** isn't there (an exception is when `'binary'` is set, it works differently then).

The **'fixendofline'** option can be used to choose what to write. You can also change the option values to write the file differently than how it was read.

Here are some examples how to use them.

If you want files in Unix format (every line NL terminated):

```
setl ff=unix fixeol
```

You should probably do this on any Unix-like system. Also modern MS-Windows systems tend to work well with this. It is recommended to always use this format for Vim scripts.

If you want to use an old MS-DOS file in a modern environment, fixing line endings and dropping **CTRL-Z**, but keeping the `<CR><NL>` style `<EOL>`:

```
setl ff=dos fixeol
```

This is useful for many MS-Windows programs, they regularly expect the `<CR><NL>` line endings.

If you want to drop the final `<EOL>` and add a final **CTRL-Z** (e.g. for an old system like CP/M):

```
setl ff=dos nofixeol noeol eof
```

If you want to preserve the fileformat exactly as-is, including any final `<EOL>` and final **CTRL-Z**:

```
setl nofixeol
```

=====

3. The argument list argument-list arglist

If you give more than one file name when starting Vim, this list is remembered as the argument list. You can jump to each file in this list.

Do not confuse this with the buffer list, which you can see with the `:buffers` command. The argument list was already present in Vi, the buffer list is new in Vim. Every file name in the argument list will also be present in the buffer list (unless it was deleted with `:bdel` or `:bwipe`). But it's common that names in the buffer list are not in the argument list.

This subject is introduced in section [07.2](#) of the user manual.

There is one global argument list, which is used for all windows by default. It is possible to create a new argument list local to a window, see `:arglocal` .

You can use the argument list with the following commands, and with the expression functions `argc()` and `argv()` . These all work on the argument list of the current window.

```
:ar[gs] :ar :arg :args  
Print the argument list, with the current file in square brackets.
```

```
:ar[gs] [++opt] [+cmd] {arglist} :args_f  
Define {arglist} as the new argument list and edit
```


the first one. This fails when changes have been made and Vim does not want to `abandon` the current buffer. Also see `++opt` and `+cmd` .

`:ar[gs]! [++opt] [+cmd] {arglist} :args_f!`
Define `{arglist}` as the new argument list and edit the first one. Discard any changes to the current buffer.
Also see `++opt` and `+cmd` .

`:[count]arge[dit][!] [++opt] [+cmd] {name} .. :arge :argedit`
Add `{name}`s to the argument list and edit it. When `{name}` already exists in the argument list, this entry is edited.
This is like using `:argadd` and then `:edit` .
Spaces in filenames have to be escaped with `"\"`.
`[count]` is used like with `:argadd` .
If the current file cannot be `abandoned` `{name}`s will still be added to the argument list, but won't be edited. No check for duplicates is done.
Also see `++opt` and `+cmd` .

`:[count]arga[dd] {name} .. :arga :argadd E479 E1156`
`:[count]arga[dd]`

Add the `{name}`s to the argument list. When `{name}` is omitted add the current buffer name to the argument list.

If `[count]` is omitted, the `{name}`s are added just after the current entry in the argument list. Otherwise they are added after the `[count]`'th file. If the argument list is "a b c", and "b" is the current argument, then these commands result in:

command	new argument list
<code>:argadd x</code>	a b x c
<code>:0argadd x</code>	x a b c
<code>:1argadd x</code>	a x b c
<code>:\$argadd x</code>	a b c x

And after the last one:

`:+2argadd y` a b c x y

There is no check for duplicates, it is possible to add a file to the argument list twice. You can use `:argdedupe` to fix it afterwards:

`:argadd *.txt | argdedupe`

The currently edited file is not changed.

Note: you can also use this method:

`:args ## x`

This will add the "x" item and sort the new list.

`:argded[upe] :argded :argdedupe`
Remove duplicate filenames from the argument list. If your current file is a duplicate, your current file will change to the original file index.

`:argd[ele]te {pattern} .. :argd :argdelete E480 E610`

Delete files from the argument list that match the `{pattern}s`. `{pattern}` is used like a file pattern, see `file-pattern`. `"%"` can be used to delete the current entry.

This command keeps the currently edited file, also when it's deleted from the argument list.

Example:

```
:argdel *.obj
```

`:[range]argd[elete]` Delete the `[range]` files from the argument list.

Example:

```
:10,$argdel
```

Deletes arguments 10 and further, keeping 1-9.

```
:$argd
```

Deletes just the last one.

```
:argd
```

```
:.argd
```

Deletes the current argument.

```
:%argd
```

Removes all the files from the arglist.

When the last number in the range is too high, up to the last argument is deleted.

`:argu` `:argument`

`:[count]argu[ment]` `[count]` `[++opt]` `[+cmd]`

Edit file `[count]` in the argument list. When `[count]` is omitted the current entry is used. This fails when changes have been made and Vim does not want to `abandon` the current buffer.

Also see `++opt` and `+cmd`.

`:[count]argu[ment]!` `[count]` `[++opt]` `[+cmd]`

Edit file `[count]` in the argument list, discard any changes to the current buffer. When `[count]` is omitted the current entry is used.

Also see `++opt` and `+cmd`.

`:[count]n[ext]` `[++opt]` `[+cmd]` `:n` `:ne` `:next` `E165` `E163`

Edit `[count]` next file. This fails when changes have been made and Vim does not want to `abandon` the current buffer. Also see `++opt` and `+cmd`.

`:[count]n[ext]!` `[++opt]` `[+cmd]`

Edit `[count]` next file, discard any changes to the buffer. Also see `++opt` and `+cmd`.

`:n[ext]` `[++opt]` `[+cmd]` `{arglist}` `:next_f`

Same as `:args_f`.

`:n[ext]!` `[++opt]` `[+cmd]` `{arglist}`

Same as `:args_f!`.

`:[count]N[ext]` `[count]` `[++opt]` `[+cmd]` `:Next` `:N` `E164`

Edit `[count]` previous file in argument list. This

fails when changes have been made and Vim does not want to `abandon` the current buffer. Also see `++opt` and `+cmd` .

`:[count]N[ext]! [count] [++opt] [+cmd]`
Edit `[count]` previous file in argument list. Discard any changes to the buffer. Also see `++opt` and `+cmd` .

`:[count]prev[ious] [count] [++opt] [+cmd]` `:prev` `:previous`
Same as `:Next`. Also see `++opt` and `+cmd` .

`:rew[ind] [++opt] [+cmd]` `:rew` `:rewind`
Start editing the first file in the argument list. This fails when changes have been made and Vim does not want to `abandon` the current buffer. Also see `++opt` and `+cmd` .

`:rew[ind]! [++opt] [+cmd]`
Start editing the first file in the argument list. Discard any changes to the buffer. Also see `++opt` and `+cmd` .

`:fir[st][!] [++opt] [+cmd]` `:fir` `:first`
Other name for `":rewind"`.

`:la[st] [++opt] [+cmd]` `:la` `:last`
Start editing the last file in the argument list. This fails when changes have been made and Vim does not want to `abandon` the current buffer. Also see `++opt` and `+cmd` .

`:la[st]! [++opt] [+cmd]`
Start editing the last file in the argument list. Discard any changes to the buffer. Also see `++opt` and `+cmd` .

`:[count]wn[ext] [++opt]` `:wn` `:wnext`
Write current file and start editing the `[count]` next file. Also see `++opt` and `+cmd` .

`:[count]wn[ext] [++opt] {file}`
Write current file to `{file}` and start editing the `[count]` next file, unless `{file}` already exists and the `'writeany'` option is off. Also see `++opt` and `+cmd` .

`:[count]wn[ext]! [++opt] {file}`
Write current file to `{file}` and start editing the `[count]` next file. Also see `++opt` and `+cmd` .

```
:[count]wN[ext][!] [++opt] [file]           :wN    :wNext
:[count]wp[revious][!] [++opt] [file]       :wp    :wprevious
      Same as :wnext, but go to previous file instead of
      next.
```

The [count] in the commands above defaults to one. For some commands it is possible to use two counts. The last one (rightmost one) is used.

If no [+cmd] argument is present, the cursor is positioned at the last known cursor position for the file. If 'startofline' is set, the cursor will be positioned at the first non-blank in the line, otherwise the last known column is used. If there is no last known cursor position the cursor will be in the first line (the last line in Ex mode).

{arglist}

The wildcards in the argument list are expanded and the file names are sorted. Thus you can use the command "vim *.c" to edit all the C files. From within Vim the command ":n *.c" does the same.

White space is used to separate file names. Put a backslash before a space or tab to include it in a file name. E.g., to edit the single file "foo bar":

```
:next foo\ bar
```

On Unix and a few other systems you can also use backticks, for example:

```
:next `find . -name \*.c -print`
```

The backslashes before the star are required to prevent "*.c" to be expanded by the shell before executing the find program.

arglist-position

When there is an argument list you can see which file you are editing in the title of the window (if there is one and 'title' is on) and with the file message you get with the "CTRL-G" command. You will see something like

```
(file 4 of 11)
```

If 'shortmess' contains 'f' it will be

```
(4 of 11)
```

If you are not really editing the file at the current position in the argument list it will be

```
(file (4) of 11)
```

This means that you are position 4 in the argument list, but not editing the fourth file in the argument list. This happens when you do ":e file".

LOCAL ARGUMENT LIST

:arglocal

```
:argl[ocal]           Make a local copy of the global argument list.
                      Doesn't start editing another file.
```

```
:argl[ocal][!] [++opt] [+cmd] {arglist}
                      Define a new argument list, which is local to the
                      current window. Works like :args_f otherwise.
```

:argglobal

`:argg[lobal]` Use the global argument list for the current window. Doesn't start editing another file.

`:argg[lobal][!] [+opt] [+cmd] {arglist}`
Use the global argument list for the current window. Define a new global argument list like `:args_f`. All windows using the global argument list will see this new list.

There can be several argument lists. They can be shared between windows. When they are shared, changing the argument list in one window will also change it in the other window.

When a window is split the new window inherits the argument list from the current window. The two windows then share this list, until one of them uses `:arglocal` or `:argglobal` to use another argument list.

USING THE ARGUMENT LIST

`:[range]argdo[!] {cmd}` Execute `{cmd}` for each file in the argument list or if `[range]` is specified only for arguments in that range. It works like doing this:
`:rewind`
`:{cmd}`
`:next`
`:{cmd}`
etc.

When the current file can't be abandoned and the `[!]` is not present, the command fails.

When an error is detected on one file, further files in the argument list will not be visited.

The last file in the argument list (or where an error occurred) becomes the current file.

`{cmd}` can contain `|` to concatenate several commands.

`{cmd}` must not change the argument list.

Note: While this command is executing, the Syntax autocommand event is disabled by adding it to `'eventignore'`. This considerably speeds up editing each file.

Also see `:windo`, `:tabdo`, `:bufdo`, `:cdo`, `:ldo`, `:cfdo` and `:lfdo`

Example:

```
:args *.c
:argdo set ff=unix | update
```

This sets the `'fileformat'` option to "unix" and writes the file if it is now changed. This is done for all *.c files.

Example:

```
:args *. [ch]
:argdo %s/\<my_foo\>/My_Foo/ge | update
```

This changes the word "my_foo" to "My_Foo" in all *.c and *.h files. The "e"

flag is used for the ":substitute" command to avoid an error for files where "my_foo" isn't used. ":update" writes the file only if changes were made.

4. Writing

writing save-file

Note: When the 'write' option is off, you are not able to write any file.

	:w	:write		
	E502	E503	E504	E505
	E512	E514	E667	E949

:w[rite] [++opt] Write the whole buffer to the current file. This is the normal way to save changes to a file. It fails when the 'readonly' option is set or when there is another reason why the file can't be written. For ++opt see ++opt , but only ++bin, ++nabin, ++ff and ++enc are effective.

:w[rite]! [++opt] Like ":write", but forcefully write when 'readonly' is set or there is another reason why writing was refused.
Note: This may change the permission and ownership of the file and break (symbolic) links. Add the 'W' flag to 'coptions' to avoid this.

:[range]w[rite][!] [++opt] Write the specified lines to the current file. This is unusual, because the file will not contain all lines in the buffer.

	:w_f	:write_f
--	------	----------

:[range]w[rite] [++opt] {file} Write the specified lines to {file}, unless it already exists and the 'writeany' option is off.

	:w!
--	-----

:[range]w[rite]! [++opt] {file} Write the specified lines to {file}. Overwrite an existing file.

	:w_a	:write_a	E494
--	------	----------	------

:[range]w[rite][!] [++opt] >> Append the specified lines to the current file.

:[range]w[rite][!] [++opt] >> {file} Append the specified lines to {file}. '!' forces the write even if file does not exist.

	:w_c	:write_c
--	------	----------

:[range]w[rite] [++opt] !{cmd} Execute {cmd} with [range] lines as standard input (note the space in front of the '!'). {cmd} is executed like with ":{cmd}", any '!' is replaced with the previous command :! .

The default `[range]` for the `":w"` command is the whole buffer (1,\$). If you write the whole buffer, it is no longer considered changed. When you write it to a different file with `":w somefile"` it depends on the `"+"` flag in `'coptions'`. When included, the write command will reset the `'modified'` flag, even though the buffer itself may still be different from its file.

If a file name is given with `":w"` it becomes the alternate file. This can be used, for example, when the write fails and you want to try again later with `":w #"`. This can be switched off by removing the `'A'` flag from the `'coptions'` option.

Note that the `'fsync'` option matters here. If it's set it may make writes slower (but safer).

`:sav[eas][!] [++opt] {file}` `:sav` `:saveas`
Save the current buffer under the name `{file}` and set the filename of the current buffer to `{file}`. The previous name is used for the alternate file name. The `!` is needed to overwrite an existing file. When `'filetype'` is empty filetype detection is done with the new name, before the file is written. When the write was successful `'readonly'` is reset.

`:[range]up[date][!] [++opt] [>>] [file]` `:up` `:update`
Like `":write"`, but only write when the buffer has been modified.

WRITING WITH MULTIPLE BUFFERS

`buffer-write`

`:wa[ll]` `:wa` `:wall`
Write all changed buffers. Buffers without a file name cause an error message. Buffers which are `readonly` are not written.

`:wa[ll]!`
Write all changed buffers, even the ones that are `readonly`. Buffers without a file name are not written and cause an error message.

Vim will warn you if you try to overwrite a file that has been changed elsewhere. See `timestamp`.

`backup` `E207` `E506` `E507` `E508` `E509` `E510`
If you write to an existing file (but do not append) while the `'backup'`, `'writebackup'` or `'patchmode'` option is on, a backup of the original file is made. The file is either copied or renamed (see `'backupcopy'`). After the file has been successfully written and when the `'writebackup'` option is on and the `'backup'` option is off, the backup file is deleted. When the `'patchmode'` option is on the backup file may be renamed.

backup-table

'backup'	'writebackup'	action
off	off	no backup made
off	on	backup current file, deleted afterwards (default)
on	off	delete old backup, backup current file
on	on	delete old backup, backup current file

When the **'backupskip'** pattern matches with the name of the file which is written, no backup file is made. The values of **'backup'** and **'writebackup'** are ignored then.

When the **'backup'** option is on, an old backup file (with the same name as the new backup file) will be deleted. If **'backup'** is not set, but **'writebackup'** is set, an existing backup file will not be deleted. The backup file that is made while the file is being written will have a different name.

On some filesystems it's possible that in a crash you lose both the backup and the newly written file (it might be there but contain bogus data). In that case try recovery, because the swap file is synced to disk and might still be there. **:recover**

The directories given with the **'backupdir'** option are used to put the backup file in. (default: same directory as the written file).

Whether the backup is a new file, which is a copy of the original file, or the original file renamed depends on the **'backupcopy'** option. See there for an explanation of when the copy is made and when the file is renamed.

If the creation of a backup file fails, the write is not done. If you want to write anyway add a **'!'** to the command.

write-permissions

When writing a new file the permissions are read-write. For unix the mask is **0o666** with additionally **umask** applied. When writing a file that was read Vim will preserve the permissions, but clear the **s-bit**.

write-readonly

When the **'cptions'** option contains **'W'**, Vim will refuse to overwrite a readonly file. When **'W'** is not present, **":w!"** will overwrite a readonly file, if the system allows it (the directory must be writable).

write-fail

If the writing of the new file fails, you have to be careful not to lose your changes AND the original file. If there is no backup file and writing the new file failed, you have already lost the original file! **DON'T EXIT VIM UNTIL YOU WRITE OUT THE FILE!** If a backup was made, it is put back in place of the original file (if possible). If you exit Vim, and lose the changes you made, the original file will mostly still be there. If putting back the original file fails, there will be an error message telling you that you lost the original file.

DOS-format-write

If the **'fileformat'** is **"dos"**, **<CR><NL>** is used for **<EOL>**. This is default for Win32. On other systems the message **"[dos format]"** is shown to remind you

that an unusual `<EOL>` was used.

Unix-format-write

If the `'fileformat'` is "unix", `<NL>` is used for `<EOL>`. On Win32 the message "[unix format]" is shown.

Mac-format-write

If the `'fileformat'` is "mac", `<CR>` is used for `<EOL>`. On non-Mac systems the message "[mac format]" is shown.

See also `file-formats` and the `'fileformat'` and `'fileformats'` options.

ACL

ACL stands for Access Control List. It is an advanced way to control access rights for a file. It is used on new MS-Windows and Unix systems, but only when the filesystem supports it.

Vim attempts to preserve the ACL info when writing a file. The backup file will get the ACL info of the original file.

The ACL info is also used to check if a file is read-only (when opening the file).

xattr E1506 E1508 E1509

xattr stands for Extended Attributes. It is an advanced way to save metadata alongside the file in the filesystem. It depends on the actual filesystem being used and Vim supports it only on a Linux system.

Vim attempts to preserve the extended attribute info when writing a file. The backup file will get the extended attribute of the original file.

read-only-share

When MS-Windows shares a drive on the network it can be marked as read-only. This means that even if the file read-only attribute is absent, and the ACL settings on NT network shared drives allow writing to the file, you can still not write to the file. Vim on Win32 platforms will detect read-only network drives and will mark the file as read-only. You will not be able to override it with `:write`.

write-device

When the file name is actually a device name, Vim will not make a backup (that would be impossible). You need to use "!", since the device already exists.

Example for Unix:

```
:w! /dev/lpt0
```

and for MS-Windows:

```
:w! lpt0
```

For Unix a device is detected when the name doesn't refer to a normal file or a directory. A fifo or named pipe also looks like a device to Vim.

For MS-Windows the device is detected by its name:

```
AUX
CON
CLOCK$
NUL
PRN
COMn    n=1,2,3... etc
LPTn    n=1,2,3... etc
```

The names can be in upper- or lowercase.

=====

5. Writing and quitting

`write-quit`

`:q` `:quit`

`:q[uit]` Quit the current window. Quit Vim if this is the last `edit-window`. This fails when changes have been made and Vim refuses to `abandon` the current buffer, and when the last file in the argument list has not been edited.

If there are other tab pages and quitting the last window in the current tab page the current tab page is closed `tab-page`.

Triggers the `QuitPre` autocommand event.

See `CTRL-W_q` for quitting another window.

`:conf[irm] q[uit]` Quit, but give prompt when changes have been made, or the last file in the argument list has not been edited. See `:confirm` and `'confirm'`.

`:q[uit]!` Quit without writing, also when the current buffer has changes. The buffer is unloaded, also when it has `'hidden'` set.

If this is the last window and there is a modified hidden buffer, the current buffer is abandoned and the first changed hidden buffer becomes the current buffer.

Use `":qall!"` to exit always.

`:cq[uit]` Quit always, without writing, and return an error code. See `:cq`. Used for Manx's QuickFix mode (see `quickfix`).

`:wq`

`:wq` `[++opt]` Write the current file and close the window. If this was the last `edit-window` Vim quits. Writing fails when the file is read-only or the buffer does not have a name. Quitting fails when the last file in the argument list has not been edited.

`:wq!` `[++opt]` Write the current file and close the window. If this was the last `edit-window` Vim quits. Writing fails when the current buffer does not have a name.

`:wq` `[++opt] {file}` Write to `{file}` and close the window. If this was the last `edit-window` Vim quits. Quitting fails when the last file in the argument list has not been edited.

`:wq!` `[++opt] {file}` Write to `{file}` and close the current window. Quit Vim if this was the last `edit-window`.

`:[range]wq[!]` `[++opt] [file]` Same as above, but only write the lines in `[range]`.

`:x` `:xit`

`:[range]x[it][!]` `[++opt] [file]`

Like `":wq"`, but write only when changes have been made.
When `'hidden'` is set and there are more windows, the current buffer becomes hidden, after writing the file.
This command is not supported in `Vim9` script, because it is too easily confused with a variable name.

`:exi` `:exit`

`:[range]exi[t][!]` `[++opt]` `[file]`
Same as `:xit`.

`ZZ`

`ZZ` Write current file, if modified, and close the current window (same as `":x"`).
If there are several windows for the current file, only the current window is closed.

`ZQ`

`ZQ` Quit without checking for changes (same as `":q!"`).

MULTIPLE WINDOWS AND BUFFERS

`window-exit`

`:qa` `:qall`

`:qa[ll]` Exit Vim, unless there are some buffers which have been changed. (Use `":bmod"` to go to the next modified buffer).
When `'autowriteall'` is set all changed buffers will be written, like `:wqall`.

`:conf[irm] qa[ll]`
Exit Vim. Bring up a prompt when some buffers have been changed. See `:confirm`.

`:qa[ll]!` Exit Vim. Any changes to buffers are lost.
Also see `:cquit`, it does the same but exits with a non-zero value.

`:quita` `:quitall`

`:quita[ll][!]` Same as `":qall"`.

`:wqa[ll]` `[++opt]` `:wqa` `:wqall` `:xa` `:xall`
`:xa[ll]` Write all changed buffers and exit Vim. If there are buffers without a file name, which are readonly or which cannot be written for another reason, Vim will not quit.

`:conf[irm] wqa[ll]` `[++opt]`
`:conf[irm] xa[ll]`
Write all changed buffers and exit Vim. Bring up a prompt when some buffers are readonly or cannot be written for another reason. See `:confirm`.

`:wqa[ll]!` `[++opt]`
`:xa[ll]!` Write all changed buffers, even the ones that are readonly,

and exit Vim. If there are buffers without a file name or which cannot be written for another reason, or there is a terminal with a running job, Vim will not quit.

6. Dialogs

edit-dialogs

`:confirm` `:conf`
`:conf[irm] {command}` Execute `{command}`, and use a dialog when an operation has to be confirmed. Can be used on the `:q`, `:qa` and `:w` commands (the latter to override a read-only setting), and any other command that can fail in such a way, such as `:only`, `:buffer`, `:bdelete`, etc.

Examples:

`:confirm w foo`

Will ask for confirmation when "foo" already exists.

`:confirm q`

Will ask for confirmation when there are changes.

`:confirm qa`

If any modified, unsaved buffers exist, you will be prompted to save or abandon each one. There are also choices to "save all" or "abandon all".

If you want to always use `:confirm`, set the `'confirm'` option.

`:browse` `:bro` E338
`:bro[wse] {command}` Open a file selection dialog for an argument to `{command}`. At present this works for `:e`, `:w`, `:wall`, `:wq`, `:wqall`, `:x`, `:xall`, `:exit`, `:view`, `:sview`, `:r`, `:saveas`, `:sp`, `:mkexrc`, `:mkvimrc`, `:mksession`, `:mkview`, `:split`, `:vsplit`, `:tabe`, `:tabnew`, `:cfile`, `:cgetfile`, `:caddfile`, `:lfile`, `:lgetfile`, `:laddfile`, `:diffsplit`, `:diffpatch`, `:open`, `:pedit`, `:redir`, `:source`, `:update`, `:visual`, `:vsplit`, and `:qall` if `'confirm'` is set.
{only in Win32, Motif, GTK and Mac GUI, in console `browse edit` works if the FileExplorer autocommand group exists}
When `:browse` is not possible you get an error message. If the `+browse` feature is missing or the `{command}` doesn't support browsing, the `{command}` is executed without a dialog.
`:browse set` works like `:options`.
See also `:oldfiles` for `:browse oldfiles`.

The syntax is best shown via some examples:

`:browse e $vim/foo`

Open the browser in the `$vim/foo` directory, and edit the file chosen.

`:browse e`

Open the browser in the directory specified with `'browsedir'`,

and edit the file chosen.

`:browse w`

Open the browser in the directory of the current buffer, with the current buffer filename as default, and save the buffer under the filename chosen.

`:browse w C:/bar`

Open the browser in the C:/bar directory, with the current buffer filename as default, and save the buffer under the filename chosen.

Also see the `'browsedir'` option.

For versions of Vim where browsing is not supported, the command is executed unmodified.

`browsefilter`

For MS-Windows and GTK, you can modify the filters that are used in the browse dialog. By setting the `g:browsefilter` or `b:browsefilter` variables, you can change the filters globally or locally to the buffer. The variable is set to a string in the format `"{filter label}\t{pattern};{pattern}\n"` where `{filter label}` is the text that appears in the "Files of Type" comboBox, and `{pattern}` is the pattern which filters the filenames. Several patterns can be given, separated by `';`.

For Motif the same format is used, but only the very first pattern is actually used (Motif only offers one pattern, but you can edit it).

For example, to have only Vim files in the dialog, you could use the following command:

```
let g:browsefilter = "Vim Scripts\t*.vim\nVim Startup Files\t*vimrc\n"
```

You can override the filter setting on a per-buffer basis by setting the `b:browsefilter` variable. You would most likely set `b:browsefilter` in a filetype plugin, so that the browse dialog would contain entries related to the type of file you are currently editing. Disadvantage: This makes it difficult to start editing a file of a different type. To overcome this, you may want to add "All Files (*.*)\t*\n" as the final filter on Windows or "All Files (*)\t*\n" on other platforms, so that the user can still access any desired file.

To avoid setting `browsefilter` when Vim does not actually support it, you can use `has("browsefilter")`:

```
if has("browsefilter")
  let g:browsefilter = "whatever"
endif
```

7. The current directory `current-directory`

You can use the `:cd`, `:tcd` and `:lcd` commands to change to another directory, so you will not have to type that directory name in front of the file names. It also makes a difference for executing external commands, e.g. `!:ls`.

Changing directory fails when the current buffer is modified, the '.' flag is present in '**coptions**' and '!' is not used in the command.

<code>:cd[!]</code>	<code>:cd</code> E747 E472 On non-Unix systems when ' cdhome ' is off: Print the current directory name. Otherwise: Change the current directory to the home directory. Clear any window-local directory. Use <code>:pwd</code> to print the current directory on all systems.
<code>:cd[!] {path}</code>	Change the current directory to <code>{path}</code> . If <code>{path}</code> is relative, it is searched for in the directories listed in ' cdpath '. Clear any window-local directory. Does not change the meaning of an already opened file, because its full path name is remembered. Files from the <code>arglist</code> may change though! On MS-Windows this also changes the active drive. To change to the directory of the current file: <code>:cd %:h</code>
<code>:cd[!] -</code>	<code>:cd-</code> E186 Change to the previous current directory (before the previous " <code>:cd {path}</code> " command).
<code>:chd[ir][!] {path}</code>	<code>:chd</code> <code>:chdir</code> Same as <code>:cd</code> .
<code>:tc[d][!] {path}</code>	<code>:tc</code> <code>:tcd</code> Like <code>:cd</code> , but only set the directory for the current tab. The current window will also use this directory. The current directory is not changed for windows in other tabs and for windows in the current tab that have their own window-local directory.
<code>:tc[d][!] -</code>	<code>:tcd-</code> Change to the previous current directory, before the last " <code>:tcd {path}</code> " command.
<code>:tch[dir][!]</code>	<code>:tch</code> <code>:tchdir</code> Same as <code>:tcd</code> .
<code>:lc[d][!] {path}</code>	<code>:lc</code> <code>:lcd</code> Like <code>:cd</code> , but only set the current directory when the cursor is in the current window. The current directory for other windows is not changed, switching to another window will stop using <code>{path}</code> .
<code>:lcd[!] -</code>	<code>:lcd-</code> Change to the previous current directory, before the last " <code>:lcd {path}</code> " command.
	<code>:lch</code> <code>:lchdir</code>

```

:lcd[dir][!]          Same as :lcd .

                                :pw  :pwd  E187
:pw[d]               Print the current directory name.
                    Also see getcwd() .

                                :pwd-verbose
When 'verbose' is non-zero, :pwd will also display
what scope the current directory was set. Example:

    " Set by :cd
    :verbose pwd
    [global] /path/to/current

    " Set by :lcd
    :verbose pwd
    [window] /path/to/current

    " Set by :tcd
    :verbose pwd
    [tabpage] /path/to/current

```

So long as no `:lcd` or `:tcd` command has been used, all windows share the same current directory. Using a command to jump to another window doesn't change anything for the current directory.

When a `:lcd` command has been used for a window, the specified directory becomes the current directory for that window. Windows where the `:lcd` command has not been used stick to the global or tab-local current directory. When jumping to another window the current directory is changed to the last specified local current directory. If none was specified, the global or tab-local current directory is used. When creating a new window it inherits the local directory of the current window.

When a `:tcd` command has been used for a tab page, the specified directory becomes the current directory for the current tab page and the current window. The current directory of other tab pages is not affected. When jumping to another tab page, the current directory is changed to the last specified local directory for that tab page. If the current tab has no local current directory the global current directory is used.

When a `:cd` command is used, the current window and tab page will lose the local current directory and will use the global current directory from now on.

After using `:cd` the full path name will be used for reading and writing files. On some networked file systems this may cause problems. The result of using the full path name is that the file names currently in use will remain referring to the same file. Example: If you have a file `a:test` and a directory `a:vim` the commands `":e test" ":cd vim" ":w"` will overwrite the file `a:test` and not write `a:vim/test`. But if you do `":w test"` the file `a:vim/test` will be written, because you gave a new file name and did not refer to a filename before the `":cd"`.

```

=====
8. Editing binary files                                edit-binary

```

Although Vim was made to edit text files, it is possible to edit binary files. The `-b` Vim argument (b for binary) makes Vim do file I/O in binary mode, and sets some options for editing binary files ('binary' on, 'textwidth' to 0, 'modeline' off, 'expandtab' off). Setting the 'binary' option has the same effect. Don't forget to do this before reading the file.

There are a few things to remember when editing binary files:

- When editing executable files the number of bytes must not change. Use only the "R" or "r" command to change text. Do not delete characters with "x" or by backspacing.
- Set the 'textwidth' option to 0. Otherwise lines will unexpectedly be split in two.
- When there are not many <EOL>s, the lines will become very long. If you want to edit a line that does not fit on the screen reset the 'wrap' option. Horizontal scrolling is used then. If a line becomes too long (more than about 32767 bytes on the Amiga, much more on 32-bit and 64-bit systems, see [limits](#)) you cannot edit that line. The line will be split when reading the file. It is also possible that you get an "out of memory" error when reading the file.
- Make sure the 'binary' option is set BEFORE loading the file. Otherwise both <CR><NL> and <NL> are considered to end a line and when the file is written the <NL> will be replaced with <CR><NL>.
- <Nul> characters are shown on the screen as ^@. You can enter them with "**CTRL-V** CTRL-@" or "**CTRL-V** 000"
- To insert a <NL> character in the file split a line. When writing the buffer to a file a <NL> will be written for the <EOL>.
- Vim normally appends an <EOL> at the end of the file if there is none. Setting the 'binary' option prevents this. If you want to add the final <EOL>, set the 'endofline' option. You can also read the value of this option to see if there was an <EOL> for the last line (you cannot see this in the text).

9. Encryption encryption

Vim is able to write files encrypted, and read them back. The encrypted text cannot be read without the right key.

{only available when compiled with the |+cryptv| feature} E833

The text in the swap file and the undo file is also encrypted. E843

However, this is done block-by-block and may reduce the time needed to crack a password. You can disable the swap file, but then a crash will cause you to lose your work. The undo file can be disabled without too much disadvantage.

```
:set noundofile
:noswapfile edit secrets
```

Note: The text in memory is not encrypted. A system administrator may be able to see your text while you are editing it. When filtering text with ":%!filter" or using ":w !command" the text is also not encrypted, this may reveal it to others. The 'viminfo' file is not encrypted.

You could do this to edit very secret text:

```
:set noundofile viminfo=
```



```
:noswapfile edit secrets.txt
```

Keep in mind that without a swap file you risk losing your work in the event of a crash or a power failure.

WARNING: If you make a typo when entering the key and then write the file and exit, the text will be lost!

The normal way to work with encryption, is to use the ":X" command, which will ask you to enter a key. A following write command will use that key to encrypt the file. If you later edit the same file, Vim will ask you to enter a key. If you type the same key as that was used for writing, the text will be readable again. If you use a wrong key, it will be a mess.

```
:X
```

Prompt for an encryption key. The typing is done without showing the actual text, so that someone looking at the display won't see it. The typed key is stored in the **'key'** option, which is used to encrypt the file when it is written. The file will remain unchanged until you write it. **Note** that commands such as ``:xit`` and ``ZZ`` will NOT write the file unless there are other changes. See also `-x` .

The value of the **'key'** options is used when text is written. When the option is not empty, the written file will be encrypted, using the value as the encryption key. A magic number is prepended, so that Vim can recognize that the file is encrypted.

To disable the encryption, reset the **'key'** option to an empty value:

```
:set key=
```

You can use the **'cryptmethod'** option to select the type of encryption, use one of these:

```
:setlocal cm=zip           " weak method, backwards compatible
:setlocal cm=blowfish      " method with flaws, do not use
:setlocal cm=blowfish2    " medium strong method
:setlocal cm=xchacha20v2  " medium strong method using libsodium
```

Do this before writing the file. When reading an encrypted file it will be set automatically to the method used when that file was written. You can change **'cryptmethod'** before writing that file to change the method.

To set the default method, used for new files, use this in your `vimrc` file:

```
set cm=blowfish2
```

Using "blowfish2" is highly recommended. Only use another method if you must use an older **Vim version that** does not support it.

The message given for reading and writing a file will show "[crypted]" when using zip, "[blowfish]" when using blowfish, etc.

When writing an undo file, the same key and method will be used for the text in the undo file. `persistent-undo` .

To test for blowfish support you can use these conditions:

```
has('crypt-blowfish')
has('crypt-blowfish2')
```

This works since Vim 7.4.1099 while blowfish support was added earlier. Thus the condition failing doesn't mean blowfish is not supported. You can test for blowfish with:

```
v:version >= 703
```

And for blowfish2 with:

```
v:version > 704 || (v:version == 704 && has('patch401'))
```

If you are sure Vim includes patch 7.4.237 a simpler check is:

```
has('patch-7.4.401')
```

E817 E818 E819 E820

When encryption does not work properly, you would be able to write your text to a file and never be able to read it back. Therefore a test is performed to check if the encryption works as expected. If you get one of these errors don't write the file encrypted! You need to rebuild the Vim binary to fix this.

E831 This is an internal error, "cannot happen". If you can reproduce it, please report to the developers.

When reading a file that has been encrypted and the **'key'** option is not empty, it will be used for decryption. If the value is empty, you will be prompted to enter the key. If you don't enter a key, or you enter the wrong key, the file is edited without being decrypted. There is no warning about using the wrong key (this makes brute force methods to find the key more difficult).

If want to start reading a file that uses a different key, set the **'key'** option to an empty string, so that Vim will prompt for a new one. Don't use the **":set"** command to enter the value, other people can read the command over your shoulder.

Since the value of the **'key'** option is supposed to be a secret, its value can never be viewed. You should not set this option in a vimrc file.

An encrypted file can be recognized by the "file" command, if you add these lines to "/etc/magic", "/usr/share/misc/magic" or wherever your system has the "magic" file:

```
0 string VimCrypt~ Vim encrypted file
>9 string 01 - "zip" cryptmethod
>9 string 02 - "blowfish" cryptmethod
>9 string 03 - "blowfish2" cryptmethod
```

Notes:

- Encryption is not possible when doing conversion with **'charconvert'**.
- Text you copy or delete goes to the numbered registers. The registers can be saved in the .viminfo file, where they could be read. Change your **'viminfo'** option to be safe.
- Someone can type commands in Vim when you walk away for a moment, he should not be able to get the key.
- If you make a typing mistake when entering the key, you might not be able to get your text back!
- If you type the key with a **":set key=value"** command, it can be kept in the

- history, showing the **'key'** value in a viminfo file.
- There is never 100% safety. The encryption in Vim has not been tested for robustness.
 - The algorithm used for **'cryptmethod'** "zip" is breakable. A 4 character key in about one hour, a 6 character key in one day (on a Pentium 133 PC). This requires that you know some text that must appear in the file. An expert can break it for any key. When the text has been decrypted, this also means that the key can be revealed, and other files encrypted with the same key can be decrypted.
 - Pkzip uses the same encryption as **'cryptmethod'** "zip", and US Govt has no objection to its export. Pkzip's public file APPNOTE.TXT describes this algorithm in detail.
 - The implementation of **'cryptmethod'** "blowfish" has a flaw. It is possible to crack the first 64 bytes of a file and in some circumstances more of the file. Use of it is not recommended, but it's still the strongest method supported by Vim 7.3 and 7.4. The "zip" method is even weaker.
 - Vim originates from the Netherlands. That is where the sources come from. Thus the encryption code is not exported from the USA.

10. Timestamps

timestamp timestamps

Vim remembers the modification timestamp, mode and size of a file when you begin editing it. This is used to avoid that you have two different versions of the same file (without you knowing this).

After a shell command is run (`:!cmd suspend :read! K`) timestamps, file modes and file sizes are compared for all buffers in a window. Vim will run any associated `FileChangedShell` autocommands or display a warning for any files that have changed. In the GUI this happens when Vim regains input focus.

E321 E462

If you want to automatically reload a file when it has been changed outside of Vim, set the **'autoread'** option. This doesn't work at the moment you write the file though, only when the file wasn't changed inside of Vim.

ignore-timestamp

If you do not want to be asked or automatically reload the file, you can use this:

```
set buftype=nofile
```

Or, when starting gvim from a shell:

```
gvim file.log -c "set buftype=nofile"
```

Note that if a `FileChangedShell` autocommand is defined you will not get a warning message or prompt. The autocommand is expected to handle this.

There is no warning for a directory (e.g., with `netrw-browse`). But you do get warned if you started editing a new file and it was created as a directory later.

When Vim notices the timestamp of a file has changed, and the file is being edited in a buffer but has not changed, Vim checks if the contents of the file is equal. This is done by reading the file again (into a hidden buffer, which

is immediately deleted again) and comparing the text. If the text is equal, you will get no warning.

If you don't get warned often enough you can use the following command.

```
                :checkt    :checktime
:checkt[ime]    Check if any buffers were changed outside of Vim.
                This checks and warns you if you would end up with two
                versions of a file.
                If this is called from an autocommand, a ":global"
                command or is not typed the actual check is postponed
                until a moment the side effects (reloading the file)
                would be harmless.
                Each loaded buffer is checked for its associated file
                being changed. If the file was changed Vim will take
                action. If there are no changes in the buffer and
                'autoread' is set, the buffer is reloaded. Otherwise,
                you are offered the choice of reloading the file. If
                the file was deleted you get an error message.
                If the file previously didn't exist you get a warning
                if it exists now.
                Once a file has been checked the timestamp is reset,
                you will not be warned again.
                Syntax highlighting, marks, diff status,
                'fileencoding', 'fileformat' and 'binary' options
                are not changed. See v:fcs_choice to reload these
                too (for example, if a code formatting tools has
                changed the file).
```

```
:[N]checkt[ime] {filename}
:[N]checkt[ime] [N]
```

Check the timestamp of a specific buffer. The buffer may be specified by name, number or with a pattern.

E813 E814

Vim will reload the buffer if you chose to. If a window is visible that contains this buffer, the reloading will happen in the context of this window. Otherwise a special window is used, so that most autocommands will work. You can't close this window. A few other restrictions apply. Best is to make sure nothing happens outside of the current buffer. E.g., setting window-local options may end up in the wrong window. Splitting the window, doing something there and closing it should be OK (if there are no side effects from other autocommands). Closing unrelated windows and buffers will get you into trouble.

Before writing a file the timestamp is checked. If it has changed, Vim will ask if you really want to overwrite the file:

```
WARNING: The file has been changed since reading it!!!
Do you really want to write to it (y/n)?
```

If you hit 'y' Vim will continue writing the file. If you hit 'n' the write is aborted. If you used ":wq" or "ZZ" Vim will not exit, you will get another

chance to write the file.

The message would normally mean that somebody has written to the file after the edit session started. This could be another person, in which case you probably want to check if your changes to the file and the changes from the other person should be merged. Write the file under another name and check for differences (the "diff" program can be used for this).

It is also possible that you modified the file yourself, from another edit session or with another command (e.g., a filter command). Then you will know which version of the file you want to keep.

The accuracy of the time check depends on the filesystem. On Unix it is usually sub-second. With old file systems and on MS-Windows it is normally one second. Use ``has('nanotime')`` to check if sub-second time stamp checks are available.

There is one situation where you get the message while there is nothing wrong: On a Win32 system on the day daylight saving time starts. There is something in the Win32 libraries that confuses Vim about the hour time difference. The problem goes away the next day.

11. File Searching

file-searching

The file searching is currently used for the `'path'`, `'cdpath'` and `'tags'` options, for `finddir()` and `findfile()`. Other commands use `wildcards` which is slightly different.

There are three different types of searching:

1) Downward search:

starstar

Downward search uses the wildcards `'*'`, `'**'` and possibly others supported by your operating system. `'*'` and `'**'` are handled inside Vim, so they work on all operating systems. **Note** that `"**"` only acts as a special wildcard when it is at the start of a name.

The usage of `'*'` is quite simple: It matches 0 or more characters. In a search pattern this would be `".*"`. **Note** that the `"."` is not used for file searching.

`'**'` is more sophisticated:

- It ONLY matches directories.
- It matches up to 30 directories deep by default, so you can use it to search an entire directory tree
- The maximum number of levels matched can be given by appending a number to `'**'`.

Thus `'/usr/**2'` can match:

```
/usr
/usr/include
/usr/include/sys
/usr/include/g++
/usr/lib
/usr/lib/X11
```

.....
It does NOT match '/usr/include/g++/std' as this would be three levels.
The allowed number range is 0 ('**0' is removed) to 100
If the given number is smaller than 0 it defaults to 30, if it's bigger than 100 then 100 is used. The system also has a limit on the path length, usually 256 or 1024 bytes.
- '**' can only be at the end of the path or be followed by a path separator or by a number and a path separator.

You can combine '*' and '**' in any order:

```
/usr/**/sys/*  
/usr/*tory/sys/**  
/usr/**2/sys/*
```

2) Upward search:

Here you can give a directory and then search the directory tree upward for a file. You could give stop-directories to limit the upward search. The stop-directories are appended to the path (for the '**path**' option) or to the filename (for the '**tags**' option) with a ';'. If you want several stop-directories separate them with ';'. If you want no stop-directory ("search upward till the root directory) just use ';'.

```
/usr/include/sys;/usr
```

will search in:

```
/usr/include/sys  
/usr/include  
/usr
```

If you use a relative path the upward search is started in Vim's current directory or in the directory of the current file (if the relative path starts with './' and 'd' is not included in '**coptions**').

If Vim's current path is /u/user_x/work/release and you do

```
:set path=include;/u/user_x
```

and then search for a file with **gf** the file is searched in:

```
/u/user_x/work/release/include  
/u/user_x/work/include  
/u/user_x/include
```

Note: If your '**path**' setting includes a non-existing directory, Vim will skip the non-existing directory, and also does not search in the parent of the non-existing directory if upwards searching is used.

3) Combined up/downward search:

If Vim's current path is /u/user_x/work/release and you do

```
set path=**;/u/user_x
```

and then search for a file with **gf** the file is searched in:

```
/u/user_x/work/release/**  
/u/user_x/work/**  
/u/user_x/**
```

BE CAREFUL! This might consume a lot of time, as the search of '/u/user_x/**' includes '/u/user_x/work/**' and '/u/user_x/work/release/**'. So '/u/user_x/work/release/**' is searched

three times and '/u/user_x/work/**' is searched twice.

In the above example you might want to set path to:

```
:set path=**,/u/user_x/**
```

This searches:

```
/u/user_x/work/release/**
```

```
/u/user_x/**
```

This searches the same directories, but in a different order.

Note that completion for ":", ":", and ":" commands do not currently work with '**path**' items that contain a URL or use the double star with depth limiter (/usr/**2) or upward search (;) notations.

Cursor motions

cursor-motions navigation

These commands move the cursor position. If the new position is off of the screen, the screen is scrolled to show the cursor (see also **'scrolljump'** and **'scrolloff'** options).

- | | |
|--------------------------|--------------------|
| 1. Motions and operators | operator |
| 2. Left-right motions | left-right-motions |
| 3. Up-down motions | up-down-motions |
| 4. Word motions | word-motions |
| 5. Text object motions | object-motions |
| 6. Text object selection | object-select |
| 7. Marks | mark-motions |
| 8. Jumps | jump-motions |
| 9. Various motions | various-motions |

General remarks:

If you want to know where you are in the file use the **"CTRL-G"** command **CTRL-G** or the "g **CTRL-G**" command **g_CTRL-G**. If you set the **'ruler'** option, the cursor position is continuously shown in the status line (which slows down Vim a little).

Experienced users prefer the hjkl keys because they are always right under their fingers. Beginners often prefer the arrow keys, because they do not know what the hjkl keys do. The mnemonic value of hjkl is clear from looking at the keyboard. Think of j as an arrow pointing downwards.

The **'virtualedit'** option can be set to make it possible to move the cursor to positions where there is no character or within a multi-column character (like a tab).

=====

operator

The motion commands can be used after an operator command, to have the command operate on the text that was moved over. That is the text between the cursor position before and after the motion. Operators are generally used to delete or change text. The following operators are available:

c	c	change
d	d	delete
y	y	yank into register (does not change the text)
~	~	swap case (only if 'tildeop' is set)
g~	g~	swap case
gu	gu	make lowercase
gU	gU	make uppercase
!	!	filter through an external program

=	=	filter through 'equalprg' or C-indenting if empty
gq	gq	text formatting
gw	gw	text formatting with no cursor movement
g?	g?	ROT13 encoding
>	>	shift right
<	<	shift left
zf	zf	define a fold
g@	g@	call function set with the 'operatorfunc' option

motion-count-multiplied

If the motion includes a count and the operator also had a count before it, the two counts are multiplied. For example: "2d3w" deletes six words.

operator-doubled

When doubling the operator it operates on a line. When using a count, before or after the first character, that many lines are operated upon. Thus `3dd` deletes three lines. A count before and after the first character is multiplied, thus `2y3y` yanks six lines.

After applying the operator the cursor is mostly left at the start of the text that was operated upon. For example, "yfe" doesn't move the cursor, but "yFe" moves the cursor leftwards to the "e" where the yank started.

linewise characterwise

The operator either affects whole lines, or the characters between the start and end position. Generally, motions that move between lines affect lines (are linewise), and motions that move within a line affect characters (are characterwise). However, there are some exceptions.

exclusive inclusive

A character motion is either inclusive or exclusive. When inclusive, the start and end position of the motion are included in the operation. When exclusive, the last character towards the end of the buffer is not included. Linewise motions always include the start and end position.

Which motions are linewise, inclusive or exclusive is mentioned with the command. There are however, two general exceptions:

1. If the motion is exclusive and the end of the motion is in column 1, the end of the motion is moved to the end of the previous line and the motion becomes inclusive. Example: "}" moves to the first line after a paragraph, but "d}" will not include that line.

exclusive-linewise

2. If the motion is exclusive, the end of the motion is in column 1 and the start of the motion was at or before the first non-blank in the line, the motion becomes linewise. Example: If a paragraph begins with some blanks and you do "d}" while standing on the first non-blank, all the lines of the paragraph are deleted, including the blanks. If you do a put now, the deleted lines will be inserted below the cursor position.

Note that when the operator is pending (the operator command is typed, but the motion isn't yet), a special set of mappings can be used. See **:omap** .

Instead of first giving the operator and then a motion you can use Visual mode: mark the start of the text with "v", move the cursor to the end of the text that is to be affected and then hit the operator. The text between the start and the cursor position is highlighted, so you can see what text will

be operated upon. This allows much more freedom, but requires more key strokes and has limited redo functionality. See the chapter on Visual mode `Visual-mode` .

You can use a ":" command for a motion. For example "d:call FindEnd()". But this can't be repeated with "." if the command is more than one line. This can be repeated:

```
d:call search("f")<CR>
```

This cannot be repeated:

```
d:if 1<CR>
  call search("f")<CR>
endif<CR>
```

Note that when using ":" any motion becomes characterwise exclusive.

forced-motion

FORCING A MOTION TO BE LINEWISE, CHARACTERWISE OR BLOCKWISE

When a motion is not of the type you would like to use, you can force another type by using "v", "V" or **CTRL-V** just after the operator.

Example:

```
dj
```

deletes two lines

```
dvj
```

deletes from the cursor position until the character below the cursor

```
d<C-V>j
```

deletes the character under the cursor and the character below the cursor.

Be careful with forcing a linewise movement to be used characterwise or blockwise, the column may not always be defined.

o_v

v When used after an operator, before the motion command: Force the operator to work characterwise, also when the motion is linewise. If the motion was linewise, it will become **exclusive** . If the motion already was characterwise, toggle inclusive/exclusive. This can be used to make an exclusive motion inclusive and an inclusive motion exclusive.

o_V

V When used after an operator, before the motion command: Force the operator to work linewise, also when the motion is characterwise.

o_CTRL-V

CTRL-V When used after an operator, before the motion command: Force the operator to work blockwise. This works like Visual block mode selection, with the corners defined by the cursor position before and after the motion.

2. Left-right motions **left-right-motions**

These commands move the cursor to the specified column in the current line.

They stop at the first column and at the end of the line, except "\$", which may move to one of the next lines. See **'whichwrap'** option to make some of the commands move across line boundaries.

h	or	h
<Left>	or	<Left>
CTRL-H	or	CTRL-H <BS>
<BS>		
		[count] characters to the left. exclusive motion.
		Note: If you prefer <BS> to delete a character, use the mapping:
		:map CTRL-V <BS> X
		(to enter " CTRL-V <BS>" type the CTRL-V key, followed by the <BS> key)
		See :fixdel if the <BS> key does not do what you want.
l	or	l
<Right>	or	<Right> <Space>
<Space>		
		[count] characters to the right. exclusive motion.
		See the 'whichwrap' option for adjusting the behavior at end of line
0		0
		To the first character of the line. exclusive motion.
<Home>		<Home> <kHome>
		To the first character of the line. exclusive motion. When moving up or down next, stay in same TEXT column (if possible). Most other commands stay in the same SCREEN column. <Home> works like "1 ", which differs from "0" when the line starts with a <Tab>.
^		^
		To the first non-blank character of the line. exclusive motion. Any count is ignored.
\$ or <End>		\$ <End> <kEnd>
		To the end of the line. When a count is given also go [count - 1] lines downward, or as far is possible. inclusive motion. If a count of 2 or larger is given and the cursor is on the last line, that is an error and the cursor doesn't move. In Visual mode the cursor goes to just after the last character in the line. When 'virtualedit' is active, "\$" may move the cursor back from past the end of the line to the last character in the line.
g_		g_
		To the last non-blank character of the line and [count - 1] lines downward inclusive .

g0 or g<Home>	<p style="text-align: right;">g0 g<Home></p> <p>When lines wrap ('wrap' on): To the first character of the screen line. exclusive motion. Differs from "0" when a line is wider than the screen.</p> <p>When lines don't wrap ('wrap' off): To the leftmost character of the current line that is on the screen. Differs from "0" when the first character of the line is not on the screen.</p>
g^	<p style="text-align: right;">g^</p> <p>When lines wrap ('wrap' on): To the first non-blank character of the screen line. exclusive motion. Differs from "^" when a line is wider than the screen.</p> <p>When lines don't wrap ('wrap' off): To the leftmost non-blank character of the current line that is on the screen. Differs from "^" when the first non-blank character of the line is not on the screen.</p>
gm	<p style="text-align: right;">gm</p> <p>Like "g0", but half a screenwidth to the right (or as much as possible).</p>
gM	<p style="text-align: right;">gM</p> <p>Like "g0", but to halfway the text of the line. With a count: to this percentage of text in the line. Thus "10gM" is near the start of the text and "90gM" is near the end of the text.</p>
g\$	<p style="text-align: right;">g\$</p> <p>When lines wrap ('wrap' on): To the last character of the screen line and [count - 1] screen lines downward inclusive . Differs from "\$" when a line is wider than the screen.</p> <p>When lines don't wrap ('wrap' off): To the rightmost character of the current line that is visible on the screen. Differs from "\$" when the last character of the line is not on the screen or when a count is used. Additionally, vertical movements keep the column, instead of going to the end of the line.</p> <p>When 'virtualedit' is enabled moves to the end of the screen line.</p>
g<End>	<p style="text-align: right;">g<End> g<kEnd></p> <p>Like g\$ but to the last non-blank character instead of the last character.</p>
	<p style="text-align: right;">bar</p> <p>To screen column [count] in the current line. exclusive motion. Ceci n'est pas une pipe.</p>
f{char}	<p style="text-align: right;">f</p> <p>To [count]'th occurrence of {char} to the right. The cursor is placed on {char} inclusive . {char} can be entered as a digraph digraph-arg .</p>

When **'encoding'** is set to Unicode, composing characters may be used, see [utf-8-char-arg](#) .
:lmap mappings apply to **{char}**. The **CTRL-^** command in Insert mode can be used to switch this on/off **i_CTRL-^** .

F{char}	To the [count] 'th occurrence of {char} to the left. The cursor is placed on {char} exclusive . {char} can be entered like with the f command.
t{char}	Till before [count] 'th occurrence of {char} to the right. The cursor is placed on the character left of {char} inclusive . {char} can be entered like with the f command.
T{char}	Till after [count] 'th occurrence of {char} to the left. The cursor is placed on the character right of {char} exclusive . {char} can be entered like with the f command.
;	Repeat latest f, t, F or T [count] times. See cpo- ;
,	Repeat latest f, t, F or T in opposite direction [count] times. See also cpo- ;

3. Up-down motions up-down-motions

k <Up> CTRL-P	or or	[count] lines upward linewise .	k <Up> CTRL-P
j <Down> CTRL-J <NL> CTRL-N	or or or or	[count] lines downward linewise .	j <Down> CTRL-J <NL> CTRL-N
gk g<Up>	or	[count] display lines upward. exclusive motion. Differs from 'k' when lines wrap, and when used with an operator, because it's not linewise.	gk g<Up>
gj g<Down>	or	[count] display lines downward. exclusive motion. Differs from 'j' when lines wrap, and when used with an operator, because it's not linewise.	gj g<Down>

-

- <minus> [count] lines upward, on the first non-blank character `linewise` .

+ or +
CTRL-M or **CTRL-M** <CR>
<CR> [count] lines downward, on the first non-blank character `linewise` .

_ <underscore> [count] - 1 lines downward, on the first non-blank character `linewise` .

G **G**
Goto line [count], default last line, on the first non-blank character `linewise` . If '**startofline**' not set, keep the same column. G is one of the `jump-motions` .

<C-End> **<C-End>**
Goto line [count], default last line, on the last character `inclusive` .

<C-Home> or **gg** <C-Home>
gg Goto line [count], default first line, on the first non-blank character `linewise` . If '**startofline**' not set, keep the same column.

:[range] **:[range]**
Set the cursor on the last line number in [range]. [range] can also be just one line number, e.g., ":1" or ":'m". In contrast with **G** this command does not modify the `jumplist` .

{count}% **N%**
Go to {count} percentage in the file, on the first non-blank in the line `linewise` . To compute the new line number this formula is used:
 $(\{count\} * \text{number-of-lines} + 99) / 100$
See also '**startofline**' option.

:[range]go[to] [count] **:go :goto go**
[**count**]go Go to [count] byte in the buffer. Default [count] is one, start of the file. When giving [range], the last number in it used as the byte count. End-of-line characters are counted depending on the current '**fileformat**' setting. Also see the `line2byte()` function, and the 'o' option in '**statusline**'. {not available when compiled without the `+byte_offset` feature}

These commands move to the specified line. They stop when reaching the first or the last line. The first two commands put the cursor in the same column (if possible) as it was after the last command that changed the column,

except after the "\$" command, then the cursor will be put on the last character of the line.

If "k", "-" or **CTRL-P** is used with a [count] and there are less than [count] lines above the cursor and the 'cpo' option includes the "-" flag it is an error. cpo-- .

4. Word motions

word-motions

<S-Right> or <S-Right> w
w [count] words forward. exclusive motion.

<C-Right> or <C-Right> W
W [count] WORDS forward. exclusive motion.
If <C-Right> does not work, check out
arrow_modifiers .

e Forward to the end of word [count] inclusive .
Does not stop in an empty line.

E Forward to the end of WORD [count] inclusive .
Does not stop in an empty line.

<S-Left> or <S-Left> b
b [count] words backward. exclusive motion.

<C-Left> or <C-Left> B
B [count] WORDS backward. exclusive motion.
If <C-Left> does not work, check out
arrow_modifiers .

ge Backward to the end of word [count] inclusive .

gE Backward to the end of WORD [count] inclusive .

These commands move over words or WORDS.

word

A word consists of a sequence of letters, digits and underscores, or a sequence of other non-blank characters, separated with white space (spaces, tabs, <EOL>). This can be changed with the 'iskeyword' option. An empty line is also considered to be a word.

WORD

A WORD consists of a sequence of non-blank characters, separated with white space. An empty line is also considered to be a WORD.

A sequence of folded lines is counted for one word of a single character. "w" and "W", "e" and "E" move to the start/end of the first word or WORD after a range of folded lines. "b" and "B" move to the start of the first word or WORD before the fold.

Special case: "cw" and "cW" are treated like "ce" and "cE" if the cursor is on a non-blank. This is because "cw" is interpreted as change-word, and a word does not include the following white space.

Another special case: When using the "w" motion in combination with an operator and the last word moved over is at the end of a line, the end of that word becomes the end of the operated text, not the first word in the next line.

The original Vi implementation of "e" is buggy. For example, the "e" command will stop on the first character of a line if the previous line was empty. But when you use "2e" this does not happen. In Vim "ee" and "2e" are the same, which is more logical. However, this causes a small incompatibility between Vi and Vim.

5. Text object motions object-motions

- ([count] sentence s backward. (exclusive motion.
-) [count] sentence s forward.) exclusive motion.
- { [count] paragraph s backward. { exclusive motion.
- } [count] paragraph s forward. } exclusive motion.
-]] [count] section s forward or to the next '{' in the first column. When used after an operator, then also stops below a '}' in the first column.]] exclusive
Note that exclusive-linewise often applies.
-][[count] section s forward or to the next '}' in the first column.][exclusive
Note that exclusive-linewise often applies.
- [[[count] section s backward or to the previous '{' in the first column. [[exclusive
Note that exclusive-linewise often applies.
- [] [count] section s backward or to the previous '}' in the first column. [] exclusive
Note that exclusive-linewise often applies.

These commands move over three kinds of text objects.

sentence

A sentence is defined as ending at a '.', '!' or '?' followed by either the end of a line, or by a space or tab. Any number of closing ')', ']', ''' and ''' characters may appear after the '.', '!' or '?' before the spaces, tabs or end of line. A paragraph and section boundary is also a sentence boundary.

If the 'J' flag is present in **'coptions'**, at least two spaces have to follow the punctuation mark; <Tab>s are not recognized as white space. The definition of a sentence cannot be changed.

paragraph

A paragraph begins after each empty line, and also at each of a set of paragraph macros, specified by the pairs of characters in the **'paragraphs'** option. The default is "IPLPPPQPP TPHPLIPpLpItpplpipbp", which corresponds to the macros ".IP", ".LP", etc. (These are nroff macros, so the dot must be in the first column). A section boundary is also a paragraph boundary.

Note that a blank line (only containing white space) is NOT a paragraph boundary.

Also **note** that this does not include a '{' or '}' in the first column. When the '{' flag is in **'coptions'** then '{' in the first column is used as a paragraph boundary **posix** .

section

A section begins after a form-feed (<C-L>) in the first column and at each of a set of section macros, specified by the pairs of characters in the **'sections'** option. The default is "SHNHH HUnhsh", which defines a section to start at the nroff macros ".SH", ".NH", ".H", ".HU", ".nh" and ".sh".

The "]]" and "[[" commands stop at the '{' in the first column. This is useful to find the start of a function in a C program. To search for a '}' in the first column, the end of a C function, use "]][" (forward) or "[[" (backward). **Note** that the first character of the command determines the search direction.

If your '{' or '}' are not in the first column, and you would like to use "[[" and "]]" anyway, try these mappings:

```
:map [[ ?{<CR>w99[{
:map ][ /}<CR>b99}]
:map ]] j0[[%//{<CR>
:map [] k$][%?}<CR>
```

[type these literally, see <>]

6. Text object selection

object-select **text-objects**
v_a v_i

This is a series of commands that can only be used while in Visual mode or after an operator. The commands that start with "a" select "a"n object including white space, the commands starting with "i" select an "inner" object without white space, or just the white space. Thus the "inner" commands always select less text than the "a" commands.

Also see `gn` and `gN`, operating on the last search pattern.

aw v_aw aw
 "a word", select [count] words (see [word](#)).
 Leading or trailing white space is included, but not counted.
 When used in Visual linewise mode "aw" switches to Visual characterwise mode.

iw v_iw iw
 "inner word", select [count] words (see [word](#)).
 White space between words is counted too.
 When used in Visual linewise mode "iw" switches to Visual characterwise mode.

aW v_aW aW
 "a WORD", select [count] WORDs (see [WORD](#)).
 Leading or trailing white space is included, but not counted.
 When used in Visual linewise mode "aW" switches to Visual characterwise mode.

iW v_iW iW
 "inner WORD", select [count] WORDs (see [WORD](#)).
 White space between words is counted too.
 When used in Visual linewise mode "iW" switches to Visual characterwise mode.

as v_as as
 "a sentence", select [count] sentences (see [sentence](#)).
 When used in Visual mode it is made characterwise.

is v_is is
 "inner sentence", select [count] sentences (see [sentence](#)).
 When used in Visual mode it is made characterwise.

ap v_ap ap
 "a paragraph", select [count] paragraphs (see [paragraph](#)).
 Exception: a blank line (only containing white space) is also a paragraph boundary.
 When used in Visual mode it is made linewise.

ip v_ip ip
 "inner paragraph", select [count] paragraphs (see [paragraph](#)).
 Exception: a blank line (only containing white space) is also a paragraph boundary.
 When used in Visual mode it is made linewise.

a] v_a] v_a[a] a[
 a[
 "a [] block", select [count] '[' ']' blocks. This goes backwards to the [count] unclosed '[' , and finds the matching ']'. The enclosed text is selected,

including the '[' and ']'. The `cpo-M` option flag is used to handle escaped brackets. When used in Visual mode it is made characterwise.

i] v_i v_i[i] i[
i["inner [] block", select `[count]` '[' ']' blocks. This goes backwards to the `[count]` unclosed '[', and finds the matching ']'. The enclosed text is selected, excluding the '[' and ']'. It's an error to select an empty inner block like "[]". The `cpo-M` option flag is used to handle escaped brackets. When used in Visual mode it is made characterwise.

a) v_a) a) a(
a(v_ab v_ab v_a(ab
ab "a block", select `[count]` blocks, from "`[count]` [" to the matching ')", including the '(' and ')' (see `[()`). Does not include white space outside of the parenthesis. The `cpo-M` option flag is used to handle escaped parenthesis. When used in Visual mode it is made characterwise.

i) v_i) i) i(
i(v_ib v_ib v_i(ib
ib "inner block", select `[count]` blocks, from "`[count]` [" to the matching ')", excluding the '(' and ')' (see `[()`). If the cursor is not inside a () block, then find the next "(" . It's an error to select an empty inner block like "()". The `cpo-M` option flag is used to handle escaped parenthesis. When used in Visual mode it is made characterwise.

a> v_a> v_a< a> a<
a< "a <> block", select `[count]` <> blocks, from the `[count]`'th unmatched '<' backwards to the matching '>', including the '<' and '>'. The `cpo-M` option flag is used to handle escaped '<' and '>'. When used in Visual mode it is made characterwise.

i> v_i> v_i< i> i<
i< "inner <> block", select `[count]` <> blocks, from the `[count]`'th unmatched '<' backwards to the matching '>', excluding the '<' and '>'. It's an error to select an empty inner block like "<>". The `cpo-M` option flag is used to handle escaped '<' and '>'. When used in Visual mode it is made characterwise.

at v_at at
"a tag block", select `[count]` tag blocks, from the `[count]`'th unmatched "<aaa>" backwards to the matching "</aaa>", including the "<aaa>" and "</aaa>". See [tag-blocks](#) about the details. When used in Visual mode it is made characterwise. Only available when compiled with the `+eval` feature.

`v_it` `it`
`it` "inner tag block", select `[count]` tag blocks, from the `[count]`'th unmatched "`<aaa>`" backwards to the matching "`</aaa>`", excluding the "`<aaa>`" and "`</aaa>`". See `tag-blocks` about the details. When used in Visual mode it is made characterwise.

`v_a}` `a}` `a{`
`v_aB` `v_a{` `aB`
`a}`
`a{`
`aB` "a Block", select `[count]` Blocks, from "`[count]` [`{`" to the matching '`}`', including the '`{`' and '`}`' (see `[{`). The `cpo-M` option flag is used to handle escaped braces. When used in Visual mode it is made characterwise.

`v_i}` `i}` `i{`
`v_iB` `v_i{` `iB`
`i}`
`i{`
`iB` "inner Block", select `[count]` Blocks, from "`[count]` [`{`" to the matching '`}`', excluding the '`{`' and '`}`' (see `[{`). It's an error to select an empty inner block like "`{}`". The `cpo-M` option flag is used to handle escaped braces. When used in Visual mode it is made characterwise.

`v_aquote` `aquote`
`v_a'` `a'`
`v_a`` `a``
`a"`
`a'`
`a`` "a quoted string". Selects the text from the previous quote until the next quote. The `'quotesescape'` option is used to skip escaped quotes. Only works within one line. When the cursor starts on a quote, Vim will figure out which quote pairs form a string by searching from the start of the line. Any trailing white space is included, unless there is none, then leading white space is included. When used in Visual mode it is made characterwise. Repeating this object in Visual mode another string is included. A count is currently not used.

`v_iquote` `iquote`
`v_i'` `i'`
`v_i`` `i``
`i"`
`i'`
`i`` Like `a"`, `a'` and `a``, but exclude the quotes and repeating won't extend the Visual selection. Special case: With a count of 2 the quotes are included, but no extra white space as with `a"/a'/a``.

`o_object-select`

When used after an operator:
 For non-block objects:

For the `"a` commands: The operator applies to the object and the white space after the object. If there is no white space after the object

or when the cursor was in the white space before the object, the white space before the object is included.

For the "inner" commands: If the cursor was on the object, the operator applies to the object. If the cursor was on white space, the operator applies to the white space.

For a block object:

The operator applies to the block where the cursor is in, or the block on which the cursor is on one of the braces. For the "inner" commands the surrounding braces are excluded. For the "a" commands, the braces are included.

v_object-select

When used in Visual mode:

When start and end of the Visual area are the same (just after typing "v"):

One object is selected, the same as for using an operator.

When start and end of the Visual area are not the same:

For non-block objects the area is extended by one object or the white space up to the next object, or both for the "a" objects. The direction in which this happens depends on which side of the Visual area the cursor is. For the block objects the block is extended one level outwards.

For illustration, here is a list of delete commands, grouped from small to big objects. **Note** that for a single character and a whole line the existing vi movement commands are used.

"dl"	delete character (alias: "x")	dl
"diw"	delete inner word	diw
"daw"	delete a word	daw
"diW"	delete inner WORD (see WORD)	diW
"daW"	delete a WORD (see WORD)	daW
"dgn"	delete the next search pattern match	dgn
"dd"	delete one line	dd
"dis"	delete inner sentence	dis
"das"	delete a sentence	das
"dib"	delete inner '(' ')' block	dib
"dab"	delete a '(' ')' block	dab
"dip"	delete inner paragraph	dip
"dap"	delete a paragraph	dap
"diB"	delete inner '{' }' block	diB
"daB"	delete a '{' }' block	daB

Note the difference between using a movement command and an object. The movement command operates from here (cursor position) to where the movement takes us. When using an object the whole object is operated upon, no matter where on the object the cursor is. For example, compare "dw" and "daw": "dw" deletes from the cursor position to the start of the next word, "daw" deletes the word under the cursor and the space after or before it.

Tag blocks

tag-blocks

For the "it" and "at" text objects an attempt is done to select blocks between matching tags for HTML and XML. But since these are not completely compatible there are a few restrictions.

The normal method is to select a `<tag>` until the matching `</tag>`. For "at" the tags are included, for "it" they are excluded. But when "it" is repeated the tags will be included (otherwise nothing would change). Also, "it" used on a tag block with no contents will select the leading tag.

"`<aaa/>`" items are skipped. Case is ignored, also for XML where case does matter.

In HTML it is possible to have a tag like `
` or `<meta ...>` without a matching end tag. These are ignored.

The text objects are tolerant about mistakes. Stray end tags are ignored.

7. Marks mark-motions E20 E78

Jumping to a mark can be done in two ways:

1. With ``` (backtick): The cursor is positioned at the specified location and the motion is **exclusive**.
2. With `'` (single quote): The cursor is positioned on the first non-blank character in the line of the specified location and the motion is **linewise**.

m mark Mark

`m{a-zA-Z}` Set mark `{a-zA-Z}` at cursor position (does not move the cursor, this is not a motion command).

m' m`

`m' or m`` Set the previous context mark. This can be jumped to with the `'''` or ````` command (does not move the cursor, this is not a motion command).

m[m]

`m[or m]` Set the `'[or ']` mark. Useful when an operator is to be simulated by multiple commands. (does not move the cursor, this is not a motion command).

m< m>

`m< or m>` Set the `'< or '>` mark. Useful to change what the ``gv`` command selects. (does not move the cursor, this is not a motion command).
Note that the Visual mode cannot be set, only the start and end position.

:ma :mark E191

`:[range]ma[rk] {a-zA-Z'}` Set mark `{a-zA-Z'}` at last line number in `[range]`, column 0. Default is cursor line.

:k

`:[range]k{a-zA-Z'}` Same as `:mark`, but the space before the mark name can be omitted.
This command is not supported in **Vim9** script,

because it is too easily confused with a variable name.

`'{a-z}` ``{a-z}` Jump to the mark `{a-z}` in the current buffer.

`'{A-Z0-9}` ``{A-Z0-9}` To the mark `{A-Z0-9}` in the file where it was set (not a motion command when in another file).

`g'{mark}` `g`{mark}` Jump to the `{mark}`, but don't change the jumplist when jumping within the current buffer. Example:
`g"`
jumps to the last known position in a file. See `$VIMRUNTIME/vimrc_example.vim`.
Also see `:keepjumps`.

`:marks` `:marks` List all the current marks (not a motion command). The `'(`, `'`), `'{` and `'}` marks are not listed. The first column has number zero.

`:marks {arg}` `E283` List the marks that are mentioned in `{arg}` (not a motion command). For example:
`:marks aB`
to list marks 'a' and 'B'.

`:delm[arks] {marks}` `:delm` `:delmarks` Delete the specified marks. Marks that can be deleted include A-Z and 0-9. You cannot delete the ' mark. They can be specified by giving the list of mark names, or with a range, separated with a dash. Spaces are ignored. Examples:
`:delmarks a` deletes mark a
`:delmarks a b 1` deletes marks a, b and 1
`:delmarks Aa` deletes marks A and a
`:delmarks p-z` deletes marks in the range p to z
`:delmarks ^.[]` deletes marks ^ . []
`:delmarks \"` deletes mark "

`:delm[arks]!` Delete all marks for the current buffer, but not marks A-Z or 0-9.

A mark is not visible in any way. It is just a position in the file that is remembered. Do not confuse marks with named registers, they are totally unrelated.

`'a - 'z` lowercase marks, valid within one file
`'A - 'Z` uppercase marks, also called file marks, valid between files
`'0 - '9` numbered marks, set from `.viminfo` file

Lowercase marks 'a to 'z are remembered as long as the file remains in the buffer list. If you remove the file from the buffer list, all its marks are lost. If you delete a line that contains a mark, that mark is erased.

Lowercase marks can be used in combination with operators. For example: "d't" deletes the lines from the cursor position to mark 't'. Hint: Use mark 't' for Top, 'b' for Bottom, etc.. Lowercase marks are restored when using undo and redo.

Uppercase marks 'A to 'Z include the file name. You can use them to jump from file to file. You can only use an uppercase mark with an operator if the mark is in the current file. The line number of the mark remains correct, even if you insert/delete lines or edit another file for a moment. When the **'viminfo'** option is not empty, uppercase marks are kept in the .viminfo file. See [viminfo-file-marks](#) .

Numbered marks '0 to '9 are quite different. They can not be set directly. They are only present when using a viminfo file [viminfo-file](#) . Basically '0 is the location of the cursor when you last exited Vim, '1 the last but one time, etc. Use the "r" flag in **'viminfo'** to specify files for which no Numbered mark should be stored. See [viminfo-file-marks](#) .

`'[`[` `'[`[`
To the first character of the previously changed or yanked text.

`']`]` `'] `]`
To the last character of the previously changed or yanked text.

After executing an operator the Cursor is put at the beginning of the text that was operated upon. After a put command ("p" or "P") the cursor is sometimes placed at the first inserted line and sometimes on the last inserted character. The four commands above put the cursor at either end. Example: After yanking 10 lines you want to go to the last one of them: "10Y']". After inserting several lines with the "p" command you want to jump to the lowest inserted line: "p']". This also works for text that has been inserted.

Note: After deleting text, the start and end positions are the same, except when using blockwise Visual mode. These commands do not work when no change was made yet in the current file.

`'<`<` `'< `<`
To the first line or character of the last selected Visual area in the current buffer. For block mode it may also be the last character in the first line (to be able to define the block).

`'>`>` `'> `>`
To the last line or character of the last selected Visual area in the current buffer. For block mode it may also be the first character of the last line (to be able to define the block). **Note** that **'selection'**

applies, the position may be just after the Visual area.

`'' ''`
To the position before the latest jump, or where the last "m'" or "m'" command was given. Not set when the `:keepjumps` command modifier was used. Also see `restore-position` .

`'" '"`
To the cursor position when last exiting the current buffer. Defaults to the first character of the first line. See `last-position-jump` for how to use this for each opened file. Only one position is remembered per buffer, not one for each window. As long as the buffer is visible in a window the position won't be changed.

`'^ '^`
To the position where the cursor was the last time when Insert mode was stopped. This is used by the `gi` command. Not set when the `:keepjumps` command modifier was used.

`'.' .'`
To the position where the last change was made. The position is at or near where the change started. Sometimes a command is executed as several changes, then the position can be near the end of what the command changed. For example when inserting a word, the position will be on the last character. To jump to older changes use `g;` .

`'('(`
To the start of the current sentence, like the `(` command.

`') ')`
To the end of the current sentence, like the `)` command.

`'{ '{`
To the start of the current paragraph, like the `{` command.

`'} '}`
To the end of the current paragraph, like the `}` command.

These commands are not marks themselves, but jump to a mark:

`]'`
`[count]` times to next line with a lowercase mark below the cursor, on the first non-blank character in the

line.

]``` `[count]` times to lowercase mark after the cursor.

[`'` `[count]` times to previous line with a lowercase mark before the cursor, on the first non-blank character in the line.

[``` `[count]` times to lowercase mark before the cursor.

`:loc[kmarks] {command}` `:loc` `:lock` `:lockmarks`
Execute `{command}` without adjusting marks. This is useful when changing text in a way that the line count will be the same when the change has completed.
WARNING: When the line count does change, marks below the change will keep their line number, thus move to another text line.

These items will not be adjusted for deleted/inserted lines:

- lower case letter marks 'a - 'z
- upper case letter marks 'A - 'Z
- numbered marks '0 - '9
- last insert position '^
- last change position '.'
- last affected text area '[' and ']'
- the Visual area '<' and '>'
- line numbers in placed signs
- line numbers in quickfix positions
- positions in the jumplist
- positions in the tagstack

These items will still be adjusted:

- previous context mark ''
- the cursor position
- the view of a window on a buffer
- folds
- diffs

`:kee[pmarks] {command}` `:kee` `:keep` `:keepmarks`

Currently only has effect for the filter command `:range!` :

- When the number of lines after filtering is equal to or larger than before, all marks are kept at the same line number.
- When the number of lines decreases, the marks in the lines that disappeared are deleted.

In any case the marks below the filtered text have their line numbers adjusted, thus stick to the text, as usual.

When the 'R' flag is missing from `'coptions'` this has the same effect as using `":keepmarks"`.

`:keepj[umps] {command}` `:keepj` `:keepjumps`

Moving around in `{command}` does not change the `'`, `'.` and `'^` marks, the `jumplist` or the `changelist`.

Useful when making a change or inserting text automatically and the user doesn't want to go to this position. E.g., when updating a "Last change" timestamp in the first line:

```
:let lnum = line(".")
:keepjumps normal gg
:call SetLastChange()
:keepjumps exe "normal " .. lnum .. "G"
```

Note that `":keepjumps"` must be used for every command. When invoking a function the commands in that function can still change the `jumplist`. Also, for `":keepjumps exe 'command '"` the "command" won't keep jumps. Instead use: `":exe 'keepjumps command'"`

8. Jumps

jump-motions

A "jump" is a command that normally moves the cursor several lines away. If you make the cursor "jump" the position of the cursor before the jump is remembered. You can return to that position with the `''` and ```` commands, unless the line containing that position was changed or deleted. The following commands are "jump" commands: `''`, ````, `"G"`, `"/"`, `"?"`, `"n"`, `"N"`, `"%"`, `"("`, `)"`, `"["`, `"]"`, `"{"`, `"}"`, `":s"`, `":tag"`, `"L"`, `"M"`, `"H"` and the commands that start editing a new file.

CTRL-O

CTRL-O Go to `[count]` Older cursor position in jump list (not a motion command).

`<Tab>`

or

CTRL-I <Tab>

CTRL-I

Go to `[count]` newer cursor position in jump list (not a motion command).

NOTE: In the GUI and in a terminal supporting `modifyOtherKeys`, **CTRL-I** can be mapped separately from `<Tab>`, on the condition that **CTRL-I** is mapped before `<Tab>`, otherwise the mapping applies to both.

:ju :jumps

`:ju[mps]` Print the jump list (not a motion command).

:cle :clearjumps

`:cle[arjumps]` Clear the jump list of the current window.

jumplist

Jumps are remembered in a jump list. With the **CTRL-O** and **CTRL-I** command you can go to cursor positions before older jumps, and back again. Thus you can move up and down the list. There is a separate jump list for each window. The maximum number of entries is fixed at 100.

For example, after three jump commands you have this jump list:

```
jump line  col file/text
  3     1    0 some text
  2    70    0 another line
  1  1154   23 end.
>
```

The "file/text" column shows the file name, or the text at the jump if it is in the current file (an indent is removed and a long line is truncated to fit in the window).

The marker ">" indicates the current position in the jumplist. It may not be shown when filtering the `:jumps` command using `:filter`

You are currently in line 1167. If you then use the **CTRL-O** command, the cursor is put in line 1154. This results in:

```
jump line  col file/text
  2     1    0 some text
  1    70    0 another line
> 0  1154   23 end.
  1  1167    0 foo bar
```

The pointer will be set at the last used jump position. The next **CTRL-O** command will use the entry above it, the next **CTRL-I** command will use the entry below it. If the pointer is below the last entry, this indicates that you did not use a **CTRL-I** or **CTRL-O** before. In this case the **CTRL-O** command will cause the cursor position to be added to the jump list, so you can get back to the position before the **CTRL-O**. In this case this is line 1167.

With more **CTRL-O** commands you will go to lines 70 and 1. If you use **CTRL-I** you can go back to 1154 and 1167 again. **Note** that the number in the "jump" column indicates the count for the **CTRL-O** or **CTRL-I** command that takes you to this position.

If you use a jump command, the current line number is inserted at the end of the jump list. If the same line was already in the jump list, it is removed. The result is that when repeating **CTRL-O** you will get back to old positions only once.

When the `:keepjumps` command modifier is used, jumps are not stored in the jumplist. Jumps are also not stored in other cases, e.g., in a `:global` command. You can explicitly add a jump by setting the ' mark with "m". **Note** that calling `setpos()` does not do this.

After the **CTRL-O** command that got you into line 1154 you could give another jump command (e.g., "G"). The jump list would then become:

```

jump line  col file/text
  4     1    0 some text
  3    70    0 another line
  2  1167    0 foo bar
  1  1154   23 end.
>

```

The line numbers will be adjusted for deleted and inserted lines. This fails if you stop editing a file without writing, like with ":n!".

When you split a window, the jumplist will be copied to the new window.

If you have included the ' item in the **'viminfo'** option the jumplist will be stored in the viminfo file and restored when starting Vim.

jumplist-stack

When **'jumpoptions'** option includes "stack", the jumplist behaves like the tag stack. When jumping to a new location from the middle of the jumplist, the locations after the current position will be discarded. With this option set you can move through a tree of jump locations. When going back up a branch and then down another branch, **CTRL-O** still takes you further up the tree.

Given a jumplist like the following in which **CTRL-O** has been used to move back three times to location X:

```

jump line  col file/text
  2  1260    8 mark.c           <-- location X-2
  1   685    0 eval.c           <-- location X-1
>  0   462   36 eval.c           <-- location X
  1   479   39 eval.c
  2   213    2 mark.c
  3   181    0 mark.c

```

jumping to (new) location Y results in the locations after the current locations being removed:

```

jump line  col file/text
  3  1260    8 mark.c           <-- location X-2
  2   685    0 eval.c           <-- location X-1
  1   462   36 eval.c           <-- location X
>

```

Then, when yet another location Z is jumped to, the new location Y appears directly after location X in the jumplist and location X remains in the same position relative to the locations (X-1, X-2, etc., ...) that had been before it prior to the original jump from X to Y:

```

jump line  col file/text
  4  1260    8 mark.c           <-- location X-2
  3   685    0 eval.c           <-- location X-1
  2   462   36 eval.c           <-- location X
  1   100    0 buffer.c         <-- location Y
>

```

CHANGE LIST JUMPS

changelist change-list-jumps E664

When making a change the cursor position is remembered. One position is remembered for every change that can be undone, unless it is close to a previous change. Two commands can be used to jump to positions of changes, also those that have been undone:

`g;` g; E662
Go to [count] older position in change list.
If [count] is larger than the number of older change positions go to the oldest change.
If there is no older change an error message is given.
(not a motion command)

`g,` g, E663
Go to [count] newer position in change list.
Just like `g;` but in the opposite direction.
(not a motion command)

When using a count you jump as far back or forward as possible. Thus you can use "999g;" to go to the first change for which the position is still remembered. The number of entries in the change list is fixed and is the same as for the `jumplist`.

When two undo-able changes are in the same line and at a column position less than '`textwidth`' apart only the last one is remembered. This avoids that a sequence of small changes in a line, for example "xxxxx", adds many positions to the change list. When '`textwidth`' is zero '`wrapmargin`' is used. When that also isn't set a fixed number of 79 is used. Detail: For the computations bytes are used, not characters, to avoid a speed penalty (this only matters for multibyte encodings).

Note that when text has been inserted or deleted the cursor position might be a bit different from the position of the change. Especially when lines have been deleted.

When the `:keepjumps` command modifier is used the position of a change is not remembered.

`:changes` :changes
Print the change list. A ">" character indicates the current position. Just after a change it is below the newest entry, indicating that `g;` takes you to the newest entry position. The first column indicates the count needed to take you to this position. Example:

```
change line col text
      3     9   8 bla bla bla
      2    11  57 foo is a bar
      1    14  54 the latest changed line
>
```

The `3g;` command takes you to line 9. Then the output of `:changes` is:

```

change line col text
> 0 9 8 bla bla bla
  1 11 57 foo is a bar
  2 14 54 the latest changed line

```

Now you can use "g," to go to line 11 and "2g," to go to line 14.

9. Various motions

various-motions

```

% Find the next item in this line after or under the
% cursor and jump to its match. inclusive motion.
% Items can be:
% ([{}]) parenthesis or (curly/square) brackets
% (this can be changed with the
% 'matchpairs' option)
% /* */ start or end of C-style comment
% #if, #ifdef, #else, #elif, #endif
% C preprocessor conditionals (when the
% cursor is on the # or no ([{
% is following)

```

For other items the `matchit` plugin can be used, see [matchit-install](#). This plugin also helps to skip matches in comments.

When **'coptions'** contains "M" `cpo-M` backslashes before parens and braces are ignored. Without "M" the number of backslashes matters: an even number doesn't match with an odd number. Thus in "(\))" and "\((\)" the first and last parenthesis match.

When the '%' character is not present in **'coptions'** `cpo-%`, parens and braces inside double quotes are ignored, unless the number of parens/braces in a line is uneven and this line and the previous one does not end in a backslash. '(', '{', '[', ']', '}' and ')' are also ignored (parens and braces inside single quotes). **Note** that this works fine for C, but not for Perl, where single quotes are used for strings.

Nothing special is done for matches in comments. You can either use the `matchit` plugin [matchit-install](#) or put quotes around matches.

No count is allowed, `{count}%` jumps to a line `{count}` percentage down the file `N%`. Using '%' on `#if/#else/#endif` makes the movement linewise.

```

[( Go to [count] previous unmatched '('.
[( exclusive motion.

```



```

[{{ Go to [count] previous unmatched '{'.
    exclusive motion.
}}
[}) Go to [count] next unmatched ')'.
    exclusive motion.
}}
[}] Go to [count] next unmatched '}'.
    exclusive motion.

```

The above four commands can be used to go to the start or end of the current code block. It is like doing "%" on the '(', ')', '{' or '}' at the other end of the code block, but you can do this from anywhere in the code block. Very useful for C programs. Example: When standing on "case x:", "{{" will bring you back to the switch statement.

```

]m Go to [count] next start of a method (for Java or
    similar structured language). When not before the
    start of a method, jump to the start or end of the
    class. exclusive motion.
]M Go to [count] next end of a method (for Java or
    similar structured language). When not before the end
    of a method, jump to the start or end of the class.
    exclusive motion.
[m Go to [count] previous start of a method (for Java or
    similar structured language). When not after the
    start of a method, jump to the start or end of the
    class. When no '{' is found before the cursor this is
    an error. exclusive motion.
[M Go to [count] previous end of a method (for Java or
    similar structured language). When not after the
    end of a method, jump to the start or end of the
    class. When no '}' is found before the cursor this is
    an error. exclusive motion.

```

The above four commands assume that the file contains a class with methods. The class definition is surrounded in '{' and '}'. Each method in the class is also surrounded with '{' and '}'. This applies to the Java language. The file looks like this:

```

// comment
class foo {
    int method_one() {
        body_one();
    }
    int method_two() {
        body_two();
    }
}

```

[To try this out copy the text and put it in a new buffer, the help text above confuses the jump commands]

Starting with the cursor on "body_two()", using "[m" will jump to the '{' at the start of "method_two()" (obviously this is much more useful when the method is long!). Using "2[m" will jump to the start of "method_one()". Using "3[m" will jump to the start of the class.

[# [#
Go to [count] previous unmatched "#if" or "#else".
exclusive motion.

]#]#
Go to [count] next unmatched "#else" or "#endif".
exclusive motion.

These two commands work in C programs that contain #if/#else/#endif constructs. It brings you to the start or end of the #if/#else/#endif where the current line is included. You can then use "%" to go to the matching line.

[* or [/ [star [/
Go to [count] previous start of a C comment "/*".
exclusive motion.

]# or]/]star]/
Go to [count] next end of a C comment "*/".
exclusive motion.

H H
To line [count] from top (Home) of window (default: first line on the window) on the first non-blank character **linewise** . See also '**startofline**' option. Cursor is adjusted for '**scrolloff**' option, unless an operator is pending, in which case the text may scroll. E.g. "yH" yanks from the first visible line until the cursor line (inclusive).

M M
To Middle line of window, on the first non-blank character **linewise** . See also '**startofline**' option.

L L
To line [count] from bottom of window (default: Last line on the window) on the first non-blank character **linewise** . See also '**startofline**' option. Cursor is adjusted for '**scrolloff**' option, unless an operator is pending, in which case the text may scroll. E.g. "yL" yanks from the cursor to the last visible line.

<LeftMouse> Moves to the position on the screen where the mouse click is **exclusive** . See also <LeftMouse> . If the position is in a status line, that window is made the

active window and the cursor is not moved.

Scrolling

scrolling

These commands move the contents of the window. If the cursor position is moved off of the window, the cursor is moved onto the window (with **'scrolloff'** screen lines around it). A page is the number of lines in the window minus two. The mnemonics for these commands may be a bit confusing. Remember that the commands refer to moving the window (the part of the buffer that you see) upwards or downwards in the buffer. When the window moves upwards in the buffer, the text in the window moves downwards on your screen.

See section 03.7 of the user manual for an introduction.

- | | |
|---------------------------------|---------------------------------|
| 1. Scrolling downwards | <code>scroll-down</code> |
| 2. Scrolling upwards | <code>scroll-up</code> |
| 3. Scrolling relative to cursor | <code>scroll-cursor</code> |
| 4. Scrolling horizontally | <code>scroll-horizontal</code> |
| 5. Scrolling synchronously | <code>scroll-binding</code> |
| 6. Scrolling with a mouse wheel | <code>scroll-mouse-wheel</code> |

=====

1. Scrolling downwards

scroll-down

The following commands move the edit window (the part of the buffer that you see) downwards (this means that more lines downwards in the text buffer can be seen):

CTRL-E

CTRL-E Scroll window [count] lines downwards in the buffer. The text moves upwards on the screen. Mnemonic: Extra lines.

CTRL-D

CTRL-D Scroll window Downwards in the buffer. The number of lines comes from the **'scroll'** option (default: half a screen). If [count] given, first set **'scroll'** option to [count]. The cursor is moved the same number of lines down in the file (if possible; when lines wrap and when hitting the end of the file there may be a difference). When the cursor is on the last line of the buffer nothing happens and a beep is produced. See also **'startofline'** option.

<S-Down>	or	<S-Down>	<kPageDown>
<PageDown>	or	<PageDown>	CTRL-F

CTRL-F Scroll window [count] pages Forwards (downwards) in the buffer. See also **'startofline'** option. When there is only one window the **'window'** option might be used.

z+ Without [count]: Redraw with the line just below the window at the top of the window. Put the cursor in that line, at the first non-blank in the line.
With [count]: just like "z<CR>".

2. Scrolling upwards

scroll-up

The following commands move the edit window (the part of the buffer that you see) upwards (this means that more lines upwards in the text buffer can be seen):

CTRL-Y Scroll window [count] lines upwards in the buffer. The text moves downwards on the screen.
Note: When using the MS-Windows key bindings **CTRL-Y** is remapped to redo.

CTRL-U Scroll window Upwards in the buffer. The number of lines comes from the '**scroll**' option (default: half a screen). If [count] given, first set the '**scroll**' option to [count]. The cursor is moved the same number of lines up in the file (if possible; when lines wrap and when hitting the end of the file there may be a difference). When the cursor is on the first line of the buffer nothing happens and a beep is produced. See also '**startofline**' option.

<S-Up> or <S-Up> <kPageUp>
<PageUp> or <PageUp> CTRL-B
CTRL-B Scroll window [count] pages Backwards (upwards) in the buffer. See also '**startofline**' option.
When there is only one window the '**window**' option might be used.

z^ Without [count]: Redraw with the line just above the window at the bottom of the window. Put the cursor in that line, at the first non-blank in the line.
With [count]: First scroll the text to put the [count] line at the bottom of the window, then redraw with the line which is now at the top of the window at the bottom of the window. Put the cursor in that line, at the first non-blank in the line.

3. Scrolling relative to cursor

scroll-cursor

The following commands reposition the edit window (the part of the buffer that you see) while keeping the cursor on the same line. **Note** that the '**scrolloff**' option may cause context lines to show above and below the cursor.

right, thus scroll the text half a screenwidth to the left. This only works when **'wrap'** is off.

zH

zH Move the view on the text half a screenwidth to the left, thus scroll the text half a screenwidth to the right. This only works when **'wrap'** is off.

For the following two commands the cursor is not moved in the text, only the text scrolls on the screen.

zS

zS Scroll the text horizontally to position the cursor at the start (left side) of the screen. This only works when **'wrap'** is off.

zE

zE Scroll the text horizontally to position the cursor at the end (right side) of the screen. This only works when **'wrap'** is off.

5. Scrolling synchronously

scroll-binding

Occasionally, it is desirable to bind two or more windows together such that when one window is scrolled, the other windows are also scrolled. In Vim, windows can be given this behavior by setting the (window-specific) **'scrollbind'** option. When a window that has **'scrollbind'** set is scrolled, all other **'scrollbind'** windows are scrolled the same amount, if possible. The behavior of **'scrollbind'** can be modified by the **'scrollopt'** option.

When using the scrollbars or the mouse wheel, the binding only happens when scrolling the window with focus (where the cursor is). You can use this to avoid scroll-binding for a moment without resetting options.

When a window also has the **'diff'** option set, the scroll-binding uses the differences between the two buffers to synchronize the position precisely. Otherwise the following method is used.

scrollbind-relative

Each **'scrollbind'** window keeps track of its "relative offset", which can be thought of as the difference between the current window's vertical scroll position and the other window's vertical scroll position. When one of the **'scrollbind'** windows is asked to vertically scroll past the beginning or end limit of its text, the window no longer scrolls, but remembers how far past the limit it wishes to be. The window keeps this information so that it can maintain the same relative offset, regardless of its being asked to scroll past its buffer's limits.

However, if a **'scrollbind'** window that has a relative offset that is past its buffer's limits is given the cursor focus, the other **'scrollbind'** windows must jump to a location where the current window's relative offset is valid. This behavior can be changed by clearing the "jump" flag from the **'scrollopt'** option.

`:syncbind` `syncbind` `:syncbind` `:sync`
 Force all `'scrollbind'` windows to have the same relative offset. I.e., when any of the `'scrollbind'` windows is scrolled to the top of its buffer, all of the `'scrollbind'` windows will also be at the top of their buffers.

`scrollbind-quickadj`

The `'scrollbind'` flag is meaningful when using keyboard commands to vertically scroll a window, and is also meaningful when using the vertical scrollbar or the mouse wheel in the window which has the cursor focus. However, when using the vertical scrollbar or the mouse wheel in a window which doesn't have the cursor focus, `'scrollbind'` is ignored. This allows quick adjustment of the relative offset of `'scrollbind'` windows.

6. Scrolling with a mouse wheel `scroll-mouse-wheel`

When your mouse has a scroll wheel, it should work with Vim in the GUI. How it works depends on your system. It might also work in an xterm `xterm-mouse-wheel`. By default only vertical scroll wheels are supported, but some GUIs also support horizontal scroll wheels.

On MS-Windows, if the scroll action causes input focus -problems, see `intellimouse-wheel-problems`.

For Win32 and the X11 GUIs (Motif and GTK) scrolling the wheel generates key presses `<ScrollWheelUp>`, `<ScrollWheelDown>`, `<ScrollWheelLeft>` and `<ScrollWheelRight>`. For example, if you push the scroll wheel upwards a `<ScrollWheelUp>` key press is generated causing the window to scroll upwards (while the text is actually moving downwards). The default action for these keys are:

<code><ScrollWheelUp></code>	scroll N lines up	<code><ScrollWheelUp></code>
<code><S-ScrollWheelUp></code>	scroll one page up	<code><S-ScrollWheelUp></code>
<code><C-ScrollWheelUp></code>	scroll one page up	<code><C-ScrollWheelUp></code>
<code><ScrollWheelDown></code>	scroll N lines down	<code><ScrollWheelDown></code>
<code><S-ScrollWheelDown></code>	scroll one page down	<code><S-ScrollWheelDown></code>
<code><C-ScrollWheelDown></code>	scroll one page down	<code><C-ScrollWheelDown></code>
<code><ScrollWheelLeft></code>	scroll N columns left	<code><ScrollWheelLeft></code>
<code><S-ScrollWheelLeft></code>	scroll one page left	<code><S-ScrollWheelLeft></code>
<code><C-ScrollWheelLeft></code>	scroll one page left	<code><C-ScrollWheelLeft></code>
<code><ScrollWheelRight></code>	scroll N columns right	<code><ScrollWheelRight></code>
<code><S-ScrollWheelRight></code>	scroll one page right	<code><S-ScrollWheelRight></code>
<code><C-ScrollWheelRight></code>	scroll one page right	<code><C-ScrollWheelRight></code>

This should work in all modes, except when editing the command line.

The value of N depends on the system. By default Vim scrolls three lines when moving vertically, and six columns when moving horizontally. On MS-Windows the amount of lines and columns for each scroll action is taken from the system-wide settings.

Note that horizontal scrolling only works if `'nowrap'` is set. Also, unless the "h" flag in `'guioptions'` is set, the cursor moves to the longest visible

line if the cursor line is about to be scrolled off the screen (similarly to how the horizontal scrollbar works).

You can modify the default behavior by mapping the keys. For example, to make the scroll wheel move one line or half a page in Normal mode:

```
:map <ScrollWheelUp> <C-Y>
:map <S-ScrollWheelUp> <C-U>
:map <ScrollWheelDown> <C-E>
:map <S-ScrollWheelDown> <C-D>
```

You can also use Alt and Ctrl modifiers.

This only works when Vim gets the scroll wheel events, of course. You can check if this works with the "xev" program.

mouse-scrolling-off

If you do not want the mouse to cause scrolling (e.g. because resting your palm on the touchpad causes scroll events), you can disable that with:

```
:map <ScrollWheelDown> <Nop>
:map! <ScrollWheelDown> <Nop>
:map <ScrollWheelUp> <Nop>
:map! <ScrollWheelUp> <Nop>
:map <ScrollWheelLeft> <Nop>
:map! <ScrollWheelLeft> <Nop>
:map <ScrollWheelRight> <Nop>
:map! <ScrollWheelRight> <Nop>
```

When using XFree86, the /etc/XF86Config file should have the correct entry for your mouse. For FreeBSD, this entry works for a Logitech scrollmouse:

```
Protocol      "MouseMan"
Device        "/dev/psm0"
ZAxisMapping  4 5
```

See the XFree86 documentation for information.

<MouseDown> <MouseUp>

The keys <MouseDown> and <MouseUp> have been deprecated. Use <ScrollWheelUp> instead of <MouseDown> and use <ScrollWheelDown> instead of <MouseUp>.

xterm-mouse-wheel

To use the mouse wheel in a new xterm you only have to make the scroll wheel work in your Xserver, as mentioned above.

To use the mouse wheel in an older xterm you must do this:

1. Make it work in your Xserver, as mentioned above.
2. Add translations for the xterm, so that the xterm will pass a scroll event to Vim as an escape sequence.
3. Add mappings in Vim, to interpret the escape sequences as <ScrollWheelDown> or <ScrollWheelUp> keys.

You can do the translations by adding this to your ~/.Xdefaults file (or other file where your X resources are kept):

```
XTerm*VT100.Translations:      #override \n
                                s<Btn4Down>: string("\0x9b") string("[64~") \n
                                s<Btn5Down>: string("\0x9b") string("[65~") \n
                                <Btn4Down>: string("\0x9b") string("[62~") \n
```

```
<Btn5Down>: string("0x9b") string("[63~") \n\  
<Btn4Up>: \n\  
<Btn5Up>:
```

Add these mappings to your vimrc file:

```
:map <M-Esc>[62~ <ScrollWheelUp>  
:map! <M-Esc>[62~ <ScrollWheelUp>  
:map <M-Esc>[63~ <ScrollWheelDown>  
:map! <M-Esc>[63~ <ScrollWheelDown>  
:map <M-Esc>[64~ <S-ScrollWheelUp>  
:map! <M-Esc>[64~ <S-ScrollWheelUp>  
:map <M-Esc>[65~ <S-ScrollWheelDown>  
:map! <M-Esc>[65~ <S-ScrollWheelDown>
```


Insert Insert-mode
mode-ins-repl

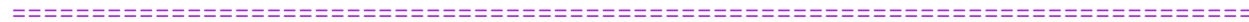
Inserting and replacing text

Most of this file is about Insert and Replace mode. At the end are a few commands for inserting text in other ways.

An overview of the most often used commands can be found in chapter 24 of the user manual [usr_24.txt](#) .

- 1. Special keys [ins-special-keys](#)
- 2. Special special keys [ins-special-special](#)
- 3. **'textwidth'** and **'wrapmargin'** options [ins-textwidth](#)
- 4. **'expandtab'**, **'smarttab'** and **'softtabstop'** options [ins-expandtab](#)
- 5. Replace mode [Replace-mode](#)
- 6. Virtual Replace mode [Virtual-Replace-mode](#)
- 7. Insert mode completion [ins-completion](#)
- 8. Insert mode commands [inserting](#)
- 9. Ex insert commands [inserting-ex](#)
- 10. Inserting a file [inserting-file](#)

Also see **'virtualedit'**, for moving the cursor to positions where there is no character. Useful for editing a table.



- 1. Special keys [ins-special-keys](#)

In Insert and Replace mode, the following characters have a special meaning; other characters are inserted directly. To insert one of these special characters into the buffer, precede it with **CTRL-V**. To insert a **<Nul>** character use "**CTRL-V CTRL-@**" or "**CTRL-V 000**". On some systems, you have to use "**CTRL-V 003**" to insert a **CTRL-C**. **Note:** When **CTRL-V** is mapped you can often use **CTRL-Q** instead [i_CTRL-Q](#) .

If you are working in a special language mode when inserting text, see the **'langmap'** option, ['langmap'](#) , on how to avoid switching this mode on and off all the time.

If you have **'insertmode'** set, **<Esc>** and a few other keys get another meaning. See ['insertmode'](#) .

char	action
------	--------



[i_CTRL-\[](#) [i_<Esc>](#)

<Esc> or **CTRL-[** End insert or Replace mode, go back to Normal mode. Finish abbreviation.

Note: If your **<Esc>** key is hard to hit on your keyboard, train yourself to use **CTRL-[**.

If **Esc** doesn't work and you are using a Mac, try **CTRL-Esc**.

Or disable Listening under Accessibility preferences.

- CTRL-C** [i_CTRL-C](#)
Quit insert mode, go back to Normal mode. Do not check for abbreviations. Does not trigger the [InsertLeave](#) autocommand event.
- CTRL-@** [i_CTRL-@](#)
Insert previously inserted text and stop insert.
- CTRL-A** [i_CTRL-A](#)
Insert previously inserted text.
- <BS> or CTRL-H** [i_CTRL-H](#) [i_<BS>](#) [i_BS](#)
Delete the character before the cursor (see [i_backspacing](#) about joining lines).
See [:fixdel](#) if your **<BS>** key does not do what you want.
- ** [i_](#) [i_DEL](#)
Delete the character under the cursor. If the cursor is at the end of the line, and the **'backspace'** option includes "eol", delete the **<EOL>**; the next line is appended after the current one.
See [:fixdel](#) if your **** key does not do what you want.
- CTRL-W** [i_CTRL-W](#)
Delete the word before the cursor (see [i_backspacing](#) about joining lines). See the section "word motions", [word-motions](#), for the definition of a word.
- CTRL-U** [i_CTRL-U](#)
Delete all entered characters before the cursor in the current line. If there are no newly entered characters and **'backspace'** is not empty, delete all characters before the cursor in the current line.
If C-indenting is enabled the indent will be adjusted if the line becomes blank.
See [i_backspacing](#) about joining lines.
- <Tab> or CTRL-I** [i_CTRL-I](#) [i_<Tab>](#) [i_Tab](#)
Insert a tab. If the **'expandtab'** option is on, the equivalent number of spaces is inserted (use **CTRL-V <Tab>** to avoid the expansion; use **CTRL-Q <Tab>** if **CTRL-V** is mapped [i_CTRL-Q](#)). See also the **'smarttab'** option and [ins-expandtab](#) .
- <NL> or CTRL-J** [i_CTRL-J](#) [i_<NL>](#)
Begin new line.
- <CR> or CTRL-M** [i_CTRL-M](#) [i_<CR>](#)
Begin new line.
- CTRL-K {char1} [char2]** [i_CTRL-K](#)
Enter digraph (see [digraphs](#)). When {char1} is a special key, the code for that key is inserted in **<>** form. For example, the string **"<S-Space>"** can be entered by typing **<C-K><S-Space>** (two keys). Neither char is considered for mapping.
- CTRL-N** [i_CTRL-N](#)
Find next keyword (see [i_CTRL-N](#)).

CTRL-P Find previous keyword (see [i_CTRL-P](#)).

CTRL-R {register} [i_CTRL-R](#)
Insert the contents of a register. Between typing **CTRL-R** and the second character, `''` will be displayed to indicate that you are expected to enter the name of a register. The text is inserted as if you typed it, but mappings and abbreviations are not used. If you have options like **'textwidth'**, **'formatoptions'**, or **'autoindent'** set, this will influence what will be inserted. This is different from what happens with the "p" command and pasting with the mouse.

Special registers:

<code>''</code>	the unnamed register, containing the text of the last delete or yank
<code>'%</code>	the current file name
<code>'#'</code>	the alternate file name
<code>'*'</code>	the clipboard contents (X11: primary selection)
<code>'+'</code>	the clipboard contents
<code> '/'</code>	the last search pattern
<code> ':'</code>	the last command-line
<code> '.'</code>	the last inserted text

`'-'` the last small (less than a line) delete register. This is repeatable using `.` since it remembers the register to put instead of the literal text to insert.

`'='` the expression register: you are prompted to enter an expression (see [expression](#))
Note that 0x80 (128 decimal) is used for special keys. E.g., you can use this to move the cursor up:

CTRL-R = "\<Up>"

Use **CTRL-R CTRL-R** to insert text literally. When the result is a [List](#) the items are used as lines. They can have line breaks inside too.

When the result is a Float it's automatically converted to a String.

When `append()` or `setline()` is invoked the undo sequence will be broken.

See [registers](#) about registers.

CTRL-R CTRL-R {register} [i_CTRL-R_CTRL-R](#)
Insert the contents of a register. Works like using a single **CTRL-R**, but the text is inserted literally, not as if typed. This differs when the register contains characters like `<BS>`. Example, where register a contains "ab^Hc":

`CTRL-R a` results in "ac".

`CTRL-R CTRL-R a` results in "ab^Hc".

Options **'textwidth'**, **'formatoptions'**, etc. still apply. If you also want to avoid these, use **CTRL-R CTRL-O**, see below. The `'.'` register (last inserted text) is still inserted as typed.

After this command, the '.' register contains the text from the register as if it was inserted by typing it.

CTRL-R CTRL-O {register} i_CTRL-R_CTRL-O
Insert the contents of a register literally and don't auto-indent. Does the same as pasting with the mouse `<MiddleMouse>`. When the register is linewise this will insert the text above the current line, like with ``P``. The '.' register (last inserted text) is still inserted as typed.
After this command, the '.' register contains the command typed and not the text. I.e., the literals `"^R^O"` and not the text from the register.
Does not replace characters in `Replace-mode` !

CTRL-R CTRL-P {register} i_CTRL-R_CTRL-P
Insert the contents of a register literally and fix the indent, like `[<MiddleMouse>`. The '.' register (last inserted text) is still inserted as typed.
After this command, the '.' register contains the command typed and not the text. I.e., the literals `"^R^P"` and not the text from the register.
Does not replace characters in `Replace-mode` !

CTRL-T i_CTRL-T
Insert one shiftwidth of indent at the start of the current line. The indent is always rounded to a `'shiftwidth'` (this is vi compatible).

CTRL-D i_CTRL-D
Delete one shiftwidth of indent at the start of the current line. The indent is always rounded to a `'shiftwidth'` (this is vi compatible).

0 CTRL-D i_0_CTRL-D
Delete all indent in the current line.

^ CTRL-D i_^_CTRL-D
Delete all indent in the current line. The indent is restored in the next line. This is useful when inserting a label.

CTRL-V i_CTRL-V
Insert next non-digit literally. For special keys, the terminal code is inserted. It's also possible to enter the decimal, octal or hexadecimal value of a character `i_CTRL-V_digit`.
The characters typed right after **CTRL-V** are not considered for mapping.
Note: When **CTRL-V** is mapped (e.g., to paste text) you can often use **CTRL-Q** instead `i_CTRL-Q`.
When `modifyOtherKeys` is enabled then special Escape sequence is converted back to what it was without `modifyOtherKeys`, unless the Shift key is also pressed.

CTRL-Q Same as **CTRL-V**. i_CTRL-Q
Note: Some terminal connections may eat **CTRL-Q**, it doesn't work then. It does work in the GUI.

CTRL-SHIFT-V i_CTRL-SHIFT-V
CTRL-SHIFT-Q i_CTRL-SHIFT-Q
Works just like **CTRL-V**, unless `modifyOtherKeys` is active, then it inserts the Escape sequence for a key with modifiers.

CTRL-X Enter **CTRL-X** mode. This is a sub-mode where commands can be given to complete words or scroll the window. See `i_CTRL-X` and `ins-completion`.

CTRL-E i_CTRL-E
Insert the character which is below the cursor.

CTRL-Y i_CTRL-Y
Insert the character which is above the cursor.
Note that for **CTRL-E** and **CTRL-Y** `'textwidth'` is not used, to be able to copy characters from a long line.

CTRL-_ i_CTRL-_
Switch between languages, as follows:
- When in a rightleft window, `revins` and `nohkmap` are toggled, since English will likely be inserted in this case.
- When in a norightleft window, `revins` and `hkmap` are toggled, since Hebrew will likely be inserted in this case.

`CTRL-_` moves the cursor to the end of the typed text.

This command is only available when the `'allowrevins'` option is set.
Please refer to `rileft.txt` for more information about right-to-left mode.
Only if compiled with the `+rightleft` feature.

CTRL-^ i_CTRL-^
Toggle the use of typing language characters.
When language `:lmap` mappings are defined:
- If `'iminsert'` is 1 (langmap mappings used) it becomes 0 (no langmap mappings used).
- If `'iminsert'` has another value it becomes 1, thus langmap mappings are enabled.
When no language mappings are defined:
- If `'iminsert'` is 2 (Input Method used) it becomes 0 (no Input Method used).
- If `'iminsert'` has another value it becomes 2, thus the Input Method is enabled.
When set to 1, the value of the `"b:keymap_name"` variable, the `'keymap'` option or `"<lang>"` appears in the status line.
The language mappings are normally used to type characters that are different from what the keyboard produces. The `'keymap'` option can be used to install a whole number of them.

i_CTRL-]

CTRL-] Trigger abbreviation, without inserting a character.

i_<Insert>

<Insert> Toggle between Insert and Replace mode.

i_backspacing

The effect of the **<BS>**, **CTRL-W**, and **CTRL-U** depend on the **'backspace'** option (unless **'revins'** is set). This is a comma-separated list of items:

item	action
indent	allow backspacing over autoindent
eol	allow backspacing over end-of-line (join lines)
start	allow backspacing over the start position of insert; CTRL-W and CTRL-U stop once at the start position

When **'backspace'** is empty, Vi compatible backspacing is used. You cannot backspace over autoindent, before column 1 or before where insert started.

For backwards compatibility the values "0", "1", "2" and "3" are also allowed, see **'backspace'** .

If the **'backspace'** option does contain "eol" and the cursor is in column 1 when one of the three keys is used, the current line is joined with the previous line. This effectively deletes the **<EOL>** in front of the cursor.

i_CTRL-V_digit

With **CTRL-V** the decimal, octal or hexadecimal value of a character can be entered directly. This way you can enter any character, except a line break (**<NL>**, value 10). There are five ways to enter the character value:

first char	mode	max nr of chars	max value
(none)	decimal	3	255
o or O	octal	3	377 (255)
x or X	hexadecimal	2	ff (255)
u	hexadecimal	4	ffff (65535)
U	hexadecimal	8	7fffffff (2147483647)

Normally you would type the maximum number of characters. Thus to enter a space (value 32) you would type **<C-V>032**. You can omit the leading zero, in which case the character typed after the number must be a non-digit. This happens for the other modes as well: As soon as you type a character that is invalid for the mode, the value before it will be used and the "invalid" character is dealt with in the normal way.

If you enter a value of 10, it will end up in the file as a 0. The 10 is a **<NL>**, which is used internally to represent the **<Nul>** character. When writing the buffer to a file, the **<NL>** character is translated into **<Nul>**. The **<NL>** character is written at the end of each line. Thus if you want to insert a **<NL>** character in a file you will have to make a line break.

Also see **'fileformat'**.

i_CTRL-X insert_expand

CTRL-X enters a sub-mode where several commands can be used. Most of these

commands do keyword completion; see [ins-completion](#) .

Two commands can be used to scroll the window up or down, without exiting insert mode:

CTRL-X CTRL-E i_CTRL-X_CTRL-E
scroll window one line up.
When doing completion look here: [complete_CTRL-E](#)

CTRL-X CTRL-Y i_CTRL-X_CTRL-Y
scroll window one line down.
When doing completion look here: [complete_CTRL-Y](#)

After **CTRL-X** is pressed, each **CTRL-E** (CTRL-Y) scrolls the window up (down) by one line unless that would cause the cursor to move from its current position in the file. As soon as another key is pressed, **CTRL-X** mode is exited and that key is interpreted as in Insert mode.

2. Special special keys

[ins-special-special](#)

The following keys are special. They stop the current insert, do something, and then restart insertion. This means you can do something without getting out of Insert mode. This is very handy if you prefer to use the Insert mode all the time, just like editors that don't have a separate Normal mode. You may also want to set the **'backspace'** option to "indent,eol,start" and set the **'insertmode'** option. You can use **CTRL-O** if you want to map a function key to a command.

The changes (inserted or deleted characters) before and after these keys can be undone separately. Only the last change can be redone and always behaves like an "i" command.

char	action	
<Up>	cursor one line up	i_<Up>
<Down>	cursor one line down	i_<Down>
CTRL-G <Up>	cursor one line up, insert start column	i_CTRL-G_<Up>
CTRL-G k	cursor one line up, insert start column	i_CTRL-G_k
CTRL-G CTRL-K	cursor one line up, insert start column	i_CTRL-G_CTRL-K
CTRL-G <Down>	cursor one line down, insert start column	i_CTRL-G_<Down>
CTRL-G j	cursor one line down, insert start column	i_CTRL-G_j
CTRL-G CTRL-J	cursor one line down, insert start column	i_CTRL-G_CTRL-J
<Left>	cursor one character left	i_<Left>
<Right>	cursor one character right	i_<Right>
<S-Left>	cursor one word back (like "b" command)	i_<S-Left>
<C-Left>	cursor one word back (like "b" command)	i_<C-Left>
<S-Right>	cursor one word forward (like "w" command)	i_<S-Right>
<C-Right>	cursor one word forward (like "w" command)	i_<C-Right>
<Home>	cursor to first char in the line	i_<Home>
<End>	cursor to after last char in the line	i_<End>
<C-Home>	cursor to first char in the file	i_<C-Home>
<C-End>	cursor to after last char in the file	i_<C-End>

<LeftMouse>	cursor to position of mouse click	i_<LeftMouse>
<S-Up>	move window one page up	i_<S-Up>
<PageUp>	move window one page up	i_<PageUp>
<S-Down>	move window one page down	i_<S-Down>
<PageDown>	move window one page down	i_<PageDown>
<ScrollWheelDown>	move window three lines down	i_<ScrollWheelDown>
<S-ScrollWheelDown>	move window one page down	i_<S-ScrollWheelDown>
<ScrollWheelUp>	move window three lines up	i_<ScrollWheelUp>
<S-ScrollWheelUp>	move window one page up	i_<S-ScrollWheelUp>
<ScrollWheelLeft>	move window six columns left	i_<ScrollWheelLeft>
<S-ScrollWheelLeft>	move window one page left	i_<S-ScrollWheelLeft>
<ScrollWheelRight>	move window six columns right	i_<ScrollWheelRight>
<S-ScrollWheelRight>	move window one page right	i_<S-ScrollWheelRight>
CTRL-O	execute one command, return to Insert mode	i_CTRL-O
CTRL-\ CTRL-O	like CTRL-O but don't move the cursor	i_CTRL-_CTRL-O
CTRL-L	when ' insertmode ' is set: go to Normal mode	i_CTRL-L
CTRL-G u	close undo sequence, start new change	i_CTRL-G_u
CTRL-G U	don't start a new undo block with the next left/right cursor movement, if the cursor stays within the same line	i_CTRL-G_U

Note: If the cursor keys take you out of Insert mode, check the '**noesckey**' option.

The **CTRL-O** command sometimes has a side effect: If the cursor was beyond the end of the line, it will be put on the last character in the line. In mappings it's often better to use <Esc> (first put an "x" in the text, <Esc> will then always put the cursor on it). Or use CTRL-\ **CTRL-O**, but then beware of the cursor possibly being beyond the end of the line. **Note** that the command following CTRL-\ **CTRL-O** can still move the cursor, it is not restored to its original position.

The **CTRL-O** command takes you to Normal mode. If you then use a command enter Insert mode again it normally doesn't nest. Thus when typing "a<C-O>a" and then <Esc> takes you back to Normal mode, you do not need to type <Esc> twice. An exception is when not typing the command, e.g. when executing a mapping or sourcing a script. This makes mappings work that briefly switch to Insert mode.

The shifted cursor keys are not available on all terminals.

Another side effect is that a count specified before the "i" or "a" command is ignored. That is because repeating the effect of the command after **CTRL-O** is too complicated.

An example for using **CTRL-G u**:

```
:inoremap <C-H> <C-G>u<C-H>
```

This redefines the backspace key to start a new undo sequence. You can now undo the effect of the backspace key, without changing what you typed before that, with **CTRL-O u**. Another example:

```
:inoremap <CR> <C-]><C-G>u<CR>
```

This starts a new undo block at each line break. It also expands abbreviations before this.

An example for using **CTRL-G U**:

```
inoremap <Left> <C-G>U<Left>
inoremap <Right> <C-G>U<Right>
inoremap <expr> <Home> col('.') == match(getline('.'), '\S') + 1 ?
  \ repeat('<C-G>U<Left>', col('.') - 1) :
  \ (col('.') < match(getline('.'), '\S') ?
  \   \ repeat('<C-G>U<Right>', match(getline('.'), '\S') + 0) :
  \   \ repeat('<C-G>U<Left>', col('.') - 1 - match(getline('.'), '\S')))
inoremap <expr> <End> repeat('<C-G>U<Right>', col('$') - col('.'))
inoremap ( )<C-G>U<Left>
```

This makes it possible to use the cursor keys in Insert mode, without starting a new undo block and therefore using `.` (redo) will work as expected. Also entering a text like (with the "(" mapping from above):

```
  Lorem ipsum (dolor
```

will be repeatable by using `.` to the expected

```
  Lorem ipsum (dolor)
```

Using **CTRL-O** splits undo: the text typed before and after it is undone separately. If you want to avoid this (e.g., in a mapping) you might be able to use **CTRL-R** = `i_CTRL-R`. E.g., to call a function:

```
:imap <F2> <C-R>=MyFunc()<CR>
```

When the **'whichwrap'** option is set appropriately, the `<Left>` and `<Right>` keys on the first/last character in the line make the cursor wrap to the previous/next line.

The **CTRL-G j** and **CTRL-G k** commands can be used to insert text in front of a column. Example:

```
int i;
int j;
```

Position the cursor on the first "int", type "istatic `<C-G>j` ". The result is:

```
static int i;
int j;
```

When inserting the same text in front of the column in every line, use the Visual blockwise command "I" `v_b_I`.

3. **'textwidth'** and **'wrapmargin'** options ins-textwidth

The **'textwidth'** option can be used to automatically break a line before it gets too long. Set the **'textwidth'** option to the desired maximum line length. If you then type more characters (not spaces or tabs), the last word will be put on a new line (unless it is the only word on the

line). If you set `'textwidth'` to 0, this feature is disabled.

The `'wrapmargin'` option does almost the same. The difference is that `'textwidth'` has a fixed width while `'wrapmargin'` depends on the width of the screen. When using `'wrapmargin'` this is equal to using `'textwidth'` with a value equal to $(\text{columns} - \text{'wrapmargin'})$, where columns is the width of the screen.

When `'textwidth'` and `'wrapmargin'` are both set, `'textwidth'` is used.

If you don't really want to break the line, but view the line wrapped at a convenient place, see the `'linebreak'` option.

The line is only broken automatically when using Insert mode, or when appending to a line. When in replace mode and the line length is not changed, the line will not be broken.

Long lines are broken if you enter a non-white character after the margin. The situations where a line will be broken can be restricted by adding characters to the `'formatoptions'` option:

- "l" Only break a line if it was not longer than `'textwidth'` when the insert started.
- "v" Only break at a white character that has been entered during the current insert command. This is mostly Vi-compatible.
- "lv" Only break if the line was not longer than `'textwidth'` when the insert started and only at a white character that has been entered during the current insert command. Only differs from "l" when entering non-white characters while crossing the `'textwidth'` boundary.

Normally an internal function will be used to decide where to break the line. If you want to do it in a different way set the `'formatexpr'` option to an expression that will take care of the line break.

If you want to format a block of text, you can use the "gq" operator. Type "gq" and a movement command to move the cursor to the end of the block. In many cases, the command "gq}" will do what you want (format until the end of paragraph). Alternatively, you can use "gqap", which will format the whole paragraph, no matter where the cursor currently is. Or you can use Visual mode: hit "v", move to the end of the block, and type "gq". See also [gq](#) .

4. `'expandtab'`, `'smarttab'` and `'softtabstop'` options ins-expandtab

If the `'expandtab'` option is on, spaces will be used to fill the amount of whitespace of the tab. If you want to enter a real `<Tab>`, type `CTRL-V` first (use `CTRL-Q` when `CTRL-V` is mapped `i_CTRL-Q`).

The `'expandtab'` option is off by default. **Note** that in Replace mode, a single character is replaced with several spaces. The result of this is that the number of characters in the line increases. Backspacing will delete one space at a time. The original character will be put back for only one space that you backspace over (the last one).

ins-smarttab

When the `'smarttab'` option is on, a `<Tab>` inserts `'shiftwidth'` positions at

the beginning of a line and **'tabstop'** positions in other places. This means that often spaces instead of a `<Tab>` character are inserted. When **'smarttab'** is off, a `<Tab>` always inserts **'tabstop'** positions, and **'shiftwidth'** is only used for `>>` and the like.

`ins-softtabstop`

When the **'softtabstop'** option is non-zero, a `<Tab>` inserts **'softtabstop'** positions, and a `<BS>` used to delete white space, will delete **'softtabstop'** positions. This feels like **'tabstop'** was set to **'softtabstop'**, but a real `<Tab>` character still takes **'tabstop'** positions, so your file will still look correct when used by other applications.

If **'softtabstop'** is non-zero, a `<BS>` will try to delete as much white space to move to the previous **'softtabstop'** position, except when the previously inserted character is a space, then it will only delete the character before the cursor. Otherwise you cannot always delete a single character before the cursor. You will have to delete **'softtabstop'** characters first, and then type extra spaces to get where you want to be.

5. Replace mode

`Replace` `Replace-mode` `mode-replace`

Enter Replace mode with the `"R"` command in normal mode.

In Replace mode, one character in the line is deleted for every character you type. If there is no character to delete (at the end of the line), the typed character is appended (as in Insert mode). Thus the number of characters in a line stays the same until you get to the end of the line. If a `<NL>` is typed, a line break is inserted and no character is deleted.

Be careful with `<Tab>` characters. If you type a normal printing character in its place, the number of characters is still the same, but the number of columns will become smaller.

If you delete characters in Replace mode (with `<BS>`, `CTRL-W`, or `CTRL-U`), what happens is that you delete the changes. The characters that were replaced are restored. If you had typed past the existing text, the characters you added are deleted. This is effectively a character-at-a-time undo.

If the **'expandtab'** option is on, a `<Tab>` will replace one character with several spaces. The result of this is that the number of characters in the line increases. Backspacing will delete one space at a time. The original character will be put back for only one space that you backspace over (the last one).

6. Virtual Replace mode

`vreplace-mode` `Virtual-Replace-mode`

Enter Virtual Replace mode with the `"gR"` command in normal mode.
{not available when compiled without the `|+vreplace|` feature}

Virtual Replace mode is similar to Replace mode, but instead of replacing actual characters in the file, you are replacing screen real estate, so that characters further on in the file never appear to move.

So if you type a `<Tab>` it may replace several normal characters, and if you type a letter on top of a `<Tab>` it may not replace anything at all, since the `<Tab>` will still line up to the same place as before.

Typing a `<NL>` still doesn't cause characters later in the file to appear to move. The rest of the current line will be replaced by the `<NL>` (that is, they are deleted), and replacing continues on the next line. A new line is NOT inserted unless you go past the end of the file.

Interesting effects are seen when using **CTRL-T** and **CTRL-D**. The characters before the cursor are shifted sideways as normal, but characters later in the line still remain still. **CTRL-T** will hide some of the old line under the shifted characters, but **CTRL-D** will reveal them again.

As with Replace mode, using `<BS>` etc will bring back the characters that were replaced. This still works in conjunction with `'smartindent'`, **CTRL-T** and **CTRL-D**, `'expandtab'`, `'smarttab'`, `'softtabstop'`, etc.

In `'list'` mode, Virtual Replace mode acts as if it was not in `'list'` mode, unless "L" is in `'coptions'`.

Note that the only situations for which characters beyond the cursor should appear to move are in List mode `'list'`, and occasionally when `'wrap'` is set (and the line changes length to become shorter or wider than the width of the screen). In other cases spaces may be inserted to avoid following characters to move.

This mode is very useful for editing `<Tab>` separated columns in tables, for entering new data while keeping all the columns aligned.

7. Insert mode completion ins-completion

In Insert and Replace mode, there are several commands to complete part of a keyword or line that has been typed. This is useful if you are using complicated keywords (e.g., function names with capitals and underscores).

Completion can be done for:

- | | |
|---|---|
| 1. Whole lines | <code>i_CTRL-X_CTRL-L</code> |
| 2. keywords in the current file | <code>i_CTRL-X_CTRL-N</code> |
| 3. keywords in <code>'dictionary'</code> | <code>i_CTRL-X_CTRL-K</code> |
| 4. keywords in <code>'thesaurus'</code> , thesaurus-style | <code>i_CTRL-X_CTRL-T</code> |
| 5. keywords in the current and included files | <code>i_CTRL-X_CTRL-I</code> |
| 6. tags | <code>i_CTRL-X_CTRL-]</code> |
| 7. file names | <code>i_CTRL-X_CTRL-F</code> |
| 8. definitions or macros | <code>i_CTRL-X_CTRL-D</code> |
| 9. Vim command-line | <code>i_CTRL-X_CTRL-V</code> |
| 10. User defined completion | <code>i_CTRL-X_CTRL-U</code> |
| 11. omni completion | <code>i_CTRL-X_CTRL-O</code> |
| 12. Spelling suggestions | <code>i_CTRL-X_s</code> |
| 13. keywords in <code>'complete'</code> | <code>i_CTRL-N</code> <code>i_CTRL-P</code> |

Additionally, `i_CTRL-X_CTRL-Z` stops completion without changing the text.

All these, except **CTRL-N** and **CTRL-P**, are done in **CTRL-X** mode. This is a sub-mode of Insert and Replace modes. You enter **CTRL-X** mode by typing **CTRL-X** and one of the **CTRL-X** commands. You exit **CTRL-X** mode by typing a key that is not a valid **CTRL-X** mode command. Valid keys are the **CTRL-X** command itself, **CTRL-N** (next), and **CTRL-P** (previous).

To get the current completion information, `complete_info()` can be used. Also see the `'infercase'` option if you want to adjust the case of the match.

`complete_CTRL-E`

When completion is active you can use **CTRL-E** to stop it and go back to the originally typed text. The **CTRL-E** will not be inserted.

`complete_CTRL-Y`

When the popup menu is displayed you can use **CTRL-Y** to stop completion and accept the currently selected entry. The **CTRL-Y** is not inserted. Typing a space, Enter, or some other unprintable character will leave completion mode and insert that typed character.

When the popup menu is displayed there are a few more special keys, see `popupmenu-keys`.

Note: The keys that are valid in **CTRL-X** mode are not mapped. This allows for ``:map <C-F> <C-X><C-F>`` to work (assuming "<" is not in `'cpo'`). The key that ends **CTRL-X** mode (any key that is not a valid **CTRL-X** mode command) is mapped. Also, when doing completion with `'complete'` mappings apply as usual.

`E565`

Note: While completion is active Insert mode can't be used recursively and buffer text cannot be changed. Mappings that somehow invoke `":normal i.."` will generate an E565 error.

The following mappings are suggested to make typing the completion commands a bit easier (although they will hide other commands; this requires "<" is not in `'cpo'`):

```
:inoremap <C-]> <C-X><C-]>
:inoremap <C-F> <C-X><C-F>
:inoremap <C-D> <C-X><C-D>
:inoremap <C-L> <C-X><C-L>
```

As a special case, typing **CTRL-R** to perform register insertion (see `i_CTRL-R`) will not exit **CTRL-X** mode. This is primarily to allow the use of the '=' register to call some function to determine the next operation. If the contents of the register (or result of the '=' register evaluation) are not valid **CTRL-X** mode keys, then **CTRL-X** mode will be exited as if those keys had been typed.

For example, the following will map `<Tab>` to either actually insert a `<Tab>` if the current line is currently only whitespace, or start/continue a **CTRL-N** completion operation:

```
function! CleverTab()
```

```

if strpart( getline('.'), 0, col('.')-1 ) =~ '^\\s*$'
    return "\\<Tab>"
else
    return "\\<C-N>"
endif
endfunction
inoremap <Tab> <C-R>=CleverTab()<CR>

```

Completing whole lines

`compl-whole-line`

`i_CTRL-X_CTRL-L`

CTRL-X CTRL-L

Search backwards for a line that starts with the same characters as those in the current line before the cursor. Indent is ignored. The matching line is inserted in front of the cursor. The '**complete**' option is used to decide which buffers are searched for a match. Both loaded and unloaded buffers are used.

CTRL-L or
CTRL-P

Search backwards for next matching line. This line replaces the previous matching line.

CTRL-N

Search forward for next matching line. This line replaces the previous matching line.

CTRL-X CTRL-L

After expanding a line you can additionally get the line next to it by typing **CTRL-X CTRL-L** again, unless a double **CTRL-X** is used. Only works for loaded buffers.

Completing keywords in current file

`compl-current`

`i_CTRL-X_CTRL-P`

`i_CTRL-X_CTRL-N`

CTRL-X CTRL-N

Search forwards for words that start with the keyword in front of the cursor. The found keyword is inserted in front of the cursor.

CTRL-X CTRL-P

Search backwards for words that start with the keyword in front of the cursor. The found keyword is inserted in front of the cursor.

CTRL-N

Search forward for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-P

Search backwards for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-X CTRL-N or

CTRL-X CTRL-P

Further use of **CTRL-X CTRL-N** or **CTRL-X CTRL-P** will copy the words following the previous expansion in other contexts unless a double **CTRL-X** is used.

If there is a keyword in front of the cursor (a name made out of alphabetic characters and characters in **'iskeyword'**), it is used as the search pattern, with "**<**" prepended (meaning: start of a word). Otherwise "**<kk**" is used as search pattern (start of any keyword of at least two characters).

In Replace mode, the number of characters that are replaced depends on the length of the matched string. This works like typing the characters of the matched string in Replace mode.

If there is not a valid keyword character before the cursor, any keyword of at least two characters is matched.

```
e.g., to get:
    printf("(%g, %g, %g)", vector[0], vector[1], vector[2]);
just type:
    printf("(%g, %g, %g)", vector[0], ^P[1], ^P[2]);
```

The search wraps around the end of the file, the value of **'wrapscan'** is not used here.

Multiple repeats of the same completion are skipped; thus a different match will be inserted at each **CTRL-N** and **CTRL-P** (unless there is only one matching keyword).

Single character matches are never included, as they usually just get in the way of what you were really after.

```
e.g., to get:
    printf("name = %s\n", name);
just type:
    printf("name = %s\n", n^P);
or even:
    printf("name = %s\n", ^P);
```

The 'n' in '\n' is skipped.

After expanding a word, you can use **CTRL-X CTRL-P** or **CTRL-X CTRL-N** to get the word following the expansion in other contexts. These sequences search for the text just expanded and further expand by getting an extra word. This is useful if you need to repeat a sequence of complicated words. Although **CTRL-P** and **CTRL-N** look just for strings of at least two characters, **CTRL-X CTRL-P** and **CTRL-X CTRL-N** can be used to expand words of just one character.

```
e.g., to get:
    M&eacute;xico
you can type:
    M^N^P^X^P^X^P
```

CTRL-N starts the expansion and then **CTRL-P** takes back the single character "M", the next two **CTRL-X CTRL-P**'s get the words "é" and ";xico".

If the previous expansion was split, because it got longer than **'textwidth'**, then just the text in the current line will be used.

If the match found is at the end of a line, then the first word in the next line will be inserted and the message "Word from other line" displayed, if this word is accepted the next **CTRL-X CTRL-P** or **CTRL-X CTRL-N** will search for those lines starting with this word.

Completing keywords in **'dictionary'**

compl-dictionary

i_CTRL-X_CTRL-K

CTRL-X CTRL-K

Search the files given with the **'dictionary'** option for words that start with the keyword in front of the cursor. This is like **CTRL-N**, but only the dictionary files are searched, not the current file. The found keyword is inserted in front of the cursor. This could potentially be pretty slow, since all matches are found before the first match is used. By default, the **'dictionary'** option is empty.

For suggestions where to find a list of words, see the **'dictionary'** option.

'ignorecase', **'smartcase'** and **'infercase'** apply.

CTRL-K or

CTRL-N

Search forward for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-P

Search backwards for next matching keyword. This keyword replaces the previous matching keyword.

Completing words in **'thesaurus'**

compl-thesaurus

i_CTRL-X_CTRL-T

CTRL-X CTRL-T

Works as **CTRL-X** CTRL-K, but in a special way. It uses the **'thesaurus'** option instead of **'dictionary'**. If a match is found in the thesaurus file, all the remaining words on the same line are included as matches, even though they don't complete the word. Thus a word can be completely replaced.

CTRL-T or

CTRL-N

Search forward for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-P

Search backwards for next matching keyword. This keyword replaces the previous matching keyword.

In the file used by the **'thesaurus'** option each line in the file should contain words with similar meaning, separated by non-keyword characters (white space is preferred). Maximum line length is 510 bytes.

For an example, imagine the **'thesaurus'** file has a line like this:

angry furious mad enraged

Placing the cursor after the letters "ang" and typing **CTRL-X CTRL-T** would complete the word "angry"; subsequent presses would change the word to "furious", "mad" etc.

Other uses include translation between two languages, or grouping API functions by keyword.

An English word list was added to this github issue:
<https://github.com/vim/vim/issues/629#issuecomment-443293282>
Unpack thesaurus_pkg.zip, put the thesaurus.txt file somewhere, e.g.
~/.vim/thesaurus/english.txt, and the **'thesaurus'** option to this file name.

Completing keywords with **'thesaurusfunc'** compl-thesaurusfunc

If the **'thesaurusfunc'** option is set, then the user specified function is invoked to get the list of completion matches and the **'thesaurus'** option is not used. See [complete-functions](#) for an explanation of how the function is invoked and what it should return.

Here is an example that uses the "aiksaurus" command (provided by Magnus Groß):

```
func Thesaur(findstart, base)
  if a:findstart
    return searchpos('\<', 'bnW', line('.'))[1] - 1
  endif
  let res = []
  let h = ''
  for l in systemlist('aiksaurus ' .. shellescape(a:base))
    if l[:3] == '=== '
      let h = '(' .. substitute(l[4:], ' =*$', '), ')
    elseif l ==# 'Alphabetically similar known words are: '
      let h = "\U0001f52e"
    elseif l[0] =~ '\a' || (h ==# "\U0001f52e" && l[0] ==# "\t")
      call extend(res, map(split(substitute(l, '^\\t', '', ''), ', '), {_, val -> {'word'
    endif
  endfor
  return res
endfunc

if exists('+thesaurusfunc')
  set thesaurusfunc=Thesaur
endif
```

Completing keywords in the current and included files compl-keyword

The **'include'** option is used to specify a line that contains an include file name. The **'path'** option is used to search for include files.

i_CTRL-X_CTRL-I

CTRL-X CTRL-I Search for the first keyword in the current and included files that starts with the same characters as those before the cursor. The matched keyword is inserted in front of the cursor.

CTRL-N Search forwards for next matching keyword. This keyword replaces the previous matching keyword.
Note: **CTRL-I** is the same as <Tab>, which is likely to

be typed after a successful completion, therefore **CTRL-I** is not used for searching for the next match.

CTRL-P Search backward for previous matching keyword. This keyword replaces the previous matching keyword.

CTRL-X CTRL-I Further use of **CTRL-X CTRL-I** will copy the words following the previous expansion in other contexts unless a double **CTRL-X** is used.

Completing tags

`compl-tag`
`i_CTRL-X_CTRL-]`

CTRL-X CTRL-] Search for the first tag that starts with the same characters as before the cursor. The matching tag is inserted in front of the cursor. Alphabetic characters and characters in **'iskeyword'** are used to decide which characters are included in the tag name (same as for a keyword). See also **CTRL-]**. The **'showfulltag'** option can be used to add context from around the tag definition.

CTRL-] or
CTRL-N Search forwards for next matching tag. This tag replaces the previous matching tag.

CTRL-P Search backward for previous matching tag. This tag replaces the previous matching tag.

Completing file names

`compl-filename`
`i_CTRL-X_CTRL-F`

CTRL-X CTRL-F Search for the first file name that starts with the same characters as before the cursor. The matching file name is inserted in front of the cursor. Alphabetic characters and characters in **'isfname'** are used to decide which characters are included in the file name. **Note:** the **'path'** option is not used here (yet).

CTRL-F or
CTRL-N Search forwards for next matching file name. This file name replaces the previous matching file name.

CTRL-P Search backward for previous matching file name. This file name replaces the previous matching file name.

Completing definitions or macros

`compl-define`

The **'define'** option is used to specify a line that contains a definition. The **'include'** option is used to specify a line that contains an include file name. The **'path'** option is used to search for include files.

CTRL-X CTRL-D `i_CTRL-X_CTRL-D` Search in the current and included files for the

first definition (or macro) name that starts with the same characters as before the cursor. The found definition name is inserted in front of the cursor.

- CTRL-D** or **CTRL-N** Search forwards for next matching macro name. This macro name replaces the previous matching macro name.
- CTRL-P** Search backward for previous matching macro name. This macro name replaces the previous matching macro name.
- CTRL-X CTRL-D** Further use of **CTRL-X CTRL-D** will copy the words following the previous expansion in other contexts unless a double **CTRL-X** is used.

Completing Vim commands

[compl-vim](#)

Completion is context-sensitive. It works like on the Command-line. It completes an Ex command as well as its arguments. This is useful when writing a Vim script.

- CTRL-X CTRL-V** Guess what kind of item is in front of the cursor and find the first match for it.
Note: When **CTRL-V** is mapped you can often use **CTRL-Q** instead of [i_CTRL-Q](#).
- CTRL-V** or **CTRL-N** Search forwards for next match. This match replaces the previous one.
- CTRL-P** Search backwards for previous match. This match replaces the previous one.
- CTRL-X CTRL-V** Further use of **CTRL-X CTRL-V** will do the same as **CTRL-V**. This allows mapping a key to do Vim command completion, for example:
`:imap <Tab> <C-X><C-V>`

User defined completion

[compl-function](#)

Completion is done by a function that can be defined by the user with the '**completefunc**' option. See below for how the function is called and an example [complete-functions](#).

- CTRL-X CTRL-U** Guess what kind of item is in front of the cursor and find the first match for it.
- CTRL-U** or **CTRL-N** Use the next match. This match replaces the previous one.
- CTRL-P** Use the previous match. This match replaces the

previous one.

Omni completion

[compl-omni](#)

Completion is done by a function that can be defined by the user with the **'omnifunc'** option. This is to be used for filetype-specific completion.

See below for how the function is called and an example [complete-functions](#) . For remarks about specific filetypes see [compl-omni-filetypes](#) . More completion scripts will appear, check www.vim.org. Currently there is a first version for C++.

[i_CTRL-X_CTRL-O](#)

CTRL-X CTRL-O

Guess what kind of item is in front of the cursor and find the first match for it.

CTRL-O or
CTRL-N

Use the next match. This match replaces the previous one.

CTRL-P

Use the previous match. This match replaces the previous one.

Spelling suggestions

[compl-spelling](#)

A word before or at the cursor is located and correctly spelled words are suggested to replace it. If there is a badly spelled word in the line, before or under the cursor, the cursor is moved to after it. Otherwise the word just before the cursor is used for suggestions, even though it isn't badly spelled.

NOTE: **CTRL-S** suspends display in many Unix terminals. Use 's' instead. Type **CTRL-Q** to resume displaying.

[i_CTRL-X_CTRL-S](#) [i_CTRL-X_s](#)

CTRL-X CTRL-S or
CTRL-X s

Locate the word in front of the cursor and find the first spell suggestion for it.

CTRL-S or
CTRL-N

Use the next suggestion. This replaces the previous one. **Note** that you can't use 's' here.

CTRL-P

Use the previous suggestion. This replaces the previous one.

Completing keywords from different sources

[compl-generic](#)

[i_CTRL-N](#)

CTRL-N

Find next match for words that start with the keyword in front of the cursor, looking in places specified with the **'complete'** option. The found keyword is inserted in front of the cursor.

CTRL-P i_CTRL-P
 Find previous match for words that start with the keyword in front of the cursor, looking in places specified with the '**complete**' option. The found keyword is inserted in front of the cursor.

CTRL-N Search forward for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-P Search backwards for next matching keyword. This keyword replaces the previous matching keyword.

CTRL-X CTRL-N or **CTRL-X CTRL-P** Further use of **CTRL-X CTRL-N** or **CTRL-X CTRL-P** will copy the words following the previous expansion in other contexts unless a double **CTRL-X** is used.

Stop completion compl-stop

CTRL-X CTRL-Z i_CTRL-X_CTRL-Z
 Stop completion without changing the text.

FUNCTIONS FOR FINDING COMPLETIONS complete-functions

This applies to '**completefunc**', '**thesaurusfunc**' and '**omnifunc**'.

The function is called in two different ways:

- First the function is called to find the start of the text to be completed.
- Later the function is called to actually find the matches.

On the first invocation the arguments are:

```
a:findstart 1
a:base      empty
```

The function must return the column where the completion starts. It must be a number between zero and the cursor column "col('.')". This involves looking at the characters just before the cursor and including those characters that could be part of the completed item. The text between this column and the cursor column will be replaced with the matches. If the returned value is larger than the cursor column, the cursor column is used.

Negative return values:

- 2 To cancel silently and stay in completion mode.
 - 3 To cancel silently and leave completion mode.
- Another negative value: completion starts at the cursor column

On the second invocation the arguments are:

```
a:findstart 0
a:base      the text with which matches should match; the text that was
             located in the first call (can be empty)
```

The function must return a List with the matching words. These matches

usually include the "a:base" text. When there are no matches return an empty List. **Note** that the cursor may have moved since the first invocation, the text may have been changed.

In order to return more information than the matching words, return a Dict that contains the List. The Dict can have these items:

words	The List of matching words (mandatory).
refresh	A string to control re-invocation of the function (optional). The only value currently recognized is "always", the effect is that the function is called whenever the leading text is changed.

If you want to suppress the warning message for an empty result, return `v:none` . This is useful to implement asynchronous completion with `complete()` .

Other items are ignored.

For acting upon end of completion, see the `CompleteDonePre` and `CompleteDone` autocommand event.

For example, the function can contain this:

```
let matches = ... list of words ...  
return {'words': matches, 'refresh': 'always'}
```

complete-items

Each list item can either be a string or a Dictionary. When it is a string it is used as the completion. When it is a Dictionary it can contain these items:

word	the text that will be inserted, mandatory
abbr	abbreviation of "word"; when not empty it is used in the menu instead of "word"
menu	extra text for the popup menu, displayed after "word" or "abbr"
info	more information about the item, can be displayed in a preview or popup window
kind	single letter indicating the type of completion
icase	when non-zero case is to be ignored when comparing items to be equal; when omitted zero is used, thus items that only differ in case are added
equal	when non-zero, always treat this item to be equal when comparing. Which means, "equal=1" disables filtering of this item.
dup	when non-zero this match will be added even when an item with the same word is already present.
empty	when non-zero this match will be added even when it is an empty string
user_data	custom data which is associated with the item and available in <code>v:completed_item</code> ; it can be any type; defaults to an empty string

All of these except "icase", "equal", "dup" and "empty" must be a string. If an item does not meet these requirements then an error message is given and

further items in the list are not used. You can mix string and Dictionary items in the returned list.

The "menu" item is used in the popup menu and may be truncated, thus it should be relatively short. The "info" item can be longer, it will be displayed in the preview window when "preview" appears in **'completeopt'** or in a popup window when "popup" appears in **'completeopt'**. In the preview window the "info" item will also remain displayed after the popup menu has been removed. This is useful for function arguments. Use a single space for "info" to remove existing text in the preview window. The size of the preview window is three lines, but **'previewheight'** is used when it has a value of 1 or 2.

complete-popup

When "popup" is in **'completeopt'** a popup window is used to display the "info". Then the **'completepopup'** option specifies the properties of the popup. This is used when the info popup is created. The option is a comma-separated list of values:

height	maximum height of the popup
width	maximum width of the popup
highlight	highlight group of the popup (default is PmenuSel)
align	"item" (default) or "menu"
border	"on" (default) or "off"

Example:

```
:set completepopup=height:10,width:60,highlight:InfoPopup
```

When the "align" value is "item" then the popup is positioned close to the selected item. Changing the selection will also move the popup. When "align" is "menu" then the popup is aligned with the top of the menu if the menu is below the text, and the bottom of the menu otherwise.

After the info popup is created it can be found with `popup_findinfo()` and properties can be changed with `popup_setoptions()`.

complete-popuphidden

If the information for the popup is obtained asynchronously, use "popuphidden" in **'completeopt'**. The info popup will then be initially hidden and `popup_show()` must be called once it has been filled with the info. This can be done with a `CompleteChanged` autocommand, something like this:

```
set completeopt+=popuphidden
au CompleteChanged * call UpdateCompleteInfo()
func UpdateCompleteInfo()
  " Cancel any pending info fetch
  let item = v:event.completed_item
  " Start fetching info for the item then call ShowCompleteInfo(info)
endfunc
func ShowCompleteInfo(info)
  let id = popup_findinfo()
  if id
    call popup_settext(id, 'async info: ' .. a:info)
    call popup_show(id)
  endif
endfunc
```

complete-item-kind

The "kind" item uses a single letter to indicate the kind of completion. This may be used to show the completion differently (different color or icon). Currently these types can be used:

v	variable
f	function or method
m	member of a struct or class
t	typedef
d	#define or macro

When searching for matches takes some time call `complete_add()` to add each match to the total list. These matches should then not appear in the returned list! Call `complete_check()` now and then to allow the user to press a key while still searching for matches. Stop searching when it returns non-zero.

E840

The function is allowed to move the cursor, it is restored afterwards. The function is not allowed to move to another window or delete text.

An example that completes the names of the months:

```
fun! CompleteMonths(findstart, base)
  if a:findstart
    " locate the start of the word
    let line = getline('.')
    let start = col('.') - 1
    while start > 0 && line[start - 1] =~ '\a'
      let start -= 1
    endwhile
    return start
  else
    " find months matching with "a:base"
    let res = []
    for m in split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec")
      if m =~ '^' .. a:base
        call add(res, m)
      endif
    endfor
    return res
  endif
endfun
set completefunc=CompleteMonths
```

The same, but now pretending searching for matches is slow:

```
fun! CompleteMonths(findstart, base)
  if a:findstart
    " locate the start of the word
    let line = getline('.')
    let start = col('.') - 1
    while start > 0 && line[start - 1] =~ '\a'
      let start -= 1
    endwhile
    return start
  else
    " find months matching with "a:base"
    for m in split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec")
```

```

        if m =~ '^' .. a:base
            call complete_add(m)
        endif
        sleep 300m          " simulate searching for next match
        if complete_check()
            break
        endif
    endfor
    return []
endif
endfun
set completefunc=CompleteMonths

```

INSERT COMPLETION POPUP MENU

[ins-completion-menu](#)
[popupmenu-completion](#)

Vim can display the matches in a simplistic popup menu.

The menu is used when:

- The '**completeopt**' option contains "menu" or "menuone".
- The terminal supports at least 8 colors.
- There are at least two matches. One if "menuone" is used.

The '**pumheight**' option can be used to set a maximum height. The default is to use all space available.

The '**pumwidth**' option can be used to set a minimum width. The default is 15 characters.

There are three states:

1. A complete match has been inserted, e.g., after using **CTRL-N** or **CTRL-P**.
2. A cursor key has been used to select another match. The match was not inserted then, only the entry in the popup menu is highlighted.
3. Only part of a match has been inserted and characters were typed or the backspace key was used. The list of matches was then adjusted for what is in front of the cursor.

You normally start in the first state, with the first match being inserted. When "longest" is in '**completeopt**' and there is more than one match you start in the third state.

If you select another match, e.g., with **CTRL-N** or **CTRL-P**, you go to the first state. This doesn't change the list of matches.

When you are back at the original text then you are in the third state. To get there right away you can use a mapping that uses **CTRL-P** right after starting the completion:

```
:imap <F7> <C-N><C-P>
```

[popupmenu-keys](#)

In the first state these keys have a special meaning:

<BS> and **CTRL-H** Delete one character, find the matches for the word before the cursor. This reduces the list of matches, often to one entry, and switches to the second state.

Any non-special character:

Stop completion without changing the match and insert the typed character.

In the second and third state these keys have a special meaning:

<BS> and **CTRL-H** Delete one character, find the matches for the shorter word before the cursor. This may find more matches.

CTRL-L Add one character from the current match, may reduce the number of matches.

any printable, non-white character:

Add this character and reduce the number of matches.

In all three states these can be used:

CTRL-Y Yes: Accept the currently selected match and stop completion.

CTRL-E End completion, go back to what was there before selecting a match (what was typed or longest common string).

<PageUp> Select a match several entries back, but don't insert it.

<PageDown> Select a match several entries further, but don't insert it.

<Up> Select the previous match, as if **CTRL-P** was used, but don't insert it.

<Down> Select the next match, as if **CTRL-N** was used, but don't insert it.

<Space> or **<Tab>** Stop completion without changing the match and insert the typed character.

The behavior of the **<Enter>** key depends on the state you are in:

first state: Use the text as it is and insert a line break.

second state: Insert the currently selected match.

third state: Use the text as it is and insert a line break.

In other words: If you used the cursor keys to select another entry in the list of matches then the **<Enter>** key inserts that match. If you typed something else then **<Enter>** inserts a line break.

The colors of the menu can be changed with these highlight groups:

Pmenu	normal item	hl-Pmenu
PmenuSel	selected item	hl-PmenuSel
PmenuSbar	scrollbar	hl-PmenuSbar
PmenuThumb	thumb of the scrollbar	hl-PmenuThumb

There are no special mappings for when the popup menu is visible. However, you can use an Insert mode mapping that checks the `pumvisible()` function to do something different. Example:

```
:inoremap <Down> <C-R>=pumvisible() ? "\<lt>C-N" : "\<lt>Down"<CR>
```

You can use of `<expr>` in mapping to have the popup menu used when typing a character and some condition is met. For example, for typing a dot:

```
inoremap <expr> . MayComplete()  
func MayComplete()  
  if (can complete)  
    return ".\<C-X>\<C-O>"  
  endif  
  return '.'  
endfunc
```

See `:map-<expr>` for more info.

FILETYPE-SPECIFIC REMARKS FOR OMNI COMPLETION

`compl-omni-filetypes`

The file used for `{filetype}` should be `autoload/{filetype}complete.vim` in `'runtimepath'`. Thus for "java" it is `autoload/javacomplete.vim`.

C

`ft-c-omni`

Completion of C code requires a tags file. You should use Universal/Exuberant ctags, because it adds extra information that is needed for completion. You can find it here:

Universal Ctags: <https://ctags.io>

Exuberant Ctags: <http://ctags.sourceforge.net>

Universal Ctags is preferred, Exuberant Ctags is no longer being developed.

For Exuberant ctags, version 5.6 or later is recommended. For version 5.5.4 you should add a patch that adds the "typename:" field:

<ftp://ftp.vim.org/pub/vim/unstable/patches/ctags-5.5.4.patch>

A compiled .exe for MS-Windows can be found at:

<http://ctags.sourceforge.net/>

<https://github.com/universal-ctags/ctags-win32>

If you want to complete system functions you can do something like this. Use ctags to generate a tags file for all the system header files:

```
% ctags -R -f ~/.vim/systags /usr/include /usr/local/include
```

In your vimrc file add this tags file to the `'tags'` option:

```
set tags+=~/.vim/systags
```

When using **CTRL-X CTRL-O** after a name without any "." or "->" it is completed from the tags file directly. This works for any identifier, also function names. If you want to complete a local variable name, which does not appear in the tags file, use **CTRL-P** instead.

When using **CTRL-X CTRL-O** after something that has "." or "->" Vim will attempt to recognize the type of the variable and figure out what members it has. This means only members valid for the variable will be listed.

When a member name already was complete, **CTRL-X CTRL-O** will add a "." or "->" for composite types.

Vim doesn't include a C compiler, only the most obviously formatted declarations are recognized. Preprocessor stuff may cause confusion. When the same structure name appears in multiple places all possible members are included.

CSS

`ft-css-omni`

Complete properties and their appropriate values according to CSS 2.1

specification.

HTML
XHTML

[ft-html-omni](#)
[ft-xhtml-omni](#)

CTRL-X CTRL-O provides completion of various elements of (X)HTML files. It is designed to support writing of XHTML 1.0 Strict files but will also work for other versions of HTML. Features:

- after "<" complete tag name depending on context (no div suggestion inside of an a tag); '/>' indicates empty tags
- inside of tag complete proper attributes (no width attribute for an a tag); show also type of attribute; '*' indicates required attributes
- when attribute has limited number of possible values help to complete them
- complete names of entities
- complete values of "class" and "id" attributes with data obtained from `<style>` tag and included CSS files
- when completing value of "style" attribute or working inside of "style" tag switch to [ft-css-omni](#) completion
- when completing values of events attributes or working inside of "script" tag switch to [ft-javascript-omni](#) completion
- when used after "</" **CTRL-X CTRL-O** will close the last opened tag

Note: When used first time completion menu will be shown with little delay - this is time needed for loading of data file.

Note: Completion may fail in badly formatted documents. In such case try to run `:make` command to detect formatting problems.

HTML flavor

[html-flavor](#)

The default HTML completion depends on the filetype. For HTML files it is HTML 4.01 Transitional ('filetype' is "html"), for XHTML it is XHTML 1.0 Strict ('filetype' is "xhtml").

When doing completion outside of any other tag you will have possibility to choose DOCTYPE and the appropriate data file will be loaded and used for all next completions.

More about format of data file in [xml-omni-datafile](#) . Some of the data files may be found on the Vim website ([www](#)).

Note that `b:html_omni_flavor` may point to a file with any XML data. This makes possible to mix PHP ([ft-php-omni](#)) completion with any XML dialect (assuming you have data file for it). Without setting that variable XHTML 1.0 Strict will be used.

JAVASCRIPT

[ft-javascript-omni](#)

Completion of most elements of JavaScript language and DOM elements.

Complete:

- variables
- function name; show function arguments
- function arguments
- properties of variables trying to detect type of variable
- complete DOM objects and properties depending on context
- keywords of language

Completion works in separate JavaScript files (&ft==javascript), inside of `<script>` tag of (X)HTML and in values of event attributes (including scanning of external files).

DOM compatibility

At the moment (beginning of 2006) there are two main browsers - MS Internet Explorer and Mozilla Firefox. These two applications are covering over 90% of market. Theoretically standards are created by W3C organisation (<http://www.w3c.org>) but they are not always followed/implemented.

IE	FF	W3C	Omni completion
+/-	+/-	+	+
+	+	-	+
+	-	-	-
-	+	-	-

Regardless from state of implementation in browsers but if element is defined in standards, completion plugin will place element in suggestion list. When both major engines implemented element, even if this is not in standards it will be suggested. All other elements are not placed in suggestion list.

PHP

ft-php-omni

Completion of PHP code requires a tags file for completion of data from external files and for class aware completion. You should use Universal/Exuberant ctags version 5.5.4 or newer. You can find it here:

Universal Ctags: <https://ctags.io>
 Exuberant Ctags: <http://ctags.sourceforge.net>

Script completes:

- after \$ variables name
 - if variable was declared as object add "->", if tags file is available show name of class
 - after "->" complete only function and variable names specific for given class. To find class location and contents tags file is required. Because PHP isn't strongly typed language user can use @var tag to declare class:

```
/* @var $myVar myClass */
$myVar->
```

Still, to find myClass contents tags file is required.

- function names with additional info:
 - in case of built-in functions list of possible arguments and after | type data returned by function
 - in case of user function arguments and name of file where function was defined (if it is not current file)
- constants names
- class names after "new" declaration

Note: when doing completion first time Vim will load all necessary data into memory. It may take several seconds. After next use of completion delay should not be noticeable.

Script detects if cursor is inside `<?php ?>` tags. If it is outside it will automatically switch to HTML/CSS/JavaScript completion. **Note:** contrary to original HTML files completion of tags (and only tags) isn't context aware.

RUBY

ft-ruby-omni

Completion of Ruby code requires that vim be built with `+ruby`.

Ruby completion will parse your buffer on demand in order to provide a list of completions. These completions will be drawn from modules loaded by `'require'` and modules defined in the current buffer.

The completions provided by **CTRL-X CTRL-O** are sensitive to the context:

CONTEXT	COMPLETIONS PROVIDED
1. Not inside a class definition	Classes, constants and globals
2. Inside a class definition	Methods or constants defined in the class
3. After '.', '::' or ':'	Methods applicable to the object being dereferenced
4. After ':' or ':foo'	Symbol name (beginning with <code>'foo'</code>)

Notes:

- Vim will load/evaluate code in order to provide completions. This may cause some code execution, which may be a concern. This is no longer enabled by default, to enable this feature add


```
let g:rubycomplete_buffer_loading = 1
```
- In context 1 above, Vim can parse the entire buffer to add a list of classes to the completion results. This feature is turned off by default, to enable it add


```
let g:rubycomplete_classes_in_global = 1
```

 to your vimrc
- In context 2 above, anonymous classes are not supported.
- In context 3 above, Vim will attempt to determine the methods supported by the object.
- Vim can detect and load the Rails environment for files within a rails

project. The feature is disabled by default, to enable it add
`let g:rubycomplete_rails = 1`
to your vimrc

SYNTAX

ft-syntax-omni

Vim has the ability to color syntax highlight nearly 500 languages. Part of this highlighting includes knowing what keywords are part of a language. Many filetypes already have custom completion scripts written for them, the `syntaxcomplete` plugin provides basic completion for all other filetypes. It does this by populating the omni completion list with the text Vim already knows how to color highlight. It can be used for any filetype and provides a minimal language-sensitive completion.

To enable syntax code completion you can run:

```
setlocal omnifunc=syntaxcomplete#Complete
```

You can automate this by placing the following in your `.vimrc` (after any `":filetype"` command):

```
if has("autocmd") && exists("+omnifunc")
    autocmd Filetype *
        \   if &omnifunc == "" |
        \       setlocal omnifunc=syntaxcomplete#Complete |
        \   endif
endif
```

The above will set completion to this script only if a specific plugin does not already exist for that filetype.

Each filetype can have a wide range of syntax items. The plugin allows you to customize which syntax groups to include or exclude from the list. Let's have a look at the PHP filetype to see how this works.

If you edit a file called, `index.php`, run the following command:

```
syntax list
```

The first thing you will notice is that there are many different syntax groups. The PHP language can include elements from different languages like HTML, JavaScript and many more. The syntax plugin will only include syntax groups that begin with the filetype, "php", in this case. For example these syntax groups are included by default with the PHP: `phpEnvVar`, `phpIntVar`, `phpFunctions`.

If you wish non-filetype syntax items to also be included, you can use a regular expression syntax (added in version 13.0 of `autoload/syntaxcomplete.vim`) to add items. Looking at the output from `":syntax list"` while editing a PHP file I can see some of these entries:

```
htmlArg,htmlTag,htmlTagName,javascriptStatement,javascriptGlobalObjects
```

To pick up any JavaScript and HTML keyword syntax groups while editing a PHP file, you can use 3 different regexs, one for each language. Or you can simply restrict the include groups to a particular value, without using a regex string:

```
let g:omni_syntax_group_include_php = 'php\w\+,javaScript\w\+,html\w\+'
let g:omni_syntax_group_include_php = 'phpFunctions,phpMethods'
```

The basic form of this variable is:

```
let g:omni_syntax_group_include_{filetype} = 'regex,comma,separated'
```

The PHP language has an enormous number of items which it knows how to syntax highlight. These items will be available within the omni completion list.

Some people may find this list unwieldy or are only interested in certain items. There are two ways to prune this list (if necessary). If you find certain syntax groups you do not wish displayed you can use two different methods to identify these groups. The first specifically lists the syntax groups by name. The second uses a regular expression to identify both syntax groups. Simply add one the following to your vimrc:

```
let g:omni_syntax_group_exclude_php = 'phpCoreConstant,phpConstant'
let g:omni_syntax_group_exclude_php = 'php\w*Constant'
```

Add as many syntax groups to this list by comma separating them. The basic form of this variable is:

```
let g:omni_syntax_group_exclude_{filetype} = 'regex,comma,separated'
```

You can create as many of these variables as you need, varying only the filetype at the end of the variable name.

The plugin uses the `isKeyword` option to determine where word boundaries are for the syntax items. For example, in the Scheme language completion should include the `"-"`, `call-with-output-file`. Depending on your filetype, this may not provide the words you are expecting. Setting the `g:omni_syntax_use_iskeyword` option to `0` will force the syntax plugin to break on word characters. This can be controlled adding the following to your vimrc:

```
let g:omni_syntax_use_iskeyword = 0
```

For plugin developers, the plugin exposes a public function `OmniSyntaxList`. This function can be used to request a List of syntax items. When editing a SQL file (`:e syntax.sql`) you can use the `":syntax list"` command to see the various groups and syntax items. For example:

```
syntax list
```

Yields data similar to this:

```
sqlOperator    xxx some prior all like and any escape exists in is not
                or intersect minus between distinct
                links to Operator
sqlType        xxx varbit varchar nvarchar bigint int uniqueidentifier
                date money long tinyint unsigned xml text smalldate
                double datetime nchar smallint numeric time bit char
                varbinary binary smallmoney
                image float integer timestamp real decimal
```

There are two syntax groups listed here: `sqlOperator` and `sqlType`. To retrieve a List of syntax items you can call `OmniSyntaxList` a number of different ways. To retrieve all syntax items regardless of syntax group:

```
echo OmniSyntaxList( [] )
```

To retrieve only the syntax items for the sqlOperator syntax group:
`echo OmniSyntaxList(['sqlOperator'])`

To retrieve all syntax items for both the sqlOperator and sqlType groups:
`echo OmniSyntaxList(['sqlOperator', 'sqlType'])`

A regular expression can also be used:
`echo OmniSyntaxList(['sql\w\+')]`

From within a plugin, you would typically assign the output to a List:
`let myKeywords = []
let myKeywords = OmniSyntaxList(['sqlKeyword'])`

SQL

[ft-sql-omni](#)

Completion for the SQL language includes statements, functions, keywords. It will also dynamically complete tables, procedures, views and column lists with data pulled directly from within a database. For detailed instructions and a tutorial see [omni-sql-completion](#) .

The SQL completion plugin can be used in conjunction with other completion plugins. For example, the PHP filetype has its own completion plugin. Since PHP is often used to generate dynamic website by accessing a database, the SQL completion plugin can also be enabled. This allows you to complete PHP code and SQL code at the same time.

XML

[ft-xml-omni](#)

Vim 7 provides a mechanism for context aware completion of XML files. It depends on a special [xml-omni-datafile](#) and two commands: `:XMLns` and `:XMLent` . Features are:

- after "<" complete the tag name, depending on context
- inside of a tag complete proper attributes
- when an attribute has a limited number of possible values help to complete them
- complete names of entities (defined in [xml-omni-datafile](#) and in the current file with "<!ENTITY" declarations)
- when used after "</" **CTRL-X CTRL-O** will close the last opened tag

Format of XML data file

[xml-omni-datafile](#)

XML data files are stored in the "autoload/xml" directory in **'runtimepath'**. Vim distribution provides examples of data files in the "\$VIMRUNTIME/autoload/xml" directory. They have a meaningful name which will be used in commands. It should be a unique name which will not create conflicts. For example, the name xhtml10s.vim means it is the data file for XHTML 1.0 Strict.

Each file contains a variable with a name like g:xmldata_xhtml10s . It is a compound from two parts:

1. "g:xmldata_" general prefix, constant for all data files
2. "xhtml10s" the name of the file and the name of the described XML dialect; it will be used as an argument for the `:XMLns` command

Part two must be exactly the same as name of file.

The variable is a `Dictionary`. Keys are tag names and each value is a two element `List`. The first element of the List is also a List with the names of possible children. The second element is a `Dictionary` with the names of attributes as keys and the possible values of attributes as values. Example:

```
let g:xmldata_crippled = {
  \ "vimxmlentities": ["amp", "lt", "gt", "apos", "quot"],
  \ 'vimxmlroot': ['tag1'],
  \ 'tag1':
  \ [ ['chldoftag1a', 'chldoftag1b'], {'attroftag1a': [],
  \ 'attroftag1b': ['valueofattr1', 'valueofattr2']}],
  \ 'chldoftag1a':
  \ [ [], {'attrofchild': ['attrofchild']}],
  \ 'chldoftag1b':
  \ [ ['chldoftag1a'], {'attrofchild': []}],
  \ "vimxmltaginfo": {
  \ 'tag1': ['Menu info', 'Long information visible in preview window']},
  \ 'vimxmlattrinfo': {
  \ 'attrofchild': ['Menu info', 'Long information visible in preview window']}]}
```

This example would be put in the "autoload/xml/crippled.vim" file and could help to write this file:

```
<tag1 attroftag1b="valueofattr1">
  <chldoftag1a attrofchild>
    &amp; &lt;
  </chldoftag1a>
  <chldoftag1b attrofchild="5">
    <chldoftag1a>
      &gt; &apos; &quot;
    </chldoftag1a>
  </chldoftag1b>
</tag1>
```

In the example four special elements are visible:

1. "vimxmlentities" - a special key with List containing entities of this XML dialect.
2. If the list containing possible values of attributes has one element and this element is equal to the name of the attribute this attribute will be treated as boolean and inserted as `'attrname'` and not as `'attrname="'`
3. "vimxmltaginfo" - a special key with a Dictionary containing tag names as keys and two element List as values, for additional menu info and the long description.
4. "vimxmlattrinfo" - special key with Dictionary containing attribute names as keys and two element List as values, for additional menu info and long

description.

Note: Tag names in the data file MUST not contain a namespace description. Check `xsl.vim` for an example.

Note: All data and functions are publicly available as global variables/functions and can be used for personal editing functions.

DTD -> Vim

`dtd2vim`

On [www](http://www.vim.org) is the script `dtd2vim` which parses DTD and creates an XML data file for Vim XML omni completion.

`dtd2vim`: http://www.vim.org/scripts/script.php?script_id=1462

Check the beginning of that file for usage details. The script requires perl and:

perlSGML: <http://savannah.nongnu.org/projects/perlsgml>

Commands

`:XMLns {name} [{namespace}]`

`:XMLns`

Vim has to know which data file should be used and with which namespace. For loading of the data file and connecting data with the proper namespace use `:XMLns` command. The first (obligatory) argument is the name of the data (`xhtml10s`, `xsl`). The second argument is the code of namespace (`h`, `xsl`). When used without a second argument the dialect will be used as default - without namespace declaration. For example to use XML completion in `.xsl` files:

```
:XMLns xhtml10s
:XMLns xsl xsl
```

`:XMLent {name}`

`:XMLent`

By default entities will be completed from the data file of the default namespace. The `XMLent` command should be used in case when there is no default namespace:

```
:XMLent xhtml10s
```

Usage

While used in this situation (after declarations from previous part, `|` is cursor position):

```
<|
```

Will complete to an appropriate XHTML tag, and in this situation:

```
<xsl:|
```

Will complete to an appropriate XSL tag.

The script `xmlcomplete.vim`, provided through the `autoload` mechanism, has the `xmlcomplete#GetLastOpenTag()` function which can be used in XML files to get the name of the last open tag (`b:unaryTagsStack` has to be defined):

```
:echo xmlcomplete#GetLastOpenTag("b:unaryTagsStack")
```

8. Insert mode commands

inserting

The following commands can be used to insert new text into the buffer. They can all be undone and repeated with the `."` command.

`a` Append text after the cursor `[count]` times. If the cursor is in the first column of an empty line Insert starts there. But not when `'virtualedit'` is set!

`A` Append text at the end of the line `[count]` times.
For using "A" in Visual block mode see `v_b_A`.

`<insert>` or `i` Insert text before the cursor `[count]` times.
When using `CTRL-O` in Insert mode `i_CTRL-O` the count is not supported.

`I` Insert text before the first non-blank in the line `[count]` times.
When the 'H' flag is present in `'coptions'` and the line only contains blanks, insert start just before the last blank.
For using "I" in Visual block mode see `v_b_I`.

`gI` Insert text in column 1 `[count]` times.

`gi` Insert text in the same position as where Insert mode was stopped last time in the current buffer.
This uses the `'^` mark. It's different from `"^i`" when the mark is past the end of the line.
The position is corrected for inserted/deleted lines, but NOT for inserted/deleted characters.
When the `:keepjumps` command modifier is used the `'^` mark won't be changed.

o

o Begin a new line below the cursor and insert text, repeat [count] times. When the '#' flag is in 'coptions' the count is ignored.

0 Begin a new line above the cursor and insert text, repeat [count] times. When the '#' flag is in 'coptions' the count is ignored.

These commands are used to start inserting text. You can end insert mode with <Esc>. See [mode-ins-repl](#) for the other special characters in Insert mode. The effect of [count] takes place after Insert mode is exited.

When 'autoindent' is on, the indent for a new line is obtained from the previous line. When 'smartindent' or 'cindent' is on, the indent for a line is automatically adjusted for C programs.

'formatoptions' can be set to copy the comment leader when opening a new line.

'textwidth' can be set to the maximum width for a line. When a line becomes too long when appending characters a line break is automatically inserted.

9. Ex insert commands inserting-ex

:a :append

:{range}a[ppend][!] Insert several lines of text below the specified line. If the {range} is missing, the text will be inserted after the current line. Adding [!] toggles 'autoindent' for the time this command is executed. This command is not supported in Vim9 script, because it is too easily confused with a variable name.

:i :in :insert

:{range}i[nsert][!] Insert several lines of text above the specified line. If the {range} is missing, the text will be inserted before the current line. Adding [!] toggles 'autoindent' for the time this command is executed. This command is not supported in Vim9 script, because it is too easily confused with a variable name.

These two commands will keep on asking for lines, until you type a line containing only a ".". Watch out for lines starting with a backslash, see [line-continuation](#) .

When in Ex mode (see [-e](#)) a backslash at the end of the line can be used to

insert a NUL character. To be able to have a line ending in a backslash use two backslashes. This means that the number of backslashes is halved, but only at the end of the line.

NOTE: These commands cannot be used with `:global` or `:vglobal`.
":append" and ":insert" don't work properly in between ":if" and ":endif", ":for" and ":endfor", ":while" and ":endwhile".

`:start` `:startinsert`
`:star[tinsert][!]` Start Insert mode just after executing this command. Works like typing "i" in Normal mode. When the ! is included it works like "A", append to the line. Otherwise insertion starts at the cursor position. **Note** that when using this command in a function or script, the insertion only starts after the function or script is finished. This command does not work from `:normal`.

`:stopi` `:stopinsert`
`:stopi[nsert]` Stop Insert mode as soon as possible. Works like typing <Esc> in Insert mode. Can be used in an autocommand, example:
`:au BufEnter scratch stopinsert`

`replacing-ex` `:startreplace`
`:startr[eplace][!]` Start Replace mode just after executing this command. Works just like typing "R" in Normal mode. When the ! is included it acts just like "\$R" had been typed (ie. begin replace mode at the end-of-line). Otherwise replacement begins at the cursor position. **Note** that when using this command in a function or script that the replacement will only start after the function or script is finished.

`:startgreplace`
`:startg[replace][!]` Just like `:startreplace`, but use Virtual Replace mode, like with `gR`.

10. Inserting a file

inserting-file

`:r` `:re` `:read`
`:r[ead] [++opt] [name]` Insert the file `[name]` (default: current file) below the cursor. See `++opt` for the possible values of `[++opt]`.

`:{range}r` `:read`
`:{range}r[ead] [++opt] [name]` Insert the file `[name]` (default: current file) below the specified line. See `++opt` for the possible values of `[++opt]`.

`:r!` `:read!`
`:[range]r[ead] [++opt] !{cmd}`

Execute `{cmd}` and insert its standard output below the cursor or the specified line. A temporary file is used to store the output of the command which is then read into the buffer. `'shellredir'` is used to save the output of the command, which can be set to include `stderr` or not. `{cmd}` is executed like with `:{!}{cmd}`, any `!` is replaced with the previous command `:! .`. See `++opt` for the possible values of `[++opt]`.

These commands insert the contents of a file, or the output of a command, into the buffer. They can be undone. They cannot be repeated with the `."` command. They work on a line basis, insertion starts below the line in which the cursor is, or below the specified line. To insert text above the first line use the command `":0r {name}"`.

After the `":read"` command, the cursor is left on the first non-blank in the first new line. Unless in Ex mode, then the cursor is left on the last new line (sorry, this is Vi compatible).

If a file name is given with `":r"`, it becomes the alternate file. This can be used, for example, when you want to edit that file instead: `":e! #"`. This can be switched off by removing the `'a'` flag from the `'coptions'` option.

Of the `[++opt]` arguments one is specifically for `":read"`, the `++edit` argument. This is useful when the `":read"` command is actually used to read a file into the buffer as if editing that file. Use this command in an empty buffer:

```
:read ++edit filename
```

The effect is that the `'fileformat'`, `'fileencoding'`, `'bomb'`, etc. options are set to what has been detected for "filename". Note that a single empty line remains, you may want to delete it.

file-read

The `'fileformat'` option sets the `<EOL>` style for a file:

<code>'fileformat'</code>	characters	name
"dos"	<code><CR><NL></code> or <code><NL></code>	DOS format
"unix"	<code><NL></code>	Unix format
"mac"	<code><CR></code>	Mac format

Previously `'textmode'` was used. It is obsolete now.

If `'fileformat'` is "dos", a `<CR>` in front of an `<NL>` is ignored and a **CTRL-Z** at the end of the file is ignored.

If `'fileformat'` is "mac", a `<NL>` in the file is internally represented by a `<CR>`. This is to avoid confusion with a `<NL>` which is used to represent a `<NUL>`. See `CR-used-for-NL`.

If the `'fileformats'` option is not empty Vim tries to recognize the type of `<EOL>` (see `file-formats`). However, the `'fileformat'` option will not be changed, the detected format is only used while reading the file. A similar thing happens with `'fileencodings'`.

On non-Win32 systems the message "[dos format]" is shown if a file is read in DOS format, to remind you that something unusual is done.

On Macintosh and Win32 the message "[unix format]" is shown if a file is read

in Unix format.

On non-Macintosh systems, the message "[mac format]" is shown if a file is read in Mac format.

An example on how to use ":r !":

```
:r !uuencode binfile binfile
```

This command reads "binfile", uuencodes it and reads it into the current buffer. Useful when you are editing e-mail and want to include a binary file.

read-messages

When reading a file Vim will display a message with information about the read file. In the table is an explanation for some of the items. The others are self explanatory. Using the long or the short version depends on the '**shortmess**' option.

long	short	meaning
[readonly]	{RO}	the file is write protected
[fifo/socket]		using a stream
[fifo]		using a fifo stream
[socket]		using a socket stream
[CR missing]		reading with "dos" ' fileformat ' and a NL without a preceding CR was found.
[NL found]		reading with "mac" ' fileformat ' and a NL was found (could be "unix" format)
[long lines split]		at least one line was split in two
[NOT converted]		conversion from ' fileencoding ' to ' encoding ' was desired but not possible
[converted]		conversion from ' fileencoding ' to ' encoding ' done
[crypted]		file was decrypted
[READ ERRORS]		not all of the file could be read

VIM REFERENCE MANUAL by Bram Moolenaar

This file describes commands that delete or change text. In this context, changing text means deleting the text and replacing it with other text using one command. You can undo all of these commands. You can repeat the non-Ex commands with the "." command.

- | | | |
|----------------------------|----------------|----------|
| 1. Deleting text | deleting | |
| 2. Delete and insert | delete-insert | |
| 3. Simple changes | simple-change | changing |
| 4. Complex changes | complex-change | |
| 4.1 Filter commands | filter | |
| 4.2 Substitute | :substitute | |
| 4.3 Search and replace | search-replace | |
| 4.4 Changing tabs | change-tabs | |
| 5. Copying and moving text | copy-move | |
| 6. Formatting text | formatting | |
| 7. Sorting text | sorting | |

For inserting text see [insert.txt](#) .

-
- | | | |
|-------------------------|--|------|
| 1. Deleting text | deleting | E470 |
| ["x] or | | x dl |
| ["x]x | Delete [count] characters under and after the cursor [into register x] (not linewise). Does the same as "dl". | |
| | The key does not take a [count]. Instead, it deletes the last character of the count. | |
| | See :fixdel if the key does not do what you want. See ' whichwrap ' for deleting a line break (join lines). | |
| ["x]X | Delete [count] characters before the cursor [into register x] (not linewise). Does the same as "dh". Also see ' whichwrap ' . | X dh |
| ["x]d{motion} | Delete text that {motion} moves over [into register x]. See below for exceptions. | d |
| ["x]dd | Delete [count] lines [into register x] linewise . | dd |
| ["x]D | Delete the characters under the cursor until the end of the line and [count]-1 more lines [into register x]; synonym for "d\$". | D |

(not `linewise`)

When the '#' flag is in '`coptions`' the count is ignored.

`{Visual}["x]x` or `v_x` `v_d` `v_`
`{Visual}["x]d` or
`{Visual}["x]` Delete the highlighted text [into register x] (for `{Visual}` see `Visual-mode`).

`{Visual}["x]CTRL-H` or `v_CTRL-H` `v_<BS>`
`{Visual}["x]<BS>` When in Select mode: Delete the highlighted text [into register x].

`{Visual}["x]X` or `v_X` `v_D` `v_b_D`
`{Visual}["x]D` Delete the highlighted lines [into register x] (for `{Visual}` see `Visual-mode`). In Visual block mode, "D" deletes the highlighted text plus all text until the end of the line.

`:[range]d[el]e [x]` `:d` `:de` `:del` `:delete` `:dl` `:dp`
Delete `[range]` lines (default: current line) [into register x].

Note these weird abbreviations:

<code>:dl</code>	delete and list
<code>:dell</code>	idem
<code>:delel</code>	idem
<code>:deletl</code>	idem
<code>:deletel</code>	idem
<code>:dp</code>	delete and print
<code>:dep</code>	idem
<code>:delp</code>	idem
<code>:delep</code>	idem
<code>:deletp</code>	idem
<code>:deletp</code>	idem

`:[range]d[el]e [x] {count}`
Delete `{count}` lines, starting with `[range]` (default: current line `cmdline-ranges`) [into register x].

These commands delete text. You can repeat them with the ``.`` command (except `:`:d``) and undo them. Use Visual mode to delete blocks of text. See `registers` for an explanation of registers.

An exception for the `d{motion}` command: If the motion is not `linewise`, the start and end of the motion are not in the same line, and there are only blanks before the start and there are no non-blanks after the end of the motion, the delete becomes `linewise`. This means that the delete also removes the line of blanks that you might expect to remain. Use the `o_v` operator to force the motion to be `characterwise`.

Trying to delete an empty region of text (e.g., "d0" in the first column) is an error when '`coptions`' includes the 'E' flag.

`J` Join `[count]` lines, with a minimum of two lines. Remove the indent and insert up to two spaces (see below). Fails when on the last line of the buffer. If `[count]` is too big it is reduced to the number of lines available.

`{Visual}J` Join the highlighted lines, with a minimum of two lines. Remove the indent and insert up to two spaces (see below).

`gJ` Join `[count]` lines, with a minimum of two lines. Don't insert or remove any spaces.

`{Visual}gJ` Join the highlighted lines, with a minimum of two lines. Don't insert or remove any spaces.

`:[range]j[oin][!] [flags]` Join `[range]` lines. Same as "J", except with `[!]` the join does not insert or delete any spaces. If a `[range]` has equal start and end values, this command does nothing. The default behavior is to join the current line with the line below it. See `ex-flags` for `[flags]`.

`:[range]j[oin][!] {count} [flags]` Join `{count}` lines, starting with `[range]` (default: current line `cmdline-ranges`). Same as "J", except with `[!]` the join does not insert or delete any spaces. See `ex-flags` for `[flags]`.

These commands delete the `<EOL>` between lines. This has the effect of joining multiple lines into one line. You can repeat these commands (except ``:j``) and undo them.

These commands, except "gJ", insert one space in place of the `<EOL>` unless there is trailing white space or the next line starts with a ')'. These commands, except "gJ", delete any leading white space on the next line. If the `'joinspaces'` option is on, these commands insert two spaces after a '.', '!' or '?' (but if `'coptions'` includes the 'j' flag, they insert two spaces only after a '.').

The 'B' and 'M' flags in `'formatoptions'` change the behavior for inserting spaces before and after a multibyte character `fo-table`.

The '[' mark is set at the end of the first line that was joined, ']' at the end of the resulting line.

=====

2. Delete and insert

delete-insert replacing

	R	
R		Enter Replace mode: Each character you type replaces an existing character, starting with the character under the cursor. Repeat the entered text [count]-1 times. See Replace-mode for more details.
	gR	
gR		Enter Virtual Replace mode: Each character you type replaces existing characters in screen space. So a <Tab> may replace several characters at once. Repeat the entered text [count]-1 times. See Virtual-Replace-mode for more details.
	c	
["x]c{motion}		Delete {motion} text [into register x] and start insert. When ' coptions ' includes the 'E' flag and there is no text to delete (e.g., with "cTx" when the cursor is just after an 'x'), an error occurs and insert mode does not start (this is Vi compatible). When ' coptions ' does not include the 'E' flag, the "c" command always starts insert mode, even if there is no text to delete.
	cc	
["x]cc		Delete [count] lines [into register x] and start insert linewise . If ' autoindent ' is on, preserve the indent of the first line.
	C	
["x]C		Delete from the cursor position to the end of the line and [count]-1 more lines [into register x], and start insert. Synonym for c\$ (not linewise).
	s	
["x]s		Delete [count] characters [into register x] and start insert (s stands for Substitute). Synonym for "cl" (not linewise).
	S	
["x]S		Delete [count] lines [into register x] and start insert. Synonym for "cc" linewise .
{Visual}["x]c or {Visual}["x]s		v_c v_s Delete the highlighted text [into register x] and start insert (for {Visual} see Visual-mode).
	v_r	
{Visual}r{char}		Replace all selected characters by {char}. CTRL-C will be inserted literally.
	v_C	
{Visual}["x]C		Delete the highlighted lines [into register x] and

start insert. In Visual block mode it works differently `v_b_C` .

`{Visual}["x]S` Delete the highlighted lines [into register x] and start insert (for `{Visual}` see `Visual-mode`).

`{Visual}["x]R` Currently just like `{Visual}["x]S`. In a next version it might work differently.

Notes:

- You can end Insert and Replace mode with `<Esc>`.
- See the section "Insert and Replace mode" `mode-ins-repl` for the other special characters in these modes.
- The effect of `[count]` takes place after Vim exits Insert or Replace mode.
- When the `'coptions'` option contains '\$' and the change is within one line, Vim continues to show the text to be deleted and puts a '\$' at the last deleted character.

See `registers` for an explanation of registers.

Replace mode is just like Insert mode, except that every character you enter deletes one character. If you reach the end of a line, Vim appends any further characters (just like Insert mode). In Replace mode, the backspace key restores the original text (if there was any). (See section "Insert and Replace mode" `mode-ins-repl`).

Special case: When the cursor is in a word, `"cw` and `"cW` do not include the white space after a word, they only change up to the end of the word. This is because Vim interprets `"cw` as change-word, and a word does not include the following white space.

{Vi: `"cw` when on a blank followed by other blanks changes only the first blank; this is probably a bug, because `"dw` deletes all the blanks; use the 'w' flag in `'coptions'` to make it work like Vi anyway}

If you prefer `"cw` to include the space after a word, use this mapping:

```
:map cw dwi
```

Or use `"caw` (see `aw`).

`:{range}c[hange][!]` Replace lines of text with some different text. Type a line containing only "." to stop replacing. Without `{range}`, this command changes only the current line. Adding `[!]` toggles `'autoindent'` for the time this command is executed. This command is not supported in Vim9 script, because it is too easily confused with a variable name.

3. Simple changes

`simple-change`

`r`

`r{char}` Replace the character under the cursor with `{char}`. If `{char}` is a `<CR>` or `<NL>`, a line break replaces the character. To replace with a real `<CR>`, use **CTRL-V** `<CR>`. **CTRL-V** `<NL>` replaces with a `<Nul>`.

If `{char}` is **CTRL-E** or **CTRL-Y** the character from the line below or above is used, just like with `i_CTRL-E` and `i_CTRL-Y`. This also works with a count, thus ``10r<C-E>`` copies 10 characters from the line below.

If you give a `[count]`, Vim replaces `[count]` characters with `[count]` `{char}`s. When `{char}` is a `<CR>` or `<NL>`, however, Vim inserts only one `<CR>`: `"5r<CR>"` replaces five characters with a single line break. When `{char}` is a `<CR>` or `<NL>`, Vim performs autoindenting. This works just like deleting the characters that are replaced and then doing `"i<CR><Esc>"`.

`{char}` can be entered as a digraph `digraph-arg`. `:lmap` mappings apply to `{char}`. The **CTRL-^** command in Insert mode can be used to switch this on/off `i_CTRL-^`. See `utf-8-char-arg` about using composing characters when `'encoding'` is Unicode.

`gr{char}` Replace the virtual characters under the cursor with `{char}`. This replaces in screen space, not file space. See `gR` and `Virtual-Replace-mode` for more details. As with `r` a count may be given. `{char}` can be entered like with `r`, but characters that have a special meaning in Insert mode, such as most `CTRL-keys`, cannot be used.

The argument for Normal mode commands like `r` and `t` is a single character. When `'cpo'` doesn't contain the `'D'` flag, this character can also be entered like `digraphs`. First type `CTRL-K` and then the two digraph characters. `{not available when compiled without the |+digraphs| feature}`

The following commands change the case of letters. The currently active `locale` is used. See `:language`. The `LC_CTYPE` value matters here.

`~` **'notildeop'** option: Switch case of the character under the cursor and move the cursor to the right. If a `[count]` is given, do that many characters.

`~{motion}` **'tildeop'** option: switch case of `{motion}` text.

`g~{motion}` Switch case of `{motion}` text.

`g~g~` `g~g~` `g~~`

<code>g~~</code>	Switch case of current line.	
<code>{Visual}~</code>	Switch case of highlighted text (for <code>{Visual}</code> see <code>Visual-mode</code>).	<code>v_~</code>
<code>{Visual}U</code>	Make highlighted text uppercase (for <code>{Visual}</code> see <code>Visual-mode</code>).	<code>v_U</code>
<code>gU{motion}</code>	Make <code>{motion}</code> text uppercase. Example: <code>:map! <C-F> <Esc>gUiw`]a</code> This works in Insert mode: press CTRL-F to make the word before the cursor uppercase. Handy to type words in lowercase and then make them uppercase.	<code>gU uppercase</code>
<code>gUgU</code> <code>gUU</code>	Make current line uppercase.	<code>gUgU gUU</code>
<code>{Visual}u</code>	Make highlighted text lowercase (for <code>{Visual}</code> see <code>Visual-mode</code>).	<code>v_u</code>
<code>gu{motion}</code>	Make <code>{motion}</code> text lowercase.	<code>gu lowercase</code>
<code>gugu</code> <code>guu</code>	Make current line lowercase.	<code>gugu guu</code>
<code>g?{motion}</code>	Rot13 encode <code>{motion}</code> text.	<code>g? rot13</code>
<code>{Visual}g?</code>	Rot13 encode the highlighted text (for <code>{Visual}</code> see <code>Visual-mode</code>).	<code>v_g?</code>
<code>g?g?</code> <code>g??</code>	Rot13 encode current line.	<code>g?g? g??</code>

To turn one line into title caps, make every first letter of a word uppercase:

```
:s/\v<(.)\w*/\u\1\L2/g
```

Adding and subtracting

CTRL-A	Add <code>[count]</code> to the number or alphabetic character at or after the cursor.	<code>CTRL-A</code>
<code>{Visual}CTRL-A</code>	Add <code>[count]</code> to the number or alphabetic character in	<code>v_CTRL-A</code>

the highlighted text.

`{Visual}g CTRL-A` `v_g_CTRL-A`
Add `[count]` to the number or alphabetic character in the highlighted text. If several lines are highlighted, each one will be incremented by an additional `[count]` (so effectively creating a `[count]` incrementing sequence).
For Example, if you have this list of numbers:

```
1.  
1.  
1.  
1.
```

Move to the second "1." and Visually select three lines, pressing `g CTRL-A` results in:

```
1.  
2.  
3.  
4.
```

`CTRL-X` `CTRL-X`
Subtract `[count]` from the number or alphabetic character at or after the cursor.

`{Visual}CTRL-X` `v_CTRL-X`
Subtract `[count]` from the number or alphabetic character in the highlighted text.

On MS-Windows, this is mapped to cut Visual text `dos-standard-mappings`. If you want to disable the mapping, use this:

```
silent! vunmap <C-X>
```

`{Visual}g CTRL-X` `v_g_CTRL-X`
Subtract `[count]` from the number or alphabetic character in the highlighted text. If several lines are highlighted, each value will be decremented by an additional `[count]` (so effectively creating a `[count]` decrementing sequence).

The `CTRL-A` and `CTRL-X` commands can work for:

- signed and unsigned decimal numbers
- unsigned binary, octal and hexadecimal numbers
- alphabetic characters

This depends on the `'nrformats'` option:

- When `'nrformats'` includes "bin", Vim assumes numbers starting with '0b' or '0B' are binary.
- When `'nrformats'` includes "octal", Vim considers numbers starting with a '0' to be octal, unless the number includes a '8' or '9'. Other numbers are decimal and may have a preceding minus sign.
If the cursor is on a number, the commands apply to that number; otherwise Vim uses the number to the right of the cursor.
- When `'nrformats'` includes "hex", Vim assumes numbers starting with '0x' or

'0X' are hexadecimal. The case of the rightmost letter in the number determines the case of the resulting hexadecimal number. If there is no letter in the current number, Vim uses the previously detected case.

- When **'nrformats'** includes "alpha", Vim will change the alphabetic character under or after the cursor. This is useful to make lists with an alphabetic index.

For decimals a leading negative sign is considered for incrementing/decrementing, for binary, octal and hex values, it won't be considered. To ignore the sign Visually select the number before using **CTRL-A** or **CTRL-X**.

For numbers with leading zeros (including all octal and hexadecimal numbers), Vim preserves the number of characters in the number when possible. **CTRL-A** on "0077" results in "0100", **CTRL-X** on "0x100" results in "0x0ff".

There is one exception: When a number that starts with a zero is found not to be octal (it contains a '8' or '9'), but **'nrformats'** does include "octal", leading zeros are removed to avoid that the result may be recognized as an octal number.

Note that when **'nrformats'** includes "octal", decimal numbers with leading zeros cause mistakes, because they can be confused with octal numbers.

Note similarly, when **'nrformats'** includes both "bin" and "hex", binary numbers with a leading '0x' or '0X' can be interpreted as hexadecimal rather than binary since '0b' are valid hexadecimal digits. **CTRL-A** on "0x0b11" results in "0x0b12", not "0x0b100".

When **'nrformats'** includes "bin" and doesn't include "hex", **CTRL-A** on "0b11" in "0x0b11" results in "0x0b100".

When the number under the cursor is too big to fit into 32 or 64 bit (depending on how Vim was build), it will be rounded off to the nearest number that can be represented, and the addition/subtraction is skipped. E.g. with 64 bit support using **CTRL-X** on 18446744073709551616 results in 18446744073709551615. Same for larger numbers, such as 18446744073709551618.

The **CTRL-A** command is very useful in a macro. Example: Use the following steps to make a numbered list.

1. Create the first list entry, make sure it starts with a number.
2. qa - start recording into register 'a'
3. Y - yank the entry
4. p - put a copy of the entry below the first one
5. **CTRL-A** - increment the number
6. q - stop recording
7. <count>@a - repeat the yank, put and increment <count> times

SHIFTING LINES LEFT OR RIGHT

shift-left-right

<{motion}> Shift {motion} lines one **'shiftwidth'** leftwards.

If the **'vartabstop'** feature is enabled, and the **'shiftwidth'** option is set to zero, the amount of

indent is calculated at the first non-blank character in the line.

	<<
<<	Shift [count] lines one 'shiftwidth' leftwards.
	v_<
{Visual}[count]<	Shift the highlighted lines [count] 'shiftwidth' leftwards (for {Visual} see Visual-mode).
	>
>{motion}	Shift {motion} lines one 'shiftwidth' rightwards.
	If the 'vartabstop' feature is enabled, and the 'shiftwidth' option is set to zero, the amount of indent is calculated at the first non-blank character in the line.
	>>
>>	Shift [count] lines one 'shiftwidth' rightwards.
	v_>
{Visual}[count]>	Shift the highlighted lines [count] 'shiftwidth' rightwards (for {Visual} see Visual-mode).
	:<
:[range]<	Shift [range] lines one 'shiftwidth' left. Repeat '<' for shifting multiple 'shiftwidth' 's.
:[range]< {count}	Shift {count} lines one 'shiftwidth' left, starting with [range] (default current line cmdline-ranges). Repeat '<' for shifting multiple 'shiftwidth' 's.
:[range]le[ft] [indent]	left align lines in [range]. Sets the indent in the lines to [indent] (default 0).
	:>
:[range]> [flags]	Shift [range] lines one 'shiftwidth' right. Repeat '>' for shifting multiple 'shiftwidth' 's. See ex-flags for [flags].
:[range]> {count} [flags]	Shift {count} lines one 'shiftwidth' right, starting with [range] (default current line cmdline-ranges). Repeat '>' for shifting multiple 'shiftwidth' 's. See ex-flags for [flags].

The ">" and "<" commands are handy for changing the indentation within programs. Use the **'shiftwidth'** option to set the size of the white space which these commands insert or delete. Normally the **'shiftwidth'** option is 8, but you can set it to, say, 3 to make smaller indents. The shift leftwards stops when there is no indent. The shift right does not affect empty lines.

If the **'shiftround'** option is on, the indent is rounded to a multiple of **'shiftwidth'**.

If the **'smartindent'** option is on, or **'cindent'** is on and **'cinkeys'** contains '#' with a zero value, shift right does not affect lines starting with '#' (these are supposed to be C preprocessor lines that must stay in column 1). This can be changed with the **'cino'** option, see [cino-#](#) .

When the **'expandtab'** option is off (this is the default) Vim uses <Tab>s as much as possible to make the indent. You can use ">><<" to replace an indent made out of spaces with the same indent made out of <Tab>s (and a few spaces if necessary). If the **'expandtab'** option is on, Vim uses only spaces. Then you can use ">><<" to replace <Tab>s in the indent by spaces (or use ``:retab!``).

To move a line several **'shiftwidth'**s, use Visual mode or the ``:`` commands. For example:

```
Vjj4>      move three lines 4 indents to the right
: <<<      move current line 3 indents to the left
:>> 5      move 5 lines 2 indents to the right
:5>>      move line 5 2 indents to the right
```

4. Complex changes

[complex-change](#)

4.1 Filter commands

[filter](#)

A filter is a program that accepts text at standard input, changes it in some way, and sends it to standard output. You can use the commands below to send some text through a filter, so that it is replaced by the filter output. Examples of filters are "sort", which sorts lines alphabetically, and "indent", which formats C program files (you need a version of indent that works like a filter; not all versions do). The **'shell'** option specifies the shell Vim uses to execute the filter command (See also the **'shelltype'** option). You can repeat filter commands with ".". Vim does not recognize a comment (starting with '"') after the ``:!'`` command.

<code>!{motion}{filter}</code>	Filter <code>{motion}</code> text lines through the external program <code>{filter}</code> .	!
<code>!!{filter}</code>	Filter <code>[count]</code> lines through the external program <code>{filter}</code> .	!!
<code>{Visual}!{filter}</code>	Filter the highlighted lines through the external program <code>{filter}</code> (for <code>{Visual}</code> see Visual-mode).	v_!
<code>:{range}![!]{filter} [!][arg]</code>	Filter <code>{range}</code> lines through the external program <code>{filter}</code> . Vim replaces the optional bangs with the latest given command and appends the optional <code>[arg]</code> . Vim saves the output of the filter command in a temporary file and then reads the file into the buffer <code>tempfile</code> . Vim uses the 'shellredir' option to redirect the filter output to the temporary file.	:range!

However, if the `'shelltemp'` option is off then pipes are used when possible (on Unix).

When the 'R' flag is included in `'coptions'` marks in the filtered lines are deleted, unless the `:keepmarks` command is used. Example:

```
:keepmarks '<,>!sort
```

When the number of lines after filtering is less than before, marks in the missing lines are deleted anyway.

```
=  
={motion} Filter {motion} lines through the external program  
given with the 'equalprg' option. When the 'equalprg'  
option is empty (this is the default), use the  
internal formatting function C-indenting and  
'lisp'. But when 'indentexpr' is not empty, it will  
be used instead indent-expression. When Vim was  
compiled without internal formatting then the "indent"  
program is used as a last resort.
```

```
==  
== Filter [count] lines like with = {motion}.
```

```
v_  
{Visual}= Filter the highlighted lines like with = {motion}.
```

```
tempfile setuid  
Vim uses temporary files for filtering, generating diffs and also for  
tempname(). For Unix, the file will be in a private directory (only  
accessible by the current user) to avoid security problems (e.g., a symlink  
attack or other people reading your file). When Vim exits the directory and  
all files in it are deleted. When Vim has the setuid bit set this may cause  
problems, the temp file is owned by the setuid user but the filter command  
probably runs as the original user.
```

Directory for temporary files is created in the first of these directories that works:

Unix: \$TMPDIR, /tmp, current-dir, \$HOME.

Windows: \$TMP, \$TEMP, c:\TMP, c:\TEMP

For MS-Windows the GetTempFileName() system function is used.

For other systems the tmpnam() library function is used.

4.2 Substitute

```
:substitute  
:s :su  
:[range]s[substitute]/ {pattern} / {string} / [flags] [count]  
For each line in [range] replace a match of {pattern}  
with {string}.  
For the {pattern} see pattern.  
{string} can be a literal string, or something  
special; see sub-replace-special.  
When [range] and [count] are omitted, replace in the  
current line only. When [count] is given, replace in  
[count] lines, starting with the last line in [range].
```


When `[range]` is omitted start in the current line.

`E939` `E1510`

`[count]` must be a positive number (max 2147483647)

Also see `cmdline-ranges` .

See `:s_flags` for `[flags]`.

The delimiter doesn't need to be `/`, see `pattern-delimiter` .

`:[range]s[substitute] [flags] [count]`

`:[range]&&[flags] [count]`

`:&`

Repeat last `:substitute` with same search pattern and substitute string, but without the same flags. You may add `[flags]`, see `:s_flags` .

Note that after `:substitute` the `'&'` flag can't be used, it's recognized as a pattern separator.

The space between `:substitute` and the `'c'`, `'g'`, `'i'`, `'I'` and `'r'` flags isn't required, but in scripts it's a good idea to keep it to avoid confusion.

Also see the two and three letter commands to repeat `:substitute` below `:substitute-repeat` .

`:[range]~[&][flags] [count]`

`:~`

Repeat last substitute with same substitute string but with last used search pattern. This is like `:`&r``. See `:s_flags` for `[flags]`.

`&`

Synonym for `:`s`` (repeat last substitute). **Note** that the flags are not remembered, thus it might actually work differently. You can use `:`&&`` to keep the flags.

`g&`

Synonym for `:`%s//~/&`` (repeat last substitute with last search pattern on all lines with the same flags). For example, when you first do a substitution with `:`s/pattern/repl/flags`` and then `:`/search`` for something else, `g&` will do `:`%s/search/repl/flags``. Mnemonic: global substitute.

`:[range]sno[magic] ...`

`:snomagic` `:sno`
Same as `:`substitute``, but always use `'nomagic'`.

`:[range]sm[agic] ...`

`:smagic` `:sm`
Same as `:`substitute``, but always use `'magic'`.

The flags that you can use for the substitute commands:

`[&]`

Must be the first one: Keep the flags from the previous substitute command. Examples:

`:`&&``

`:s/this/that/&`

Note that `:s` and `:&` don't keep the flags.

[c] Confirm each substitution. Vim highlights the matching string (with `hl-IncSearch`). You can type: `:s_c`

'y' to substitute this match
'l' to substitute this match and then quit ("last")
'n' to skip this match
<Esc> to quit substituting
'a' to substitute this and all remaining matches
'q' to quit substituting
CTRL-E to scroll the screen up
CTRL-Y to scroll the screen down

If the `'edcompatible'` option is on, Vim remembers the [c] flag and toggles it each time you use it, but resets it when you give a new search pattern.

[e] When the search pattern fails, do not issue an error message and, in particular, continue in maps as if no error occurred. This is most useful to prevent the "No match" error from breaking a mapping. Vim does not suppress the following error messages, however: `:s_e`

Regular expressions can't be delimited by letters
\ should be followed by /, ? or &
No previous substitute regular expression
Trailing characters
Interrupted

[g] Replace all occurrences in the line. Without this argument, replacement occurs only for the first occurrence in each line. If the `'edcompatible'` option is on, Vim remembers this flag and toggles it each time you use it, but resets it when you give a new search pattern. If the `'gdefault'` option is on, this flag is on by default and the [g] argument switches it off. `:s_g`

[i] Ignore case for the pattern. The `'ignorecase'` and `'smartcase'` options are not used. `:s_i`

[I] Don't ignore case for the pattern. The `'ignorecase'` and `'smartcase'` options are not used. `:s_I`

[n] Report the number of matches, do not actually substitute. The [c] flag is ignored. The matches are reported as if `'report'` is zero. Useful to `count-items`.
If `\= sub-replace-expression` is used, the expression will be evaluated in the `sandbox` at every match. `:s_n`

[p] Print the line containing the last substitute. `:s_p`

[#] Like [p] and prepend the line number. `:s_#`

[l] Like [p] but print the text like `:list . :s_l`

[r] Only useful in combination with `:`&`` or `:`s`` without arguments. `:`&r`` works the same way as `:`~``: When the search pattern is empty, use the previously used search pattern instead of the search pattern from the last substitute or `:`global``. If the last command that did a search was a substitute or `:`global``, there is no effect. If the last command was a search command such as `/`, use the pattern from that command.

For `:`s`` with an argument this already happens:

```
:s/blue/red/
/green
:s//red/ or :~ or :&r
```

The last commands will replace "green" with "red".

```
:s/blue/red/
/green
:&
```

The last command will replace "blue" with "red".

Note that there is no flag to change the "magicness" of the pattern. A different command is used instead, or you can use `/\v` and friends. The reason is that the flags can only be found by skipping the pattern, and in order to skip the pattern the "magicness" must be known. Catch 22!

If the `{pattern}` for the substitute command is empty, the command uses the pattern from the last substitute or `:`global`` command. If there is none, but there is a previous search pattern, that one is used. With the [r] flag, the command uses the pattern from the last substitute, `:`global``, or search command.

If the `{string}` is omitted the substitute is done as if it's empty. Thus the matched pattern is deleted. The separator after `{pattern}` can also be left out then. Example:

```
:%s/TESTING
```

This deletes "TESTING" from all lines, but only one per line.

E1270

For compatibility with Vi these two exceptions are allowed in legacy script:

`"\/{string}/"` and `"\?{string}?"` do the same as `"//{string}/r"`.

`"\&{string}&"` does the same as `"//{string}/"`.

pattern-delimiter E146 E1241 E1242

Instead of the `/` which surrounds the pattern and replacement string, you can use another single-byte character. This is useful if you want to include a `/` in the search pattern or replacement string. Example:

```
:s/+//+
```

You can use most characters, but not an alphanumeric character, `\`, `'` or `|`. In Vim9 script you should not use `#` because it may be recognized as the start of a comment.

For the definition of a pattern, see [pattern](#). In Visual block mode, use `/\%V` in the pattern to have the substitute work in the block only.

Otherwise it works on whole lines anyway.

sub-replace-special :s\=

When the `{string}` starts with `"\"` it is evaluated as an expression, see [sub-replace-expression](#). You can use that for complex replacement or special characters.

The substitution is limited in recursion to 4 levels. [E1290](#)

Otherwise these characters in `{string}` have a special meaning:

:s%

When `{string}` is equal to `"%"` and `'/'` is included with the `'cptions'` option, then the `{string}` of the previous substitute command is used, see [cpo-](#)

magic	nomagic	action	
&	\&	replaced with the whole matched pattern	s/\&
\&	&	replaced with &	
	\0	replaced with the whole matched pattern	\0 s/\0
	\1	replaced with the matched pattern in the first pair of ()	s/\1
	\2	replaced with the matched pattern in the second pair of ()	s/\2
	s/\3
	\9	replaced with the matched pattern in the ninth pair of ()	s/\9
~	\~	replaced with the <code>{string}</code> of the previous substitute	s~
\~	~	replaced with ~	s/\~
	\u	next character made uppercase	s/\u
	\U	following characters made uppercase, until \E	s/\U
	\l	next character made lowercase	s/\l
	\L	following characters made lowercase, until \E	s/\L
	\e	end of \u, \U, \l and \L (NOTE: not <Esc>!)	s/\e
	\E	end of \u, \U, \l and \L	s/\E
<CR>		split line in two at this point (Type the <CR> as CTRL-V <Enter>)	s<CR>
\r		idem	s/\r
\<CR>		insert a carriage-return (CTRL-M) (Type the <CR> as CTRL-V <Enter>)	s/\<CR>
\n		insert a <NL> (<NUL> in the file) (does NOT break the line)	s/\n
\b		insert a <BS>	s/\b
\t		insert a <Tab>	s/\t
\\		insert a single backslash	s/\\
\x		where x is any character not mentioned above: Reserved for future expansion	

The special meaning is also used inside the third argument `{sub}` of the `substitute()` function with the following exceptions:

- A % inserts a percent literally without regard to `'cptions'`.
- magic is always set without regard to `'magic'`.
- A ~ inserts a tilde literally.
- <CR> and \r inserts a carriage-return (CTRL-M).
- \<CR> does not have a special meaning. It's just one of \x.

Examples:

```
:s/a\|b/xxx\0xxx/g      modifies "a b"      to "xxxaxxx xxxbxxx"
:s/\([abc]\)\([efg]\)/\2\1/g  modifies "af fa bg" to "fa fa gb"
:s/abcde/abc^Mde/          modifies "abcde"    to "abc", "de" (two lines)
:s/$/\^M/                  modifies "abcde"    to "abcde^M"
:s/\w\+/\u\0/g             modifies "bla bla"  to "Bla Bla"
:s/\w\+/\L\u\0/g           modifies "BLA bla"  to "Bla Bla"
```

Note: "\L\u" can be used to capitalize the first letter of a word. This is not compatible with Vi and older versions of Vim, where the "\u" would cancel out the "\L". Same for "\U\l".

Note: In previous versions **CTRL-V** was handled in a special way. Since this is not Vi compatible, this was removed. Use a backslash instead.

command	text	result
:s/aa/a^Ma/	aa	a<line-break>a
:s/aa/a\^Ma/	aa	a^Ma
:s/aa/a\\^Ma/	aa	a<line-break>a

(you need to type **CTRL-V** <CR> to get a ^M here)

The numbering of "\1", "\2" etc. is done based on which "(" comes first in the pattern (going left to right). When a parentheses group matches several times, the last one will be used for "\1", "\2", etc. Example:

```
:s/\(\(a[a-d] \)*\)/\2/      modifies "aa ab x" to "ab x"
```

The "\2" is for "\(a[a-d] \)". At first it matches "aa ", secondly "ab ".

When using parentheses in combination with '|', like in "\([ab]\)\|([cd]\)", either the first or second pattern in parentheses did not match, so either \1 or \2 is empty. Example:

```
:s/\([ab]\)\|([cd]\)/\1x/g  modifies "a b c d" to "ax bx x x"
```

```
:sc  :sce  :scg  :sci  :scI  :scl  :scp  :sg  :sgc
:sge  :sgi  :sgI  :sgl  :sgn  :sgp  :sgr  :sI  :si
:sic  :sIc  :sie  :sIe  :sIg  :sIl  :sin  :sIn  :sIp
:sip  :sIr  :sir  :sr  :src  :srg  :sri  :srI  :srl
:srn  :srp  :substitute-repeat
```

2-letter and 3-letter :substitute commands

These commands repeat the previous `:substitute` command with the given flags. The first letter is always "s", followed by one or two of the possible flag characters. For example `:sce` works like `:s///ce`. The table lists the possible combinations, not all flags are possible, because the command is short for another command.

List of :substitute commands

	c	e	g	i	I	n	p	l	r
c	:sc	:sce	:scg	:sci	:scI	:scn	:scp	:scl	
e									
g	:sgc	:sge	:sg	:sgi	:sgI	:sgn	:sgp	:sgl	:sgr
i	:sic	:sie		:si	:siI	:sin	:sip		:sir
I	:sIc	:sIe	:sIg	:sIi	:sI	:sIn	:sIp	:sIl	:sIr

```

| n
| p
| l
| r :src      :srg :sri :srI :srn :srp :srl :sr

```

Exceptions:

```

:scr is `:scriptnames`
:se  is `:set`
:sig is `:sign`
:sil is `:silent`
:sn  is `:snext`
:sp  is `:split`
:sl  is `:sleep`
:sre is `:srewind`

```

Substitute with an expression

```

sub-replace-expression
sub-replace-\= s/\=

```

When the substitute string starts with "\=" the remainder is interpreted as an expression.

The special meaning for characters as mentioned at [sub-replace-special](#) does not apply except for "<CR>". A <NL> character is used as a line break, you can get one with a double-quote string: "\n". Prepend a backslash to get a real <NL> character (which will be a NUL in the file).

The "\=" notation can also be used inside the third argument {sub} of [substitute\(\)](#) function. In this case, the special meaning for characters as mentioned at [sub-replace-special](#) does not apply at all. Especially, <CR> and <NL> are interpreted not as a line break but as a carriage-return and a new-line respectively.

When the result is a [List](#) then the items are joined with separating line breaks. Thus each item becomes a line, except that they can contain line breaks themselves.

The [submatch\(\)](#) function can be used to obtain matched text. The whole matched text can be accessed with "submatch(0)". The text matched with the first pair of () with "submatch(1)". Likewise for further sub-matches in ().

Be careful: The separation character must not appear in the expression! Consider using a character like "@" or ":". There is no problem if the result of the expression contains the separation character.

Examples:

```

:s@\n@="\r" .. expand("$HOME") .. "\r"@

```

This replaces an end-of-line with a new line containing the value of \$HOME.

```

s/E/\="\<Char-0x20ac>"/g

```

This replaces each 'E' character with a euro sign. Read more in [<Char->](#) .

4.3 Search and replace

[search-replace](#)

:pro :promptfind

`:promptf[ind] [string]`
 Put up a Search dialog. When `[string]` is given, it is used as the initial search string.
 {only for Win32, Motif and GTK GUI}

:promptr :promptrepl

`:promptr[ep] [string]`
 Put up a Search/Replace dialog. When `[string]` is given, it is used as the initial search string.
 {only for Win32, Motif and GTK GUI}

4.4 Changing tabs

change-tabs
:ret :retab :retab!

`:[range]ret[ab][!] [new_tabstop]`
 Replace all sequences of white-space containing a `<Tab>` with new strings of white-space using the new `tabstop` value given. If you do not specify a new `tabstop` size or it is zero, Vim uses the current value of `'tabstop'`.
 The current value of `'tabstop'` is always used to compute the width of existing tabs.
 With `!`, Vim also replaces strings of only normal spaces with tabs where appropriate.
 With `'expandtab'` on, Vim replaces all tabs with the appropriate number of spaces.
 This command sets `'tabstop'` to the new value given, and if performed on the whole file, which is default, should not make any visible change.
 Careful: This command modifies any `<Tab>` characters inside of strings in a C program. Use `"\t"` to avoid this (that's a good habit anyway).
`':retab!'` may also change a sequence of spaces by <Tab> characters, which can mess up a printf().
 If the +vartabs feature is enabled then a list of tab widths separated by commas may be used in place of a single tabstop. Each value in the list represents the width of one tabstop, except the final value which applies to all following tabstops.`

retab-example

Example for using autocommands and `":retab"` to edit a file which is stored with `tabstops` at 8 but edited with `tabstops` set at 4. Warning: white space inside of strings can change! Also see `'softtabstop'` option.

```
:auto BufReadPost    *.xx    retab! 4
:auto BufWritePre    *.xx    retab! 8
:auto BufWritePost   *.xx    retab! 4
:auto BufNewFile     *.xx    set ts=4
```

5. Copying and moving text copy-move

<code>"{register}</code>	Use <code>{register}</code> for next delete, yank or put. Use an uppercase character to append with delete and yank. Registers ".", "%", "# and ":" only work with put.
<code>:reg[isters]</code>	Display the type and contents of all numbered and named registers. If a register is written to for <code>:redir</code> it will not be listed. Type can be one of: " c " for <code>characterwise</code> text " \ " for <code>linewise</code> text " b " for <code>blockwise-visual</code> text
<code>:reg[isters] {arg}</code>	Display the contents of the numbered and named registers that are mentioned in <code>{arg}</code> . For example: <code>:reg 1a</code> to display registers '1' and 'a'. Spaces are allowed in <code>{arg}</code> .
<code>:di[splay] [arg]</code>	Same as <code>:registers</code> .
<code>["x]y{motion}</code>	Yank <code>{motion}</code> text [into register x]. When no characters are to be yanked (e.g., "y0" in column 1), this is an error when <code>'coptions'</code> includes the 'E' flag.
<code>["x]yy</code>	Yank <code>[count]</code> lines [into register x] <code>linewise</code> .
<code>["x]Y</code>	yank <code>[count]</code> lines [into register x] (synonym for <code>yy</code> , <code>linewise</code>). If you like "Y" to work from the cursor to the end of line (which is more logical, but not Vi-compatible) use <code>":map Y y\$"</code> .
<code>["x]zy{motion}</code>	Yank <code>{motion}</code> text [into register x]. Only differs from <code>`y`</code> when selecting a block of text, see <code>v_zy</code> .
<code>{Visual}["x]y</code>	Yank the highlighted text [into register x] (for <code>{Visual}</code> see <code>Visual-mode</code>).
<code>{Visual}["x]Y</code>	Yank the highlighted lines [into register x] (for <code>{Visual}</code> see <code>Visual-mode</code>).
<code>{Visual}["x]zy</code>	Yank the highlighted text [into register x]. Trailing whitespace at the end of each line of a selected block

won't be yanked. Especially useful in combination with `zp`. (for {Visual} see [Visual-mode](#))

- `:\[range\]y[ank] [x]` [:y](#) [:yank](#) [E850](#)
Yank [\[range\]](#) lines [into register [x](#)]. Yanking to the "[*](#)" or "[+](#)" registers is possible only when the [+clipboard](#) feature is included.
- `:\[range\]y[ank] [x] {count}`
Yank [{count}](#) lines, starting with last line number in [\[range\]](#) (default: current line [cmdline-ranges](#)), [into register [x](#)].
- `["x"]p` [p](#) [put](#) [E353](#) [E1240](#)
Put the text [from register [x](#)] after the cursor [\[count\]](#) times.
- `["x"]P` [P](#)
Put the text [from register [x](#)] before the cursor [\[count\]](#) times.
- `["x"]<MiddleMouse>` [<MiddleMouse>](#)
Put the text from a register before the cursor [\[count\]](#) times. Uses the "[*](#)" register, unless another is specified.
Leaves the cursor at the end of the new text.
Using the mouse only works when '[mouse](#)' contains '[n](#)' or '[a](#)'.
If you have a scrollwheel and often accidentally paste text, you can use these mappings to disable the pasting with the middle mouse button:
 [:map](#) <[MiddleMouse](#)> <[Nop](#)>
 [:imap](#) <[MiddleMouse](#)> <[Nop](#)>
You might want to disable the multi-click versions too, see [double-click](#) .
- `["x"]gp` [gp](#)
Just like "p", but leave the cursor just after the new text.
- `["x"]gP` [gP](#)
Just like "P", but leave the cursor just after the new text.
- `:\[line\]pu[t] [x]` [:pu](#) [:put](#)
Put the text [from register [x](#)] after [\[line\]](#) (default current line). This always works [linewise](#) , thus this command can be used to put a yanked block as new lines.
If no register is specified, it depends on the '[cb](#)' option: If '[cb](#)' contains "unnamedplus", paste from the + register [quoteplus](#) . Otherwise, if '[cb](#)' contains "unnamed", paste from the * register [quotestar](#) .
Otherwise, paste from the unnamed register

`quote_quote .`

The register can also be '=' followed by an optional expression. The expression continues until the end of the command. You need to escape the '|' and '"' characters to prevent them from terminating the command. Example:

```
:put ='path' .. \"/test\"
```

If there is no expression after '=', Vim uses the previous expression. You can see it with `":dis ="`.

`:[line]pu[t]! [x]` Put the text [from register x] before [line] (default current line).

`["x]]p` or `]p]<MiddleMouse>`
`["x]]<MiddleMouse>` Like "p", but adjust the indent to the current line. Using the mouse only works when 'mouse' contains 'n' or 'a'.

`["x][P` or `[P`
`["x]]P` or `]P`
`["x][p` or `[p [<MiddleMouse>`
`["x]][<MiddleMouse>` Like "P", but adjust the indent to the current line. Using the mouse only works when 'mouse' contains 'n' or 'a'.

`["x]zp` or `zp zP`
`["x]zP` Like "p" and "P", except without adding trailing spaces when pasting a block. Thus the inserted text will not always be a rectangle. Especially useful in combination with `v_zy`.

You can use these commands to copy text from one place to another. Do this by first getting the text into a register with a yank, delete or change command, then inserting the register contents with a put command. You can also use these commands to move text from one file to another, because Vim preserves all registers when changing buffers (the **CTRL-^** command is a quick way to toggle between two files).

`linewise-register` `characterwise-register`

You can repeat the put commands with "." (except for `:put`) and undo them. If the command that was used to get the text into the register was `linewise`, Vim inserts the text below ("p") or above ("P") the line where the cursor is. Otherwise Vim inserts the text after ("p") or before ("P") the cursor. With the `":put"` command, Vim always inserts the text in the next line. You can exchange two characters with the command sequence "xp". You can exchange two lines with the command sequence "ddp". You can exchange two words with the command sequence "deep" (start with the cursor in the blank space before the first word). You can use the `']]"` or ``]]"` command after the put command to move the cursor to the end of the inserted text, or use `'['` or ``["` to move the cursor to the start.

`put-Visual-mode` `v_p` `v_P`

When using a put command like `p` or `P` in Visual mode, Vim will try to replace the selected text with the contents of the register. Whether this

works well depends on the type of selection and the type of the text in the register. With blockwise selection it also depends on the size of the block and whether the corners are on an existing character. (Implementation detail: it actually works by first putting the register after the selection and then deleting the selection.)

With `p` the previously selected text is put in the unnamed register (and possibly the selection and/or clipboard). This is useful if you want to put that text somewhere else. But you cannot repeat the same change.

With `P` the unnamed register is not changed (and neither the selection or clipboard), you can repeat the same change. But the deleted text cannot be used. If you do need it you can use `p` with another register. E.g., yank the text to copy, Visually select the text to replace and use `"0p` . You can repeat this as many times as you like, and the unnamed register will be changed each time.

blockwise-put

When a register contains text from one line (characterwise), using a blockwise Visual selection, putting that register will paste that text repeatedly in each of the selected lines, thus replacing the blockwise selected region by multiple copies of the register text. For example:

- yank the word "TEXT" into a register with ``yw``
- select a visual block, marked with "v" in this text:
aaavva
bbbvbbb
cccvccc
- press ``p``, results in:
aaaTEXTaaa
bbbTEXTbbb
cccTEXTccc

blockwise-register

If you use a blockwise Visual mode command to get the text into the register, the block of text will be inserted before ("`P`") or after ("`p`") the cursor column in the current and next lines. Vim makes the whole block of text start in the same column. Thus the inserted text looks the same as when it was yanked or deleted. Vim may replace some `<Tab>` characters with spaces to make this happen. However, if the width of the block is not a multiple of a `<Tab>` width and the text after the inserted block contains `<Tab>`s, that text may be misaligned.

Use `zP|/|zp` to paste a blockwise yanked register without appending trailing spaces.

Note that after a characterwise yank command, Vim leaves the cursor on the first yanked character that is closest to the start of the buffer. This means that "`yl`" doesn't move the cursor, but "`yh`" moves the cursor one character left.

Rationale: In Vi the "`y`" command followed by a backwards motion would sometimes not move the cursor to the first yanked character, because redisplaying was skipped. In Vim it always moves to the first character, as specified by Posix.

With a linewise yank command the cursor is put in the first line, but the column is unmodified, thus it may not be on the first yanked character.

There are ten types of registers: `registers {register} E354`

1. The unnamed register ""
2. 10 numbered registers "0 to "9
3. The small delete register "-
4. 26 named registers "a to "z or "A to "Z
5. Three read-only registers ":", ".", "%
6. Alternate buffer register "#
7. The expression register "="
8. The selection and drop registers "*", "+ and "~
9. The black hole register "_
10. Last search pattern register "/"

1. Unnamed register ""

quote_quote quotequote

Vim fills this register with text deleted with the "d", "c", "s", "x" commands or copied with the yank "y" command, regardless of whether or not a specific register was used (e.g. "xdd). This is like the unnamed register is pointing to the last used register. Thus when appending using an uppercase register name, the unnamed register contains the same text as the named register. An exception is the '_' register: "_dd does not store the deleted text in any register.

Vim uses the contents of the unnamed register for any put command (p or P) which does not specify a register. Additionally you can access it with the name ''. This means you have to type two double quotes. Writing to the "" register writes to register "0.

{Vi: register contents are lost when changing files, no ''}

2. Numbered registers "0 to "9

quote_number quote0 quote1
quote2 quote3 quote4 quote9

Vim fills these registers with text from yank and delete commands.

Numbered register 0 contains the text from the most recent yank command, unless the command specified another register with ["x].

Numbered register 1 contains the text deleted by the most recent delete or change command, unless the command specified another register or the text is less than one line (the small delete register is used then). An exception is made for the delete operator with these movement commands: %, (,), ` , / , ? , n , N , { and } . Register "1 is always used then (this is Vi compatible). The "- register is used as well if the delete is within a line. **Note** that these characters may be mapped. E.g. % is mapped by the matchit plugin.

With each successive deletion or change, Vim shifts the previous contents of register 1 into register 2, 2 into 3, and so forth, losing the previous contents of register 9.

{Vi: numbered register contents are lost when changing files; register 0 does not exist}

3. Small delete register "-

quote_- quote-

This register contains text from commands that delete less than one line, except when the command specifies a register with ["x].

4. Named registers "a to "z or "A to "Z

quote_alpha quotea

Vim fills these registers only when you say so. Specify them as lowercase letters to replace their previous contents or as uppercase letters to append to their previous contents. When the '>' flag is present in '**cpoptions**' then a line break is inserted before the appended text.

5. Read-only registers ":", ".", and "%"

These are '%', ':', and '.'. You can use them only with the "p", "P", and ":put" commands and with **CTRL-R**.

	<code>quote_.</code>	<code>quote.</code>	<code>E29</code>
".	Contains the last inserted text (the same as what is inserted with the insert mode commands CTRL-A and CTRL-@). Note: this doesn't work with CTRL-R on the command-line. It works a bit differently, like inserting the text instead of putting it ('textwidth' and other options affect what is inserted).		
	<code>quote_%</code>	<code>quote%</code>	
"%	Contains the name of the current file.		
	<code>quote_:</code>	<code>quote:</code>	<code>E30</code>
":	Contains the most recent executed command-line. Example: Use "@:" to repeat the previous command-line command. The command-line is only stored in this register when at least one character of it was typed. Thus it remains unchanged if the command was completely from a mapping. {not available when compiled without the <code>+cmdline_hist</code> feature}		
	<code>quote_#</code>	<code>quote#</code>	

6. Alternate file register "#"

Contains the name of the alternate file for the current window. It will change how the **CTRL-^** command works. This register is writable, mainly to allow for restoring it after a plugin has changed it. It accepts buffer number:

```
let altbuf = bufnr(@#)
```

```
...
```

```
let @# = altbuf
```

It will give error `E86` if you pass buffer number and this buffer does not exist.

It can also accept a match with an existing buffer name:

```
let @# = 'buffer_name'
```

Error `E93` if there is more than one buffer matching the given name or `E94` if none of buffers matches the given name.

7. Expression register "="

`quote_ =` `quote =` `@ =`

This is not really a register that stores text, but is a way to use an expression in commands which use a register. The expression register is read-write.

When typing the '=' after " or **CTRL-R** the cursor moves to the command-line, where you can enter any expression (see [expression](#)). All normal command-line editing commands are available, including a special history for expressions. When you end the command-line by typing `<CR>`, Vim computes the result of the expression. If you end it with `<Esc>`, Vim abandons the expression. If you do not enter an expression, Vim uses the previous expression (like with the "/" command).

The expression must evaluate to a String. A Number is always automatically converted to a String. For the "p" and ":put" command, if the result is a Float it's converted into a String. If the result is a List each element is turned into a String and used as a line. A Dictionary or FuncRef results in an error message (use `string()` to convert).

If the "=" register is used for the "p" command, the String is split up at <NL> characters. If the String ends in a <NL>, it is regarded as a linewise register.

8. Selection and drop registers "*", "+ and "~"

Use these registers for storing and retrieving the selected text for the GUI. See [quotestar](#) and [quoteplus](#). When the clipboard is not available or not working, the unnamed register is used instead. For Unix systems the clipboard is only available when the [+xterm_clipboard](#) feature is present.

Note that there is only a distinction between "*" and "+" for X11 systems. For an explanation of the difference, see [x11-selection](#). Under MS-Windows, use of "*" and "+" is actually synonymous and refers to the [gui-clipboard](#).

[quote_~](#) [quote_~](#) <Drop>

The read-only "~" register stores the dropped text from the last drag'n'drop operation. When something has been dropped onto Vim, the "~" register is filled in and the <Drop> pseudo key is sent for notification. You can remap this key if you want; the default action (for all modes) is to insert the contents of the "~" register at the cursor position.

{only available when compiled with the [+dnd](#) feature, currently only with the GTK GUI}

Note: The "~" register is only used when dropping plain text onto Vim. Drag'n'drop of URI lists is handled internally.

9. Black hole register "_"

[quote_](#)

When writing to this register, nothing happens. This can be used to delete text without affecting the normal registers. When reading from this register, nothing is returned.

10. Last search pattern register "/"

[quote_/](#) [quote/](#)

Contains the most recent search-pattern. This is used for "n" and '[hlsearch](#)'. It is writable with `:let`, you can change it to have '[hlsearch](#)' highlight other matches without actually searching. You can't yank or delete into this register. The search direction is available in [v:searchforward](#).

Note that the value is restored when returning from a function [function-search-undo](#).

[@/](#)

You can write to a register with a `:let` command `:let-@`. Example:

```
:let @/ = "the"
```

If you use a put command without specifying a register, Vim uses the register that was last filled (this is also the contents of the unnamed register). If you are confused, use the `:dis` command to find out what Vim will put (this command displays all named and numbered registers; the unnamed register is labelled '').

The next three commands always work on whole lines.

```
:[range]co[py] {address}
```

[:co](#) [:copy](#)

Copy the lines given by [range] to below the line given by {address}.

`:t` `:t`
Synonym for copy.
This command is not supported in Vim9 script, because it is too easily confused with a variable name.

`:[range]m[ove] {address}` `:m` `:mo` `:move` E134
Move the lines given by `[range]` to below the line given by `{address}`.

6. Formatting text

`formatting`

`:[range]ce[nter] [width]` `:ce` `:center`
Center lines in `[range]` between `[width]` columns (default `'textwidth'` or 80 when `'textwidth'` is 0).

`:[range]ri[ght] [width]` `:ri` `:right`
Right-align lines in `[range]` at `[width]` columns (default `'textwidth'` or 80 when `'textwidth'` is 0).

`:[range]le[ft] [indent]` `:le` `:left`
Left-align lines in `[range]`. Sets the indent in the lines to `[indent]` (default 0).

`gq{motion}` `gq`
Format the lines that `{motion}` moves over. Formatting is done with one of three methods:
1. If `'formatexpr'` is not empty the expression is evaluated. This can differ for each buffer.
2. If `'formatprg'` is not empty an external program is used.
3. Otherwise formatting is done internally.

In the third case the `'textwidth'` option controls the length of each formatted line (see below).

If the `'textwidth'` option is 0, the formatted line length is the screen width (with a maximum width of 79).

The `'formatoptions'` option controls the type of formatting `fo-table`.

The cursor is left on the first non-blank of the last formatted line.

NOTE: The "Q" command formerly performed this function. If you still want to use "Q" for formatting, use this mapping:

```
:nnoremap Q gq
```

`gqgq` `gqgq` `gqg`
`gqgq`
Format the current line. With a count format that many lines.

`{Visual}gq` `v_gq`
Format the highlighted text. (for `{Visual}` see `Visual-mode`).

`gw{motion}` `gw`
Format the lines that `{motion}` moves over. Similar to `gq` but puts the cursor back at the same position in the text. However, `'formatprg'` and `'formatexpr'` are not used.

`gwgw` `gwgw` `gww`
`gww` Format the current line as with "gw".

`{Visual}gw` `v_gw`
Format the highlighted text as with "gw". (for `{Visual}` see `Visual-mode`).

Example: To format the current paragraph use: `gqap`
`gqap`

The "gq" command leaves the cursor in the line where the motion command takes the cursor. This allows you to repeat formatting repeated with ".". This works well with "gqj" (format current and next line) and "gq}" (format until end of paragraph). **Note:** When `'formatprg'` is set, "gq" leaves the cursor on the first formatted line (as with using a filter command).

If you want to format the current paragraph and continue where you were, use: `gwap`

If you always want to keep paragraphs formatted you may want to add the 'a' flag to `'formatoptions'`. See `auto-format` .

If the `'autoindent'` option is on, Vim uses the indent of the first line for the following lines.

Formatting does not change empty lines (but it does change lines with only white space!).

The `'joinspaces'` option is used when lines are joined together.

You can set the `'formatexpr'` option to an expression or the `'formatprg'` option to the name of an external program for Vim to use for text formatting. The `'textwidth'` and other options have no effect on formatting by an external program.

`format-formatexpr`
The `'formatexpr'` option can be set to a Vim script function that performs reformatting of the buffer. This should usually happen in an `ftplugin` , since formatting is highly dependent on the type of file. It makes sense to use an `autoload` script, so the corresponding script is only loaded when actually needed and the script should be called `<filetype>format.vim`.

For example, the XML filetype plugin distributed with Vim in the `$VIMRUNTIME` directory, sets the `'formatexpr'` option to:


```
setlocal formatexpr=xmlformat#Format()
```

That means, you will find the corresponding script, defining the `xmlformat#Format()` function, in the directory:
``$VIMRUNTIME/autoload/xmlformat.vim``

Here is an example script that removes trailing whitespace from the selected text. Put it in your autoload directory, e.g. `~/.vim/autoload/format.vim`:

```
func! format#Format()  
  " only reformat on explicit gq command  
  if mode() != 'n'  
    " fall back to Vim's internal reformatting  
    return 1  
  endif  
  let lines = getline(v:lnum, v:lnum + v:count - 1)  
  call map(lines, {key, val -> substitute(val, '\s\+$', '', 'g')})  
  call setline('.', lines)  
  
  " do not run internal formatter!  
  return 0  
endfunc
```

You can then enable the formatting by executing:

```
setlocal formatexpr=format#Format()
```

Note: this function explicitly returns non-zero when called from insert mode (which basically means, text is inserted beyond the `'textwidth'` limit). This causes Vim to fall back to reformat the text by using the internal formatter.

However, if the `gq` command is used to reformat the text, the function will receive the selected lines, trim trailing whitespace from those lines and put them back in place. If you are going to split single lines into multiple lines, be careful not to overwrite anything.

If you want to allow reformatting of text from insert or replace mode, one has to be very careful, because the function might be called recursively. For debugging it helps to set the `'debug'` option.

right-justify

There is no command in Vim to right justify text. You can do it with an external command, like `"par"` (e.g.: `"!}par"` to format until the end of the paragraph) or set `'formatprg'` to `"par"`.

format-comments

An overview of comment formatting is in section [30.6](#) of the user manual.

Vim can automatically insert and format comments in a special way. Vim recognizes a comment by a specific string at the start of the line (ignoring white space). Three types of comments can be used:

- A comment string that repeats at the start of each line. An example is the type of comment used in shell scripts, starting with `"#"`.
- A comment string that occurs only in the first line, not in the following

lines. An example is this list with dashes.

- Three-piece comments that have a start string, an end string, and optional lines in between. The strings for the start, middle and end are different. An example is the C style comment:

```
/*  
 * this is a C comment  
*/
```

The **'comments'** option is a comma-separated list of parts. Each part defines a type of comment string. A part consists of:

{flags}:**{string}**

{string} is the literal text that must appear.

{flags}:

- n Nested comment. Nesting with mixed parts is allowed. If **'comments'** is "n:),n:>" a line starting with ">) >" is a comment.
- b Blank (<Space>, <Tab> or <EOL>) required after **{string}**.
- f Only the first line has the comment string. Do not repeat comment on the next line, but preserve indentation (e.g., a bullet-list).
- s Start of three-piece comment
- m Middle of a three-piece comment
- e End of a three-piece comment
- l Left align. Used together with 's' or 'e', the leftmost character of start or end will line up with the leftmost character from the middle. This is the default and can be omitted. See below for more details.
- r Right align. Same as above but rightmost instead of leftmost. See below for more details.
- 0 Don't consider this comment for the "0" command.
- x Allows three-piece comments to be ended by just typing the last character of the end-comment string as the first action on a new line when the middle-comment string has been inserted automatically. See below for more details.

{digits}

When together with 's' or 'e': add **{digit}** amount of offset to an automatically inserted middle or end comment leader. The offset begins from a left alignment. See below for more details.

-{digits}

Like **{digits}** but reduce the indent. This only works when there is some indent for the start or end part that can be removed.

When a string has none of the 'f', 's', 'm' or 'e' flags, Vim assumes the comment string repeats at the start of each line. The flags field may be

empty.

Any blank space in the text before and after the `{string}` is part of the `{string}`, so do not include leading or trailing blanks unless the blanks are a required part of the comment string.

When one comment leader is part of another, specify the part after the whole. For example, to include both `"-"` and `"->"`, use

```
:set comments=f:->,f:-
```

A three-piece comment must always be given as start,middle,end, with no other parts in between. An example of a three-piece comment is

```
sr:/*,mb:*,ex:*/
```

for C-comments. To avoid recognizing `"*ptr"` as a comment, the middle string includes the `'b'` flag. For three-piece comments, Vim checks the text after the start and middle strings for the end string. If Vim finds the end string, the comment does not continue on the next line. Three-piece comments must have a middle string because otherwise Vim can't recognize the middle lines.

Notice the use of the `"x"` flag in the above three-piece comment definition. When you hit Return in a C-comment, Vim will insert the middle comment leader for the new line: `" * "`. To close this comment you just have to type `"/` before typing anything else on the new line. This will replace the middle-comment leader with the end-comment leader and apply any specified alignment, leaving just `" */`. There is no need to hit Backspace first.

When there is a match with a middle part, but there also is a matching end part which is longer, the end part is used. This makes a C style comment work without requiring the middle part to end with a space.

Here is an example of alignment flags at work to make a comment stand out (kind of looks like a 1 too). Consider comment string:

```
:set comments=sr:/***,m:**,ex-2:*****/
```

```
/***
 **<--right aligned from "r" flag
 **
offset 2 spaces for the "-2" flag-->**
 *****/
```

In this case, the first comment was typed, then return was pressed 4 times, then `"/` was pressed to end the comment.

Here are some finer points of three part comments. There are three times when alignment and offset flags are taken into consideration: opening a new line after a start-comment, opening a new line before an end-comment, and automatically ending a three-piece comment. The end alignment flag has a backwards perspective; the result is that the same alignment flag used with `"s"` and `"e"` will result in the same indent for the starting and ending pieces. Only one alignment per comment part is meant to be used, but an offset number will override the `"r"` and `"l"` flag.

Enabling `'cindent'` will override the alignment flags in many cases. Reindenting using a different method like `gq` or `=` will not consult alignment flags either. The same behaviour can be defined in those other

formatting options. One consideration is that **'cindent'** has additional options for context based indenting of comments but cannot replicate many three piece indent alignments. However, **'indentexpr'** has the ability to work better with three piece comments.

Other examples:

```
"b:*"      Includes lines starting with "*", but not if the "*" is
           followed by a non-blank. This avoids a pointer dereference
           like "*str" to be recognized as a comment.
"n:>"      Includes a line starting with ">", ">>", ">>>", etc.
"fb:~"     Format a list that starts with "~".
```

By default, "b:#" is included. This means that a line that starts with "#include" is not recognized as a comment line. But a line that starts with "# define" is recognized. This is a compromise.

fo-table

You can use the **'formatoptions'** option to influence how Vim formats text. **'formatoptions'** is a string that can contain any of the letters below. The default setting is "tcq". You can separate the option letters with commas for readability.

letter meaning when present in 'formatoptions'

t	Auto-wrap text using 'textwidth'	fo-t
c	Auto-wrap comments using 'textwidth' , inserting the current comment leader automatically.	fo-c
r	Automatically insert the current comment leader after hitting <Enter> in Insert mode.	fo-r
o	Automatically insert the current comment leader after hitting 'o' or 'O' in Normal mode. In case comment is unwanted in a specific place use CTRL-U to quickly delete it. i_CTRL-U	fo-o
/	When 'o' is included: do not insert the comment leader for a // comment after a statement, only when // is at the start of the line.	fo-/
q	Allow formatting of comments with "gq". Note that formatting will not change blank lines or lines containing only the comment leader. A new paragraph starts after such a line, or when the comment leader changes.	fo-q
w	Trailing white space indicates a paragraph continues in the next line. A line that ends in a non-white character ends a paragraph.	fo-w
a	Automatic formatting of paragraphs. Every time text is inserted or deleted the paragraph will be reformatted. See auto-format . When the 'c' flag is present this only happens for recognized comments.	fo-a
n	When formatting text, recognize numbered lists. This actually uses the 'formatlistpat' option, thus any kind of list can be used. The	fo-n

indent of the text after the number is used for the next line. The default is to find a number, optionally followed by '.', ':', ')', ']' or '}'. Note that **'autoindent'** must be set too. Doesn't work well together with "2".

Example:

1. the first item
wraps
2. the second item

fo-2

- 2 When formatting text, use the indent of the second line of a paragraph for the rest of the paragraph, instead of the indent of the first line. This supports paragraphs in which the first line has a different indent than the rest. Note that **'autoindent'** must be set too. Example:

```
    first line of a paragraph
    second line of the same paragraph
    third line.
```

This also works inside comments, ignoring the comment leader.

fo-v

- v Vi-compatible auto-wrapping in insert mode: Only break a line at a blank that you have entered during the current insert command. (Note: this is not 100% Vi compatible. Vi has some "unexpected features" or bugs in this area. It uses the screen column instead of the line column.)

fo-b

- b Like 'v', but only auto-wrap if you enter a blank at or before the wrap margin. If the line was longer than **'textwidth'** when you started the insert, or you do not enter a blank in the insert before reaching **'textwidth'**, Vim does not perform auto-wrapping.

fo-l

- l Long lines are not broken in insert mode: When a line was longer than **'textwidth'** when the insert command started, Vim does not automatically format it.

fo-m

- m Also break at a multibyte character above 255. This is useful for Asian text where every character is a word on its own.

fo-M

- M When joining lines, don't insert a space before or after a multibyte character. Overrides the 'B' flag.

fo-B

- B When joining lines, don't insert a space between two multibyte characters. Overruled by the 'M' flag.

fo-1

- 1 Don't break a line after a one-letter word. It's broken before it instead (if possible).

fo-]

-] Respect **'textwidth'** rigorously. With this flag set, no line can be longer than **'textwidth'**, unless line-break-prohibition rules make this impossible. Mainly for CJK scripts and works only if **'encoding'** is "utf-8".

fo-j

- j Where it makes sense, remove a comment leader when joining lines. For example, joining:

```
int i; // the index
```

```

// in the list
Becomes:
int i; // the index in the list
fo-p
p Don't break lines at single spaces that follow periods. This is
intended to complement 'joinspaces' and cpo-J , for prose with
sentences separated by two spaces. For example, with 'textwidth' set
to 28:
Surely you're joking, Mr. Feynman!
Becomes:
Surely you're joking,
Mr. Feynman!
Instead of:
Surely you're joking, Mr.
Feynman!

```

With 't' and 'c' you can specify when Vim performs auto-wrapping:

```

value  action
""     no automatic formatting (you can use "gq" for manual formatting)
"t"    automatic formatting of text, but not comments
"c"    automatic formatting for comments, but not text (good for C code)
"tc"   automatic formatting for text and comments

```

Note that when 'textwidth' is 0, Vim does no automatic formatting anyway (but does insert comment leaders according to the 'comments' option). An exception is when the 'a' flag is present. [auto-format](#)

Note that when 'paste' is on, Vim does no formatting at all.

Note that 'textwidth' can be non-zero even if Vim never performs auto-wrapping; 'textwidth' is still useful for formatting with "gq".

If the 'comments' option includes "/*", "*" and/or "*/", then Vim has some built in stuff to treat these types of comments a bit more cleverly. Opening a new line before or after "/*" or "*/" (with 'r' or 'o' present in 'formatoptions') gives the correct start of the line automatically. The same happens with formatting and auto-wrapping. Opening a line after a line starting with "/*" or "*" and containing "*/", will cause no comment leader to be inserted, and the indent of the new line is taken from the line containing the start of the comment.

E.g.:

```

/*
 * Your typical comment.
 */

```

The indent on this line is the same as the start of the above comment.

All of this should be really cool, especially in conjunction with the new :autocmd command to prepare different settings for different types of file.

Some examples:

```

for C code (only format comments):
:set fo=croq

```

for Mail/news (format all, don't start comment with "o" command):
:set fo=tcrcq

Automatic formatting

auto-format autoformat

When the 'a' flag is present in '**formatoptions**' text is formatted automatically when inserting text or deleting text. This works nicely for editing text paragraphs. A few hints on how to use this:

- You need to properly define paragraphs. The simplest is paragraphs that are separated by a blank line. When there is no separating blank line, consider using the 'w' flag and adding a space at the end of each line in the paragraphs except the last one.
- You can set the '**formatoptions**' based on the type of file `filetype` or specifically for one file with a `modeline` .
- Set '**formatoptions**' to "aw2tq" to make text with indents like this:

```
    bla bla foobar bla
  bla foobar bla foobar bla
    bla bla foobar bla
  bla foobar bla bla foobar
```

- Add the 'c' flag to only auto-format comments. Useful in source code.
- Set '**textwidth**' to the desired width. If it is zero then 79 is used, or the width of the screen if this is smaller.

And a few warnings:

- When part of the text is not properly separated in paragraphs, making changes in this text will cause it to be formatted anyway. Consider doing

```
:set fo-=a
```

- When using the 'w' flag (trailing space means paragraph continues) and deleting the last line of a paragraph with `dd` , the paragraph will be joined with the next one.
- Changed text is saved for undo. Formatting is also a change. Thus each format action saves text for undo. This may consume quite a lot of memory.
- Formatting a long paragraph and/or with complicated indenting may be slow.

=====
7. Sorting text sorting

Vim has a sorting function and a sorting command. The sorting function can be found here: `sort()` , `uniq()` .

```
:[range]sor[t][!] [b][f][i][l][n][o][r][u][x] [/ {pattern} /]
:sor :sort
```

Sort lines in `[range]`. When no range is given all lines are sorted.

With `!` the order is reversed.

With `i` case is ignored.

With `l` sort uses the current collation locale. Implementation details: `strcoll()` is used to compare strings. See `:language` to check or set the collation locale. Example:

```
:language collate en_US.UTF-8
:%sort l
```

`v:collate` can also be used to check the current locale. Sorting using the locale typically ignores case. This does not work properly on Mac.

Options `[n][f][x][o][b]` are mutually exclusive.

With `[n]` sorting is done on the first decimal number in the line (after or inside a `{pattern}` match). One leading `'-'` is included in the number.

With `[f]` sorting is done on the Float in the line. The value of Float is determined similar to passing the text (after or inside a `{pattern}` match) to `str2float()` function. This option is available only if Vim was compiled with Floating point support.

With `[x]` sorting is done on the first hexadecimal number in the line (after or inside a `{pattern}` match). A leading `"0x"` or `"0X"` is ignored. One leading `'-'` is included in the number.

With `[o]` sorting is done on the first octal number in the line (after or inside a `{pattern}` match).

With `[b]` sorting is done on the first binary number in the line (after or inside a `{pattern}` match).

With `[u]` (`u` stands for unique) only keep the first of a sequence of identical lines (ignoring case when `[i]` is used). Without this flag, a sequence of identical lines will be kept in their original order.

Note that leading and trailing white space may cause lines to be different.

When `/ {pattern} /` is specified and there is no `[r]` flag the text matched with `{pattern}` is skipped, so that you sort on what comes after the match.

'ignorecase' applies to the pattern, but **'smartcase'** is not used.

Instead of the slash any non-letter can be used.

For example, to sort on the second comma-separated

field:

```
:sort /^[^,]*,/
```

To sort on the text at virtual column 10 (thus ignoring the difference between tabs and spaces):

```
:sort /*\%10v/
```

To sort on the first number in the line, no matter what is in front of it:

```
:sort /\{-}\ze\d/
```

(Explanation: "`\{-}`" matches any text, "`\ze`" sets the end of the match and `\d` matches a digit.)

With `[r]` sorting is done on the matching `{pattern}` instead of skipping past it as described above.

For example, to sort on only the first three letters of each line:

```
:sort /\a\a\a/ r
```

If a `{pattern}` is used, any lines which don't have a match for `{pattern}` are kept in their current order, but separate from the lines which do match `{pattern}`. If you sorted in reverse, they will be in reverse order after the sorted lines, otherwise they will be in their original order, right before the sorted lines.

If `{pattern}` is empty (e.g. `//` is specified), the last search pattern is used. This allows trying out a pattern first.

Note that using `:sort` with `:global` doesn't sort the matching lines, it's quite useless.

`:sort` does not use the current locale unless the `l` flag is used. Vim does do a "stable" sort.

The sorting can be interrupted, but if you interrupt it too late in the process you may end up with duplicated lines. This also depends on the system library function used.

Undo and redo

undo-redo

The basics are explained in section 02.5 of the user manual.

- 1. Undo and redo commands [undo-commands](#)
- 2. Two ways of undo [undo-two-ways](#)
- 3. Undo blocks [undo-blocks](#)
- 4. Undo branches [undo-branches](#)
- 5. Undo persistence [undo-persistence](#)
- 6. Remarks about undo [undo-remarks](#)

1. Undo and redo commands		undo-commands
<code><Undo></code>	or	<code>undo <Undo> u</code>
<code>u</code>	Undo <code>[count]</code> changes.	
<code>:u[ndo]</code>	Undo one change.	<code>:u :un :undo</code>
<code>:u[ndo] {N}</code>	Jump to after change number <code>{N}</code> . See undo-branches for the meaning of <code>{N}</code> .	E830
CTRL-R	Redo <code>[count]</code> changes which were undone.	CTRL-R
<code>:red[o]</code>	Redo one change which was undone.	<code>:red :redo redo</code>
<code>U</code>	Undo all latest changes on one line, the line where the latest change was made. <code>U</code> itself also counts as a change, and thus <code>U</code> undoes a previous <code>U</code> .	<code>U</code>

The last changes are remembered. You can use the undo and redo commands above to revert the text to how it was before each change. You can also apply the changes again, getting back the text before the undo.

The "U" command is treated by undo/redo just like any other command. Thus a "u" command undoes a "U" command and a 'CTRL-R' command redoes it again. When mixing "U", "u" and 'CTRL-R' you will notice that the "U" command will restore the situation of a line to before the previous "U" command. This may be confusing. Try it out to get used to it. The "U" command will always mark the buffer as changed. When "U" changes the buffer back to how it was without changes, it is still considered changed. Use "u" to undo changes until the buffer becomes unchanged.

2. Two ways of undo

undo-two-ways

How undo and redo commands work depends on the 'u' flag in '**coptions**'. There is the Vim way ('u' excluded) and the Vi-compatible way ('u' included). In the Vim way, "uu" undoes two changes. In the Vi-compatible way, "uu" does nothing (undoes an undo).

'u' excluded, the Vim way:

You can go back in time with the undo command. You can then go forward again with the redo command. If you make a new change after the undo command, the redo will not be possible anymore.

'u' included, the Vi-compatible way:

The undo command undoes the previous change, and also the previous undo command. The redo command repeats the previous undo command. It does NOT repeat a change command, use "." for that.

Examples	Vim way	Vi-compatible way
"uu"	two times undo	no-op
"u CTRL-R "	no-op	two times undo

Rationale: Nvi uses the "." command instead of **CTRL-R**. Unfortunately, this is not Vi compatible. For example "dwdwu." in Vi deletes two words, in Nvi it does nothing.

3. Undo blocks

undo-blocks

One undo command normally undoes a typed command, no matter how many changes that command makes. This sequence of undo-able changes forms an undo block. Thus if the typed key(s) call a function, all the commands in the function are undone together.

If you want to write a function or script that doesn't create a new undoable change but joins in with the previous change use this command:

```
:undoj[oin]                                :undoj  :undojoin  E790
Join further changes with the previous undo block.
Warning: Use with care, it may prevent the user from
properly undoing changes. Don't use this after undo
or redo.
```

This is most useful when you need to prompt the user halfway through a change. For example in a function that calls `getchar()`. Do make sure that there was a related change before this that you must join with.

This doesn't work by itself, because the next key press will start a new change again. But you can do something like this:

```
:undojoin | delete
```

After this a "u" command will undo the delete command and the previous change.

undo-break undo-close-block

g- Go to older text state. With a count repeat that many times.

:earlier {count} Go to older text state {count} times.

:earlier {N}s Go to older text state about {N} seconds before.

:earlier {N}m Go to older text state about {N} minutes before.

:earlier {N}h Go to older text state about {N} hours before.

:earlier {N}d Go to older text state about {N} days before.

:earlier {N}f Go to older text state {N} file writes before. When changes were made since the last write ":earlier 1f" will revert the text to the state when it was written. Otherwise it will go to the write before that. When at the state of the first file write, or when the file was not written, ":earlier 1f" will go to before the first change.

g+ Go to newer text state. With a count repeat that many times.

:later {count} Go to newer text state {count} times.

:later {N}s Go to newer text state about {N} seconds later.

:later {N}m Go to newer text state about {N} minutes later.

:later {N}h Go to newer text state about {N} hours later.

:later {N}d Go to newer text state about {N} days later.

:later {N}f Go to newer text state {N} file writes later. When at the state of the last file write, ":later 1f" will go to the newest text state.

Note that text states will become unreachable when undo information is cleared for '**undolevels**'.

Don't be surprised when moving through time shows multiple changes to take place at a time. This happens when moving through the undo tree and then making a new change.

EXAMPLE

Start with this text:

one two three

Delete the first word by pressing "x" three times:

ne two three

e two three

two three

Now undo that by pressing "u" three times:

e two three

ne two three

```
one two three
```

Delete the second word by pressing "x" three times:

```
one wo three
one o three
one three
```

Now undo that by using "g-" three times:

```
one o three
one wo three
two three
```

You are now back in the first undo branch, after deleting "one". Repeating "g-" will now bring you back to the original text:

```
e two three
ne two three
one two three
```

Jump to the last change with ":later 1h":

```
one three
```

And back to the start again with ":earlier 1h":

```
one two three
```

Note that using "u" and **CTRL-R** will not get you to all possible text states while repeating "g-" and "g+" does.

5. Undo persistence

`undo-persistence` `persistent-undo`

When unloading a buffer Vim normally destroys the tree of undos created for that buffer. By setting the `'undofile'` option, Vim will automatically save your undo history when you write a file and restore undo history when you edit the file again.

The `'undofile'` option is checked after writing a file, before the `BufWritePost` autocommands. If you want to control what files to write undo information for, you can use a `BufWritePre` autocommand:

```
au BufWritePre /tmp/* setlocal noundofile
```

Vim saves undo trees in a separate undo file, one for each edited file, using a simple scheme that maps filesystem paths directly to undo files. Vim will detect if an undo file is no longer synchronized with the file it was written for (with a hash of the file contents) and ignore it when the file was changed after the undo file was written, to prevent corruption. An undo file is also ignored if its owner differs from the owner of the edited file, except when the owner of the undo file is the current user. Set `'verbose'` to get a message about that when opening a file.

Undo files are normally saved in the same directory as the file. This can be changed with the `'undodir'` option.

When the file is encrypted, the text in the undo file is also encrypted. The

same key and method is used. [encryption](#)

Note that text properties are not stored in the undo file. You can restore text properties so long as a buffer is loaded, but you cannot restore them from an undo file. Rationale: It would require the associated text property types to be defined in exactly the same way as before, which cannot be guaranteed.

You can also save and restore undo histories by using `":wundo"` and `":rundo"` respectively:

```
                                :wundo  :rundo
:wundo[!] {file}
    Write undo history to {file}.
    When {file} exists and it does not look like an undo file
    (the magic number at the start of the file is wrong), then
    this fails, unless the ! was added.
    If it exists and does look like an undo file it is
    overwritten. If there is no undo-history, nothing will be
    written.
    Implementation detail: Overwriting happens by first deleting
    the existing file and then creating a new file with the same
    name. So it is not possible to overwrite an existing undofile
    in a write-protected directory.

:rundo {file}  Read undo history from {file}.
```

You can use these in autocommands to explicitly specify the name of the history file. E.g.:

```
au BufReadPost * call ReadUndo()
au BufWritePost * call WriteUndo()
func ReadUndo()
  if filereadable(expand('%:h') .. '/UNDO/' .. expand('%:t'))
    rundo %:h/UNDO/%:t
  endif
endfunc
func WriteUndo()
  let dirname = expand('%:h') .. '/UNDO'
  if !isdirectory(dirname)
    call mkdir(dirname)
  endif
  wundo %:h/UNDO/%:t
endfunc
```

You should keep `'undofile'` off, otherwise you end up with two undo files for every write.

You can use the `undofile()` function to find out the file name that Vim would use.

Note that while reading/writing files and `'undofile'` is set most errors will be silent, unless `'verbose'` is set. With `:wundo` and `:rundo` you will get more error messages, e.g., when the file cannot be read or written.

NOTE: undo files are never deleted by Vim. You need to delete them yourself.

Reading an existing undo file may fail for several reasons:

- E822** It cannot be opened, because the file permissions don't allow it.
- E823** The magic number at the start of the file doesn't match. This usually means it is not an undo file.
- E824** The version number of the undo file indicates that it's written by a newer version of Vim. You need that newer version to open it. Don't write the buffer if you want to keep the undo info in the file.

"File contents changed, cannot use undo info"

The file text differs from when the undo file was written. This means the undo file cannot be used, it would corrupt the text. This also happens when **'encoding'** differs from when the undo file was written.

- E825** The undo file does not contain valid contents and cannot be used.
- E826** The undo file is encrypted but decryption failed.
- E827** The undo file is encrypted but this version of Vim does not support encryption. Open the file with another Vim.
- E832** The undo file is encrypted but **'key'** is not set, the text file is not encrypted. This would happen if the text file was written by Vim encrypted at first, and later overwritten by not encrypted text. You probably want to delete this undo file.

"Not reading undo file, owner differs"

The undo file is owned by someone else than the owner of the text file. For safety the undo file is not used.

Writing an undo file may fail for these reasons:

- E828** The file to be written cannot be created. Perhaps you do not have write permissions in the directory.

"Cannot write undo file in any directory in **'undodir'**"

None of the directories in **'undodir'** can be used.

"Will not overwrite with undo file, cannot read"

A file exists with the name of the undo file to be written, but it cannot be read. You may want to delete this file or rename it.

"Will not overwrite, this is not an undo file"

A file exists with the name of the undo file to be written, but it does not start with the right magic number. You may want to delete this file or rename it.

"Skipping undo file write, nothing to undo"

There is no undo information to be written, nothing has been changed or **'undolevels'** is negative.

- E829** An error occurred while writing the undo file. You may want to try again.

=====

6. Remarks about undo

undo-remarks

The number of changes that are remembered is set with the **'undolevels'** option. If it is zero, the Vi-compatible way is always used. If it is negative no undo is possible. Use this if you are running out of memory.

clear-undo

When you set **'undolevels'** to -1 the undo information is not immediately cleared, this happens at the next change. To force clearing the undo information you can use these commands:

```
:let old_undolevels = &l:undolevels
:setlocal undolevels=-1
:exe "normal a \<BS>\<Esc>"
:let &l:undolevels = old_undolevels
:unlet old_undolevels
```

Note use of ``&l:undolevels`` to explicitly read the local value of `'undolevels'` and the use of ``:setlocal`` to change only the local option (which takes precedence over the corresponding global option value). Saving the option value via the use of ``&undolevels`` is unpredictable; it reads either the local value (if one has been set) or the global value (otherwise). Also, if a local value has been set, changing the option via ``:set undolevels`` will change both the global and local values, requiring extra work to save and restore both values.

Marks for the buffer ('a to 'z) are also saved and restored, together with the text.

When all changes have been undone, the buffer is not considered to be changed. It is then possible to exit Vim with `":q"` instead of `":q!"`. **Note** that this is relative to the last write of the file. Typing "u" after `":w"` actually changes the buffer, compared to what was written, so the buffer is considered changed then.

When manual **fold**ing is being used, the folds are not saved and restored. Only changes completely within a fold will keep the fold as it was, because the first and last line of the fold don't change.

The numbered registers can also be used for undoing deletes. Each time you delete text, it is put into register "1. The contents of register "1 are shifted to "2, etc. The contents of register "9 are lost. You can now get back the most recent deleted text with the put command: `""1P'`. (also, if the deleted text was the result of the last delete or copy operation, 'P' or 'p' also works as this puts the contents of the unnamed register). You can get back the text of three deletes ago with `""3P'`.

redo-register

If you want to get back more than one part of deleted text, you can use a special feature of the repeat command `."`. It will increase the number of the register used. So if you first do `""1P'`, the following `."` will result in a `""2P'`. Repeating this will result in all numbered registers being inserted.

Example: If you deleted text with `'dd....'` it can be restored with `""1P....'`.

If you don't know in which register the deleted text is, you can use the `:display` command. An alternative is to try the first register with `""1P'`, and if it is not what you want do `'u.'`. This will remove the contents of the first put, and repeat the put command for the second register. Repeat the `'u.'` until you got what you want.

Repeating commands, Vim scripts and debugging

repeating

Chapter 26 of the user manual introduces repeating [usr_26.txt](#) .

- | | |
|--------------------------|--------------------------------|
| 1. Single repeats | single-repeat |
| 2. Multiple repeats | multi-repeat |
| 3. Complex repeats | complex-repeat |
| 4. Using Vim scripts | using-scripts |
| 5. Using Vim packages | packages |
| 6. Creating Vim packages | package-create |
| 7. Debugging scripts | debug-scripts |
| 8. Profiling | profiling |

1. Single repeats

[single-repeat](#)

`.` Repeat last change, with count replaced with `[count]`. Also repeat a yank command, when the 'y' flag is included in '`coptions`'. Does not repeat a command-line command.

Simple changes can be repeated with the `."` command. Without a count, the count of the last change is used. If you enter a count, it will replace the last one. `v:count` and `v:count1` will be set.

If the last change included a specification of a numbered register, the register number will be incremented. See [redo-register](#) for an example how to use this.

Note that when repeating a command that used a Visual selection, the same SIZE of area is used, see [visual-repeat](#) .

`@:` Repeat last command-line `[count]` times.
{not available when compiled without the `+cmdline_hist` feature}

2. Multiple repeats

[multi-repeat](#)

`:[range]g[lobal]/{pattern}/[cmd]` [:g](#) [:global](#) E148
Execute the Ex command `[cmd]` (default `":p"`) on the lines within `[range]` where `{pattern}` matches.

`:[range]g[lobal]!/{pattern}/[cmd]`

Execute the Ex command `[cmd]` (default `":p"`) on the lines within `[range]` where `{pattern}` does NOT match.

`:v` `:vglobal`

`:[range]v[global]/{pattern}/[cmd]`
Same as `:g!`.

Example:

`:g/^Obsolete/d _`

Using the underscore after ``:d`` avoids clobbering registers or the clipboard. This also makes it faster.

Instead of the `'/'` which surrounds the `{pattern}`, you can use any other single byte character, but not an alphabetic character, `'\'`, `'"'`, `'|'` or `'!'`. This is useful if you want to include a `'/'` in the search pattern or replacement string.

For the definition of a pattern, see [pattern](#) .

NOTE `[cmd]` may contain a range; see [collapse](#) and [edit-paragraph-join](#) for examples.

The global commands work by first scanning through the `[range]` lines and marking each line where a match occurs (for a multi-line pattern, only the start of the match matters).

In a second scan the `[cmd]` is executed for each marked line, as if the cursor was in that line. For `":v"` and `":g!"` the command is executed for each not marked line. If a line is deleted its mark disappears.

The default for `[range]` is the whole buffer (1,\$). Use **"CTRL-C"** to interrupt the command. If an error message is given for a line, the command for that line is aborted and the global command continues with the next marked or unmarked line.

E147

When the command is used recursively, it only works on one line. Giving a range is then not allowed. This is useful to find all lines that match a pattern and do not match another pattern:

`:g/found/v/notfound/{cmd}`

This first finds all lines containing "found", but only executes `{cmd}` when there is no match for "notfound".

Any Ex command can be used, see [ex-cmd-index](#) . To execute a Normal mode command, you can use the ``:normal`` command:

`:g/pat/normal {commands}`

Make sure that `{commands}` ends with a whole command, otherwise Vim will wait for you to type the rest of the command for each match. The screen will not have been updated, so you don't know what you are doing. See [:normal](#) .

The undo/redo command will undo/redo the whole global command at once. The previous context mark will only be set once (with `""` you go back to where the cursor was before the global command).

The global command sets both the last used search pattern and the last used substitute pattern (this is vi compatible). This makes it easy to globally replace a string:

```
:g/pat/s//PAT/g
```

This replaces all occurrences of "pat" with "PAT". The same can be done with:

```
:%s/pat/PAT/g
```

Which is two characters shorter!

When using "global" in Ex mode, a special case is using ":visual" as a command. This will move to a matching line, go to Normal mode to let you execute commands there until you use `Q` to return to Ex mode. This will be repeated for each matching line. While doing this you cannot use ":global". To abort this type **CTRL-C** twice.

3. Complex repeats

complex-repeat

q recording

`q{0-9a-zA-Z"}`

Record typed characters into register `{0-9a-zA-Z"}` (uppercase to append). The 'q' command is disabled while executing a register, and it doesn't work inside a mapping and `:normal`.

Note: If the register being used for recording is also used for `y` and `p` the result is most likely not what is expected, because the put will paste the recorded macro and the yank will overwrite the recorded macro.

Note: The recording happens while you type, replaying the register happens as if the keys come from a mapping. This matters, for example, for undo, which only syncs when commands were typed.

`q`

Stops recording. (Implementation **note:** The 'q' that stops recording is not stored in the register, unless it was the result of a mapping)

@

`@{0-9a-z".=*+}`

Execute the contents of register `{0-9a-z".=*+}` [count] times. **Note** that register '%' (name of the current file) and '#' (name of the alternate file) cannot be used.

The register is executed like a mapping, that means that the difference between 'wildchar' and 'wildcharm' applies, and undo might not be synced in the same way. For "@=" you are prompted to enter an expression. The result of the expression is then executed. See also `@:`.

@@ E748

`@@`

Repeat the previous `@{0-9a-z":*}` [count] times.

:@

`:[addr]@{0-9a-z".=*+}`

Execute the contents of register `{0-9a-z".=*+}` as an Ex command. First set cursor at line [addr] (default is current line). When the last line in the register does

not have a <CR> it will be added automatically when the 'e' flag is present in 'coptions'. For ":@" the last used expression is used. The result of evaluating the expression is executed as an Ex command.

Mappings are not recognized in these commands. When the `line-continuation` character (\) is present at the beginning of a line in a linewise register, then it is combined with the previous line. This is useful for yanking and executing parts of a Vim script.

Future: Will execute the register for each line in the address range.

`:[addr]*{0-9a-z".=+}` :star-compatible
When '*' is present in 'coptions' `cpo-star`, use ":*" in the same way as ":@". This is NOT the default when 'nocompatible' is used. When the '*' flag is not present in 'coptions', ":*" is an alias for ":'<,>", select the Visual area `:star`.

`:[addr]@:` :@:
Repeat last command-line. First set cursor at line `[addr]` (default is current line).

`:[addr]@` :@@
`:[addr]@@` Repeat the previous `:@{register}`. First set cursor at line `[addr]` (default is current line).

4. Using Vim scripts using-scripts

For writing a Vim script, see chapter 41 of the user manual `usr_41.txt`.

`:so[urce] {file}` :so :source load-vim-script
Read Ex commands from `{file}`. These are commands that start with a ":".
Triggers the `SourcePre` autocommand.

`:[range]so[urce] [++clear]` :source-range
Read Ex commands from the `[range]` of lines in the current buffer. When `[range]` is omitted read all lines.

When sourcing commands from the current buffer, the same script-ID `<SID>` is used even if the buffer is sourced multiple times. If a buffer is sourced more than once, then the functions in the buffer are defined again.

To source a range of lines that doesn't start with the `:vim9script` command in Vim9 script context, the `:vim9cmd` modifier can be used. If you use a Visual selection and type ":", the range in the form "'<,>"

can come before it:

```
:<,'>vim9cmd source
```

Otherwise the range goes after the modifier and must have a colon prefixed, like all Vim9 ranges:

```
:vim9cmd :5,9source
```

When a range of lines in a buffer is sourced in the Vim9 script context, the previously defined script-local variables and functions are not cleared. This works like the range started with the `":vim9script noclear"` command. The `"++clear"` argument can be used to clear the script-local variables and functions before sourcing the script. This works like the range started with the ``:vim9script`` command without the `"noclear"` argument. See [vim9-reload](#) for more information.

Examples:

```
:4,5source
```

```
:10,18source ++clear
```

```
:source!
```

`:so[urce]! {file}`

Read Vim commands from `{file}`. These are commands that are executed from Normal mode, like you type them.

When used after `:global` , `:argdo` , `:windo` , `:bufdo` , in a loop or when another command follows the display won't be updated while executing the commands.

Cannot be used in the [sandbox](#) .

```
:ru :runtime
```

`:ru[ntime][!] [where] {file} ..`

Read Ex commands from `{file}` in each directory given by `'runtimepath'` and/or `'packpath'`. There is no error for non-existing files.

Example:

```
:runtime syntax/c.vim
```

There can be multiple `{file}` arguments, separated by spaces. Each `{file}` is searched for in the first directory from `'runtimepath'`, then in the second directory, etc. Use a backslash to include a space inside `{file}` (although it's better not to use spaces in file names, it causes trouble).

When `[!]` is included, all found files are sourced. When it is not included only the first found file is sourced.

When `[where]` is omitted only `'runtimepath'` is used. Other values:

```
START search under "start" in 'packpath'  
OPT search under "opt" in 'packpath'
```

PACK search under "start" and "opt" in **'packpath'**
 ALL first use **'runtimepath'**, then search under "start" and "opt" in **'packpath'**

When {file} contains wildcards it is expanded to all matching files. Example:

```
:runtime! plugin/**/*.vim
```

This is what Vim uses to load the plugin files when starting up. This similar command:

```
:runtime plugin/**/*.vim
```

would source the first file only.

When **'verbose'** is one or higher, there is a message when no file could be found.

When **'verbose'** is two or higher, there is a message about each searched file.

```
:pa[ckadd][!] {name} :pa :packadd E919
```

Search for an optional plugin directory in **'packpath'** and source any plugin files found. The directory must match:

```
pack/*/opt/{name}
```

The directory is added to **'runtimepath'** if it wasn't there yet.

If the directory pack/*/opt/{name}/after exists it is added at the end of **'runtimepath'**.

If loading packages from "pack/*/start" was skipped, then this directory is searched first:

```
pack/*/start/{name}
```

Note that {name} is the directory name, not the name of the .vim file. All the files matching the pattern

```
pack/*/opt/{name}/plugin/**/*.vim
```

will be sourced. This allows for using subdirectories below "plugin", just like with plugins in **'runtimepath'**.

If the filetype detection was not enabled yet (this is usually done with a ``syntax enable`` or ``filetype on`` command in your .vimrc file), this will also look for "{name}/ftdetect/*.vim" files.

When the optional ! is added no plugin files or ftdetect scripts are loaded, only the matching directories are added to **'runtimepath'**. This is useful in your .vimrc. The plugins will then be loaded during initialization, see [load-plugins](#) (note that the loading order will be reversed, because each directory is inserted before others).

Note that for ftdetect scripts to be loaded you will need to write ``filetype plugin indent on`` AFTER all `!packadd!` commands.

Also see [pack-add](#) .
{only available when compiled with |+eval|}

`:packl[oadall][!]` [:packl](#) [:packloadall](#)
Load all packages in the "start" directory under each entry in '[packpath](#)'.

First all the directories found are added to '[runtimepath](#)', then the plugins found in the directories are sourced. This allows for a plugin to depend on something of another plugin, e.g. an "autoload" directory. See [packload-two-steps](#) for how this can be useful.

This is normally done automatically during startup, after loading your .vimrc file. With this command it can be done earlier.

Packages will be loaded only once. Using ``:packloadall`` a second time will have no effect. When the optional ! is added this command will load packages even when done before.

Note that when using ``:packloadall`` in the [vimrc](#) file, the '[runtimepath](#)' option is updated, and later all plugins in '[runtimepath](#)' will be loaded, which means they are loaded again. Plugins are expected to handle that.

An error only causes sourcing the script where it happens to be aborted, further plugins will be loaded. See [packages](#) .

{only available when compiled with |+eval|}

`:scripte[ncoding] [encoding]` [:scripte](#) [:scriptencoding](#) E167
Specify the character encoding used in the script. The following lines will be converted from [\[encoding\]](#) to the value of the '[encoding](#)' option, if they are different. Examples:

```
scriptencoding iso-8859-5
scriptencoding cp932
```

When [\[encoding\]](#) is empty, no conversion is done. This can be used to restrict conversion to a sequence of lines:

```
scriptencoding euc-jp
... lines to be converted ...
scriptencoding
... not converted ...
```

When conversion isn't supported by the system, there is no error message and no conversion is done. When a line can't be converted there is no error and the

original line is kept.

Don't use "ucs-2" or "ucs-4", scripts cannot be in these encodings (they would contain NUL bytes). When a sourced script starts with a BOM (Byte Order Mark) in utf-8 format Vim will recognize it, no need to use ":scriptencoding utf-8" then.

If you set the **'encoding'** option in your `.vimrc`, `:scriptencoding` must be placed after that. E.g.:`

```
set encoding=utf-8
scriptencoding utf-8
```

`:scriptv[ersion] {version}` **:scriptv** **:scriptversion**
E999 E984 E1040

Specify the version of Vim for the lines that follow in the same file. Only applies at the toplevel of sourced scripts, not inside functions.

If `{version}` is higher than what the current Vim version supports E999 will be given. You either need to rewrite the script to make it work with an older Vim version, or update Vim to a newer version. See `vimscript-version` for what changed between versions.

`:vim9s[cript] [noclear]` **:vim9s** **:vim9script**

Marks a script file as containing `Vim9-script` commands. Also see `vim9-namespace` . E1038
Must be the first command in the file. E1039
For `[noclear]` see `vim9-reload` .
Without the `+eval` feature this changes the syntax for some commands.
See `:vim9cmd` for executing one command with Vim9 syntax and semantics.

`:scr[iptnames]` **:scr** **:scriptnames**

List all sourced script names, in the order they were first encountered. The number is used for the script ID `<SID>` .
For a script that was used with ``import autoload`` but was not actually sourced yet an "A" is shown after the script ID.
For a script that was referred to by one name but after resolving symbolic links got sourced with another name the other script is after `"->"`. E.g. `"20->22"` means script 20 was sourced as script 22.
Also see ``getscriptinfo()``.
{not available when compiled without the `+eval` feature}

`:scr[iptnames][!] {scriptId}` **:script**

Edit script `{scriptId}`. Although `":scriptnames name"` works, using `":script name"` is recommended.

When the current buffer can't be `abandoned` and the `!` is not present, the command fails.

`:fini` [sh] `:fini` `:finish` E168

Stop sourcing a script. Can only be used in a Vim script file. This is a quick way to skip the rest of the file. If it is used after a `:try` but before the matching `:finally` (if present), the commands following the `:finally` up to the matching `:endtry` are executed first. This process applies to all nested `:try`s in the script. The outermost `:endtry` then stops sourcing the script.

All commands and command sequences can be repeated by putting them in a named register and then executing it. There are two ways to get the commands in the register:

- Use the record command `"q`". You type the commands once, and while they are being executed they are stored in a register. Easy, because you can see what you are doing. If you make a mistake, `"p`"ut the register into the file, edit the command sequence, and then delete it into the register again. You can continue recording by appending to the register (use an uppercase letter).
- Delete or yank the command sequence into the register.

Often used command sequences can be put under a function key with the `':map'` command.

An alternative is to put the commands in a file, and execute them with the `':source!'` command. Useful for long command sequences. Can be combined with the `':map'` command to put complicated commands under a function key.

The `':source'` command reads Ex commands from a file or a buffer line by line. You will have to type any needed keyboard input. The `':source!'` command reads from a script file character by character, interpreting each character as if you typed it.

Example: When you give the `":!ls"` command you get the `hit-enter` prompt. If you `':source'` a file with the line `"!ls"` in it, you will have to type the `<Enter>` yourself. But if you `':source!'` a file with the line `":!ls"` in it, the next characters from that file are read until a `<CR>` is found. You will not have to type `<CR>` yourself, unless `":!ls"` was the last line in the file.

It is possible to put `':source[!]'` commands in the script file, so you can make a top-down hierarchy of script files. The `':source'` command can be nested as deep as the number of files that can be opened at one time (about 15). The `':source!'` command can be nested up to 15 levels deep.

You can use the `"<sfile>"` string (literally, this is not a special key) inside of the sourced file, in places where a file name is expected. It will be replaced by the file name of the sourced file. For example, if you have a `"other.vimrc"` file in the same directory as your `".vimrc"` file, you can source it from your `".vimrc"` file with this command:

```
:source <sfile>:h/other.vimrc
```

In script files terminal-dependent key codes are represented by terminal-independent two character codes. This means that they can be used in the same way on different kinds of terminals. The first character of a key code is 0x80 or 128, shown on the screen as "~@". The second one can be found in the list [key-notation](#). Any of these codes can also be entered with **CTRL-V** followed by the three digit decimal code. This does NOT work for the `<t_xx>` termcap codes, these can only be used in mappings.

Win32: Files that are read with `":source"` normally have `<CR><NL> <EOL>s`. These always work. If you are using a file with `<NL> <EOL>s` (for example, a file made on Unix), this will be recognized if `'fileformats'` is not empty and the first line does not end in a `<CR>`. This fails if the first line has something like `":map <F1> :help^M"`, where `^M` is a `<CR>`. If the first line ends in a `<CR>`, but following ones don't, you will get an error message, because the `<CR>` from the first lines will be lost.

Mac Classic: Files that are read with `":source"` normally have `<CR> <EOL>s`. These always work. If you are using a file with `<NL> <EOL>s` (for example, a file made on Unix), this will be recognized if `'fileformats'` is not empty and the first line does not end in a `<CR>`. Be careful not to use a file with `<NL>` linebreaks which has a `<CR>` in first line.

On other systems, Vim expects `":source"`d files to end in a `<NL>`. These always work. If you are using a file with `<CR><NL> <EOL>s` (for example, a file made on MS-Windows), all lines will have a trailing `<CR>`. This may cause problems for some commands (e.g., mappings). There is no automatic `<EOL>` detection, because it's common to start with a line that defines a mapping that ends in a `<CR>`, which will confuse the automaton.

Long lines in a `":source"`d Ex command script file can be split by inserting a line continuation symbol `"\"` (backslash) at the start of the next line. There can be white space before the backslash, which is ignored.

Example: the lines

```
:set comments=sr:/*,mb:*,el:*/,
    \://,
    \b:#,
    \:%,
    \n:>,
    \fb:-
```

are interpreted as if they were given in one line:

```
:set comments=sr:/*,mb:*,el:*/,://,b:#,:%,n:>,fb:-
```

All leading whitespace characters in the line before a backslash are ignored. **Note** however that trailing whitespace in the line before it cannot be inserted freely; it depends on the position where a command is split up whether additional whitespace is allowed or not.

When a space is required it's best to put it right after the backslash. A space at the end of a line is hard to see and may be accidentally deleted.

```
:syn match Comment
    \ "very long regexp"
```

`\ keepend`

In `Vim9` script the backslash can often be omitted, but not always.
See `vim9-line-continuation` .

There is a problem with the `":append"` and `":insert"` commands:

```
:1append
\asdf
.
```

The backslash is seen as a line-continuation symbol, thus this results in the command:

```
:1appendasdf
.
```

To avoid this, add the 'C' flag to the `'coptions'` option:

```
:set cpo+=C
:1append
\asdf
.
:set cpo-=C
```

Note that when the commands are inside a function, you need to add the 'C' flag when defining the function, it is not relevant when executing it.

```
:set cpo+=C
:function Foo()
:1append
\asdf
.
:endifunction
:set cpo-=C
```

`line-continuation-comment`

To add a comment in between the lines start with `'"\ ' . Notice the space after the backslash. Example:`

```
let array = [
    "\ first entry comment
    \ 'first',
    "\ second entry comment
    \ 'second',
    \ ]
```

Rationale:

Most programs work with a trailing backslash to indicate line continuation. Using this in Vim would cause incompatibility with Vi. For example for this Vi mapping:

```
:map xx asdf\
```

Therefore the unusual leading backslash is used.

Starting a comment in a continuation line results in all following continuation lines to be part of the comment. Since it was like this for a long time, when making it possible to add a comment halfway a sequence of continuation lines, it was not possible to use `"\`, since that was a valid continuation line. Using `'"\ ' comes closest, even though it may look a bit weird. Requiring the space after the backslash is to make it very unlikely this is a normal comment line.`

5. Using Vim packages

packages

A Vim package is a directory that contains one or more plugins. The advantages over normal plugins:

- A package can be downloaded as an archive and unpacked in its own directory. Thus the files are not mixed with files of other plugins. That makes it easy to update and remove.
- A package can be a git, mercurial, etc. repository. That makes it really easy to update.
- A package can contain multiple plugins that depend on each other.
- A package can contain plugins that are automatically loaded on startup and ones that are only loaded when needed with `:packadd`.

Using a package and loading automatically

Let's assume your Vim files are in the `"~/vim"` directory and you want to add a package from a zip archive `"/tmp/foopack.zip"`:

```
% mkdir -p ~/.vim/pack/foo
% cd ~/.vim/pack/foo
% unzip /tmp/foopack.zip
```

The directory name "foo" is arbitrary, you can pick anything you like.

You would now have these files under `~/vim`:

```
pack/foo/README.txt
pack/foo/start/foobar/plugin/foo.vim
pack/foo/start/foobar/syntax/some.vim
pack/foo/opt/foodebug/plugin/debugger.vim
```

When Vim starts up, after processing your `.vimrc`, it scans all directories in `'packpath'` for plugins under the `"pack/*/start"` directory. First all those directories are added to `'runtimepath'`. Then all the plugins are loaded. See [packload-two-steps](#) for how these two steps can be useful.

To allow for calling into package functionality while parsing your `.vimrc`, `:colorscheme` and `autoload` will both automatically search under `'packpath'` as well in addition to `'runtimepath'`. See the documentation for each for details.

In the example Vim will find `"pack/foo/start/foobar/plugin/foo.vim"` and adds `"~/vim/pack/foo/start/foobar"` to `'runtimepath'`.

If the "foobar" plugin kicks in and sets the `'filetype'` to "some", Vim will find the `syntax/some.vim` file, because its directory is in `'runtimepath'`.

Vim will also load `ftdetect` files, if there are any.

Note that the files under `"pack/foo/opt"` are not loaded automatically, only the ones under `"pack/foo/start"`. See [pack-add](#) below for how the "opt" directory is used.

Loading packages automatically will not happen if loading plugins is disabled, see `load-plugins` .

To load packages earlier, so that `'runtimepath'` gets updated:

```
:packloadall
```

This also works when loading plugins is disabled. The automatic loading will only happen once.

If the package has an "after" directory, that directory is added to the end of `'runtimepath'`, so that anything there will be loaded later.

Using a single plugin and loading it automatically

If you don't have a package but a single plugin, you need to create the extra directory level:

```
% mkdir -p ~/.vim/pack/foo/start/foobar
% cd ~/.vim/pack/foo/start/foobar
% unzip /tmp/someplugin.zip
```

You would now have these files:

```
pack/foo/start/foobar/plugin/foo.vim
pack/foo/start/foobar/syntax/some.vim
```

From here it works like above.

Optional plugins

To load an optional plugin from a pack use the `pack-add` `:packadd` command:
`:packadd foodebug`

This searches for "pack/*/opt/foodebug" in `'packpath'` and will find `~/.vim/pack/foo/opt/foodebug/plugin/debugger.vim` and source it.

This could be done if some conditions are met. For example, depending on whether Vim supports a feature or a dependency is missing.

You can also load an optional plugin at startup, by putting this command in your `.vimrc` :

```
:packadd! foodebug
```

The extra "!" is so that the plugin isn't loaded if Vim was started with `--noplugin` .

It is perfectly normal for a package to only have files in the "opt" directory. You then need to load each plugin when you want to use it.

Where to put what

Since color schemes, loaded with `:colorscheme`, are found below "pack/*/start" and "pack/*/opt", you could put them anywhere. We recommend you put them below "pack/*/opt", for example `~/.vim/pack/mycolors/opt/dark/colors/very_dark.vim`.

Filetype plugins should go under "pack/*/start", so that they are always found. Unless you have more than one plugin for a file type and want to select which one to load with `:packadd`. E.g. depending on the compiler version:

```
if foo_compiler_version > 34
  packadd foo_new
else
  packadd foo_old
endif
```

The "after" directory is most likely not useful in a package. It's not disallowed though.

6. Creating Vim packages

`package-create`

This assumes you write one or more plugins that you distribute as a package.

If you have two unrelated plugins you would use two packages, so that Vim users can choose what they include or not. Or you can decide to use one package with optional plugins, and tell the user to add the preferred ones with `:packadd`.

Decide how you want to distribute the package. You can create an archive or you could use a repository. An archive can be used by more users, but is a bit harder to update to a new version. A repository can usually be kept up-to-date easily, but it requires a program like "git" to be available. You can do both, github can automatically create an archive for a release.

Your directory layout would be like this:

```
start/foobar/plugin/foo.vim      " always loaded, defines commands
start/foobar/plugin/bar.vim      " always loaded, defines commands
start/foobar/autoload/foo.vim    " loaded when foo command used
start/foobar/doc/foo.txt         " help for foo.vim
start/foobar/doc/tags            " help tags
opt/fooextra/plugin/extra.vim    " optional plugin, defines commands
opt/fooextra/autoload/extra.vim  " loaded when extra command used
opt/fooextra/doc/extra.txt       " help for extra.vim
opt/fooextra/doc/tags            " help tags
```

This allows for the user to do:

```
mkdir ~/.vim/pack
cd ~/.vim/pack
git clone https://github.com/you/foobar.git myfoobar
```

Here "myfoobar" is a name that the user can choose, the only condition is that it differs from other packages.

In your documentation you explain what the plugins do, and tell the user how to load the optional plugin:

```
:packadd! fooextra
```

You could add this packadd command in one of your plugins, to be executed when the optional plugin is needed.

Run the `:helptags` command to generate the doc/tags file. Including this generated file in the package means that the user can drop the package in the pack directory and the help command works right away. Don't forget to re-run the command after changing the plugin help:

```
:helptags path/start/foobar/doc
:helptags path/opt/foextra/doc
```

Dependencies between plugins

[packload-two-steps](#)

Suppose you have two plugins that depend on the same functionality. You can put the common functionality in an autoloader directory, so that it will be found automatically. Your package would have these files:

```
pack/foo/start/one/plugin/one.vim
    call foolib#getit()
pack/foo/start/two/plugin/two.vim
    call foolib#getit()
pack/foo/start/lib/autoload/foolib.vim
    func foolib#getit()
```

This works, because loading packages will first add all found directories to `'runtimepath'` before sourcing the plugins.

7. Debugging scripts

[debug-scripts](#)

Besides the obvious messages that you can add to your scripts to find out what they are doing, Vim offers a debug mode. This allows you to step through a sourced file or user function and set breakpoints.

NOTE: The debugging mode is far from perfect. Debugging will have side effects on how Vim works. You cannot use it to debug everything. For example, the display is messed up by the debugging messages.

An alternative to debug mode is setting the `'verbose'` option. With a bigger number it will give more verbose messages about what Vim is doing.

STARTING DEBUG MODE

[debug-mode](#)

To enter debugging mode use one of these methods:

1. Start Vim with the `-D` argument:

```
vim -D file.txt
```

Debugging will start as soon as the first vimrc file is sourced. This is useful to find out what is happening when Vim is starting up. A side effect is that Vim will switch the terminal mode before initialisations have finished, with unpredictable results.

For a GUI-only version (Windows, Macintosh) the debugging will start as soon as the GUI window has been opened. To make this happen early, add a `":gui"` command in the vimrc file.

[:debug](#)

2. Run a command with `":debug"` prepended. Debugging will only be done while

this command executes. Useful for debugging a specific script or user function. And for scripts and functions used by autocommands. Example:
`:debug edit test.txt.gz`

3. Set a breakpoint in a sourced file or user function. You could do this in the command line:

```
vim -c "breakadd file */explorer.vim" .
```

This will run Vim and stop in the first line of the "explorer.vim" script. Breakpoints can also be set while in debugging mode.

In debugging mode every executed command is displayed before it is executed. Comment lines, empty lines and lines that are not executed are skipped. When a line contains two commands, separated by "|", each command will be displayed separately.

DEBUG MODE

Once in debugging mode, the usual Ex commands can be used. For example, to inspect the value of a variable:

```
echo idx
```

When inside a user function, this will print the value of the local variable "idx". Prepend "g:" to get the value of a global variable:

```
echo g:idx
```

All commands are executed in the context of the current function or script. You can also set options, for example setting or resetting '**verbose**' will show what happens, but you might want to set it just before executing the lines you are interested in:

```
:set verbose=20
```

Commands that require updating the screen should be avoided, because their effect won't be noticed until after leaving debug mode. For example:

```
:help
```

won't be very helpful.

There is a separate command-line history for debug mode.

NOTE: In Vim9 script, if a command is written at the script level and continues on the next line, not using the old way with a backslash for line continuation, only the first line is printed before the debugging prompt.

The line number for a function line is relative to the start of the function. If you have trouble figuring out where you are, edit the file that defines the function in another Vim, search for the start of the function and do "99j". Replace "99" with the line number.

Additionally, these commands can be used:

cont	Continue execution until the next breakpoint is hit.
	<code>>cont</code>
quit	Abort execution. This is like using CTRL-C , some things might still be executed, doesn't abort everything. Still stops at the next breakpoint.
	<code>>quit</code>
	<code>>next</code>

next	Execute the command and come back to debug mode when it's finished. This steps over user function calls and sourced files.
	<code>>step</code>
step	Execute the command and come back to debug mode for the next command. This steps into called user functions and sourced files.
	<code>>interrupt</code>
interrupt	This is like using CTRL-C , but unlike " <code>>quit</code> " comes back to debug mode for the next command that is executed. Useful for testing <code>:finally</code> and <code>:catch</code> on interrupt exceptions.
	<code>>finish</code>
finish	Finish the current script or user function and come back to debug mode for the command after the one that sourced or called it.
	<code>>bt</code> <code>>backtrace</code> <code>>where</code>
backtrace bt where	Show the call stacktrace for current debugging session.
	<code>>frame</code>
frame N	Goes to N backtrace level. + and - signs make movement relative. E.g., " <code>:frame +3</code> " goes three frames up.
	<code>>up</code>
up	Goes one level up from call stacktrace.
	<code>>down</code>
down	Goes one level down from call stacktrace.

About the additional commands in debug mode:

- There is no command-line completion for them, you get the completion for the normal Ex commands only.
- You can shorten them, up to a single character, unless more than one command starts with the same letter. "f" stands for "finish", use "fr" for "frame".
- Hitting `<CR>` will repeat the previous one. When doing another command, this is reset (because it's not clear what you want to repeat).
- When you want to use the Ex command with the same name, prepend a colon: "`:cont`", "`:next`", "`:finish`" (or shorter).

`vim9-debug`

When debugging a compiled `:def` function, "step" will stop before every executed line, not every single instruction. Thus it works mostly like a not compiled function. Access to local variables is limited you can use:

`echo varname`

But not much else.

When executing a command that is not a specific bytecode instruction but executed like a normal Ex command, "step" will stop once in the compiled context, where local variables can be inspected, and once just before executing the command.

In a `:def` function variables that haven't been declared yet cannot be inspected. Variables that have been declared can be inspected, also when the block they were declared in has finished. In commands this would not be possible, thus is slightly misleading (but can be useful).

The backtrace shows the hierarchy of function calls, e.g.:

```
>bt
 3 function One[3]
 2 Two[3]
->1 Three[3]
 0 Four
line 1: let four = 4
```

The "->" points to the current frame. Use "up", "down" and "frame N" to select another frame.

In the current frame you can evaluate the local function variables. There is no way to see the command at the current line yet.

DEFINING BREAKPOINTS

```
:breaka[dd] func [lnum] {name} :breaka :breakadd
```

Set a breakpoint in a function. Example:

```
:breakadd func Explore
```

Doesn't check for a valid function name, thus the breakpoint can be set before the function is defined.

```
:breaka[dd] file [lnum] {name}
```

Set a breakpoint in a sourced file. Example:

```
:breakadd file 43 .vimrc
```

```
:breaka[dd] here
```

Set a breakpoint in the current line of the current file.

Like doing:

```
:breakadd file <cursor-line> <current-file>
```

Note that this only works for commands that are executed when sourcing the file, not for a function defined in that file.

```
:breaka[dd] expr {expression}
```

Sets a breakpoint, that will break whenever the {expression} evaluates to a different value. Example:

```
:breakadd expr g:lnum
```

Will break, whenever the global variable lnum changes.

Errors in evaluation are suppressed, you can use the name of a variable that does not exist yet. This also means you will not notice anything if the expression has a mistake.

Note if you watch a script-variable this will break when switching scripts, since the script variable is only valid in the script where it has been defined and if that script is called from several other scripts, this will stop whenever that particular variable will become visible or inaccessible again.

The [lnum] is the line number of the breakpoint. Vim will stop at or after this line. When omitted line 1 is used.

`:debug-name`

`{name}` is a pattern that is matched with the file or function name. The pattern is like what is used for autocommands. There must be a full match (as if the pattern starts with "^" and ends in "\$"). A "*" matches any sequence of characters. `'ignorecase'` is not used, but `"\c"` can be used in the pattern to ignore case `/\c`. Don't include the () for the function name!

The match for sourced scripts is done against the full file name. If no path is specified the current directory is used. Examples:

```
breakadd file explorer.vim
```

matches "explorer.vim" in the current directory.

```
breakadd file *explorer.vim
```

matches ".../plugin/explorer.vim", ".../plugin/iexplorer.vim", etc.

```
breakadd file */explorer.vim
```

matches ".../plugin/explorer.vim" and "explorer.vim" in any other directory.

The match for functions is done against the name as it's shown in the output of `:function`. However, for local functions the script-specific prefix such as `"<SNR>99_"` is ignored to make it easier to match script-local functions without knowing the ID of the script.

Note that functions are first loaded and later executed. When they are loaded the "file" breakpoints are checked, when they are executed the "func" breakpoints.

DELETING BREAKPOINTS

`:breakd` `:breakdel` E161

```
:breakd[el] {nr}
```

Delete breakpoint `{nr}`. Use `:breaklist` to see the number of each breakpoint.

```
:breakd[el] *
```

Delete all breakpoints.

```
:breakd[el] func [lnum] {name}
```

Delete a breakpoint in a function.

```
:breakd[el] file [lnum] {name}
```

Delete a breakpoint in a sourced file.

```
:breakd[el] here
```

Delete a breakpoint at the current line of the current file.

When `[lnum]` is omitted, the first breakpoint in the function or file is deleted.

The `{name}` must be exactly the same as what was typed for the `:breakadd` command. "explorer", "*explorer.vim" and "*explorer*" are different.

LISTING BREAKPOINTS

`:breakl` `:breaklist`

```
:breakl[ist]
```

List all breakpoints.

OBSCURE

`:debugg` `:debuggreedy`

`:debugg[reedy]`

Read debug mode commands from the normal input stream, instead of getting them directly from the user. Only useful for test scripts. Example:

```
echo 'q^Mq' | vim -e -s -c debuggreedy -c 'breakadd file script.vim' -S s
```

`:@debugg[reedy]`

Undo `":debuggreedy"`: get debug mode commands directly from the user, don't use typeahead for debug commands.

8. Profiling

`profile` `profiling`

Profiling means that Vim measures the time that is spent on executing functions and/or scripts. The `+profile` feature is required for this. It is included when Vim was compiled with "huge" features.

You can also use the `reltime()` function to measure time. This only requires the `+reltime` feature, which is present in more builds.

For profiling syntax highlighting see `:syntime` .

For example, to profile the `one_script.vim` script file:

```
:profile start /tmp/one_script_profile
:profile file one_script.vim
:source one_script.vim
:exit
```

`:prof[ile] start {fname}` `:prof` `:profile` E750

Start profiling, write the output in `{fname}` upon exit or when a ``:profile stop`` or ``:profile dump`` command is invoked. `"~/` and environment variables in `{fname}` will be expanded. If `{fname}` already exists it will be silently overwritten. The variable `v:profiling` is set to one.

`:prof[ile] stop`

Write the collected profiling information to the logfile and stop profiling. You can use the ``:profile start`` command to clear the profiling statistics and start profiling again.

`:prof[ile] pause`

Don't profile until the following ``:profile continue``. Can be used when doing something that should not be counted (e.g., an external command). Does not nest.

`:prof[ile] continue`

Continue profiling after ``:profile pause``.

`:prof[ile] func {pattern}`
 Profile function that matches the pattern `{pattern}`.
 See `:debug-name` for how `{pattern}` is used.

`:prof[ile][!] file {pattern}`
 Profile script file that matches the pattern `{pattern}`.
 See `:debug-name` for how `{pattern}` is used.
 This only profiles the script itself, not the functions defined in it.
 When the `!` is added then all functions defined in the script will also be profiled.
Note that profiling only starts when the script is loaded after this command. A `:profile` command in the script itself won't work.

`:prof[ile] dump`
 Write the current state of profiling to the logfile immediately. After running this command, Vim continues to collect the profiling statistics.

`:profd[el] ...` `:profd` `:profdel`
 Stop profiling for the arguments specified. See `:breakdel` for the arguments. Examples:
`profdel func MyFunc`
`profdel file MyScript.vim`
`profdel here`

You must always start with a `:profile start fname` command. The resulting file is written when Vim exits. For example, to profile one specific function:

```
profile start /tmp/vimprofile
profile func MyFunc
```

Here is an example of the output, with line numbers prepended for the explanation:

```
1 FUNCTION Test2()
2 Called 1 time
3 Total time: 0.155251
4 Self time: 0.002006
5
6 count total (s) self (s)
7 9 0.000096 for i in range(8)
8 8 0.153655 call Test3()
9 8 0.000070 endfor
10 " Ask a question
11 1 0.001341 echo input("give me an answer: ")
```

The header (lines 1-4) gives the time for the whole function. The "Total" time is the time passed while the function was executing. The "Self" time is the "Total" time reduced by time spent in:

- other user defined functions
- sourced scripts

- executed autocommands
- external (shell) commands

Lines 7-11 show the time spent in each executed line. Lines that are not executed do not count. Thus a comment line is never counted.

The Count column shows how many times a line was executed. **Note** that the "for" command in line 7 is executed one more time as the following lines. That is because the line is also executed to detect the end of the loop.

The time Vim spends waiting for user input isn't counted at all. Thus how long you take to respond to the input() prompt is irrelevant.

Profiling should give a good indication of where time is spent, but keep in mind there are various things that may clobber the results:

- The accuracy of the time measured depends on the `gettimeofday()`, or `clock_gettime()` if available, system function. The accuracy ranges from 1/100 second to nanoseconds. With `clock_gettime()` the times are displayed in nanoseconds, otherwise microseconds. You can use ``has("prof_nsec")``.
- Real elapsed time is measured, if other processes are busy they may cause delays at unpredictable moments. You may want to run the profiling several times and use the lowest results.
- If you have several commands in one line you only get one time. Split the line to see the time for the individual commands.
- The time of the lines added up is mostly less than the time of the whole function. There is some overhead in between.
- Functions that are deleted before Vim exits will not produce profiling information. You can check the `v:profiling` variable if needed:


```

      :if !v:profiling
      :   delfunc MyFunc
      :endif
      
```
- Profiling may give weird results on multi-processor systems, when sleep mode kicks in or the processor frequency is reduced to save power.
- The "self" time is wrong when a function is used recursively.

Visual mode

Visual Visual-mode visual-mode

Visual mode is a flexible and easy way to select a piece of text for an operator. It is the only way to select a block of text.

This is introduced in section 04.4 of the user manual.

- | | |
|--------------------------------------|---------------------|
| 1. Using Visual mode | visual-use |
| 2. Starting and stopping Visual mode | visual-start |
| 3. Changing the Visual area | visual-change |
| 4. Operating on the Visual area | visual-operators |
| 5. Blockwise operators | blockwise-operators |
| 6. Repeating | visual-repeat |
| 7. Examples | visual-examples |
| 8. Select mode | Select-mode |

{Since Vim 7.4.200 the |+visual| feature is always included}

1. Using Visual mode

visual-use

Using Visual mode consists of three parts:

1. Mark the start of the text with "v", "V" or **CTRL-V**.
The character under the cursor will be used as the start.
2. Move to the end of the text.
The text from the start of the Visual mode up to and including the character under the cursor is highlighted.
3. Type an operator command.
The highlighted characters will be operated upon.

The **'highlight'** option can be used to set the display mode to use for highlighting in Visual mode.

The **'virtualedit'** option can be used to allow positioning the cursor to positions where there is no actual character.

The highlighted text normally includes the character under the cursor. However, when the **'selection'** option is set to "exclusive" and the cursor is after the Visual area, the character under the cursor is not included.

With "v" the text before the start position and after the end position will not be highlighted. However, all uppercase and non-alpha operators, except "~" and "U", will work on whole lines anyway. See the list of operators below.

visual-block

With **CTRL-V** (blockwise Visual mode) the highlighted text will be a rectangle between start position and the cursor. However, some operators work on whole lines anyway (see the list below). The change and substitute operators will

delete the highlighted text and then start insertion at the top left position.

2. Starting and stopping Visual mode

`visual-start`

`v` `characterwise-visual`

`[count]v` Start Visual mode per character. With `[count]` select the same number of characters or lines as used for the last Visual operation, but at the current cursor position, multiplied by `[count]`. When the previous Visual operation was on a block both the width and height of the block are multiplied by `[count]`. When there was no previous Visual operation `[count]` characters are selected. This is like moving the cursor right $N * [count]$ characters. One less when `'selection'` is not "exclusive".

`V` `linewise-visual`

`[count]V` Start Visual mode linewise. With `[count]` select the same number of lines as used for the last Visual operation, but at the current cursor position, multiplied by `[count]`. When there was no previous Visual operation `[count]` lines are selected.

`CTRL-V` `blockwise-visual`

`[count]CTRL-V` Start Visual mode blockwise. **Note:** Under Windows `CTRL-V` could be mapped to paste text, it doesn't work to start Visual mode then, see `CTRL-V-alternative`. `[count]` is used as with ``v`` above.

If you use `<Esc>`, click the left mouse button or use any command that does a jump to another buffer while in Visual mode, the highlighting stops and no text is affected. Also when you hit `"v"` in characterwise Visual mode, `"CTRL-V"` in blockwise Visual mode or `"V"` in linewise Visual mode. If you hit `CTRL-Z` the highlighting stops and the editor is suspended or a new shell is started `CTRL-Z`.

old mode	new mode after typing: "v"	"CTRL-V"	<code>v_v</code>	<code>v_CTRL-V</code>	<code>v_V</code>
Normal	Visual	blockwise Visual		linewise Visual	
Visual	Normal	blockwise Visual		linewise Visual	
blockwise Visual	Visual	Normal		linewise Visual	
linewise Visual	Visual	blockwise Visual		Normal	

`gv` `v_gv` `reselect-Visual`

`gv` Start Visual mode with the same area as the previous area and the same mode. In Visual mode the current and the previous Visual area are exchanged. After using `"p"` or `"P"` in Visual mode the text that

was put will be selected.

`gn` `gn` `v_gn`
Search forward for the last used search pattern, like with ``n``, and start Visual mode to select the match. If the cursor is on the match, visually selects it. If an operator is pending, operates on the match. E.g., `"dgn"` deletes the text of the next match. If Visual mode is active, extends the selection until the end of the next match. `'wrapscan'` applies.
Note: Unlike ``n`` the search direction does not depend on the previous search command.

`gN` `gN` `v_gN`
Like `gn` but searches backward, like with ``N``.

`<LeftMouse>` `<LeftMouse>`
Set the current cursor position. If Visual mode is active it is stopped. Only when `'mouse'` option contains `'n'` or `'a'`. If the position is within `'so'` lines from the last line on the screen the text is scrolled up. If the position is within `'so'` lines from the first line on the screen the text is scrolled down.

`<RightMouse>` `<RightMouse>`
Start Visual mode if it is not active. The text from the cursor position to the position of the click is highlighted. If Visual mode was already active move the start or end of the highlighted text, whichever is closest, to the position of the click. Only when `'mouse'` option contains `'n'` or `'a'`.
Note: when `'mousemodel'` is set to `"popup"`, `<S-LeftMouse>` has to be used instead of `<RightMouse>`.

`<LeftRelease>` `<LeftRelease>`
This works like a `<LeftMouse>`, if it is not at the same position as `<LeftMouse>`. In an older version of xterm you won't see the selected area until the button is released, unless there is access to the display where the xterm is running (via the `DISPLAY` environment variable or the `-display` argument). Only when `'mouse'` option contains `'n'` or `'a'`.

`<LeftMouseNM>`
`<LeftReleaseNM>`
Internal mouse code, used for clicking on the status line to focus a window. NM stands for non-mappable. You cannot use these, but they might show up in some places.

If Visual mode is not active and the `"v"`, `"V"` or **CTRL-V** is preceded with a count, the size of the previously highlighted area is used for a start. You can then move the end of the highlighted area and give an operator. The type

of the old area is used (character, line or blockwise).

- Linewise Visual mode: The number of lines is multiplied with the count.
- Blockwise Visual mode: The number of lines and columns is multiplied with the count.
- Normal Visual mode within one line: The number of characters is multiplied with the count.
- Normal Visual mode with several lines: The number of lines is multiplied with the count, in the last line the same number of characters is used as in the last line in the previously highlighted area.

The start of the text is the Cursor position. If the "\$" command was used as one of the last commands to extend the highlighted text, the area will be extended to the rightmost column of the longest line.

If you want to highlight exactly the same area as the last time, you can use "gv" gv v_gv .

- `<Esc>` v_<Esc>
In Visual mode: Stop Visual mode.
- CTRL-C** v_CTRL-C
In Visual mode: Stop Visual mode. When insert mode is pending (the mode message shows "-- (insert) VISUAL --"), it is also stopped. On MS-Windows, you may need to press **CTRL-Break** dos-CTRL-Break .

3. Changing the Visual area visual-change

- `o` v_o
Go to Other end of highlighted text: The current cursor position becomes the start of the highlighted text and the cursor is moved to the other end of the highlighted text. The highlighted area remains the same.
- `O` v_O
Go to Other end of highlighted text. This is like "o", but in Visual block mode the cursor moves to the other corner in the same line. When the corner is at a character that occupies more than one position on the screen (e.g., a `<Tab>`), the highlighted text may change.

v_\$
When the "\$" command is used with blockwise Visual mode, the right end of the highlighted text will be determined by the longest highlighted line. This stops when a motion command is used that does not move straight up or down.

For moving the end of the block many commands can be used, but you cannot use Ex commands, commands that make changes or abandon the file. Commands (starting with) ".", "&", **CTRL-^**, "Z", **CTRL-]**, **CTRL-T**, **CTRL-R**, **CTRL-I** and **CTRL-O** cause a beep and Visual mode continues.

When switching to another window on the same buffer, the cursor position in that window is adjusted, so that the same Visual area is still selected. This is especially useful to view the start of the Visual area in one window, and the end in another. You can then use `<RightMouse>` (or `<S-LeftMouse>` when `'mousemodel'` is "popup") to drag either end of the Visual area.

4. Operating on the Visual area

visual-operators

The operators that can be used are:

<code>~</code>	switch case	<code>v_~</code>
<code>d</code>	delete	<code>v_d</code>
<code>c</code>	change (4)	<code>v_c</code>
<code>y</code>	yank	<code>v_y</code>
<code>></code>	shift right (4)	<code>v_></code>
<code><</code>	shift left (4)	<code>v_<</code>
<code>!</code>	filter through external command (1)	<code>v_!</code>
<code>=</code>	filter through <code>'equalprg'</code> option command (1)	<code>v_='</code>
<code>gq</code>	format lines to <code>'textwidth'</code> length (1)	<code>v_gq</code>

The objects that can be used are:

<code>aw</code>	a word (with white space)	<code>v_aw</code>
<code>iw</code>	inner word	<code>v_iw</code>
<code>aW</code>	a WORD (with white space)	<code>v_aW</code>
<code>iW</code>	inner WORD	<code>v_iW</code>
<code>as</code>	a sentence (with white space)	<code>v_as</code>
<code>is</code>	inner sentence	<code>v_is</code>
<code>ap</code>	a paragraph (with white space)	<code>v_ap</code>
<code>ip</code>	inner paragraph	<code>v_ip</code>
<code>ab</code>	a <code>()</code> block (with parentheses)	<code>v_ab</code>
<code>ib</code>	inner <code>()</code> block	<code>v_ib</code>
<code>aB</code>	a <code>{}</code> block (with braces)	<code>v_aB</code>
<code>iB</code>	inner <code>{}</code> block	<code>v_iB</code>
<code>at</code>	a <code><tag> </tag></code> block (with tags)	<code>v_at</code>
<code>it</code>	inner <code><tag> </tag></code> block	<code>v_it</code>
<code>a<</code>	a <code><></code> block (with <code><></code>)	<code>v_a<</code>
<code>i<</code>	inner <code><></code> block	<code>v_i<</code>
<code>a[</code>	a <code>[]</code> block (with <code>[]</code>)	<code>v_a[</code>
<code>i[</code>	inner <code>[]</code> block	<code>v_i[</code>
<code>a"</code>	a double quoted string (with quotes)	<code>v_aquote</code>
<code>i"</code>	inner double quoted string	<code>v_iquote</code>
<code>a'</code>	a single quoted string (with quotes)	<code>v_a'</code>
<code>i'</code>	inner simple quoted string	<code>v_i'</code>
<code>a`</code>	a string in backticks (with backticks)	<code>v_a`</code>
<code>i`</code>	inner string in backticks	<code>v_i`</code>

Additionally the following commands can be used:

<code>:</code>	start Ex command for highlighted lines (1)	<code>v_:</code>
<code>r</code>	change (4)	<code>v_r</code>
<code>s</code>	change	<code>v_s</code>
<code>C</code>	change (2)(4)	<code>v_C</code>
<code>S</code>	change (2)	<code>v_S</code>
<code>R</code>	change (2)	<code>v_R</code>
<code>x</code>	delete	<code>v_x</code>

D	delete (3)	v_D
X	delete (2)	v_X
Y	yank (2)	v_Y
p	put	v_p
P	put without overwriting registers	v_P
J	join (1)	v_J
U	make uppercase	v_U
u	make lowercase	v_u
^]	find tag	v_CTRL-]
I	block insert	v_b_I
A	block append	v_b_A

- (1): Always whole lines, see `:visual_example` .
- (2): Whole lines when not using **CTRL-V**.
- (3): Whole lines when not using **CTRL-V**, delete until the end of the line when using **CTRL-V**.
- (4): When using **CTRL-V** operates on the block only.

Note that the `:vmap` command can be used to specifically map keys in Visual mode. For example, if you would like the `/` command not to extend the Visual area, but instead take the highlighted text and search for that:

```
:vmap / y/<C-R>"<CR>
```

(In the `<>` notation `<>` , when typing it you should type it literally; you need to remove the 'B' and '<' flags from `'coptions'`.)

If you want to give a register name using the `""` command, do this just before typing the operator character: `"v{move-around}"xd`.

If you want to give a count to the command, do this just before typing the operator character: `"v{move-around}3>"` (move lines 3 indents to the right).

The `{move-around}` is any sequence of movement commands. **Note** the difference with `{motion}`, which is only ONE movement command.

Another way to operate on the Visual area is using the `/\%V` item in a pattern. For example, to replace all '(' in the Visual area with '#':

```
:'<,'>s/\%V(/#/g
```

Note that the `''<,'>` will appear automatically when you press `:"` in Visual mode.

5. Blockwise operators

blockwise-operators

Reminder: Use `'virtualedit'` to be able to select blocks that start or end after the end of a line or halfway a tab.

Visual-block Insert

v_b_I

With a blockwise selection, `I{string}<ESC>` will insert `{string}` at the start of block on every line of the block, provided that the line extends into the block. Thus lines that are short will remain unmodified. TABs are split to retain visual columns. Works only for adding text to a line, not for

deletions. See [v_b_I_example](#) .

Visual-block Append

[v_b_A](#)

With a blockwise selection, A{string}<ESC> will append {string} to the end of block on every line of the block. There is some differing behavior where the block RHS is not straight, due to different line lengths:

1. Block was created with <C-v>\$
In this case the string is appended to the end of each line.
2. Block was created with <C-v>{move-around}
In this case the string is appended to the end of the block on each line, and whitespace is inserted to pad to the end-of-block column.

See [v_b_A_example](#) .

Note: "I" and "A" behave differently for lines that don't extend into the selected block. This was done intentionally, so that you can do it the way you want.

Works only for adding text to a line, not for deletions.

Visual-block change

[v_b_c](#)

All selected text in the block will be replaced by the same text string. When using "c" the selected text is deleted and Insert mode started. You can then enter text (without a line break). When you hit <Esc>, the same string is inserted in all previously selected lines.

Visual-block Change

[v_b_C](#)

Like using "c", but the selection is extended until the end of the line for all lines.

Visual-block Shift

[v_b_<](#)

[v_b_>](#)

The block is shifted by '**shiftwidth**'. The RHS of the block is irrelevant. The LHS of the block determines the point from which to apply a right shift, and padding includes TABs optimally according to '**ts**' and '**et**'. The LHS of the block determines the point up to which to shift left.

See [v_b_>_example](#) .

See [v_b_<_example](#) .

Visual-block Replace

[v_b_r](#)

Every screen char in the highlighted region is replaced with the same char, ie TABs are split and the virtual whitespace is replaced, maintaining screen layout.

See [v_b_r_example](#) .

6. Repeating [visual-repeat](#)

When repeating a Visual mode operator, the operator will be applied to the same amount of text as the last time:

- Linewise Visual mode: The same number of lines.
- Blockwise Visual mode: The same number of lines and columns.
- Normal Visual mode within one line: The same number of characters.
- Normal Visual mode with several lines: The same number of lines, in the last line the same number of characters as in the last line the last time.

The start of the text is the Cursor position. If the "\$" command was used as one of the last commands to extend the highlighted text, the repeating will be applied up to the rightmost column of the longest line. Any count passed to the `.` command is not used.

7. Examples

visual-examples

:visual_example

Currently the ":" command works on whole lines only. When you select part of a line, doing something like "!:date" will replace the whole line. If you want only part of the line to be replaced you will have to make a mapping for it. In a future release ":" may work on partial lines.

Here is an example, to replace the selected text with the output of "date":

```
:vmap _a <Esc>`>a<CR><Esc>`<i<CR><Esc>!!date<CR>kJJ
```

(In the <> notation <> , when typing it you should type it literally; you need to remove the 'B' and '<' flags from 'coptions')

What this does is:

```
<Esc>      stop Visual mode
`>         go to the end of the Visual area
a<CR><Esc>  break the line after the Visual area
`<         jump to the start of the Visual area
i<CR><Esc>  break the line before the Visual area
!!date<CR> filter the Visual text through date
kJJ        Join the lines back together
```

visual-search

Here is an idea for a mapping that makes it possible to do a search for the selected text:

```
:vmap X y/<C-R>"<CR>
```

(In the <> notation <> , when typing it you should type it literally; you need to remove the 'B' and '<' flags from 'coptions')

Note that special characters (like '.' and '*') will cause problems.

Visual-block Examples

blockwise-examples

With the following text, I will indicate the commands to produce the block and the results below. In all cases, the cursor begins on the 'a' in the first line of the test text.

The following modeline settings are assumed ":ts=8:sw=4:".

It will be helpful to

```
:set hls
```

```
/<TAB>
```

where <TAB> is a real TAB. This helps visualise the operations.

The test text is:

```
abcdefghijklmnopqrstuvwxy
```



```
abc          defghijklmnopqrstuvwxyz
abcdef ghi   jklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

1. fo<C-v>3jISTRING<ESC>

v_b_I_example

```
abcdefghijklmnSTRINGopqrstuvwxyz
abc          STRING defghijklmnopqrstuvwxyz
abcdef ghi   STRING jklmnopqrstuvwxyz
abcdefghijklmnSTRINGopqrstuvwxyz
```

2. fo<C-v>3j\$ASTRING<ESC>

v_b_A_example

```
abcdefghijklmnopqrstuvwxyzSTRING
abc          defghijklmnopqrstuvwxyzSTRING
abcdef ghi   jklmnopqrstuvwxyzSTRING
abcdefghijklmnopqrstuvwxyzSTRING
```

3. fo<C-v>3j3l<..

v_b_<_example

```
abcdefghijklmnopqrstuvwxyz
abc          defghijklmnopqrstuvwxyz
abcdef ghi   jklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

4. fo<C-v>3j>..

v_b_>_example

```
abcdefghijklmn          opqrstuvwxyz
abc                    defghijklmnopqrstuvwxyz
abcdef ghi             jklmnopqrstuvwxyz
abcdefghijklmn          opqrstuvwxyz
```

5. fo<C-v>5l3jrX

v_b_r_example

```
abcdefghijklmnXXXXXXuvwxyz
abc          XXXXXXhijklmnopqrstuvwxyz
abcdef ghi   XXXXXX jklmnopqrstuvwxyz
abcdefghijklmnXXXXXXuvwxyz
```

=====
8. Select mode

Select Select-mode

Select mode looks like Visual mode, but the commands accepted are quite different. This resembles the selection mode in Microsoft Windows. When the '**showmode**' option is set, "-- SELECT --" is shown in the last line.

Entering Select mode:

- Using the mouse to select an area, and '**selectmode**' contains "mouse". '**mouse**' must also contain a flag for the current mode.
- Using a non-printable movement command, with the Shift key pressed, and '**selectmode**' contains "key". For example: <S-Left> and <S-End>. '**keymode**' must also contain "startsel".
- Using "v", "V" or **CTRL-V** command, and '**selectmode**' contains "cmd".
- Using "gh", "gH" or "g_CTRL-H" command in Normal mode.

- From Visual mode, press **CTRL-G**.

v_CTRL-G

Commands in Select mode:

- Printable characters, `<NL>` and `<CR>` cause the selection to be deleted, and Vim enters Insert mode. The typed character is inserted.
- Non-printable movement commands, with the Shift key pressed, extend the selection. **'keymodel'** must include "startsel".
- Non-printable movement commands, with the Shift key NOT pressed, stop Select mode. **'keymodel'** must include "stopsel".
- ESC stops Select mode.
- **CTRL-O** switches to Visual mode for the duration of one command. **v_CTRL-O**
- **CTRL-G** switches to Visual mode.
- **CTRL-R {register}** selects the register to be used for the text that is deleted when typing text. **v_CTRL-R**
Unless you specify the "_" (black hole) register, the unnamed register is also overwritten.

Otherwise, typed characters are handled as in Visual mode.

When using an operator in Select mode, and the selection is linewise, the selected lines are operated upon, but like in characterwise selection. For example, when a whole line is deleted, it can later be pasted halfway a line.

Mappings and menus in Select mode.

Select-mode-mapping

When mappings and menus are defined with the `:vmap` or `:vmenu` command they work both in Visual mode and in Select mode. When these are used in Select mode Vim automatically switches to Visual mode, so that the same behavior as in Visual mode is effective. If you don't want this use `:xmap` or `:smap`.

One particular edge case:

```
:vnoremap <C-K> <Esc>
```

This ends Visual mode when in Visual mode, but in Select mode it does not work, because Select mode is restored after executing the mapped keys. You need to use:

```
:snoremap <C-K> <Esc>
```

Users will expect printable characters to replace the selected area. Therefore avoid mapping printable characters in Select mode. Or use `:sunmap` after `:map` and `:vmap` to remove it for Select mode.

After the mapping or menu finishes, the selection is enabled again and Select mode entered, unless the selected area was deleted, another buffer became the current one or the window layout was changed.

When a character was typed that causes the selection to be deleted and Insert mode started, Insert mode mappings are applied to this character. This may cause some confusion, because it means Insert mode mappings apply to a character typed in Select mode. Language mappings apply as well.

gV

gV v_gV

Avoid the automatic reselection of the Visual area

after a Select mode mapping or menu has finished. Put this just before the end of the mapping or menu. At least it should be after any operations on the selection.

`gh` Start Select mode, characterwise. This is like "v", but starts Select mode instead of Visual mode. Mnemonic: "get highlighted".

`gH` Start Select mode, linewise. This is like "V", but starts Select mode instead of Visual mode. Mnemonic: "get Highlighted".

`g CTRL-H` Start Select mode, blockwise. This is like **CTRL-V**, but starts Select mode instead of Visual mode. Mnemonic: "get Highlighted".

Various commands various

- 1. Various commands various-cmds
- 2. Using Vim like less or more less

1. Various commands various-cmds

CTRL-L CTRL-L
 Clear and redraw the screen. The redraw may happen later, after processing typeahead.

:redr[aw][!] :redr :redraw
 Redraw the screen right now. When ! is included it is cleared first.
 Useful to update the screen halfway executing a script or function. Also when halfway a mapping and **'lazyredraw'** is set.

:redraws[tatus][!] :redraws :redrawstatus
 Redraw the status line of the current window. When ! is included all status lines are redrawn.
 Useful to update the status line(s) when **'statusline'** includes an item that doesn't cause automatic updating.
 If the command line is being edited the redraw is postponed until later.

:redrawt[abline] :redrawt :redrawtabline
 Redraw the tabline. Useful to update the tabline when **'tabline'** includes an item that doesn't trigger automatic updating.

**** N
 When entering a number: Remove the last digit.
Note: if you like to use <BS> for this, add this mapping to your .vimrc:
:map CTRL-V <BS> CTRL-V
 See **:fixdel** if your key does not do what you want.

:as[cii] ga :as :ascii
 or
ga
 Print the ascii value of the character under the cursor in decimal, hexadecimal and octal.
 Mnemonic: Get Ascii value.

For example, when the cursor is on a 'R':
<R> 82, Hex 52, Octal 122

When the character is a non-standard ASCII character, but printable according to the **'isprint'** option, the non-printable version is also given.

When the character is larger than 127, the `<M-x>` form is also printed. For example:

```
<~A> <M-^A> 129, Hex 81, Octal 201
<p> <|~> <M-~> 254, Hex fe, Octal 376
```

(where `<p>` is a special character)

The `<Nu\>` character in a file is stored internally as `<NL>`, but it will be shown as:

```
<^@> 0, Hex 00, Octal 000
```

If the character has composing characters these are also shown. The value of **'maxcombine'** doesn't matter.

If the character can be inserted as a digraph, also output the two characters that can be used to create the character:

```
<ö> 246, Hex 00f6, Oct 366, Digr o:
```

This shows you can type `CTRL-K o :` to insert `ö`.

`g8`

`g8`

Print the hex values of the bytes used in the character under the cursor, assuming it is in `UTF-8` encoding. This also shows composing characters. The value of **'maxcombine'** doesn't matter. Example of a character with two composing characters:

```
e0 b8 81 + e0 b8 b9 + e0 b9 89
```

`8g8`

`8g8`

Find an illegal UTF-8 byte sequence at or after the cursor. This works in two situations:

1. when **'encoding'** is any 8-bit encoding
2. when **'encoding'** is "utf-8" and **'fileencoding'** is any 8-bit encoding

Thus it can be used when editing a file that was supposed to be UTF-8 but was read as if it is an 8-bit encoding because it contains illegal bytes. Does not wrap around the end of the file. **Note** that when the cursor is on an illegal byte or the cursor is halfway a multibyte character the command won't move the cursor.

`:p :pr :print E749`

`:[range]p[rint] [flags]`

Print `[range]` lines (default current line). **Note:** If you are looking for a way to print your text on paper see `:hardcopy`. In the GUI you can use the File.Print menu entry. See `ex-flags` for `[flags]`. The `:filter` command can be used to only show lines matching a pattern.

`:[range]p[rint] {count} [flags]`
Print `{count}` lines, starting with `[range]` (default current line `cmdline-ranges`).
See `ex-flags` for `[flags]`.

`:P` `:Print`

`:[range]P[rint] {count} [flags]`
Just as `:"print"`. Was apparently added to Vi for people that keep the shift key pressed too long... This command is not supported in Vim9 script.
Note: A user command can overrule this command.
See `ex-flags` for `[flags]`.

`:l` `:list`

`:[range]l[ist] {count} [flags]`
Same as `:"print`, but display unprintable characters with '^' and put \$ after the line. This can be further changed with the `'listchars'` option.
See `ex-flags` for `[flags]`.

`:nu` `:number`

`:[range]nu[mber] {count} [flags]`
Same as `:"print`, but precede each line with its line number. (See also `'highlight'` and `'numberwidth'` option).
See `ex-flags` for `[flags]`.

`:#`

`:[range]# {count} [flags]`
synonym for `:"number`.

`:#!`

`:#!{anything}` Ignored, so that you can start a Vim script with:
`#!vim -S`
`echo "this is a Vim script"`
`quit`

`:z` `E144`

`:[range]z[+-^.=]{count}` Display several lines of text surrounding the line specified with `[range]`, or around the current line if there is no `[range]`.

If there is a `[count]`, that's how many lines you'll see; if there is no `[count]` and only one window then twice the value of the `'scroll'` option is used, otherwise the current window height minus 3 is used. This is the value of "scr" in the table below.

If there is a `[count]` the `'window'` option is set to its value.

`:z` can be used either alone or followed by any of several marks. These have the following effect:

mark	first line	last line	new cursor line
+	current line	1 scr forward	1 scr forward
-	1 scr back	current line	current line
^	2 scr back	1 scr back	1 scr back
.	1/2 scr back	1/2 scr fwd	1/2 scr fwd
=	1/2 scr back	1/2 scr fwd	current line

Specifying no mark at all is the same as "+".
 If the mark is "=", a line of dashes is printed
 around the current line.

- `:z!`
`:[range]z![+-^.=][count]`
 Like ":z", but when [count] is not specified, it defaults to the Vim window height minus one.
- `:z#`
`:[range]z[!][+-^.=][count]`
 Like ":z" or ":z!", but number the lines.
- `:=`
`:= [flags]`
 Print the last line number.
 See `ex-flags` for [flags].
- `:.=`
`:{range}= [flags]`
 Prints the last line number in {range}. For example, this prints the current line number:
`:.=`
 See `ex-flags` for [flags].
- `:norm` `:normal`
`:norm[al][!] {commands}`
 Execute Normal mode commands {commands}. This makes it possible to execute Normal mode commands typed on the command-line. {commands} are executed like they are typed. For undo all commands are undone together. Execution stops when an error is encountered.
- If the [!] is given, mappings will not be used. Without it, when this command is called from a non-remappable mapping (`:noremap`), the argument can be mapped anyway.
- {commands} should be a complete command. If {commands} does not finish a command, the last one will be aborted as if `<Esc>` or `<C-C>` was typed. This implies that an insert command must be completed (to start Insert mode, see `:startinsert`). A ":" command must be completed as well. And you can't use "Q" or "gQ" to start Ex mode.
- The display is not updated while ":normal" is busy.
- {commands} cannot start with a space. Put a count of 1 (one) before it, "1 " is one space.

The `'insertmode'` option is ignored for `{commands}`.

This command cannot be followed by another command, since any `'|'` is considered part of the command.

This command can be used recursively, but the depth is limited by `'maxmapdepth'`.

An alternative is to use `:execute`, which uses an expression as argument. This allows the use of printable characters to represent special characters.

Example:

```
:exe "normal \<c-w\<c-w"
```

`:{range}norm[al][!] {commands}` `:normal-range`
Execute Normal mode commands `{commands}` for each line in the `{range}`. Before executing the `{commands}`, the cursor is positioned in the first column of the range, for each line. Otherwise it's the same as the `":normal"` command without a range.

`:sh[ell]` `:sh :shell E371`
This command starts a shell. When the shell exits (after the "exit" command) you return to Vim. The name for the shell command comes from `'shell'` option. `E360`

Note: This doesn't work when Vim on the Amiga was started in QuickFix mode from a compiler, because the compiler will have set stdin to a non-interactive mode.

`:{cmd}` `!:cmd :!`
Execute `{cmd}` with the shell. See also the `'shell'` and `'shelltype'` option. `E34`
Any `'!'` in `{cmd}` is replaced with the previous external command (see also `'cptions'`). But not when there is a backslash before the `'!'`, then that backslash is removed. Example: `":!ls"` followed by `":!echo ! \! \!"` executes `"echo ls ! \!"`.

A `'|'` in `{cmd}` is passed to the shell, you cannot use it to append a Vim command. See `:bar`.

If `{cmd}` contains `"%"` it is expanded to the current file name, `"#"` is expanded to the alternate file name. Special characters in the file name are not escaped, use quotes to avoid their special meaning:

```
:!ls "%"
```

If the file name contains a `"$"` then single quotes might work better, but this only works if the file name does not contain a single quote:

```
    :!ls '%'
```

This should always work, but it's more typing:

```
    :exe "!ls " .. shellescape(expand("%"))
```

To get a literal "%" or "#" prepend it with a backslash. For example, to list all files starting with "%":

```
    :!ls \%*
```

A newline character ends `{cmd}`, what follows is interpreted as a following ":" command. However, if there is a backslash before the newline it is removed and `{cmd}` continues. It doesn't matter how many backslashes are before the newline, only one is removed.

On Unix the command normally runs in a non-interactive shell. If you want an interactive shell to be used (to use aliases) set '`shellcmdflag`' to "-ic". For Win32 also see `!start` .

After the command has been executed, the timestamp and size of the current file is checked `timestamp` .

Vim redraws the screen after the command is finished, because it may have printed any text. This requires a hit-enter prompt, so that you can read any messages. To avoid this use:

```
    :silent !{cmd}
```

The screen is not redrawn then, thus you have to use **CTRL-L** or `":redraw!"` if the command did display something. However, this depends on what the `t_ti` and `t_te` termcap entries are set to. Also see `shell-window` .

```
!!!
```

Repeat last `":!{cmd}"`.

```
!!!
```

```
:ve[rsion]
```

Print the version number of the editor. If the compiler used understands `"__DATE__"` the compilation date is mentioned. Otherwise a fixed release-date is shown.

The following lines contain information about which features were enabled when Vim was compiled. When there is a preceding '+', the feature is included, when there is a '-' it is excluded. To change this, you have to edit `feature.h` and recompile Vim.

To check for this in an expression, see `has()` .

Here is an overview of the features.

The first column shows the smallest version in which they are included:

```
T    tiny (always)
N    normal
H    huge
```

m manually enabled or depends on other features
 - never, feature was removed
 (none) system dependent
 Thus if a feature is marked with "N", it is included in the normal and huge versions of Vim.

+feature-list

	<code>+acl</code>	ACL support included
	<code>+ARP</code>	Amiga only: ARP support included
H	<code>+arabic</code>	Arabic language support
N	<code>+autochdir</code>	support <code>'autochdir'</code> option
T	<code>+autocmd</code>	<code>:autocmd</code> , automatic commands. Always enabled since 8.0.1564
H	<code>+autoservername</code>	Automatically enable <code>clientserver</code>
m	<code>+balloon_eval</code>	<code>balloon-eval</code> support in the GUI. Included when compiling with supported GUI (Motif, GTK, GUI) and either Netbeans/Sun Workshop integration or <code>+eval</code> feature.
H	<code>+balloon_eval_term</code>	<code>balloon-eval</code> support in the terminal, <code>'balloonevalterm'</code>
N	<code>+browse</code>	<code>:browse</code> command
T	<code>++builtin_terms</code>	maximal terminals builtin <code>builtin-terms</code> Always enabled since 9.0.0280
N	<code>+byte_offset</code>	support for 'o' flag in <code>'statusline'</code> option, "go" and <code>":goto"</code> commands.
m	<code>+channel</code>	inter process communication <code>channel</code>
T	<code>+cindent</code>	<code>'cindent'</code> , C indenting; Always enabled
N	<code>+clientserver</code>	Unix and Win32: Remote invocation <code>clientserver</code>
	<code>+clipboard</code>	<code>clipboard</code> support compiled-in
	<code>+clipboard_working</code>	<code>clipboard</code> support compiled-in and working
T	<code>+cmdline_compl</code>	command line completion <code>cmdline-completion</code>
T	<code>+cmdline_hist</code>	command line history <code>cmdline-history</code>
T	<code>+cmdline_info</code>	<code>'showcmd'</code> and <code>'ruler'</code> ; Always enabled since 9.0.0747
T	<code>+cmdwin</code>	<code>cmdline-window</code> support; Always enabled since 9.0.0657
T	<code>+comments</code>	<code>'comments'</code> support
N	<code>+conceal</code>	"conceal" support, see <code>conceal</code> <code>:syn-conceal</code> etc.
N	<code>+cryptv</code>	encryption support <code>encryption</code>
H	<code>+cscope</code>	<code>cscope</code> support
T	<code>+cursorbind</code>	<code>'cursorbind'</code> support
m	<code>+cursorshape</code>	<code>termcap-cursor-shape</code> support
m	<code>+debug</code>	Compiled for debugging.
N	<code>+dialog_gui</code>	Support for <code>:confirm</code> with GUI dialog.
N	<code>+dialog_con</code>	Support for <code>:confirm</code> with console dialog.
N	<code>+dialog_con_gui</code>	Support for <code>:confirm</code> with GUI and console dialog.
N	<code>+diff</code>	<code>vimdiff</code> and <code>'diff'</code>
N	<code>+digraphs</code>	<code>digraphs</code> E196
	<code>+directx</code>	Win32 GUI only: DirectX and <code>'renderoptions'</code>
	<code>+dnd</code>	Support for DnD into the <code>"~</code> register <code>quote_~</code> .
H	<code>+emacs_tags</code>	<code>emacs-tags</code> files
N	<code>+eval</code>	expression evaluation <code>eval.txt</code>
T	<code>+ex_extra</code>	always on now, used to be for Vim's extra Ex commands
N	<code>+extra_search</code>	<code>'hlsearch'</code> and <code>'incsearch'</code> options.
-	<code>+farsi</code>	Removed: <code>farsi</code> language

T	+file_in_path	gf , CTRL-W_f and <cfile> Always enabled since 9.0.265
N	+find_in_path	include file searches: [I , :isearch , CTRL-W_CTRL-I , :checkpath , etc.
N	+folding	folding
	+footer	gui-footer
	+fork	Unix only: fork shell commands
T	+float	Floating point support Always enabled since 9.0.0491
N	+gettext	message translations multi-lang
-	+GUI_Athena	Unix only: Athena GUI
	+GUI_neXtaw	Unix only: neXtaw GUI
	+GUI_GTK	Unix only: GTK+ GUI
	+GUI_Motif	Unix only: Motif GUI
	+GUI_Photon	QNX only: Photon GUI
m	+hangul_input	Hangul input support hangul
	+iconv	Compiled with the iconv() function
	+iconv/dyn	Likewise iconv-dynamic /dyn
T	+insert_expand	insert_expand Insert mode completion
m	+ipv6	Support for IPv6 networking channel
m	+job	starting and stopping jobs job
T	+jumplist	jumplist ; Always enabled since 8.2.3795
H	+keymap	'keymap'
N	+lambda	lambda and closure
H	+langmap	'langmap'
N	+libcall	libcall()
N	+linebreak	'linebreak' , 'breakat' and 'showbreak'
T	+lispindent	'lisp'
T	+listcmds	Vim commands for the list of buffers buffer-hidden and argument list :argdelete
T	+localmap	Support for mappings local to a buffer :map-local
m	+lua	Lua interface
m	+lua/dyn	Lua interface /dyn
N	+menu	:menu
N	+mksession	:mksession
T	+modify_fname	filename-modifiers
T	+mouse	Mouse handling mouse-using
N	+mousethshape	'mousethshape'
N	+mouse_dec	Unix only: Dec terminal mouse handling dec-mouse
N	+mouse_gpm	Unix only: Linux console mouse handling gpm-mouse
m	+mouse_gpm/dyn	Same as +mouse_gpm with optional library dependency /dyn
N	+mouse_jsbterm	JSB mouse handling jsbterm-mouse
N	+mouse_netterm	Unix only: netterm mouse handling netterm-mouse
N	+mouse_pterm	QNX only: pterm mouse handling qnx-terminal
N	+mouse_sysmouse	Unix only: *BSD console mouse handling sysmouse
N	+mouse_sgr	Unix only: sgr mouse handling sgr-mouse
N	+mouse_urxvt	Unix only: urxvt mouse handling urxvt-mouse
N	+mouse_xterm	Unix only: xterm mouse handling xterm-mouse
T	+multi_byte	Unicode support, 16 and 32 bit characters multibyte
	+multi_byte_ime	Win32 input method for multibyte chars multibyte-ime
N	+multi_lang	non-English language support multi-lang
m	+mzscheme	Mzscheme interface mzscheme
m	+mzscheme/dyn	Mzscheme interface mzscheme-dynamic /dyn
m	+netbeans_intg	netbeans

T **+num64** 64-bit Number support `Number`
Always enabled since 8.2.0271, use `v:numbersize` to check the actual size of a Number.

m **+ole** Win32 GUI only: `ole-interface`

N **+packages** Loading `packages`

T **+path_extra** Up/downwards search in `'path'` and `'tags'` Always enabled since 9.0.0270

m **+perl** Perl interface `perl`

m **+perl/dyn** Perl interface `perl-dynamic /dyn`

N **+persistent_undo** Persistent undo `undo-persistence`

N **+popupwin** Popup windows `popup-window`

+postscript `:hardcopy` writes a PostScript file

N **+printer** `:hardcopy` command

H **+profile** `:profile` command

m **+python** Python 2 interface `python`

m **+python/dyn** Python 2 interface `python-dynamic /dyn`

m **+python3** Python 3 interface `python`

m **+python3/dyn** Python 3 interface `python-dynamic /dyn`

m **+python3/dyn-stable** Python 3 interface `python-dynamic python-stable /dyn`

N **+quickfix** `:make` and `quickfix` commands

N **+reltime** `reltime()` function, `'hlsearch'`/`'incsearch'` timeout, `'redrawtime'` option

H **+rightleft** Right to left typing `'rightleft'`

m **+ruby** Ruby interface `ruby`

m **+ruby/dyn** Ruby interface `ruby-dynamic /dyn`

T **+scrollbind** `'scrollbind'`

N **+signs** `:sign`

T **+smartindent** `'smartindent'`

H **+sodium** compiled with libsodium for better encryption support

H **+sound** `sound_playevent()` , `sound_playfile()` functions, etc.

N **+spell** spell checking support, see `spell`

N **+startuptime** `--startuptime` argument

N **+statusline** Options `'statusline'`, `'rulerformat'` and special formats of `'titlestring'` and `'iconstring'`

- **+sun_workshop** Removed: `workshop`

N **+syntax** Syntax highlighting `syntax`

+system() Unix only: opposite of `+fork`

T **+tag_binary** binary searching in tags file `tag-binary-search`

- **+tag_old_static** Removed; method for static tags `tag-old-static`

- **+tag_any_white** Removed; was to allow any white space in tags files

m **+tcl** Tcl interface `tcl`

m **+tcl/dyn** Tcl interface `tcl-dynamic /dyn`

m **+terminal** Support for terminal window `terminal`

+terminfo uses `terminfo` instead of `termcap`

N **+termresponse** support for `t_RV` and `v:termresponse`

N **+termguicolors** 24-bit color in xterm-compatible terminals support

T **+textobjects** `text-objects` selection. Always enabled since 9.0.0222.

N **+textprop** `text-properties`

+tgetent non-Unix only: able to use external `termcap`

N **+timers** the `timer_start()` function

T **+title** Setting the window `'title'` and `'icon'`; Always enabled

N **+toolbar** `gui-toolbar`

T **+user_commands** User-defined commands. `user-commands`
 Always enabled since 8.1.1210.
H **+vartabs** Variable-width tabstops. `'vartabstop'`
T **+vertspl** Vertically split windows `:vsplit` ; Always enabled
 since 8.0.1118.
T **+vim9script** Vim9 script
N **+viminfo** `'viminfo'`
T **+virtualedit** `'virtualedit'` Always enabled since 8.1.826.
T **+visual** Visual mode `Visual-mode` Always enabled since 7.4.200.
T **+visualextra** extra Visual mode commands `blockwise-operators`
T **+vreplace** `gR` and `gr`
+vtp on MS-Windows console: support for `'termguicolors'`
T **+wildignore** `'wildignore'` Always enabled since 9.0.0278
T **+wildmenu** `'wildmenu'` Always enabled since 9.0.0279
T **+windows** more than one window; Always enabled since 8.0.1118.
m **+writebackup** `'writebackup'` is default on
m **+xim** X input method `xim`
+xfontset X fontset support `xfontset`
N **+xattr** compiled with extended attribute support (Linux only)
+xpm pixmap support
m **+xpm_w32** Win32 GUI only: pixmap support `w32-xpm-support`
+xsmp XSMP (X session management) support
+xsmp_interact interactive XSMP (X session management) support
N **+xterm_clipboard** Unix only: xterm clipboard handling
m **+xterm_save** save and restore xterm screen `xterm-screens`
N **+X11** Unix only: can restore window title `X11`

`/dyn` `E370` `E448`

To some of the features `"/dyn"` is added when the feature is only available when the related library can be dynamically loaded.

`:ve[rsion] {nr}` Is now ignored. This was previously used to check the version number of a `.vimrc` file. It was removed, because you can now use the `":if"` command for version-dependent behavior.

`:redi` `:redir`

`:redi[r][!] > {file}` Redirect messages to file `{file}`. The messages which are the output of commands are written to that file, until redirection ends. The messages are also still shown on the screen. When `[:!]` is included, an existing file is overwritten. When `[:!]` is omitted, and `{file}` exists, this command fails.

Only one `":redir"` can be active at a time. Calls to `":redir"` will close any active redirection before starting redirection to the new target. For recursive use check out `execute()` .

To stop the messages and commands from being echoed to the screen, put the commands in a function and call it with `":silent call Function()"`.

An alternative is to use the `'verbosefile'` option,

this can be used in combination with ":redir".

```
:redi[r] >> {file}      Redirect messages to file {file}. Append if {file}
                        already exists.

:redi[r] @{a-zA-Z}
:redi[r] @{a-zA-Z}>     Redirect messages to register {a-z}. Append to the
                        contents of the register if its name is given
                        uppercase {A-Z}. The ">" after the register name is
                        optional.

:redi[r] @{a-z}>>       Append messages to register {a-z}.

:redi[r] @*>
:redi[r] @+>           Redirect messages to the selection or clipboard. For
                        backward compatibility, the ">" after the register
                        name can be omitted. See quotestar and quoteplus .

:redi[r] @*>>
:redi[r] @+>>         Append messages to the selection or clipboard.

:redi[r] @">
:redi[r] @">>         Redirect messages to the unnamed register. For
                        backward compatibility, the ">" after the register
                        name can be omitted.
:redi[r] @">>         Append messages to the unnamed register.
                        E1092

:redi[r] => {var}       Redirect messages to a variable.
                        In legacy script: If the variable doesn't exist, then
                        it is created. If the variable exists, then it is
                        initialized to an empty string. After the redirection
                        starts, if the variable is removed or locked or the
                        variable type is changed, then further command output
                        messages will cause errors. When using a local
                        variable (l:var in a function or s:var in a script)
                        and another `:redir` causes the current one to end,
                        the scope might be different and the assignment fails.
                        In Vim9 script: the variable must have been declared
                        as a string.
                        The variable will remain empty until redirection ends.
                        Only string variables can be used.
                        To get the output of one command the execute\(\)
                        function can be used instead of redirection.

:redi[r] =>> {var}      Append messages to an existing variable. Only string
                        variables can be used.
                        E1185

:redi[r] END           End redirecting messages.

                        :filt  :filter

:filt[er][!] {pattern} {command}
:filt[er][!] /{pattern}/ {command}
                        Restrict the output of {command} to lines matching
                        with {pattern}. For example, to list only xml files:
                        :filter /\.xml$/ oldfiles
                        If the [!] is given, restrict the output of {command}
                        to lines that do NOT match {pattern}.
```

`{pattern}` is a Vim search pattern. Instead of enclosing it in `/` any non-ID character (see `'isident'`) can be used, so long as it does not appear in `{pattern}`. Without the enclosing character the pattern cannot include the bar character. `'ignorecase'` is not used.

The pattern is matched against the relevant part of the output, not necessarily the whole line. Only some commands support filtering, try it out to check if it works. Some of the commands that support filtering:

```
:#           - filter whole line
:clist       - filter by file name or module name
:command     - filter by command name
:files       - filter by file name
:highlight   - filter by highlight group
:jumps       - filter by file name
:let         - filter by variable name
:list        - filter whole line
:llist       - filter by file name or module name
:marks       - filter by text in the current file,
              or file name for other files
:oldfiles    - filter by file name
:registers   - filter by register contents
              (does not work multi-line)
:set         - filter by option name
```

Only normal messages are filtered, error messages are not.

```
                                :sil  :silent  :silent!
:sil[ent][!] {command} Execute {command} silently. Normal messages will not
                        be given or added to the message history.
                        When [!] is added, error messages will also be
                        skipped, and commands and mappings will not be aborted
                        when an error is detected. v:errmsg is still set.
                        When [!] is not used, an error message will cause
                        further messages to be displayed normally.
                        Redirection, started with :redir, will continue as
                        usual, although there might be small differences.
                        This will allow redirecting the output of a command
                        without seeing it on the screen. Example:
                        :redir >/tmp/foobar
                        :silent g/Aap/p
                        :redir END
```

To execute a Normal mode command silently, use the `:normal` command. For example, to search for a string without messages:

```
:silent exe "normal /path\<CR>"
```

`":silent!"` is useful to execute a command that may fail, but the failure is to be ignored. Example:

```
:let v:errmsg = ""
:silent! /^begin
:if v:errmsg != ""
```


`: ... pattern was not found`

`":silent"` will also avoid the hit-enter prompt. When using this for an external command, this may cause the screen to be messed up. Use `CTRL-L` to clean it up then.

`":silent menu ..."` defines a menu that will not echo a Command-line command. The command will still produce messages though. Use `":silent"` in the command itself to avoid that: `":silent menu :silent command"`.

`:uns` `:unsilent`

`:uns[ilent] {command}` Execute `{command}` not silently. Only makes a difference when `:silent` was used to get to this command.

Use this for giving a message even when `:silent` was used. In this example `:silent` is used to avoid the message about reading the file and `:unsilent` to be able to list the first line of each file.

```
:silent argdo unsilent echo expand('%') .. ": " .. getline(1)
```

`:verb` `:verbose`

`:[count]verb[ose] {command}`

Execute `{command}` with `'verbose'` set to `[count]`. If `[count]` is omitted one is used. `":@verbose"` can be used to set `'verbose'` to zero.

The additional use of `":silent"` makes messages generated but not displayed.

The combination of `":silent"` and `":verbose"` can be used to generate messages and check them with

`v:statusmsg` and friends. For example:

```
:let v:statusmsg = ""
:silent verbose runtime foobar.vim
:if v:statusmsg != ""
: " foobar.vim could not be found
:endif
```

When concatenating another command, the `":verbose"` only applies to the first one:

```
:4verbose set verbose | set verbose
verbose=4
verbose=0
```

For logging verbose messages in a file use the `'verbosefile'` option.

`:verbose-cmd`

When `'verbose'` is non-zero, listing the value of a Vim option or a key map or an abbreviation or a user-defined function or a command or a highlight group or an autocommand will also display where it was last defined. If it was defined manually then there will be no "Last set" message. When it was defined while executing a function, user command or autocommand, the script in which it was defined is reported.

`{not available when compiled without the |+eval| feature}`

K

K

Run a program to lookup the keyword under the cursor. The name of the program is given with the **'keywordprg'** (kp) option (default is "man"). The keyword is formed of letters, numbers and the characters in **'iskeyword'**. The keyword under or right of the cursor is used. The same can be done with the command

```
!:{program} {keyword}
```

There is an example of a program to use in the tools directory of Vim. It is called "ref" and does a simple spelling check.

Special cases:

- If **'keywordprg'** begins with ":" it is invoked as a Vim Ex command with [count].
- If **'keywordprg'** is empty, the ":help" command is used. It's a good idea to include more characters in **'iskeyword'** then, to be able to find more help.
- When **'keywordprg'** is equal to "man" or starts with ":", a [count] before "K" is inserted after keywordprg and before the keyword. For example, using "2K" while the cursor is on "mkdir", results in:

```
!man 2 mkdir
```

- When **'keywordprg'** is equal to "man -s", a count before "K" is inserted after the "-s". If there is no count, the "-s" is removed.

v_K

{Visual}K

Like "K", but use the visually highlighted text for the keyword. Only works when the highlighted text is not more than one line.

[N]gs

: [N]sl[eeep] [N][m]

gs :sl :sleep

Do nothing for [N] seconds. When [m] is included, sleep for [N] milliseconds. The count for "gs" always uses seconds. The default is one second.

```
:sleep          "sleep for one second
:5sleep         "sleep for five seconds
:sleep 100m     "sleep for 100 milliseconds
10gs           "sleep for ten seconds
```

Can be interrupted with **CTRL-C** (CTRL-Break on MS-Windows). "gs" stands for "goto sleep".

While sleeping the cursor is positioned in the text, if at a visible position.

Also process the received netbeans messages. {only available when compiled with the **+netbeans_intg** feature}

:sl! :sleep!

: [N]sl[eeep]! [N][m]

Same as above, but hide the cursor.

:xrestore :xr

:xr[estore] [display]

Reinitializes the connection to the X11 server. Useful after the X server restarts, e.g. when running Vim for

long time inside screen/tmux and connecting from different machines.
[display] should be in the format of the \$DISPLAY environment variable (e.g. "localhost:10.0")
If [display] is omitted, then it reinitializes the connection to the X11 server using the same value as was used for the previous execution of this command. If the value was never specified, then it uses the value of \$DISPLAY environment variable as it was when Vim was started.
{only available when compiled with the +clipboard feature}

g CTRL-A g_CTRL-A
Only when Vim was compiled with MEM_PROFILING defined (which is very rare): print memory usage statistics. Only useful for debugging Vim.
For incrementing in Visual mode see [v_g_CTRL-A](#) .

2. Using Vim like less or more

less

If you use the less or more program to view a file, you don't get syntax highlighting. Thus you would like to use Vim instead. You can do this by using the shell script "\$VIMRUNTIME/macros/less.sh".

This shell script uses the Vim script "\$VIMRUNTIME/macros/less.vim". It sets up mappings to simulate the commands that less supports. Otherwise, you can still use the Vim commands.

This isn't perfect. For example, when viewing a short file Vim will still use the whole screen. But it works well enough for most uses, and you get syntax highlighting.

The "h" key will give you a short overview of the available commands.

If you want to set options differently when using less, define the LessInitFunc in your vimrc, for example:

```
func LessInitFunc()  
    set nocursorcolumn nocursorline  
endfunc
```


Recovery after a crash crash-recovery

You have spent several hours typing in that text that has to be finished next morning, and then disaster strikes: Your computer crashes.

DON'T PANIC!

You can recover most of your changes from the files that Vim uses to store the contents of the file. Mostly you can recover your work with one command:

```
vim -r filename
```

- 1. The swap file swap-file
- 2. Recovery recovery

1. The swap file swap-file

Vim stores the things you changed in a swap file. Using the original file you started from plus the swap file you can mostly recover your work.

You can see the name of the current swap file being used with the command:

```
:sw[apname] :sw :swapname
```

Or you can use the `swapname()` function, which also allows for seeing the swap file name of other buffers.

The name of the swap file is normally the same as the file you are editing, with the extension ".swp".

- On Unix, a '.' is prepended to swap file names in the same directory as the edited file. This avoids that the swap file shows up in a directory listing.
- On MS-Windows machines and when the **'shortname'** option is on, any '.' in the original file name is replaced with '_'.
- If this file already exists (e.g., when you are recovering from a crash) a warning is given and another extension is used, ".sw0", ".swn", etc.
- An existing file will never be overwritten.
- The swap file is deleted as soon as Vim stops editing the file.

Technical: The replacement of '.' with '_' is done to avoid problems with MS-DOS compatible filesystems (e.g., crossdos, multidos). If Vim is able to detect that the file is on an MS-DOS-like filesystem, a flag is set that has the same effect as the **'shortname'** option. This flag is reset when you start editing another file.

E326

If the ".swp" file name already exists, the last character is decremented until there is no file with that name or ".saa" is reached. In the last case, no swap file is created.

By setting the **'directory'** option you can place the swap file in another place than where the edited file is.

Advantages:

- You will not pollute the directories with ".swp" files.
- When the **'directory'** is on another partition, reduce the risk of damaging the file system where the file is (in a crash).

Disadvantages:

- You can get name collisions from files with the same name but in different directories (although Vim tries to avoid that by comparing the path name). This will result in bogus ATTENTION warning messages.
- When you use your home directory, and somebody else tries to edit the same file, that user will not see your swap file and will not get the ATTENTION warning message.

On the Amiga you can also use a recoverable ram disk, but there is no 100% guarantee that this works. Putting swap files in a normal ram disk (like RAM: on the Amiga) or in a place that is cleared when rebooting (like /tmp on Unix) makes no sense, you will lose the swap file in a crash.

If you want to put swap files in a fixed place, put a command resembling the following ones in your .vimrc:

```
:set dir=dh2:tmp      (for Amiga)
:set dir=~ /tmp       (for Unix)
:set dir=c:\\tmp      (for Win32)
```

This is also very handy when editing files on floppy. Of course you will have to create that "tmp" directory for this to work!

For read-only files, a swap file is not used. Unless the file is big, causing the amount of memory used to be higher than given with **'maxmem'** or **'maxmemtot'**. And when making a change to a read-only file, the swap file is created anyway.

The **'swapfile'** option can be reset to avoid creating a swapfile. And the **:noswapfile** modifier can be used to not create a swapfile for a new buffer.

```
:nos[wapfile] {command}          :nos :noswapfile
Execute {command}. If it contains a command that loads a new
buffer, it will be loaded without creating a swapfile and the
'swapfile' option will be reset. If a buffer already had a
swapfile it is not removed and 'swapfile' is not reset.
```

Detecting an existing swap file

You can find this in the user manual, section [11.3](#) .

Updating the swapfile

The swap file is updated after typing 200 characters or when you have not typed anything for four seconds. This only happens if the buffer was changed, not when you only moved around. The reason why it is not kept up to date all the time is that this would slow down normal work too much. You can change the 200 character count with the **'updatecount'** option. You can set

the time with the **'updatetime'** option. The time is given in milliseconds. After writing to the swap file Vim syncs the file to disk. This takes some time, especially on busy Unix systems. If you don't want this you can set the **'swapsync'** option to an empty string. The risk of losing work becomes bigger though. On some non-Unix systems (MS-Windows, Amiga) the swap file won't be written at all.

If the writing to the swap file is not wanted, it can be switched off by setting the **'updatecount'** option to 0. The same is done when starting Vim with the "-n" option. Writing can be switched back on by setting the **'updatecount'** option to non-zero. Swap files will be created for all buffers when doing this. But when setting **'updatecount'** to zero, the existing swap files will not be removed, it will only affect files that will be opened after this.

If you want to make sure that your changes are in the swap file use this command:

```
                                :pre  :preserve  E313  E314
'coptions' the swap
                                file will not be deleted for this buffer when Vim
                                exits and the buffer is still loaded cpo-& .
```

A Vim swap file can be recognized by the first six characters: "b0VIM ". After that comes the version number, e.g., "3.0".

Links and symbolic links

On Unix it is possible to have two names for the same file. This can be done with hard links and with symbolic links (symlinks).

For hard links Vim does not know the other name of the file. Therefore, the name of the swapfile will be based on the name you used to edit the file. There is no check for editing the same file by the other name too, because Vim cannot find the other swapfile (except for searching all of your harddisk, which would be very slow).

For symbolic links Vim resolves the links to find the name of the actual file. The swap file name is based on that name. Thus it doesn't matter by what name you edit the file, the swap file name will normally be the same. However, there are exceptions:

- When the directory of the actual file is not writable the swapfile is put elsewhere.
- When the symbolic links somehow create a loop you get an **E773** error message and the unmodified file name will be used. You won't be able to save your file normally.

```
=====
2. Recovery                                recovery  E308  E311
```

Basic file recovery is explained in the user manual: [usr_11.txt](#) .

Another way to do recovery is to start Vim and use the `":recover"` command. This is easy when you start Vim to edit a file and you get the "ATTENTION: Found a swap file ..." message. In this case the single command `":recover"` will do the work. You can also give the name of the file or the swap file to the recover command:

```
                :rec  :recover  E305  E306  E307
:rec[over] [file]  Try to recover [file] from the swap file.  If [file]
                  is not given use the file name for the current
                  buffer.  The current contents of the buffer are lost.
                  This command fails if the buffer was modified.
```

```
:rec[over]! [file]  Like ":recover", but any changes in the current
                  buffer are lost.
```

```
                E312  E309  E310  E1364
Vim has some intelligence about what to do if the swap file is corrupt in
some way.  If Vim has doubt about what it found, it will give an error
message and insert lines with "???" in the text.  If you see an error message
while recovering, search in the file for "???" to see what is wrong.  You may
want to cut and paste to get the text you need.
```

The most common remark is "???LINES MISSING". This means that Vim cannot read the text from the original file. This can happen if the system crashed and parts of the original file were not written to disk.

Be sure that the recovery was successful before overwriting the original file or deleting the swap file. It is good practice to write the recovered file elsewhere and run `'diff'` to find out if the changes you want are in the recovered file. Or use `:DiffOrig` .

Once you are sure the recovery is ok delete the swap file. Otherwise, you will continue to get warning messages that the ".swp" file already exists.

ENCRYPTION AND THE SWAP FILE

```
:recover-crypt
```

When the text file is encrypted the swap file is encrypted as well. This makes recovery a bit more complicated. When recovering from a swap file and encryption has been used, you will be asked to enter one or two crypt keys.

If the text file does not exist you will only be asked to enter the crypt key for the swap file.

If the text file does exist, it may be encrypted in a different way than the swap file. You will be asked for the crypt key twice:

```
Need encryption key for "/tmp/tt"
Enter encryption key: *****
"/tmp/tt" [crypted] 23200L, 522129C
Using swap file "/tmp/.tt.swp"
Original file "/tmp/tt"
```


Swap file is encrypted: "/tmp/.tt.swp"
If you entered a new crypt key but did not write the text file,
enter the new crypt key.
If you wrote the text file after changing the crypt key press enter
to use the same key for text file and swap file
Enter encryption key:

You can be in one of these two situations:

1. The encryption key was not changed, or after changing the key the text file was written. You will be prompted for the crypt key twice. The second time you can simply press Enter. That means the same key is used for the text file and the swap file.
2. You entered a new encryption key, but did not save the text file. Vim will then use the new key for the swap file, and the text file will still be encrypted with the old key. At the second prompt enter the new key.

Note that after recovery the key of the swap file will be used for the text file. Thus if you write the text file, you need to use that new key.

Command-line mode [Cmdline-mode](#) [Command-line-mode](#)
[Cmdline](#) [Command-line](#) [mode-cmdline](#) :

Command-line mode is used to enter Ex commands (":"), search patterns ("/" and "?"), and filter commands ("!").

Basic command line editing is explained in chapter 20 of the user manual [usr_20.txt](#) .

- | | |
|----------------------------|------------------------------------|
| 1. Command-line editing | cmdline-editing |
| 2. Command-line completion | cmdline-completion |
| 3. Ex command-lines | cmdline-lines |
| 4. Ex command-line ranges | cmdline-ranges |
| 5. Ex command-line flags | ex-flags |
| 6. Ex special characters | cmdline-special |
| 7. Command-line window | cmdline-window |

1. Command-line editing [cmdline-editing](#)

Normally characters are inserted in front of the cursor position. You can move around in the command-line with the left and right cursor keys. With the [<Insert>](#) key, you can toggle between inserting and overstriking characters.

Note that if your keyboard does not have working cursor keys or any of the other special keys, you can use [":cnoremap"](#) to define another key for them. For example, to define tcsh style editing keys: [tcsh-style](#)

```
:cnoremap <C-A> <Home>
:cnoremap <C-F> <Right>
:cnoremap <C-B> <Left>
:cnoremap <Esc>b <S-Left>
:cnoremap <Esc>f <S-Right>
```

(<> notation <> ; type all this literally)

[cmdline-too-long](#)

When the command line is getting longer than what fits on the screen, only the part that fits will be shown. The cursor can only move in this visible part, thus you cannot edit beyond that.

[cmdline-history](#) [history](#)

The command-lines that you enter are remembered in a history table. You can recall them with the up and down cursor keys. There are actually five history tables:

- one for ':' commands
- one for search strings
- one for expressions
- one for input lines, typed for the [input\(\)](#) function.
- one for debug mode commands

These are completely separate. Each history can only be accessed when entering the same type of line. Use the **'history'** option to set the number of lines that are remembered (default: 50).

Notes:

- When you enter a command-line that is exactly the same as an older one, the old one is removed (to avoid repeated commands moving older commands out of the history).
- Only commands that are typed are remembered. Ones that completely come from mappings are not put in the history.
- All searches are put in the search history, including the ones that come from commands like "*" and "#". But for a mapping, only the last search is remembered (to avoid that long mappings trash the history).

{not available when compiled without the `|+cmdline_hist|` feature}

There is an automatic completion of names on the command-line; see [cmdline-completion](#) .

CTRL-V	Insert next non-digit literally. Up to three digits form the decimal value of a single byte. The non-digit and the three digits are not considered for mapping. This works the same way as in Insert mode (see above, i_CTRL-V). Note: Under MS-Windows CTRL-V is often mapped to paste text. Use CTRL-Q instead then. When modifyOtherKeys is enabled then special Escape sequence is converted back to what it was without modifyOtherKeys , unless the Shift key is also pressed.	c_CTRL-V
CTRL-Q	Same as CTRL-V . But with some terminals it is used for control flow, it doesn't work then.	c_CTRL-Q
CTRL-SHIFT-V CTRL-SHIFT-Q	Works just like CTRL-V , unless modifyOtherKeys is active, then it inserts the Escape sequence for a key with modifiers. In the GUI the key-notation is inserted without simplifying.	c_CTRL-SHIFT-V c_CTRL-SHIFT-Q
<Left>	cursor left. See 'wildmenu' for behavior during wildmenu completion mode.	c_<Left> c_Left
<Right>	cursor right. See 'wildmenu' for behavior during wildmenu completion mode.	c_<Right> c_Right
<S-Left> or <C-Left>	cursor one WORD left	c_<S-Left> c_<C-Left>
<S-Right> or <C-Right>	cursor one WORD right	c_<S-Right> c_<C-Right>
CTRL-B or <Home>	cursor to beginning of command-line	c_CTRL-B c_<Home> c_Home
CTRL-E or <End>	cursor to end of command-line. See 'wildmenu' for behavior during wildmenu completion mode.	c_CTRL-E c_<End> c_End

`<LeftMouse>` `c_<LeftMouse>`
 Move the cursor to the position of the mouse click.

`<MiddleMouse>` `c_<MiddleMouse>`
 Paste the contents of the clipboard (for X11 the primary selection). This is similar to using **CTRL-R** *, but no CR characters are inserted between lines.

CTRL-H `c_<BS>` `c_CTRL-H` `c_BS`
`<BS>` Delete the character in front of the cursor (see `:fixdel` if your `<BS>` key does not do what you want).

`` `c_` `c_Del`
 Delete the character under the cursor (at end of line: character before the cursor) (see `:fixdel` if your `` key does not do what you want).

CTRL-W `c_CTRL-W`
 Delete the `word` before the cursor. This depends on the `'iskeyword'` option.

CTRL-U `c_CTRL-U`
 Remove all characters between the cursor position and the beginning of the line. Previous versions of vim deleted all characters on the line. If that is the preferred behavior, add the following to your `.vimrc`:
`:cnoremap <C-U> <C-E><C-U>`

`<Insert>` `c_<Insert>` `c_Insert`
 Toggle between insert and overstrike.

`{char1} <BS> {char2}` or `c_digraph`
`CTRL-K {char1} {char2}` `c_CTRL-K`
 enter digraph (see `digraphs`). When `{char1}` is a special key, the code for that key is inserted in `<>` form.

CTRL-R `{register}` `c_CTRL-R` `c_<C-R>`
 Insert the contents of a numbered or named register. Between typing **CTRL-R** and the second character `''` will be displayed to indicate that you are expected to enter the name of a register.
 The text is inserted as if you typed it, but mappings and abbreviations are not used. Command-line completion through `'wildchar'` is not triggered though. And characters that end the command line are inserted literally (`<Esc>`, `<CR>`, `<NL>`, `<C-C>`). A `<BS>` or **CTRL-W** could still end the command line though, and remaining characters will then be interpreted in another mode, which might not be what you intended.
 Special registers:
`''` the unnamed register, containing the text of the last delete or yank
`'%` the current file name
`'#'` the alternate file name
`'*'` the clipboard contents (X11: primary selection)
`'+'` the clipboard contents
`'/'` the last search pattern

':' the last command-line
 '-' the last small (less than a line) delete
 '.' the last inserted text

c_CTRL-R_
 '=' the expression register: you are prompted to enter an expression (see **expression**) (doesn't work at the expression prompt; some things such as changing the buffer or current window are not allowed to avoid side effects) When the result is a **List** the items are used as lines. They can have line breaks inside too. When the result is a Float it's automatically converted to a String. **Note** that when you only want to move the cursor and not insert anything, you must make sure the expression evaluates to an empty string. E.g.:
`<C-R><C-R>=setcmdpos(2)[-1]<CR>`

See **registers** about registers.

Implementation detail: When using the **expression** register and invoking `setcmdpos()`, this sets the position before inserting the resulting string. Use **CTRL-R CTRL-R** to set the position afterwards.

CTRL-R CTRL-F	c_CTRL-R_CTRL-F	c_<C-R>_<C-F>
CTRL-R CTRL-P	c_CTRL-R_CTRL-P	c_<C-R>_<C-P>
CTRL-R CTRL-W	c_CTRL-R_CTRL-W	c_<C-R>_<C-W>
CTRL-R CTRL-A	c_CTRL-R_CTRL-A	c_<C-R>_<C-A>
CTRL-R CTRL-L	c_CTRL-R_CTRL-L	c_<C-R>_<C-L>

Insert the object under the cursor:

CTRL-F the Filename under the cursor
CTRL-P the Filename under the cursor, expanded with **'path'** as in **gf**
CTRL-W the Word under the cursor
CTRL-A the WORD under the cursor; see **WORD**
CTRL-L the line under the cursor

When **'incsearch'** is set the cursor position at the end of the currently displayed match is used. With **CTRL-W** the part of the word that was already typed is not inserted again.

	c_CTRL-R_CTRL-R	c_<C-R>_<C-R>
	c_CTRL-R_CTRL-O	c_<C-R>_<C-O>

CTRL-R CTRL-R {register CTRL-F CTRL-P CTRL-W CTRL-A CTRL-L}
CTRL-R CTRL-O {register CTRL-F CTRL-P CTRL-W CTRL-A CTRL-L}

Insert register or object under the cursor. Works like **c_CTRL-R** but inserts the text literally. For example, if register a contains "xy^Hz" (where ^H is a backspace), "**CTRL-R** a" will insert "xz" while "**CTRL-R CTRL-R** a" will insert "xy^Hz".

CTRL-\ e {expr} **c_CTRL-_e**
 Evaluate {expr} and replace the whole command line with the

result. You will be prompted for the expression, type `<Enter>` to finish it. It's most useful in mappings though. See `expression` .

See `c_CTRL-R_` for inserting the result of an expression. Useful functions are `getcmdtype()` , `getcmdline()` and `getcmdpos()` .

The cursor position is unchanged, except when the cursor was at the end of the line, then it stays at the end.

`setcmdpos()` can be used to set the cursor position.

The `sandbox` is used for evaluating the expression to avoid nasty side effects.

Example:

```
:cmap <F7> <C-\>eAppendSome()<CR>
:func AppendSome()
  :let cmd = getcmdline() .. " Some()"
  : " place the cursor on the )
  :call setcmdpos(strlen(cmd))
  :return cmd
:endfunc
```

This doesn't work recursively, thus not when already editing an expression. But it is possible to use in a mapping.

`c_CTRL-Y`

CTRL-Y

When there is a modeless selection, copy the selection into the clipboard. `modeless-selection`

If there is no selection **CTRL-Y** is inserted as a character.

See `'wildmenu'` for behavior during wildmenu completion mode.

CTRL-M or **CTRL-J**

`c_CTRL-M` `c_CTRL-J` `c_<NL>` `c_<CR>` `c_CR`

`<CR>` or `<NL>` start entered command

CTRL-[

`c_CTRL-[` `c_<Esc>` `c_Esc`

`<Esc>`

When typed and 'x' not present in `'cptions'`, quit Command-line mode without executing. In macros or when 'x' present in `'cptions'`, start entered command.

Note: If your `<Esc>` key is hard to hit on your keyboard, train yourself to use `CTRL-[`.

`c_CTRL-C`

CTRL-C

quit command-line without executing

`c_<Up>` `c_Up`

`<Up>`

recall older command-line from history, whose beginning matches the current command-line (see below). See `'wildmenu'` for behavior during wildmenu completion mode.

{not available when compiled without the `+cmdline_hist` feature}

`c_<Down>` `c_Down`

`<Down>`

recall more recent command-line from history, whose beginning matches the current command-line (see below). See `'wildmenu'` for behavior during wildmenu completion mode.

{not available when compiled without the `+cmdline_hist` feature}

`c_<S-Up>` `c_<PageUp>`

<S-Up> or **<PageUp>**
 recall older command-line from history
 {not available when compiled without the `+cmdline_hist` feature}

`c_<S-Down>` `c_<PageDown>`

<S-Down> or **<PageDown>**
 recall more recent command-line from history
 {not available when compiled without the `+cmdline_hist` feature}

CTRL-D command-line completion (see `cmdline-completion`)
'wildchar' option

 command-line completion (see `cmdline-completion`)

CTRL-N command-line completion (see `cmdline-completion`)

CTRL-P command-line completion (see `cmdline-completion`)

CTRL-A command-line completion (see `cmdline-completion`)

CTRL-L command-line completion (see `cmdline-completion`)

`c_CTRL-_`

CTRL-_ a - switch between Hebrew and English keyboard mode, which is private to the command-line and not related to `hkmap`. This is useful when Hebrew text entry is required in the command-line, searches, abbreviations, etc. Applies only if Vim is compiled with the `+rightleft` feature and the **'allowrevins'** option is set. See `rileft.txt` .

 b - switch between Farsi and English keyboard mode, which is private to the command-line and not related to `fkmap`. In Farsi keyboard mode the characters are inserted in reverse insert manner. This is useful when Farsi text entry is required in the command-line, searches, abbreviations, etc. Applies only if Vim is compiled with the `+farsi` feature. See `farsi.txt` .

`c_CTRL-^`

CTRL-^ Toggle the use of language `:lmap` mappings and/or Input Method.
 When typing a pattern for a search command and **'imsearch'** is not -1, VAL is the value of **'imsearch'**, otherwise VAL is the value of **'iminsert'**.
 When language mappings are defined:
 - If VAL is 1 (langmap mappings used) it becomes 0 (no langmap mappings used).
 - If VAL was not 1 it becomes 1, thus langmap mappings are enabled.
 When no language mappings are defined:
 - If VAL is 2 (Input Method is used) it becomes 0 (no input method used)
 - If VAL has another value it becomes 2, thus the Input Method is enabled.
 These language mappings are normally used to type characters that are different from what the keyboard produces. The **'keymap'** option can be used to install a whole number of them.

When entering a command line, langmap mappings are switched off, since you are expected to type a command. After switching it on with **CTRL-^**, the new state is not used again for the next command or Search pattern.

CTRL-] c_CTRL-]
Trigger abbreviation, without inserting a character.

For Emacs-style editing on the command-line see [emacs-keys](#) .

The **<Up>** and **<Down>** keys take the current command-line as a search string. The beginning of the next/previous command-lines are compared with this string. The first line that matches is the new command-line. When typing these two keys repeatedly, the same string is used again. For example, this can be used to find the previous substitute command: Type **":s"** and then **<Up>**. The same could be done by typing **<S-Up>** a number of times until the desired command-line is shown. (Note: the shifted arrow keys do not work on all terminals)

:his[tory] :his :history
Print the history of last entered commands.
{not available when compiled without the [+cmdline_hist](#) feature}

:his[tory] [{name}] [{first}][, [{last}]]
List the contents of history **{name}** which can be:
c[md] or : command-line history
s[earch] or / or ? search string history
e[xpr] or = expression register history
i[nput] or @ input line history
d[ebug] or > debug command history
a[ll] all of the above

If the numbers **{first}** and/or **{last}** are given, the respective range of entries from a history is listed. These numbers can be specified in the following form:

:history-indexing
A positive number represents the absolute index of an entry as it is given in the first column of a **:history** listing. This number remains fixed even if other entries are deleted. (see [E1510](#))

A negative number means the relative position of an entry, counted from the newest entry (which has index -1) backwards.

Examples:

List entries 6 to 12 from the search history:
:history / 6,12

List the penultimate entry from all histories:
:history all -2

List the most recent two entries from all histories:
:history all -2,

`:keep[atterns] {command}` `:keep` `:keepatterns`
Execute `{command}`, without adding anything to the search history

2. Command-line completion

`cmdline-completion`

When editing the command-line, a few commands can be used to complete the word before the cursor. This is available for:

- Command names: At the start of the command-line.
- `++opt` values.
- Tags: Only after the `:tag` command.
- File names: Only after a command that accepts a file name or a setting for an option that can be set to a file name. This is called file name completion.
- Shell command names: After `!:cmd`, `:r !cmd` and `:w !cmd`. `$PATH` is used.
- Options: Only after the `:set` command.
- Mappings: Only after a `:map` or similar command.
- Variable and function names: Only after a `:if`, `:call` or similar command.

The number of help item matches is limited (currently to 300) to avoid a long delay when there are very many matches.

These are the commands that can be used:

CTRL-D

`c_CTRL-D`

List names that match the pattern in front of the cursor. When showing file names, directories are highlighted (see `'highlight'` option). Names where `'suffixes'` matches are moved to the end. The `'wildoptions'` option can be set to `"tagfile"` to list the file of matching tags.

`'wildchar'` option

`c_CTRL-I` `c_wildchar` `c_Tab`

A match is done on the pattern in front of the cursor. The match (if there are several, the first match) is inserted in place of the pattern. (Note: does not work inside a macro, because `<Tab>` or `<Esc>` are mostly used as `'wildchar'`, and these have a special meaning in some macros.) When typed again and there were multiple matches, the next match is inserted. After the last match, the first is used again (wrap around).

The behavior can be changed with the `'wildmode'` option.

`<S-Tab>`

`c_S-Tab`

Like `'wildchar'` or `<Tab>`, but begin with the last match and then go to the previous match. `<S-Tab>` does not work everywhere.

CTRL-N

`c_CTRL-N`

After using `'wildchar'` which got multiple matches, go to next match. Otherwise recall more recent command-line from history.

CTRL-P

`c_CTRL-P`

After using `'wildchar'` which got multiple matches, go to

previous match. Otherwise recall older command-line from history.

CTRL-A c_CTRL-A
All names that match the pattern in front of the cursor are inserted.

CTRL-L c_CTRL-L
A match is done on the pattern in front of the cursor. If there is one match, it is inserted in place of the pattern. If there are multiple matches the longest common part is inserted in place of the pattern. If the result is shorter than the pattern, no completion is done.

/_CTRL-L
When **'incsearch'** is set, entering a search pattern for "/" or "?" and the current match is displayed then **CTRL-L** will add one character from the end of the current match. If **'ignorecase'** and **'smartcase'** are set and the command line has no uppercase characters, the added character is converted to lowercase.

CTRL-G c_CTRL-G /_CTRL-G
When **'incsearch'** is set, entering a search pattern for "/" or "?" and the current match is displayed then **CTRL-G** will move to the next match (does not take `search-offset` into account) Use **CTRL-T** to move to the previous match. Hint: on a regular keyboard T is above G.

CTRL-T c_CTRL-T /_CTRL-T
When **'incsearch'** is set, entering a search pattern for "/" or "?" and the current match is displayed then **CTRL-T** will move to the previous match (does not take `search-offset` into account). Use **CTRL-G** to move to the next match. Hint: on a regular keyboard T is above G.

The **'wildchar'** option defaults to `<Tab>` (CTRL-E when in Vi compatible mode; in a previous version `<Esc>` was used). In the pattern standard `wildcards` are accepted when matching file names.

When repeating **'wildchar'** or **CTRL-N** you cycle through the matches, eventually ending up back to what was typed. If the first match is not what you wanted, you can use `<S-Tab>` or **CTRL-P** to go straight back to what you typed.

The **'wildmenu'** option can be set to show the matches just above the command line.

The **'wildoptions'** option provides additional configuration to use a popup menu for **'wildmenu'**, and to use fuzzy matching.

The **'wildignorecase'** option can be set to ignore case in filenames. For completing other texts (e.g. command names), the **'ignorecase'** option is used instead (fuzzy matching always ignores case, however).

If you like tcsh's autolist completion, you can use this mapping:

```
:cnoremap X <C-L><C-D>
```

(Where X is the command key to use, `<C-L>` is **CTRL-L** and `<C-D>` is **CTRL-D**)
This will find the longest match and then list all matching files.

If you like tcsh's autolist completion, you can use the **'wildmode'** option to emulate it. For example, this mimics autolist=ambiguous:

```
:set wildmode=longest,list
```

This will find the longest match with the first **'wildchar'**, then list all matching files with the next.

complete-script-local-functions

When completing user function names, prepend "s:" to find script-local functions.

suffixes

For file name completion you can use the **'suffixes'** option to set a priority between files with almost the same name. If there are multiple matches, those files with an extension that is in the **'suffixes'** option are ignored. The default is ".bak,~, .o, .h, .info, .swp, .obj", which means that files ending in ".bak", "~", ".o", ".h", ".info", ".swp" and ".obj" are sometimes ignored.

An empty entry, two consecutive commas, match a file name that does not contain a ".", thus has no suffix. This is useful to ignore "prog" and prefer "prog.c".

Examples:

pattern:	files:	match:
test*	test.c test.h test.o	test.c
test*	test.h test.o	test.h and test.o
test*	test.i test.h test.c	test.i and test.c

It is impossible to ignore suffixes with two dots.

If there is more than one matching file (after ignoring the ones matching the **'suffixes'** option) the first file name is inserted. You can see that there is only one match when you type **'wildchar'** twice and the completed match stays the same. You can get to the other matches by entering **'wildchar'**, **CTRL-N** or **CTRL-P**. All files are included, also the ones with extensions matching the **'suffixes'** option.

To completely ignore files with some extension use **'wildignore'**.

To match only files that end at the end of the typed text append a "\$". For example, to match only files that end in ".c":

```
:e *.c$
```

This will not match a file ending in ".cpp". Without the "\$" it does match.

If you would like using **<S-Tab>** for **CTRL-P** in an xterm, put this command in your .cshrc:

```
xmodmap -e "keysym Tab = Tab Find"
```

And this in your .vimrc:

```
:cmap <Esc>[1~ <C-P>
```

complete-set-option

When setting an option using **:set=**, the old value of an option can be obtained by hitting **'wildchar'** just after the '='. For example, typing **'wildchar'** after **:set dir=** will insert the current value of **'dir'**. This

overrules file name completion for the options that take a file name.

When using `:set=` , `:set+=` , or `:set^=` , string options that have pre-defined names or syntax (e.g. `'diffopt'` , `'listchars'`) or are a list of single-character flags (e.g. `'shortmess'`) will also present a list of possible values for completion when using `'wildchar'`.

When using `:set-=` , comma-separated options like `'diffopt'` or `'backupdir'` will show each item separately. Flag list options like `'shortmess'` will show both the entire old value and the individual flags. Otherwise completion will just fill in with the entire old value.

3. Ex command-lines

cmdline-lines

The Ex commands have a few specialties:

`'''` at the start of a line causes the whole line to be ignored. `'''` after a command causes the rest of the line to be ignored. This can be used to add comments. Example:

```
      :set ai          "set 'autoindent' option
```

It is not possible to add a comment to a shell command `":!cmd"` or to the `":map"` command and a few others (mainly commands that expect expressions) that see the `'''` as part of their argument:

```
:argdo
:autocmd
:bufdo
:cexpr (and the like)
:cdo (and the like)
:command
:cscope (and the like)
:debug
:display
:echo (and the like)
:elseif
:execute
:folddoopen
:folddoclosed
:for
:grep (and the like)
:help (and the like)
:if
:let
:make
:map (and the like including :abbrev commands)
:menu (and the like)
:mkspell
:normal
:ownsyntax
:popup
:promptfind (and the like)
:registers
```

```
:return
:sort
:syntax
:tabdo
:tearoff
:vimgrep (and the like)
:while
:windo
```

```
:bar :|bar
```

'|' can be used to separate commands, so you can give multiple commands in one line. If you want to use '|' in an argument, precede it with '\\'.

These commands see the '|' as their argument, and can therefore not be followed by another Vim command:

```
:argdo
:autocmd
:bufdo
:cdo
:cfdo
:command
:cscope
:debug
:eval
:folddoopen
:folddoclosed
:function
:global
:help
:helpfind
:helpgrep
:lcscope
:ldo
:lfdo
:lhelppgrep
:make
:normal
:perl
:perldo
:promptfind
:promptrepl
:pyfile
:python
:registers
:read !
:scscope
:sign
:tabdo
:tcl
:tcldo
:tclfile
:terminal
:vglobal
:windo
```

```

:write !
:[range]!
a user defined command without the "-bar" argument :command

```

Note that this is confusing (inherited from Vi): With ":g" the '|' is included in the command, with ":s" it is not.

To be able to use another command anyway, use the ":execute" command.

Example (append the output of "ls" and jump to the first line):

```
:execute 'r !ls' | '['
```

There is one exception: When the 'b' flag is present in 'coptions', with the ":map" and ":abbr" commands and friends **CTRL-V** needs to be used instead of '\'. You can also use "<Bar>" instead. See also [map_bar](#) .

Examples:

```

:ls | wc          view the output of two commands
:r !ls | wc      insert the same output in the text
:%g/foo/p|>     moves all matching lines one shiftwidth
:%s/foo/bar/|>  moves one line one shiftwidth
:map q 10^V|    map "q" to "10|"
:map q 10\| map \ l    map "q" to "10\" and map "\" to "l"
                    (when 'b' is present in 'coptions')

```

You can also use <NL> to separate commands in the same way as with '|'. To insert a <NL> use **CTRL-V CTRL-J**. "^@" will be shown. Using '|' is the preferred method. But for external commands a <NL> must be used, because a '|' is included in the external command. To avoid the special meaning of <NL> it must be preceded with a backslash. Example:

```
:r !date<NL>-join
```

This reads the current date into the file and joins it with the previous line.

Note that when the command before the '|' generates an error, the following commands will not be executed.

Because of Vi compatibility the following strange commands are supported:

```

:|          print current line (like ":p")
:3|        print line 3 (like ":3p")
:3         goto line 3

```

A colon is allowed between the range and the command name. It is ignored (this is Vi compatible). For example:

```
:1,$:s/pat/string
```

When the character '%' or '#' is used where a file name is expected, they are expanded to the current and alternate file name (see the chapter "editing files" :_% :_#).

Embedded spaces in file names are allowed on the Amiga if one file name is expected as argument. Trailing spaces will be ignored, unless escaped with a backslash or **CTRL-V**. Note that the ":next" command uses spaces to separate file names. Escape the spaces to include them in a file name. Example:

```
:next foo\ bar goes\ to school\
```

starts editing the three files "foo bar", "goes to" and "school ".

When you want to use the special characters '"' or '|' in a command, or want to use '%' or '#' in a file name, precede them with a backslash. The backslash is not required in a range and in the ":substitute" command. See also `= .

The '!' (bang) character after an Ex command makes the command behave in a different way. The '!' should be placed immediately after the command, without any blanks in between. If you insert blanks the '!' will be seen as an argument for the command, which has a different meaning. For example:

```
:w! name      write the current buffer to file "name", overwriting
              any existing file
:w !name      send the current buffer as standard input to command
              "name"
```

4. Ex command-line ranges cmdline-ranges [range] E16

Some Ex commands accept a line range in front of them. This is noted as [range]. It consists of one or more line specifiers, separated with ',' or ';'.

The basics are explained in section 10.3 of the user manual.

In Vim9 script a range needs to be prefixed with a colon to avoid ambiguity with continuation lines. For example, "+" can be used for a range but is also a continuation of an expression:

```
var result = start
+ print
```

If the "+" is a range then it must be prefixed with a colon:

```
var result = start
:+ print
```

When separated with ';' the cursor position will be set to that line before interpreting the next line specifier. This doesn't happen for ','.

Examples:

```
4,/this line/
   from line 4 till match with "this line" after the cursor line.
5;/that line/
   from line 5 till match with "that line" after line 5.
```

The default line specifier for most commands is the cursor position, but the commands ":write" and ":global" have the whole file (1,\$) as default.

If more line specifiers are given than required for the command, the first one(s) will be ignored.

Line numbers may be specified with:

	:range	{address}
{number}	an absolute line number	E1247
.	the current line	::
\$	the last line in the file	::\$

%	equal to 1,\$ (the entire file)	:%
't	position of mark t (lowercase)	:'
'T	position of mark T (uppercase); when the mark is in another file it cannot be used in a range	
/{pattern}[/]	the next line where {pattern} matches	:/
	also see <code>:range-pattern</code> below	
?{pattern}[?]	the previous line where {pattern} matches	?:
	also see <code>:range-pattern</code> below	
\/	the next line where the previously used search pattern matches	
\?	the previous line where the previously used search pattern matches	
\&	the next line where the previously used substitute pattern matches	

:range-offset

Each may be followed (several times) by '+' or '-' and an optional number. This number is added or subtracted from the preceding line number. If the number is omitted, 1 is used. If there is nothing before the '+' or '-' then the current line is used.

:range-closed-fold

When a line number after the comma is in a closed fold it is adjusted to the last line of the fold, thus the whole fold is included.

When a number is added this is done after the adjustment to the last line of the fold. This means these lines are additionally included in the range. For example:

```
:3,4+2print
```

On this text:

```

1 one
2 two
3 three
4 four FOLDED
5 five FOLDED
6 six
7 seven
8 eight
```

Where lines four and five are a closed fold, ends up printing lines 3 to 7. The 7 comes from the "4" in the range, which is adjusted to the end of the closed fold, which is 5, and then the offset 2 is added.

An example for subtracting (which isn't very useful):

```
:2,4-1print
```

On this text:

```

1 one
2 two
3 three FOLDED
4 four FOLDED
5 five FOLDED
6 six FOLDED
7 seven
8 eight
```

Where lines three to six are a closed fold, ends up printing lines 2 to 6. The 6 comes from the "4" in the range, which is adjusted to the end of the

closed fold, which is 6, and then 1 is subtracted, then this is still in the closed fold and the last line of that fold is used, which is 6.

:range-pattern

The "/" and "?" after `{pattern}` are required to separate the pattern from anything that follows.

The "/" and "?" may be preceded with another address. The search starts from there. The difference from using ";" is that the cursor isn't moved.

Examples:

```
/pat1//pat2/    Find line containing "pat2" after line containing
                 "pat1", without moving the cursor.
7;/pat2/        Find line containing "pat2", after line 7, leaving
                 the cursor in line 7.
```

The `{number}` must be between 0 and the number of lines in the file. When using a 0 (zero) this is interpreted as a 1 by most commands. Commands that use it as a count do use it as a zero (`:tag`, `:pop`, etc). Some commands interpret the zero as "before the first line" (`:read`, search pattern, etc).

Examples:

```
.+3             three lines below the cursor
/that/+1        the line below the next line containing "that"
.,$            from current line until end of file
0;/that         the first line containing "that", also matches in the
                 first line.
1;/that         the first line after line 1 containing "that"
```

Some commands allow for a count after the command. This count is used as the number of lines to be used, starting with the line given in the last line specifier (the default is the cursor line). The commands that accept a count are the ones that use a range but do not have a file name argument (because a file name can also be a number). The count cannot be negative.

Examples:

```
:s/x/X/g 5      substitute 'x' by 'X' in the current line and four
                 following lines
:23d 4          delete lines 23, 24, 25 and 26
```

Folds and Range

When folds are active the line numbers are rounded off to include the whole closed fold. See [fold-behavior](#).

Reverse Range

E493

A range should have the lower line number first. If this is not the case, Vim will ask you if it should swap the line numbers.

Backwards range given, OK to swap

This is not done within the global command ":g".

You can use ":silent" before a command to avoid the question, the range will

always be swapped then.

Count and Range

N:

When giving a count before entering ":", this is translated into:

`:. ,.+ (count - 1)`

In words: The "count" lines at and after the cursor. Example: To delete three lines:

`3:d<CR>` is translated into: `:. ,.+2d<CR>`

Visual Mode and Range

v_:

`{Visual}`: Starts a command-line with the Visual selected lines as a range. The code ``:'<,'>`` is used for this range, which makes it possible to select a similar line from the command-line history for repeating a command on different Visually selected lines.

`:*` `:star` `:star-visual-range`

When Visual mode was already ended, a short way to use the Visual area for a range is ``:*``. This requires that "*" does not appear in `'cpo'`, see `cpo-star`. Otherwise you will have to type ``:'<,'>``

For when "*" is in `'cpo'` see `:star-compatible`.

5. Ex command-line flags

ex-flags

These flags are supported by a selection of Ex commands. They print the line that the cursor ends up after executing the command:

<code>l</code>	output like for	<code>:list</code>
<code>#</code>	add line number	
<code>p</code>	output like for	<code>:print</code>

The flags can be combined, thus `"l#"` uses both a line number and `:list` style output.

6. Ex special characters

cmdline-special

Note: These are special characters in the executed command line. If you want to insert special things while typing you can use the **CTRL-R** command. For example, "%" stands for the current file name, while **CTRL-R** % inserts the current file name right away. See `c_CTRL-R`.

Note: If you want to avoid the effects of special characters in a Vim script you may want to use `fnameescape()`. Also see ``=`.

In Ex commands, at places where a file name can be used, the following characters have a special meaning. These can also be used in the expression

```
function expand() .
%      Is replaced with the current file name.           :_%    c_%
#      Is replaced with the alternate file name.         :_#    c_#
      This is remembered for every window.
#n     (where n is a number) is replaced with           :_#0   :_#n
      the file name of buffer n. "#0" is the same as "#".   c_#n
##     Is replaced with all names in the argument list    :_##   c_##
      concatenated, separated by spaces. Each space in a name
      is preceded with a backslash.
#<n   (where n is a number > 0) is replaced with old      :_#<   c_#<
      file name n. See :oldfiles or v:oldfiles to get the
      number.                                             E809
      {only when compiled with the |+eval| and |+viminfo| features}
```

In `Vim9-script` # is used to start a comment, use %% for the alternate file name:

```
%      Is replaced with the current file name.
%%     Is replaced with the alternate file name.         :_%%   c_%%
%%n    (where n is a number) is replaced with           :_%%0  :_%%n
      the file name of buffer n. "%%0" is the same as "%%".  c_%%n
%%%    Is replaced with all names in the argument       :_%%%  c_%%%#
      list concatenated, separated by spaces.
%%<n   (where n is a number > 0) is replaced with old    :_%%<  c_%%<
      file name n.
```

Note that these, except "#<n", give the file name as it was typed. If an absolute path is needed (when using the file name from a different directory), you need to add ":p". See [filename-modifiers](#).

The "#<n" item returns an absolute path, but it will start with "~//" for files below your home directory.

Note that backslashes are inserted before spaces, so that the command will correctly interpret the file name. But this doesn't happen for shell commands. For those you probably have to use quotes (this fails for files that contain a quote and wildcards):

```
:!ls "%"
:r !spell "%"
```

To avoid the special meaning of '%' and '#' insert a backslash before it. Detail: The special meaning is always escaped when there is a backslash before it, no matter how many backslashes.

you type:	result
#	alternate.file
\#	#
\\#	\#

Also see ``=`.

E499 E500

Note: these are typed literally, they are not special keys!

<word>	is replaced with the word under the cursor (like <code>star</code>)	:<word> <word>
<WORD>	is replaced with the WORD under the cursor (see <code>WORD</code>)	:<WORD> <WORD>
		:<cexpr> <cexpr>

<code><cexpr></code>	is replaced with the word under the cursor, including more to form a C expression. E.g., when the cursor is on "arg" of "ptr->arg" then the result is "ptr->arg"; when the cursor is on "]" of "list[idx]" then the result is "list[idx]". This is used for <code>v:beval_text</code> . <code>:<cfile></code> <code><cfile></code>
<code><cfile></code>	is replaced with the path name under the cursor (like what <code>gf</code> uses) <code>:<afile></code> <code><afile></code>
<code><afile></code>	When executing autocommands, is replaced with the file name of the buffer being manipulated, or the file for a read or write. <code>E495</code> <code>:<abuf></code> <code><abuf></code>
<code><abuf></code>	When executing autocommands, is replaced with the currently effective buffer number. It is not set for all events, also see <code>bufnr()</code> . For ":r file" and ":so file" it is the current buffer, the file being read/sourced is not in a buffer. <code>E496</code> <code>:<amatch></code> <code><amatch></code>
<code><amatch></code>	When executing autocommands, is replaced with the match for which this autocommand was executed. <code>E497</code> It differs from <code><afile></code> when the file name isn't used to match with (for FileType, Syntax and SpellFileMissing events). When the match is with a file name, it is expanded to the full path. <code>:<sfile></code> <code><sfile></code>
<code><sfile></code>	When executing a <code>:source</code> command, is replaced with the file name of the sourced file. <code>E498</code> When executing a legacy function, is replaced with the call stack, as with <code><stack></code> (this is for backwards compatibility, using <code><stack></code> or <code><script></code> is preferred). In Vim9 script using <code><sfile></code> in a function gives error <code>E1245</code> . Note that filename-modifiers are useless when <code><sfile></code> is not used inside a script. <code>:<stack></code> <code><stack></code>
<code><stack></code>	is replaced with the call stack, using "function {function-name}[{lnum}]" for a function line and "script {file-name}[{lnum}]" for a script line, and ".." in between items. E.g.: "function {function-name1}[{lnum}]..{function-name2}[{lnum}]" If there is no call stack you get error <code>E489</code> . <code>:<script></code> <code><script></code>
<code><script></code>	When executing a <code>:source</code> command, is replaced with the file name of the sourced file. When executing a function, is replaced with the file name of the script where it is defined. If the file name cannot be determined you get error <code>E1274</code> . <code>:<slnum></code> <code><slnum></code>
<code><slnum></code>	When executing a <code>:source</code> command, is replaced with the line number. <code>E842</code> When executing a function it's the line number relative to the start of the function.

`<sflnum>` When executing a script, is replaced with the line number. It differs from `<slnum>` in that `<sflnum>` is replaced with the script line number in any situation. E961

`<client>` is replaced with the `{clinetid}` of the last received message in `server2client()`

filename-modifiers

`:_%: ::8 ::p ::. ::~ ::h ::t ::r ::e ::s ::gs ::S`
`:%:8 %:p %:. %:~ %:h %:t %:r %:e %:s %:gs %:S`

The file name modifiers can be used after "%", "#", "#n", "<cfil>", "<sfil>", "<afil>" or "<abuf>". They are also used with the `fnamemodify()` function.

These modifiers can be given, in this order:

- `:p` Make file name a full path. Must be the first modifier. Also changes "~/ (and "~user/" for Unix and VMS) to the path for the home directory. If the name is a directory a path separator is added at the end. For a file name that does not exist and does not have an absolute path the result is unpredictable. On MS-Windows an 8.3 filename is expanded to the long name.
- `:8` Converts the path to 8.3 short format (currently only on MS-Windows). Will act on as much of a path that is an existing path.
- `:~` Reduce file name to be relative to the home directory, if possible. File name is unmodified if it is not below the home directory.
- `::` Reduce file name to be relative to current directory, if possible. File name is unmodified if it is not below the current directory.
For maximum shortness, use "::~".
- `:h` Head of the file name (the last component and any separators removed). Cannot be used with `:e`, `:r` or `:t`.
Can be repeated to remove several components at the end. When the file name ends in a path separator, only the path separator is removed. Thus `:p:h` on a directory name results on the directory name itself (without trailing slash).
When the file name is an absolute path (starts with "/" for Unix; "x:\" for Win32; "drive:" for Amiga), that part is not removed. When there is no head (path is relative to current directory) the result is empty.
- `:t` Tail of the file name (last component of the name). Must precede any `:r` or `:e`.
- `:r` Root of the file name (the last extension removed). When there is only an extension (file name that starts with '.', e.g., ".vimrc"), it is not removed. Can be repeated to remove several extensions (last one first).
- `:e` Extension of the file name. Only makes sense when used alone. When there is no extension the result is empty. When there is only an extension (file name that starts with '.'), the result is empty. Can be repeated to include more extensions. If there are not enough extensions (but at least one) as much as possible are included.

```

:s?pat?sub?
    Substitute the first occurrence of "pat" with "sub". This works like the :s command. "pat" is a regular expression. Any character can be used for '?', but it must not occur in "pat" or "sub".
    After this, the previous modifiers can be used again. For example ":p", to make a full path after the substitution.

:gs?pat?sub?
    Substitute all occurrences of "pat" with "sub". Otherwise this works like ":s".

:S
    Escape special characters for use with a shell command (see shellescape() ). Must be the last one. Examples:
        :!dir <file>:S
        :call system('chmod +w -- ' . expand('%:S'))

```

Examples, when the file name is "src/version.c", current dir "/home/mool/vim":

```

:p                /home/mool/vim/src/version.c
:p:              src/version.c
:p:~             ~/vim/src/version.c
:h              src
:p:h            /home/mool/vim/src
:p:h:h         /home/mool/vim
:t              version.c
:p:t           version.c
:r            src/version
:p:r         /home/mool/vim/src/version
:t:r       version
:e                c
:s?version?main?      src/main.c
:s?version?main?:p  /home/mool/vim/src/main.c
:p:gs?/?\?          \home\mool\vim\src\version.c

```

Examples, when the file name is "src/version.c.gz":

```

:p                /home/mool/vim/src/version.c.gz
:e                gz
:e:e             c.gz
:e:e:e          c.gz
:e:e:r          c
:r              src/version.c
:r:e            c
:r:r           src/version
:r:r:r        src/version

```

extension-removal :_%<

If a "<" is appended to "%", "#", "#n" or "CTRL-V p" the extension of the file name is removed (everything after and including the last '.' in the file name). This is included for backwards compatibility with version 3.0, the ":r" form is preferred. Examples:

```

%                current file name
%<              current file name without extension
#                alternate file name for current window
#<              idem, without extension

```

```

#31          alternate file number 31
#31<        idem, without extension
<cwor>      word under the cursor
<cWORD>     WORD under the cursor (see |WORD|)
<cfile>     path name under the cursor
<cfile><    idem, without extension

```

Note: Where a file name is expected wildcards expansion is done. On Unix the shell is used for this, unless it can be done internally (for speed). Unless in `restricted-mode`, backticks work also, like in

```
:n `echo *.c`
```

But expansion is only done if there are any wildcards before expanding the '%', '#', etc.. This avoids expanding wildcards inside a file name. If you want to expand the result of `<cfile>`, add a wildcard character to it.

Examples: (alternate file name is "?readme?")

command	expands to
:e #	:e ?readme?
:e `ls #`	:e {files matching "?readme?"}
:e #.*	:e {files matching "?readme?.*"}
:cd <cfile>	:cd {file name under cursor}
:cd <cfile>*	:cd {file name under cursor plus "*" and then expanded}

Also see ``=`.

When the expanded argument contains a "!" and it is used for a shell command ("!:cmd", ":r !cmd" or ":w !cmd"), the "!" is escaped with a backslash to avoid it being expanded into a previously used command. When the `'shell'` option contains "sh", this is done twice, to avoid the shell trying to expand the "!".

filename-backslash

For filesystems that use a backslash as directory separator (MS-Windows), it's a bit difficult to recognize a backslash that is used to escape the special meaning of the next character. The general rule is: If the backslash is followed by a normal file name character, it does not have a special meaning. Therefore "\file\foo" is a valid file name, you don't have to type the backslash twice.

An exception is the '\$' sign. It is a valid character in a file name. But to avoid a file name like "\$home" to be interpreted as an environment variable, it needs to be preceded by a backslash. Therefore you need to use "/\ \$home" for the file "\$home" in the root directory. A few examples:

FILE NAME	INTERPRETED AS
\$home	expanded to value of environment var \$home
\ \$home	file "\$home" in current directory
/\ \$home	file "\$home" in root directory
\\ \$home	file "\\ ", followed by expanded \$home

Also see ``=`.

7. Command-line window

`cmdline-window` `cmdwin`
`command-line-window`

In the command-line window the command line can be edited just like editing

text in any window. It is a special kind of window, because you cannot leave it in a normal way.

OPEN

`c_CTRL-F` `q:` `q/` `q?`

There are two ways to open the command-line window:

1. From Command-line mode, use the key specified with the `'cedit'` option. The default is `CTRL-F` when `'compatible'` is not set.
2. From Normal mode, use the `"q:"`, `"q/"` or `"q?"` command. This starts editing an Ex command-line (`"q:"`) or search string (`"q/"` or `"q?"`). **Note** that this is not possible while recording is in progress (the `"q"` stops recording then).

When the window opens it is filled with the command-line history. The last line contains the command as typed so far. The left column will show a character that indicates the type of command-line being edited, see `cmdwin-char` .

Vim will be in Normal mode when the editor is opened, except when `'insertmode'` is set.

`E1292`

Once a command-line window is open it is not possible to open another one.

The height of the window is specified with `'cmdwinheight'` (or smaller if there is no room). The window is always full width and is positioned just above the command-line.

EDIT

You can now use commands to move around and edit the text in the window. Both in Normal mode and Insert mode.

It is possible to use `:"`, `/"` and other commands that use the command-line, but it's not possible to open another command-line window then. There is no nesting.

`E11` `E1188`

The command-line window is not a normal window. It is not possible to move to another window or edit another buffer. All commands that would do this are disabled in the command-line window. Of course it is possible to execute any command that you entered in the command-line window. Other text edits are discarded when closing the window.

CLOSE

`E199`

There are several ways to leave the command-line window:

- `<CR>` Execute the command-line under the cursor. Works both in Insert and in Normal mode.
- `CTRL-C` Continue in Command-line mode. The command-line under the cursor is used as the command-line. Works both in Insert and in Normal mode. There is no redraw, thus the window will

```

                remain visible.
:quit          Discard the command line and go back to Normal mode.
                ":close", CTRL-W c, ":exit", ":xit" and CTRL-\ CTRL-N also
                work.
:qall         Quit Vim, unless there are changes in some buffer.
:qall!        Quit Vim, discarding changes to any buffer.

```

Once the command-line window is closed the old window sizes are restored. The executed command applies to the window and buffer where the command-line was started from. This works as if the command-line window was not there, except that there will be an extra screen redraw.

The buffer used for the command-line window is deleted. Any changes to lines other than the one that is executed with `<CR>` are lost.

If you would like to execute the command under the cursor and then have the command-line window open again, you may find this mapping useful:

```
:autocmd CmdwinEnter * map <buffer> <F5> <CR>q:
```

VARIOUS

The command-line window cannot be used:

- when there already is a command-line window (no nesting)
- for entering an encryption key or when using `inputsecret()`

Some options are set when the command-line window is opened:

```

'filetype'    "vim", when editing an Ex command-line; this starts Vim syntax
               highlighting if it was enabled
'rightleft'   off
'modifiable' on
'buftype'     "nofile"
'swapfile'    off

```

It is allowed to write the buffer contents to a file. This is an easy way to save the command-line history and read it back later.

If the `'wildchar'` option is set to `<Tab>`, and the command-line window is used for an Ex command, then two mappings will be added to use `<Tab>` for completion in the command-line window, like this:

```
:inoremap <buffer> <Tab> <C-X><C-V>
:nnoremap <buffer> <Tab> a<C-X><C-V>
```

Note that hitting `<Tab>` in Normal mode will do completion on the next character. That way it works at the end of the line.

If you don't want these mappings, disable them with:

```
au CmdwinEnter [:>] iunmap <Tab>
au CmdwinEnter [:>] nunmap <Tab>
```

You could put these lines in your vimrc file.

While in the command-line window you cannot use the mouse to put the cursor in another window, or drag statuslines of other windows. You can drag the statusline of the command-line window itself and the statusline above it. Thus you can resize the command-line window, but not others.

The `getcldwintype()` function returns the type of the command-line being edited as described in `cmdwin-char` .

AUTOCOMMANDS

Two autocommand events are used: `CmdwinEnter` and `CmdwinLeave` . You can use the `Cmdwin` events to do settings specifically for the command-line window. Be careful not to cause side effects!

Example:

```
:au CmdwinEnter : let b:cpt_save = &cpt | set cpt=.  
:au CmdwinLeave : let &cpt = b:cpt_save
```

This sets **'complete'** to use completion in the current window for `i_CTRL-N` .

Another example:

```
:au CmdwinEnter [/?] startinsert
```

This will make Vim start in Insert mode in the command-line window.

`cmdwin-char`

The character used for the pattern indicates the type of command-line:

```
:      normal Ex command  
>     debug mode command debug-mode  
/     forward search string  
?     backward search string  
=     expression for "= expr-register"  
@     string for input()  
-     text for :insert or :append
```


Options

options

- 1. Setting options [set-option](#)
- 2. Automatically setting options [auto-setting](#)
- 3. Options summary [option-summary](#)

For an overview of options see quickref.txt [option-list](#) .

Vim has a number of internal variables and switches which can be set to achieve special effects. These options come in three forms:

- boolean can only be on or off [boolean](#) [toggle](#)
- number has a numeric value
- string has a string value

1. Setting options

[set-option](#) E764

[:se](#) [:set](#)

`:se[t][!]` Show all options that differ from their default value. When [!] is present every option is on a separate line.

`:se[t][!] all` Show all but terminal options. When [!] is present every option is on a separate line.

`:se[t] termcap` Show all terminal options. [Note](#) that in the GUI the key codes are not shown, because they are generated internally and can't be changed. Changing the terminal codes in the GUI is not useful either... The options have the form t_AB, see [terminal-options](#) .

`:se[t]! termcap` Idem, but don't use multiple columns.

E518 E519

`:se[t] {option}?` Show value of {option}.

`:se[t] {option}` Toggle option: set, switch it on.
Number option: show value.
String option: show value.

`:se[t] no{option}` Toggle option: Reset, switch it off.

[:set-!](#) [:set-inv](#)

`:se[t] {option}!` or `:se[t] inv{option}` Toggle option: Invert value.

[:set-default](#) [:set-&](#) [:set-&vi](#) [:set-&vim](#)

```
:se[t] {option}&          Reset option to its default value.  May depend on the
                          current value of 'compatible'.
:se[t] {option}&vi       Reset option to its Vi default value.
:se[t] {option}&vim      Reset option to its Vim default value.
```

```
:se[t] all&            Set all options to their default value.  The values of
                          these options are not changed:
                          all terminal options, starting with t_
                          'columns'
                          'cryptmethod'
                          'encoding'
                          'key'
                          'lines'
                          'term'
                          'ttymouse'
                          'ttytype'
Warning: This may have a lot of side effects.
```

[:set-args](#) [:set=](#) [E487](#) [E521](#)

```
:se[t] {option}={value}          or
:se[t] {option}:{value}

Set string or number option to {value}.
For numeric options the value can be given in decimal,
hex (preceded with 0x) or octal (preceded with '0').
The old value can be inserted by typing 'wildchar' (by
default this is a <Tab> or CTRL-E if 'compatible' is
set). Many string options with fixed syntax and names
also support completing known values.  See
cmdline-completion and complete-set-option .
White space between {option} and '=' is allowed and
will be ignored.  White space between '=' and {value}
is not allowed.
See option-backslash for using white space and
backslashes in {value}.
```

```
:se[t] {option}+={value}          :set+=

Add the {value} to a number option, or append the
{value} to a string option.  When the option is a
comma-separated list, a comma is added, unless the
value was empty.
If the option is a list of flags, superfluous flags
are removed.  When adding a flag that was already
present the option value doesn't change.
Also see :set-args above.
```

```
:se[t] {option}^={value}          :set^=

Multiply the {value} to a number option, or prepend
the {value} to a string option.  When the option is a
comma-separated list, a comma is added, unless the
value was empty.
Also see :set-args above.
```

```
:se[t] {option}-={value}          :set-=
```

Subtract the `{value}` from a number option, or remove the `{value}` from a string option, if it is there. If the `{value}` is not found in a string option, there is no error or warning. When the option is a comma separated list, a comma is deleted, unless the option becomes empty. When the option is a list of flags, `{value}` must be exactly as they appear in the option. Remove flags one by one to avoid problems. The individual values from a comma separated list or list of flags can be inserted by typing `'wildchar'`. See `complete-set-option`. Also see `:set-args` above.

The `{option}` arguments to `":set"` may be repeated. For example:

```
:set ai nosi sw=3 ts=3
```

If you make an error in one of the arguments, an error message will be given and the following arguments will be ignored.

`:set-verbose`

When `'verbose'` is non-zero, displaying an option value will also tell where it was last set. Example:

```
:verbose set shiftwidth cindent?
shiftwidth=4
  Last set from modeline line 1
cindent
  Last set from /usr/local/share/vim/vim60/ftplugin/c.vim line 30
```

This is only done when specific option values are requested, not for `":verbose set all"` or `":verbose set"` without an argument.

When the option was set by hand there is no "Last set" message.

When the option was set while executing a function, user command or autocommand, the script in which it was defined is reported.

Note that an option may also have been set as a side effect of setting `'compatible'`.

A few special texts:

```
Last set from modeline line 1
  Option was set in a modeline .
Last set from --cmd argument
  Option was set with command line argument --cmd or +.
Last set from -c argument
  Option was set with command line argument -c , +, -S or
  -q .
Last set from environment variable
  Option was set from an environment variable, $VIMINIT,
  $GVIMINIT or $EXINIT.
Last set from error handler
  Option was cleared when evaluating it resulted in an error.
```

{not available when compiled without the `|+eval|` feature}

`:set-termcap` E522

For `{option}` the form `"t_xx"` may be used to set a terminal option. This will override the value from the termcap. You can then use it in a mapping. If the `"xx"` part contains special characters, use the `<t_xx>` form:

```
:set <t_#4>=^[0t
```

This can also be used to translate a special code for a normal key. For example, if Alt-b produces <Esc>b, use this:

```
:set <M-b>=^[b
```

(the ^[is a real <Esc> here, use **CTRL-V** <Esc> to enter it)

The advantage over a mapping is that it works in all situations.

You can define any key codes, e.g.:

```
:set t_xy=^[foo;
```

There is no warning for using a name that isn't recognized. You can map these codes as you like:

```
:map <t_xy> something
```

E846

When a key code is not set, it's like it does not exist. Trying to get its value will result in an error:

```
:set t_kb=
```

```
:set t_kb
```

```
E846: Key code not set: t_kb
```

The t_xx options cannot be set from a `modeline` or in the `sandbox`, for security reasons.

The listing from `":set"` looks different from Vi. Long string options are put at the end of the list. The number of options is quite large. The output of `"set all"` probably does not fit on the screen, causing Vim to give the `more-prompt`.

option-backslash

To include white space in a string option value it has to be preceded with a backslash. To include a backslash you have to use two. Effectively this means that the number of backslashes in an option value is halved (rounded down).

In options `'path'`, `'cdpath'`, and `'tags'`, spaces have to be preceded with three backslashes instead for compatibility with version 3.0 where the options can be separated by either commas or spaces.

Comma-separated options like `'backupdir'` and `'tags'` will also require commas to be escaped with two backslashes, whereas this is not needed for non-comma-separated ones like `'makeprg'`.

When setting options using `:let` and `literal-string`, you need to use one fewer layer of backslash.

A few examples:

```
:set makeprg=make\ file           results in "make file"
:let &makeprg='make file'         (same as above)
:set makeprg=make\\ file         results in "make\ file"
:set tags=tags\ /usr/tags        results in "tags" and "/usr/tags"
:set tags=tags\\ file           results in "tags file"
:let &tags='tags\ file'         (same as above)

:set makeprg=make,file           results in "make,file"
:set makeprg=make\\,file        results in "make\,file"
:set tags=tags,file             results in "tags" and "file"
:set tags=tags\\,file          results in "tags,file"
:let &tags='tags\,file'         (same as above)
```


The "|" character separates a ":set" command from a following command. To include the "|" in the option value, use "\\|" instead. This example sets the **'titlestring'** option to "hi|there":

```
:set titlestring=hi\\|there
```

This sets the **'titlestring'** option to "hi" and **'iconstring'** to "there":

```
:set titlestring=hi|set iconstring=there
```

Similarly, in legacy script the double quote character starts a comment. To include the '"' in the option value, use "\\\"" instead. This example sets the **'titlestring'** option to 'hi "there"':

```
:set titlestring=hi\\ \"there\\"
```

In **Vim9** script it's simpler, comments start with a '#' character, and only when preceded by white space. A backslash is needed less often:

```
vim9script
set titlestring=hi\\ "there"
set titlestring=hi#there#
set titlestring=hi\\ \\#there#
```

For Win32 backslashes in file names are mostly not removed. More precise: For options that expect a file name (those where environment variables are expanded) a backslash before a normal file name character is not removed. But a backslash before a special character (space, backslash, comma, etc.) is used like explained above.

There is one special situation, when the value starts with "\\\"":

```
:set dir=\\machine\\path           results in "\\machine\\path"
:set dir=\\\\\\machine\\\\path       results in "\\machine\\path"
:set dir=\\path\\file               results in "\\path\\file" (wrong!)
```

For the first one the start is kept, but for the second one the backslashes are halved. This makes sure it works both when you expect backslashes to be halved and when you expect the backslashes to be kept. The third gives a result which is probably not what you want. Avoid it.

```
add-option-flags  remove-option-flags
E539  E550  E551  E552
```

Some options are a list of flags. When you want to add a flag to such an option, without changing the existing ones, you can do it like this:

```
:set guioptions+=a
```

Remove a flag from an option like this:

```
:set guioptions-=a
```

This removes the 'a' flag from **'guioptions'**.

Note that you should add or remove one flag at a time. If **'guioptions'** has the value "ab", using "set guioptions-=ba" won't work, because the string "ba" doesn't appear.

```
:set_env  expand-env  expand-environment-var
```

Environment variables in specific string options will be expanded. If the environment variable exists the '\$' and the following environment variable name is replaced with its value. If it does not exist the '\$' and the name are not modified. Any non-id character (not a letter, digit or '_') may follow the environment variable name. That character and what follows is appended to the value of the environment variable. Examples:

```
:set term=$TERM.new
:set path=/usr/$INCLUDE,$HOME/include,.
```

When adding or removing a string from an option with `":set opt-=val"` or `":set opt+=val"` the expansion is done before the adding or removing.

Handling of local options

local-options

Note: The following also applies to `global-local` options.

Some of the options only apply to a window or buffer. Each window or buffer has its own copy of this option, thus each can have its own value. This allows you to set `'list'` in one window but not in another. And set `'shiftwidth'` to 3 in one buffer and 4 in another.

The following explains what happens to these local options in specific situations. You don't really need to know all of this, since Vim mostly uses the option values you would expect. Unfortunately, doing what the user expects is a bit complicated...

When splitting a window, the local options are copied to the new window. Thus right after the split the contents of the two windows look the same.

When editing a new buffer, its local option values must be initialized. Since the local options of the current buffer might be specifically for that buffer, these are not used. Instead, for each buffer-local option there also is a global value, which is used for new buffers. With `":set"` both the local and global value is changed. With `"setlocal"` only the local value is changed, thus this value is not used when editing a new buffer.

When editing a buffer that has been edited before, the options from the window that was last closed are used again. If this buffer has been edited in this window, the values from back then are used. Otherwise the values from the last closed window where the buffer was edited last are used.

It's possible to set a local window option specifically for a type of buffer. When you edit another buffer in the same window, you don't want to keep using these local window options. Therefore Vim keeps a global value of the local window options, which is used when editing another buffer. Each window has its own copy of these values. Thus these are local to the window, but global to all buffers in the window. With this you can do:

```
:e one
:set list
:e two
```

Now the `'list'` option will also be set in "two", since with the `":set list"` command you have also set the global value.

```
:set nolist
:e one
:setlocal list
:e two
```

Now the `'list'` option is not set, because `":set nolist"` resets the global value, `":setlocal list"` only changes the local value and `":e two"` gets the global value. **Note** that if you do this next:

```
:e one
```

You will get back the `'list'` value as it was the last time you edited "one". The options local to a window are remembered for each buffer. This also

happens when the buffer is not loaded, but they are lost when the buffer is wiped out `:bwipe` .

Special local window options

`local-noglobal`

The following local window options won't be copied over when new windows are created, thus they behave slightly differently:

Option	Reason
<code>'previewwindow'</code>	there can only be a single one
<code>'scroll'</code>	specific to existing window
<code>'winfixbuf'</code>	specific to existing window
<code>'winfixheight'</code>	specific to existing window
<code>'winfixwidth'</code>	specific to existing window

Special local buffer options

The following local buffer options won't be copied over when new buffers are created, thus they behave slightly differently:

Option	Reason
<code>'filetype'</code>	explicitly set by autocommands
<code>'syntax'</code>	explicitly set by autocommands
<code>'bufhidden'</code>	denote <code>special-buffers</code>
<code>'buftype'</code>	denote <code>special-buffers</code>
<code>'readonly'</code>	will be detected automatically
<code>'modified'</code>	will be detected automatically

	<code>:setl</code> <code>:setlocal</code>
<code>:setl[ocal][!] ...</code>	Like <code>":set"</code> but set only the value local to the current buffer or window. Not all options have a local value. If the option does not have a local value the global value is set. With the "all" argument: display local values for all local options. Without argument: Display local values for all local options which are different from the default. When displaying a specific local option, show the local value. For a global/local boolean option, when the global value is being used, "--" is displayed before the option name. For a global option the global value is shown (but that might change in the future).
<code>:se[t] {option}<</code>	Set the effective value of <code>{option}</code> to its global value. For string <code>global-local</code> options, the local value is removed, so that the global value will be used. For all other options, the global value is copied to the local value.
<code>:setl[ocal] {option}<</code>	Set the effective value of <code>{option}</code> to its global value. For number and boolean <code>global-local</code> options, the

local value is removed, so that the global value will be used.
 For all other options, including string `global-local` options, the global value is copied to the local value.

Note that the behaviour for `global-local` options is slightly different between string and number-based options.

`:setg[lobal][!]` ... `:setg` `:setglobal`
 Like `":set"` but set only the global value for a local option without changing the local value.
 When displaying an option, the global value is shown.
 With the "all" argument: display global values for all local options.
 Without argument: display global values for all local options which are different from the default.

For buffer-local and window-local options:

Command	global value	local value
<code>:set option=value</code>	set	set
<code>:setlocal option=value</code>	-	set
<code>:setglobal option=value</code>	set	-
<code>:set option?</code>	-	display
<code>:setlocal option?</code>	-	display
<code>:setglobal option?</code>	display	-

Global options with a local value

`global-local`

Options are global when you mostly use one value for all buffers and windows. For some global options it's useful to sometimes have a different local value. You can set the local value with `":setlocal"`. That buffer or window will then use the local value, while other buffers and windows continue using the global value.

For example, you have two windows, both on C source code. They use the global `'makeprg'` option. If you do this in one of the two windows:

```
:set makeprg=gmake
```

then the other window will switch to the same value. There is no need to set the `'makeprg'` option in the other C source window too.

However, if you start editing a Perl file in a new window, you want to use another `'makeprg'` for it, without changing the value used for the C source files. You use this command:

```
:setlocal makeprg=perlmake
```

You can switch back to using the global value by making the local value empty:

```
:setlocal makeprg=
```

This only works for a string option. For a number or boolean option you need to use the "<" flag, like this:

```
:setlocal autoread<
```

Note that for non-boolean and non-number options using "<" copies the global value to the local value, it doesn't switch back to using the global value (that matters when the global value changes later). You can also use:

```
:set path<
```

This will make the local value of **'path'** empty, so that the global value is used. Thus it does the same as:

```
:setlocal path=
```

Note: In the future more global options can be made **global-local** . Using **":setlocal"** on a global option might work differently then.

Some options (**'completefunc'**, **'imactivatefunc'**, **'imstatusfunc'**, **'omnifunc'**, **'operatorfunc'**, **'quickfixtextfunc'**, **'tagfunc'** and **'thesaurusfunc'**) are set to a function name or a function reference or a lambda function. When using a lambda it will be converted to the name, e.g. "**<lambda>123**". Examples:

```
set opfunc=MyOpFunc
set opfunc=function('MyOpFunc')
set opfunc=funcref('MyOpFunc')
set opfunc={a\ ->\ MyOpFunc(a)}
```

Set to a script-local function:

```
set opfunc=s:MyLocalFunc
set opfunc=<SID>MyLocalFunc
```

In **Vim9** script the **"s:"** and **"<SID>"** can be omitted if the function exists in the script:

```
set opfunc=MyLocalFunc
```

Set using a funcref variable:

```
let Fn = function('MyTagFunc')
let &tagfunc = Fn
```

Set using a lambda expression:

```
let &tagfunc = {t -> MyTagFunc(t)}
```

Set using a variable with lambda expression:

```
let L = {a, b, c -> MyTagFunc(a, b , c)}
let &tagfunc = L
```

In **Vim9** script, in a compiled function, you can use a lambda, but a closure does not work, because the function will be called without the context of where it was defined.

Setting the filetype

```
:setf[iletype] [FALLBACK] {filetype} :setf :setfiletype
```

Set the **'filetype'** option to **{filetype}**, but only if not done yet in a sequence of (nested) autocommands. This is short for:

```
:if !did_filetype()
: setlocal filetype={filetype}
:endif
```

This command is used in a **filetype.vim** file to avoid setting the **'filetype'** option twice, causing different settings and syntax files to be loaded.

When the optional **FALLBACK** argument is present, a

later `:setfiletype` command will override the `'filetype'`. This is to be used for filetype detections that are just a guess. `did_filetype()` will return false after this command.

```
option-window  optwin
:set-browse    :browse-set  :opt  :options
:bro[wse] se[t]
:opt[ions]
```

Open a window for viewing and setting all options. Options are grouped by function. Offers short help for each option. Hit `<CR>` on the short help to open a help window with more help for the option. Modify the value of the option and hit `<CR>` on the "set" line to set the new value. For window and buffer specific options, the last accessed window is used to set the option value in, unless this is a help window, in which case the window below help window is used (skipping the option-window). {not available when compiled without the `+eval` feature}

`$HOME`

Using `~` is like using `"$HOME"`, but it is only recognized at the start of an option and after a space or comma.

On Unix systems `~user` can be used too. It is replaced by the home directory of user "user". Example:

```
:set path=~mool/include,/usr/include,.
```

On Unix systems the form `"${HOME}"` can be used too. The name between `{ }` can contain non-id characters then. **Note** that if you want to use this for the "gf" command, you need to add the `'{'` and `'}'` characters to `'isfname'`.

NOTE: expanding environment variables and `~/` is only done with the `":set"` command, not when assigning a value to an option with `":let"`.

`$HOME-windows`

On MS-Windows, if `$HOME` is not defined as an environment variable, then at runtime Vim will set it to the expansion of `$HOMEDRIVE$HOMEPATH`. If `$HOMEDRIVE` is not set then `$USERPROFILE` is used.

This expanded value is not exported to the environment, this matters when running an external command:

```
:echo system('set | findstr ^HOME=')
```

and

```
:echo luaeval('os.getenv("HOME")')
```

should echo nothing (an empty string) despite `exists('$HOME')` being true. When setting `$HOME` to a non-empty string it will be exported to the subprocesses.

Note the maximum length of an expanded option is limited. How much depends on the system, mostly it is something like 256 or 1024 characters.

```
:fix[del]
```

```
:fix :fixdel
```

```
Set the value of 't_kD':  
't_kb' is 't_kD' becomes  
CTRL-? CTRL-H  
not CTRL-? CTRL-?
```

(CTRL-? is 0o177 octal, 0x7f hex)

If your delete key terminal code is wrong, but the code for backspace is alright, you can put this in your .vimrc:

```
:fixdel
```

This works no matter what the actual code for backspace is.

If the backspace key terminal code is wrong you can use this:

```
:if &term == "termname"  
: set t_kb=^V<BS>  
: fixdel  
:endif
```

Where "^V" is **CTRL-V** and "<BS>" is the backspace key (don't type four characters!). Replace "termname" with your terminal name.

If your <Delete> key sends a strange key sequence (not CTRL-? or **CTRL-H**) you cannot use ":fixdel". Then use:

```
:if &term == "termname"  
: set t_kD=^V<Delete>  
:endif
```

Where "^V" is **CTRL-V** and "<Delete>" is the delete key (don't type eight characters!). Replace "termname" with your terminal name.

Linux-backspace

Note about Linux: By default the backspace key produces CTRL-?, which is wrong. You can fix it by putting this line in your rc.local:

```
echo "keycode 14 = BackSpace" | loadkeys
```

NetBSD-backspace

Note about NetBSD: If your backspace doesn't produce the right code, try this:

```
xmodmap -e "keycode 22 = BackSpace"
```

If this works, add this in your .Xmodmap file:

```
keysym 22 = BackSpace
```

You need to restart for this to take effect.

2. Automatically setting options

auto-setting

Besides changing options with the ":set" command, there are three alternatives to set options automatically for one or more files:

1. When starting Vim initializations are read from various places. See [initialization](#) . Most of them are performed for all editing sessions, and some of them depend on the directory where Vim is started. You can create an initialization file with `:mkvimrc` , `:mkview` and `:mksession` .
2. If you start editing a new file, the automatic commands are executed. This can be used to set options for files matching a particular pattern and many other things. See [autocommand](#) .
3. If you start editing a new file, and the `'modeline'` option is on, a number of lines at the beginning and end of the file are checked for modelines. This is explained here.

modeline vim: vi: ex: E520

There are two forms of modelines. The first form:

```
[text{white}]{vi: vim: ex:}[white]{options}
```

<pre>[text{white}] {vi: vim: ex:} [white] {options}</pre>	<p>empty or any text followed by at least one blank character (<Space> or <Tab>); "ex:" always requires at least one blank character</p> <p>the string "vi:", "vim:" or "ex:"</p> <p>optional white space</p> <p>a list of option settings, separated with white space or ':', where each part between ':' is the argument for a ":set" command (can be empty)</p>
---	--

Examples:

```
vi:noai:sw=3 ts=6
vim: tw=77
```

The second form (this is compatible with some versions of Vi):

```
[text{white}]{vi: vim:|Vim: ex:}[white]se[t] {options}:[text]
```

<pre>[text{white}] {vi: vim: Vim: ex:} [white] se[t] {options} : [text]</pre>	<p>empty or any text followed by at least one blank character (<Space> or <Tab>); "ex:" always requires at least one blank character</p> <p>the string "vi:", "vim:", "Vim:" or "ex:"</p> <p>optional white space</p> <p>the string "set " or "se " (note the space); When "Vim" is used it must be "set".</p> <p>a list of options, separated with white space, which is the argument for a ":set" command</p> <p>a colon</p> <p>any text or empty</p>
---	---

Examples:

```
/* vim: set ai tw=75: */
/* Vim: set ai tw=75: */
```

The white space before `{vi:|vim:|Vim:|ex:}` is required. This minimizes the chance that a normal word like "lex:" is caught. There is one exception: "vi:" and "vim:" can also be at the start of the line (for compatibility with version 3.0). Using "ex:" at the start of the line will be ignored (this could be short for "example:").

If the modeline is disabled within a modeline, subsequent modelines will be ignored. This is to allow turning off modeline on a per-file basis. This is useful when a line looks like a modeline but isn't. For example, it would be good to start a YAML file containing strings like "vim:" with

```
# vim: nomodeline
```

so as to avoid modeline misdetection. Following options on the same line after modeline deactivation, if any, are still evaluated (but you would normally not have any).

modeline-local

The options are set like with ":setlocal": The new value only applies to the buffer and window that contain the file. Although it's possible to set global options from a modeline, this is unusual. If you have two windows open and the files in it set the same global option to a different value, the result depends on which one was opened last.

When editing a file that was already loaded, only the window-local options from the modeline are used. Thus if you manually changed a buffer-local option after opening the file, it won't be changed if you edit the same buffer in another window. But window-local options will be set.

modeline-version

If the modeline is only to be used for some versions of Vim, the version number can be specified where "vim:" or "Vim:" is used:

```
vim{vers}:      version {vers} or later
vim<{vers}:     version before {vers}
vim={vers}:     version {vers}
vim>{vers}:     version after {vers}
```

{vers} is 700 for Vim 7.0 (hundred times the major version plus minor).

For example, to use a modeline only for Vim 7.0:

```
/* vim700: set foldmethod=marker */
```

To use a modeline for Vim after version 7.2:

```
/* vim>702: set cole=2: */
```

There can be no blanks between "vim" and the ":".

The number of lines that are checked can be set with the **'modelines'** option. If **'modeline'** is off or **'modelines'** is 0 no lines are checked.

Note that for the first form all of the rest of the line is used, thus a line like:

```
/* vi:ts=4: */
```

will give an error message for the trailing "*/". This line is OK:

```
/* vi:set ts=4: */
```

If an error is detected the rest of the line is skipped.

If you want to include a ':' in a set command precede it with a '\\'. The backslash in front of the ':' will be removed. Example:

```
/* vi:set fillchars=stl\:^,vert\:\|: */
```

This sets the **'fillchars'** option to "stl:^,vert:\|". Only a single backslash before the ':' is removed. Thus to include "\:" you have to specify "\\:".

E992

No other commands than "set" are supported, for security reasons (somebody might create a Trojan horse text file with modelines). And not all options can be set. For some options a flag is set, so that when the value is used the `sandbox` is effective. Some options can only be set from the modeline when `'modelineexpr'` is set (the default is off).

Still, there is always a small risk that a modeline causes trouble. E.g., when some joker sets `'textwidth'` to 5 all your lines are wrapped unexpectedly. So disable modelines before editing untrusted text. The mail ftplugin does this, for example.

Hint: If you would like to do something else than setting an option, you could define an autocommand that checks the file for a specific string. For example:

```
au BufReadPost * if getline(1) =~ "VAR" | call SetVar() | endif
```

And define a function `SetVar()` that does something with the line containing "VAR".

3. Options summary

[option-summary](#)

In the list below all the options are mentioned with their full name and with an abbreviation if there is one. Both forms may be used.

In this document when a boolean option is "set" that means that `":set option"` is entered. When an option is "reset", `":set nooption"` is used.

For some options there are two default values: The "Vim default", which is used when `'compatible'` is not set, and the "Vi default", which is used when `'compatible'` is set.

Most options are the same in all windows and buffers. There are a few that are specific to how the text is presented in a window. These can be set to a different value in each window. For example the `'list'` option can be set in one window and reset in another for the same text, giving both types of view at the same time. There are a few options that are specific to a certain file. These can have a different value for each file or buffer. For example the `'textwidth'` option can be 78 for a normal text file and 0 for a C program.

global	one option for all buffers and windows
local to window	each window has its own copy of this option
local to buffer	each buffer has its own copy of this option

When creating a new window the option values from the currently active window are used as a default value for the window-specific options. For the buffer-specific options this depends on the 's' and 'S' flags in the `'coptions'` option. If 's' is included (which is the default) the values for buffer options are copied from the currently active buffer when a buffer is first entered. If 'S' is present the options are copied each time the buffer is entered, this is almost like having global options. If 's' and 'S' are not present, the options are copied from the currently active buffer when the buffer is created.

Hidden options

hidden-options

Not all options are supported in all versions. This depends on the supported features and sometimes on the system. A remark about this is in curly braces below. When an option is not supported it may still be set without getting an error, this is called a hidden option. You can't get the value of a hidden option though, it is not stored.

To test if option "foo" can be used with ":set" use something like this:

```
if exists('&foo')
```

This also returns true for a hidden option. To test if option "foo" is really supported use something like this:

```
if exists('+foo')
```

E355

A jump table for the options with a short description can be found at [Q_op](#) .

'aleph' **'al'** ['aleph'](#) ['al'](#) [aleph](#) [Aleph](#)
number (default 128 for MS-Windows, 224 otherwise)
global
{only available when compiled with the [+rightleft](#) feature}

The ASCII code for the first letter of the Hebrew alphabet. The routine that maps the keyboard in Hebrew mode, both in Insert mode (when `hkmap` is set) and on the command-line (when hitting `CTRL_`) outputs the Hebrew characters in the range `[aleph..aleph+26]`. `aleph=128` applies to PC code, and `aleph=224` applies to ISO 8859-8. See [rileft.txt](#) .

'allowrevins' **'ari'** ['allowrevins'](#) ['ari'](#) ['noallowrevins'](#) ['noari'](#)
boolean (default off)
global
{only available when compiled with the [+rightleft](#) feature}

Allow `CTRL_` in Insert and Command-line mode. This is default off, to avoid that users that accidentally type `CTRL_` instead of `SHIFT_` get into reverse Insert mode, and don't know how to get out. See **'revins'**.

NOTE: This option is reset when **'compatible'** is set.

'altkeymap' **'akm'** ['altkeymap'](#) ['akm'](#) ['noaltkeymap'](#) ['noakm'](#)
boolean (default off)
global
{only available when compiled with the [+farsi](#) feature}

This option was for using Farsi, which has been removed. See [farsi.txt](#) .

'ambiwidth' **'ambw'** ['ambiwidth'](#) ['ambw'](#)
string (default: "single")
global

Only effective when **'encoding'** is "utf-8" or another Unicode encoding. Tells Vim what to do with characters with East Asian Width Class Ambiguous (such as Euro, Registered Sign, Copyright Sign, Greek

letters, Cyrillic letters).

There are currently two possible values:

"single": Use the same width as characters in US-ASCII. This is expected by most users.

"double": Use twice the width of ASCII characters.

E834 E835

The value "double" cannot be used if **'listchars'** or **'fillchars'** contains a character that would be double width. These errors may also be given when calling `setcellwidths()`.

The values are overruled for characters specified with `setcellwidths()` .

There are a number of CJK fonts for which the width of glyphs for those characters are solely based on how many octets they take in legacy/traditional CJK encodings. In those encodings, Euro, Registered sign, Greek/Cyrillic letters are represented by two octets, therefore those fonts have "wide" glyphs for them. This is also true of some line drawing characters used to make tables in text file. Therefore, when a CJK font is used for GUI Vim or Vim is running inside a terminal (emulators) that uses a CJK font (or Vim is run inside an xterm invoked with "-cjkwidth" option.), this option should be set to "double" to match the width perceived by Vim with the width of glyphs in the font. Perhaps it also has to be set to "double" under CJK MS-Windows when the system locale is set to one of CJK locales. See Unicode Standard Annex #11 (<http://www.unicode.org/reports/tr11>).

Vim may set this option automatically at startup time when Vim is compiled with the `+termresponse` feature and if `t_u7` is set to the escape sequence to request cursor position report. The response can be found in `v:termu7resp` .

'antialias' **'anti'** **'noantialias'** **'noanti'**
boolean (default: off)
global
{only available when compiled with GUI enabled on macOS}

This option only has an effect in the GUI version of Vim on macOS v10.2 or later. When on, Vim will use smooth ("antialiased") fonts, which can be easier to read at certain sizes on certain displays. Setting this option can sometimes cause problems if **'guifont'** is set to its default (empty string).

NOTE: This option is reset when **'compatible'** is set.

'arabic' **'arab'** **'noarabic'** **'noarab'**
boolean (default off)
local to window
{only available when compiled with the `+arabic` feature}

This option can be set to start editing Arabic text.

Setting this option will:

- Set the **'rightleft'** option, unless **'termbidi'** is set.

- Set the **'arabicshape'** option, unless **'termbidi'** is set.
 - Set the **'keymap'** option to "arabic"; in Insert mode **CTRL-^** toggles between typing English and Arabic key mapping.
 - Set the **'delcombine'** option
- Note that **'encoding'** must be "utf-8" for working with Arabic text.

Resetting this option will:

- Reset the **'rightleft'** option.
- Disable the use of **'keymap'** (without changing its value).

Note that **'arabicshape'** and **'delcombine'** are not reset (it is a global option).

NOTE: This option is reset when **'compatible'** is set.

Also see [arabic.txt](#) .

'arabicshape' **'arshape'** boolean (default on)
 global
 {only available when compiled with the **+arabic** feature}

When on and **'termbidi'** is off, the required visual character corrections that need to take place for displaying the Arabic language take effect. Shaping, in essence, gets enabled; the term is a broad one which encompasses:

- a) the changing/morphing of characters based on their location within a word (initial, medial, final and stand-alone).
- b) the enabling of the ability to compose characters
- c) the enabling of the required combining of some characters

When disabled the display shows each character's true stand-alone form.

Arabic is a complex language which requires other settings, for further details see [arabic.txt](#) .

NOTE: This option is set when **'compatible'** is set.

'autochdir' **'acd'** boolean (default off)
 global
 {only available when compiled with it, use `exists("+autochdir")` to check}

When on, Vim will change the current working directory whenever you open a file, switch buffers, delete a buffer or open/close a window. It will change to the directory containing the file which was opened or selected. When a buffer has no name it also has no directory, thus the current directory won't change when navigating to it.

Note: When this option is on some plugins may not work.

'autoindent' **'ai'** boolean (default off)
 local to buffer

Copy indent from current line when starting a new line (typing **<CR>** in Insert mode or when using the "o" or "O" command). If you do not type anything on the new line except **<BS>** or **CTRL-D** and then type **<Esc>**, **CTRL-O** or **<CR>**, the indent is deleted again. Moving the cursor to another line has the same effect, unless the 'I' flag is included

in **'coptions'**.

When **autoindent** is on, formatting (with the "gq" command or when you reach **'textwidth'** in Insert mode) uses the indentation of the first line.

When **'smartindent'** or **'cindent'** is on the indent is changed in a different way.

The **'autoindent'** option is reset when the **'paste'** option is set and restored when **'paste'** is reset.

'autoread' **'ar'** **'autoread'** **'ar'** **'noautoread'** **'noar'**
boolean (default off)
global or local to buffer **global-local**

When a file has been detected to have been changed outside of Vim and it has not been changed inside of Vim, automatically read it again. When the file has been deleted this is not done, so you have the text from before it was deleted. When it appears again then it is read.

timestamp

If this option has a local value, use this command to switch back to using the global value:

```
:set autoread<
```

'autoshelldir' **'asd'** **'autoshelldir'** **'asd'** **'noautoshelldir'** **'noasd'**
boolean (default off)
global

When on, Vim will change the current working directory whenever you change the directory of the shell running in a terminal window. You need proper setting-up, so whenever the shell's pwd changes an OSC 7 escape sequence will be emitted. For example, on Linux, you can source /etc/profile.d/vte.sh in your shell profile if you use bash or zsh. For bash this should work (put it in a bash init file):

```
if [[ -n "$VIM_TERMINAL" ]]; then
  PROMPT_COMMAND='_vim_sync_PWD'
  function _vim_sync_PWD() {
    printf '\033]7;file://%s\033\\' "$PWD"
  }
fi
```

Or, in a zsh init file:

```
if [[ -n "$VIM_TERMINAL" ]]; then
  autoload -Uz add-zsh-hook
  add-zsh-hook -Uz chpwd _vim_sync_PWD
  function _vim_sync_PWD() {
    printf '\033]7;file://%s\033\\' "$PWD"
  }
fi
```

In a fish init file:

```
if test -n "$VIM_TERMINAL"
  function _vim_sync_PWD --on-variable=PWD
    printf '\033]7;file://%s\033\\' "$PWD"
  end
end
```

You can find an alternative method at [terminal-autoshelldir](#) .
When the parsing of the OSC sequence fails you get `E1179` .

'autowrite' **'aw'** `boolean` (default off)
`global`

Write the contents of the file, if it has been modified, on each `:next`, `:rewind`, `:last`, `:first`, `:previous`, `:stop`, `:suspend`, `:tag`, `:!`, `:make`, `CTRL-]` and `CTRL-^` command; and when a `:buffer`, `CTRL-O`, `CTRL-I`, `{A-Z0-9}`, or `{A-Z0-9}` command takes one to another file.

A buffer is not written if it becomes hidden, e.g. when `'bufhidden'` is set to "hide" and `:next` is used.

Note that for some commands the `'autowrite'` option is not used, see `'autowriteall'` for that.

Some buffers will not be written, specifically when `'buftype'` is "nowrite", "nofile", "terminal" or "prompt".

USE WITH CARE: If you make temporary changes to a buffer that you don't want to be saved this option may cause it to be saved anyway. Renaming the buffer with `:"file {name}"` may help avoid this.

'autowriteall' **'awa'** `boolean` (default off)
`global`

Like `'autowrite'`, but also used for commands `:"edit"`, `:"enew"`, `:"quit"`, `:"qall"`, `:"exit"`, `:"xit"`, `:"recover"` and closing the Vim window.

Setting this option also implies that Vim behaves like `'autowrite'` has been set.

'background' **'bg'** `string` (default "dark" or "light", see below)
`global`

When set to "dark", Vim will try to use colors that look good on a dark background. When set to "light", Vim will try to use colors that look good on a light background. Any other value is illegal.

Vim tries to set the default value according to the terminal used.

This will not always be correct.

Setting this option does not change the background color, it tells Vim what the background color looks like. For changing the background color, see `:hi-normal` .

When `'background'` is changed Vim will adjust the default color groups for the new value. But the colors used for syntax highlighting will not change.

When a color scheme is loaded (the `"g:colors_name"` variable is set) changing `'background'` will cause the color scheme to be reloaded. If the color scheme adjusts to the value of `'background'` this will work. However, if the color scheme sets `'background'` itself the effect may be undone. First delete the `"g:colors_name"` variable when needed.

When setting `'background'` to the default value with:

```
:set background&
```

Vim will guess the value. In the GUI this should work correctly, in other cases Vim might not be able to guess the right value.

insert.

When the value is empty, Vi compatible backspacing is used, none of the ways mentioned for the items above are possible.

For backwards compatibility with version 5.4 and earlier:

value	effect
0	same as <code>":set backspace="</code> (Vi compatible)
1	same as <code>":set backspace=indent,eol"</code>
2	same as <code>":set backspace=indent,eol,start"</code>
3	same as <code>":set backspace=indent,eol,nostop"</code>

See `:fixdel` if your `<BS>` or `` key does not do what you want.

NOTE: This option is set to "" when `'compatible'` is set.

	<code>'backup'</code>	<code>'bk'</code>	<code>'nbackup'</code>	<code>'nobk'</code>
<code>'backup'</code> <code>'bk'</code>	boolean (default off)			
	global			

Make a backup before overwriting a file. Leave it around after the file has been successfully written. If you do not want to keep the backup file, but you do want a backup while the file is being written, reset this option and set the `'writebackup'` option (this is the default). If you do not want a backup file at all reset both options (use this if your file system is almost full). See the [backup-table](#) for more explanations.

When the `'backupskip'` pattern matches, a backup is not made anyway. When `'patchmode'` is set, the backup may be renamed to become the oldest version of a file.

NOTE: This option is reset when `'compatible'` is set.

	<code>'backupcopy'</code>	<code>'bkc'</code>
<code>'backupcopy'</code> <code>'bkc'</code>	string (Vi default for Unix: "yes", otherwise: "auto")	
	global or local to buffer global-local	

When writing a file and a backup is made, this option tells how it's done. This is a comma-separated list of words.

The main values are:

- "yes" make a copy of the file and overwrite the original one
- "no" rename the file and write a new one
- "auto" one of the previous, what works best

Extra values that can be combined with the ones above are:

- "breaksymlink" always break symlinks when writing
- "breakhardlink" always break hardlinks when writing

Making a copy and overwriting the original file:

- Takes extra time to copy the file.
- + When the file has special attributes, is a (hard/symbolic) link or has a resource fork, all this is preserved.
- When the file is a link the backup will have the name of the link, not of the real file.

Renaming the file and writing a new one:

- + It's fast.

- Sometimes not all attributes of the file can be copied to the new file.
- When the file is a link the new file will not be a link.

The "auto" value is the middle way: When Vim sees that renaming the file is possible without side effects (the attributes can be passed on and the file is not a link) that is used. When problems are expected, a copy will be made.

The "breaksymlink" and "breakhardlink" values can be used in combination with any of "yes", "no" and "auto". When included, they force Vim to always break either symbolic or hard links by doing exactly what the "no" option does, renaming the original file to become the backup and writing a new file in its place. This can be useful for example in source trees where all the files are symbolic or hard links and any changes should stay in the local source tree, not be propagated back to the original source.

crontab

One situation where "no" and "auto" will cause problems: A program that opens a file, invokes Vim to edit that file, and then tests if the open file was changed (through the file descriptor) will check the backup file instead of the newly created file. "crontab -e" is an example.

When a copy is made, the original file is truncated and then filled with the new text. This means that protection bits, owner and symbolic links of the original file are unmodified. The backup file, however, is a new file, owned by the user who edited the file. The group of the backup is set to the group of the original file. If this fails, the protection bits for the group are made the same as for others.

When the file is renamed, this is the other way around: The backup has the same attributes of the original file, and the newly written file is owned by the current user. When the file was a (hard/symbolic) link, the new file will not! That's why the "auto" value doesn't rename when the file is a link. The owner and group of the newly written file will be set to the same ones as the original file, but the system may refuse to do this. In that case the "auto" value will again not rename the file.

NOTE: This option is set to the Vi default value when 'compatible' is set and to the Vim default value when 'compatible' is reset.

```
'backupdir' 'bdir'          string (default for Amiga: ".,t:",
                               for Win32: ".,$TEMP,c:/tmp,c:/temp"
                               for Unix: ".,~/tmp,~/")
                               global
```

List of directories for the backup file, separated with commas.

- The backup file will be created in the first directory in the list where this is possible. The directory must exist, Vim will not create it for you.
- Empty means that no backup file will be created ('patchmode' is

- impossible!). Writing may fail because of this.
- A directory "." means to put the backup file in the same directory as the edited file.
 - A directory starting with "./" (or ".\" for MS-Windows) means to put the backup file relative to where the edited file is. The leading "." is replaced with the path name of the edited file. ("." inside a directory name has no special meaning).
 - Spaces after the comma are ignored, other spaces are considered part of the directory name. To have a space at the start of a directory name, precede it with a backslash.
 - To include a comma in a directory name precede it with a backslash.
 - A directory name may end in an '/'.
 - For Unix and Win32, if a directory ends in two path separators "//", the backup file name will be built from the complete path to the file with all path separators changed to percent '%' signs. This will ensure file name uniqueness in the backup directory. On Win32, it is also possible to end with "\\". However, when a separating comma is following, you must use "//", since "\\" will include the comma in the file name. Therefore it is recommended to use '//', instead of '\\'.
 - Environment variables are expanded `:set_env` .
 - Careful with '\' characters, type one before a space, type two to get one in the option (see `option-backslash`), for example:
`:set bdir=c:\\tmp,\\ dir\\,with\\,commas,\\ \\ dir\\ with\\ spaces`
 - For backwards compatibility with `Vim version 3.0` a '>' at the start of the option is removed.

See also `'backup'` and `'writebackup'` options.

If you want to hide your backup files on Unix, consider this value:

```
:set backupdir=./.backup,~/.backup,../tmp
```

You must create a ".backup" directory in each directory and in your home directory for this to work properly.

The use of `:set+=` and `:set-=` is preferred when adding or removing directories from the list. This avoids problems when a future version uses another default.

This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

```
'backupext' 'bex'          string (default "~", for VMS: "_")
                           global
                           'backupext' 'bex' E589
```

String which is appended to a file name to make the name of the backup file. The default is quite unusual, because this avoids accidentally overwriting existing files with a backup file. You might prefer using ".bak", but make sure that you don't have files with ".bak" that you want to keep.

Only normal file name characters can be used; `"/*?[]|<>"` are illegal.

If you like to keep a lot of backups, you could use a `BufWritePre` autocommand to change `'backupext'` just before writing the file to include a timestamp.

```
:au BufWritePre * let &bex = '-' .. strftime("%Y%b%d%X") .. '~'
```

Use `'backupdir'` to put the backup in a different directory.

```
'backupskip' 'bsk'
```

'backupskip' 'bsk' string (default: "\$TMPDIR/*,\$TMP/*,\$TEMP/*"
 Unix: "/tmp/*,\$TMPDIR/*,\$TMP/*,\$TEMP/*"
 Mac: "/private/tmp/*,\$TMPDIR/*,\$TMP/*,\$TEMP/*")
 global

A list of file patterns. When one of the patterns matches with the name of the file which is written, no backup file is created. Both the specified file name and the full path name of the file are used. The pattern is used like with `:autocmd`, see [autocmd-patterns](#). Watch out for special characters, see [option-backslash](#). When \$TMPDIR, \$TMP or \$TEMP is not defined, it is not used for the default value. "/tmp/*" is only used for Unix.

WARNING: Not having a backup file means that when Vim fails to write your buffer correctly and then, for whatever reason, Vim exits, you lose both the original file and what you were writing. Only disable backups if you don't care about losing the file.

Note that environment variables are not expanded. If you want to use \$HOME you must expand it explicitly, e.g.:

```
:let &backupskip = escape(expand('$HOME'), '\') .. '/tmp/*'
```

Note that the default also makes sure that "crontab -e" works (when a backup would be made by renaming the original file crontab won't see the newly created file). Also see **'backupcopy'** and [crontab](#).

'balloondelay' 'bdlay' number (default: 600)
 global
 {only available when compiled with the `+balloon_eval` feature}
 Delay in milliseconds before a balloon may pop up. See [balloon-eval](#).

'ballooneval' 'beval' boolean (default off)
 global
 {only available when compiled with the `+balloon_eval` feature}
 Switch on the [balloon-eval](#) functionality for the GUI.

'balloonevalterm' 'bevalterm' boolean (default off)
 global
 {only available when compiled with the `+balloon_eval_term` feature}
 Switch on the [balloon-eval](#) functionality for the terminal.

'balloonexpr' 'bexpr' string (default "")
 global or local to buffer [global-local](#)
 {only available when compiled with the `+balloon_eval` feature}
 Expression for text to show in evaluation balloon. It is only used when **'ballooneval'** or **'balloonevalterm'** is on. These variables can be

used:

```
v:beval_bufnr    number of the buffer in which balloon is going to show
v:beval_winnr    number of the window
v:beval_winid    ID of the window
v:beval_lnum     line number
v:beval_col      column number (byte index)
v:beval_text     word under or after the mouse pointer
```

Instead of showing a balloon, which is limited to plain text, consider using a popup window, see [popup_beval_example](#) . A popup window can use highlighting and show a border.

The evaluation of the expression must not have side effects!

Example:

```
function MyBalloonExpr()
  return 'Cursor is at line ' .. v:beval_lnum ..
        \ ', column ' .. v:beval_col ..
        \ ' of file ' .. bufname(v:beval_bufnr) ..
        \ ' on word "' .. v:beval_text .. '"'
endfunction
set bexpr=MyBalloonExpr()
set ballooneval balloonevalterm
```

Also see [balloon_show\(\)](#) , it can be used if the content of the balloon is to be fetched asynchronously. In that case evaluating **'balloonexpr'** should result in an empty string. If you get a balloon with only "" you probably didn't return anything from your function.

NOTE: The balloon is displayed only if the cursor is on a text character. If the result of evaluating **'balloonexpr'** is not empty, Vim does not try to send a message to an external debugger (Netbeans or Sun Workshop).

If the expression starts with s: or <SID> , then it is replaced with the script ID ([local-function](#)). Example:

```
set bexpr=s:MyBalloonExpr()
set bexpr=<SID>SomeBalloonExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

The expression will be evaluated in the [sandbox](#) when set from a modeline, see [sandbox-option](#) . This option cannot be set in a modeline when **'modelineexpr'** is off.

It is not allowed to change text or jump to another window while evaluating **'balloonexpr'**, see [textlock](#) .

To check whether line breaks in the balloon text work use this check:

```
if has("balloon_multiline")
```

When they are supported "\n" characters will start a new line. If the expression evaluates to a [List](#) this is equal to using each List item as a string and putting "\n" in between them.

NOTE: This option is set to "" when **'compatible'** is set.

'belloff' **'bo'** string (default "")
 global

Specifies for which events the bell will not be rung. It is a comma separated list of items. For each item that is present, the bell will be silenced. This is most useful to specify specific events in insert mode to be silenced.

You can also make it flash by using **'visualbell'**.

item	meaning when present
all	All events.
backspace	When hitting <BS> or and deleting results in an error.
cursor	Fail to move around using the cursor keys or <PageUp>/<PageDown> in Insert-mode .
complete	Error occurred when using i_CTRL-X_CTRL-K or i_CTRL-X_CTRL-T .
copy	Cannot copy char from insert mode using i_CTRL-Y or i_CTRL-E .
ctrlg	Unknown Char after <C-G> in Insert mode.
error	Other Error occurred (e.g. try to join last line) (mostly used in Normal-mode or Cmdline-mode).
esc	hitting <Esc> in Normal-mode .
ex	In Visual-mode , hitting Q results in an error.
hangul	Ignored.
insertmode	Pressing <Esc> in 'insertmode' .
lang	Calling the beep module for Lua/Mzscheme/TCL.
mess	No output available for g< .
showmatch	Error occurred for 'showmatch' function.
operator	Empty region error cpo-E .
register	Unknown register after <C-R> in Insert-mode .
shell	Bell from shell output :! .
spell	Error happened on spell suggest.
term	Bell from :terminal output.
wildmode	More matches in cmdline-completion available (depends on the 'wildmode' setting).

This is most useful to fine tune when in Insert mode the bell should be rung. For Normal mode and Ex commands, the bell is often rung to indicate that an error occurred. It can be silenced by adding the "error" keyword.

'binary' **'bin'** boolean (default off)
'nobinary' **'nobin'** local to buffer

This option should be set before editing a binary file. You can also use the **-b** Vim argument. When this option is switched on a few options will be changed (also when it already was on):

- 'textwidth'** will be set to 0
- 'wrapmargin'** will be set to 0
- 'modeline'** will be off
- 'expandtab'** will be off

Also, **'fileformat'** and **'fileformats'** options will not be used, the

file is read and written like **'fileformat'** was "unix" (a single `<NL>` separates lines).

The **'fileencoding'** and **'fileencodings'** options will not be used, the file is read without conversion.

NOTE: When you start editing a(nother) file while the **'bin'** option is on, settings from autocommands may change the settings again (e.g., **'textwidth'**), causing trouble when editing. You might want to set **'bin'** again when the file has been loaded.

The previous values of these options are remembered and restored when **'bin'** is switched from on to off. Each buffer has its own set of saved option values.

To edit a file with **'binary'** set you can use the `++bin` argument. This avoids you have to do `":set bin"`, which would have effect for all files you edit.

When writing a file the `<EOL>` for the last line is only written if there was one in the original file (normally Vim appends an `<EOL>` to the last line if there is none; this would make the file longer). See the **'endofline'** option.

	'bioskey'	'biosk'	'bioskey'	'biosk'	'nobioskey'	'nobiosk'
			boolean (default on)			
			global			
			{only for MS-DOS}			

This was for MS-DOS and is no longer supported.

'bomb'			boolean (default off)	'bomb'	'nobomb'
			local to buffer		

When writing a file and the following conditions are met, a BOM (Byte Order Mark) is prepended to the file:

- this option is on
- the **'binary'** option is off
- **'fileencoding'** is "utf-8", "ucs-2", "ucs-4" or one of the little/big endian variants.

Some applications use the BOM to recognize the encoding of the file. Often used for UCS-2 files on MS-Windows. For other applications it causes trouble, for example: `"cat file1 file2"` makes the BOM of file2 appear halfway through the resulting file. Gcc doesn't accept a BOM. When Vim reads a file and **'fileencodings'** starts with "ucs-bom", a check for the presence of the BOM is done and **'bomb'** set accordingly. Unless **'binary'** is set, it is removed from the first line, so that you don't see it when editing. When you don't change the options, the BOM will be restored when writing the file.

	'breakat'	'brk'	'breakat'	'brk'
'breakat'	'brk'		string (default " ^I!@*~+;:,./?")	
			global	
			{not available when compiled without the <code>+linebreak</code> feature}	

This option lets you choose which characters might cause a line break if **'linebreak'** is on. Only works for ASCII and also for 8-bit characters when **'encoding'** is an 8-bit encoding.

'breakindent'	'bri'	'nobreakindent'	'nobri'
----------------------	--------------	------------------------	----------------

'breakindent' **'bri'** boolean (default off)
local to window
{not available when compiled without the `+linebreak`
feature}

Every wrapped line will continue visually indented (same amount of space as the beginning of that line), thus preserving horizontal blocks of text.

NOTE: This option is reset when **'compatible'** is set.

'breakindentopt' **'briopt'** string (default empty)
local to window
{not available when compiled without the `+linebreak`
feature}

Settings for **'breakindent'**. It can consist of the following optional items and must be separated by a comma:

`min:{n}` Minimum text width that will be kept after applying **'breakindent'**, even if the resulting text should normally be narrower. This prevents text indented almost to the right window border occupying lot of vertical space when broken.
(default: 20)

`shift:{n}` After applying **'breakindent'**, the wrapped line's beginning will be shifted by the given number of characters. It permits dynamic French paragraph indentation (negative) or emphasizing the line continuation (positive).
(default: 0)

`sbr` Display the **'showbreak'** value before applying the additional indent.
(default: off)

`list:{n}` Adds an additional indent for lines that match a numbered or bulleted list (using the **'formatlistpat'** setting).

`list:-1` Uses the length of a match with **'formatlistpat'** for indentation.
(default: 0)

`column:{n}` Indent at column `{n}`. Will overrule the other sub-options. **Note:** an additional indent may be added for the **'showbreak'** setting.
(default: off)

'browseidir' **'bsdir'** string (default: "last")
global
{only for Motif, GTK, Mac and Win32 GUI}

Which directory to use for the file browser:

`last` Use same directory as with last file browser, where a file was opened or saved.

`buffer` Use the directory of the related buffer.

`current` Use the current directory.

`{path}` Use the specified directory

'bufhidden' **'bh'**

'bufhidden' 'bh' string (default: "")
 local to buffer [local-noglobal](#)

This option specifies what happens when a buffer is no longer displayed in a window:

<code><empty></code>	follow the global 'hidden' option
hide	hide the buffer (don't unload it), even if 'hidden' is not set
unload	unload the buffer, even if 'hidden' is set; the <code>:hide</code> command will also unload the buffer
delete	delete the buffer from the buffer list, even if 'hidden' is set; the <code>:hide</code> command will also delete the buffer, making it behave like <code>:bdelete</code>
wipe	wipe the buffer from the buffer list, even if 'hidden' is set; the <code>:hide</code> command will also wipe out the buffer, making it behave like <code>:bwipeout</code>

CAREFUL: when "unload", "delete" or "wipe" is used changes in a buffer are lost without a warning. Also, these values may break autocommands that switch between buffers temporarily.

This option is used together with **'buftype'** and **'swapfile'** to specify special kinds of buffers. See [special-buffers](#) .

'buflisted' 'bl' ['buflisted'](#) ['bl'](#) ['nobuflisted'](#) ['nobl'](#) E85
 boolean (default: on)
 local to buffer

When this option is set, the buffer shows up in the buffer list. If it is reset it is not used for `":bnext"`, `"ls"`, the Buffers menu, etc. This option is reset by Vim for buffers that are only used to remember a file name or marks. Vim sets it when starting to edit a buffer. But not when moving to a buffer with `":buffer"`.

'buftype' 'bt' ['buftype'](#) ['bt'](#) E382
 string (default: "")
 local to buffer [local-noglobal](#)

The value of this option specifies the type of a buffer:

<code><empty></code>	normal buffer
nofile	buffer which is not related to a file and will not be written
nowrite	buffer which will not be written
acwrite	buffer which will always be written with <code>BufWriteCmd</code> autocommands.
quickfix	quickfix buffer, contains list of errors <code>:cwindow</code> or list of locations <code>:lwindow</code>
help	help buffer (you are not supposed to set this manually)
terminal	buffer for a terminal (you are not supposed to set this manually)
prompt	buffer where only the last line can be edited, meant to be used by a plugin, see prompt-buffer {only when compiled with the <code> +channel </code> feature}
popup	buffer used in a popup window, see popup . {only when compiled with the <code> +textprop </code> feature}

This option is used together with **'bufhidden'** and **'swapfile'** to

specify special kinds of buffers. See [special-buffers](#) .
Also see [win_gettype\(\)](#) , which returns the type of the window.

Be careful with changing this option, it can have many side effects!
One such effect is that Vim will not check the timestamp of the file,
if the file is changed by another program this will not be noticed.

A "quickfix" buffer is only used for the error list and the location
list. This value is set by the [:cwindow](#) and [:lwindow](#) commands and
you are not supposed to change it.

"nofile" and "nowrite" buffers are similar:

both:	The buffer is not to be written to disk, ":w" doesn't work (":w filename" does work though).
both:	The buffer is never considered to be 'modified' . There is no warning when the changes will be lost, for example when you quit Vim.
both:	A swap file is only created when using too much memory (when 'swapfile' has been reset there is never a swap file).
nofile only:	The buffer name is fixed, it is not handled like a file name. It is not modified in response to a :cd command.
both:	When using ":e bufname" and already editing "bufname" the buffer is made empty and autocommands are triggered as usual for :edit .

E676

"acwrite" implies that the buffer name is not related to a file, like "nofile", but it will be written. Thus, in contrast to "nofile" and "nowrite", ":w" does work and a modified buffer can't be abandoned without saving. For writing there must be matching [BufWriteCmd](#) , [FileWriteCmd](#) or [FileAppendCmd](#) autocommands.

		'casemap'	'cmp'	
'casemap'	'cmp'	string (default: "internal,keepascii")		
		global		
		Specifies details about changing the case of letters. It may contain these words, separated by a comma:		
	internal	Use internal case mapping functions, the current locale does not change the case mapping. This only matters when 'encoding' is a Unicode encoding, "latin1" or "iso-8859-15". When "internal" is omitted, the toupper() and tolower() system library functions are used when available.		
	keepascii	For the ASCII characters (0x00 to 0x7f) use the US case mapping, the current locale is not effective. This probably only matters for Turkish.		

		'cdhome'	'cdh'	'nocdhome'	'nocdh'	
'cdhome'	'cdh'	boolean (default: off)				
		global				
		When on, :cd , :tcd and :lcd without an argument changes the current working directory to the \$HOME directory like in Unix.				
		When off, those commands just print the current directory name.				

On Unix this option has no effect.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

NOTE: This option is reset when `'compatible'` is set.

`'cdpath'` `'cd'` `'cdpath'` `'cd'` E344 E346
string (default: equivalent to \$CDPATH or ",,")
global

This is a list of directories which will be searched when using the `:cd`, `:tcd` and `:lcd` commands, provided that the directory being searched for has a relative path, not an absolute part starting with `"/`, `"/.` or `"/..`, the `'cdpath'` option is not used then.

The `'cdpath'` option's value has the same form and semantics as `'path'`. Also see `file-searching`.

The default value is taken from \$CDPATH, with a `,"` prepended to look in the current directory first.

If the default value taken from \$CDPATH is not what you want, include a modified version of the following command in your vimrc file to override it:

```
:let &cdpath = ', ' .. substitute(substitute($CDPATH, '[, ]', '\\\\0', 'g'), ':',
```

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

(parts of `'cdpath'` can be passed to the shell to expand file names).

`'cedit'` `'cedit'`
string (Vi default: "", Vim default: **CTRL-F**)
global

The key used in Command-line Mode to open the command-line window.

The default is **CTRL-F** when `'compatible'` is off.

Only non-printable keys are allowed.

The key can be specified as a single character, but it is difficult to type. The preferred way is to use the `<>` notation. Examples:

```
:exe "set cedit=\
```

```
:exe "set cedit=\
```

`Nvi` also has this option, but it only uses the first character.

See `cmdwin`.

NOTE: This option is set to the Vim default value when `'compatible'` is reset.

`'charconvert'` `'ccv'` `'charconvert'` `'ccv'` E202 E214 E513
string (default "")
global
{only available when compiled with the `|+eval|` feature}

An expression that is used for character encoding conversion. It is evaluated when a file that is to be read or has been written has a different encoding from what is desired.

`'charconvert'` is not used when the internal `iconv()` function is supported and is able to do the conversion. Using `iconv()` is preferred, because it is much faster.

`'charconvert'` is not used when reading `stdin --`, because there is no file to convert from. You will have to save the text in a file first. The expression must return zero, false or an empty string for success, non-zero or true for failure.

The possible encoding names encountered are in `'encoding'`.

Additionally, names given in **'fileencodings'** and **'fileencoding'** are used.

Conversion between "latin1", "unicode", "ucs-2", "ucs-4" and "utf-8" is done internally by Vim, **'charconvert'** is not used for this.

'charconvert' is also used to convert the viminfo file, if the 'c' flag is present in **'viminfo'**. Also used for Unicode conversion.

Example:

```
set charconvert=CharConvert()
fun CharConvert()
  system("recode "
    \ .. v:charconvert_from .. ".." .. v:charconvert_to
    \ .. " <" .. v:fname_in .. ">" .. v:fname_out)
  return v:shell_error
endfun
```

The related Vim variables are:

v:charconvert_from	name of the current encoding
v:charconvert_to	name of the desired encoding
v:fname_in	name of the input file
v:fname_out	name of the output file

Note that v:fname_in and v:fname_out will never be the same.

Note that v:charconvert_from and v:charconvert_to may be different from **'encoding'**. Vim internally uses UTF-8 instead of UCS-2 or UCS-4.

The advantage of using a function call without arguments is that it is faster, see [expr-option-function](#) .

Encryption is not done by Vim when using **'charconvert'**. If you want to encrypt the file after conversion, **'charconvert'** should take care of this.

If the **'charconvert'** expression starts with s: or <SID> , then it is replaced with the script ID ([local-function](#)). Example:

```
set charconvert=s:MyConvert()
set charconvert=<SID>SomeConvert()
```

Otherwise the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'cindent'	'cin'	'cindent'	'cin'	'nocindent'	'nocin'
boolean (default off)					
		local to buffer			

Enables automatic C program indenting. See **'cinkeys'** to set the keys that trigger reindenting in insert mode and **'cinoptions'** to set your preferred indent style.

If **'indentexpr'** is not empty, it overrules **'cindent'**.

If **'lisp'** is not on and both **'indentexpr'** and **'equalprg'** are empty, the "=" operator indents using this algorithm rather than calling an external program.

See [C-indenting](#) .

When you don't like the way **'cindent'** works, try the **'smartindent'** option or **'indentexpr'**.

This option is not used when **'paste'** is set.

NOTE: This option is reset when **'compatible'** is set.

'cinkeys' **'cink'** string (default "0{,0},0),0],:,0#,!^F,o,O,e")
local to buffer

A list of keys that, when typed in Insert mode, cause reindenting of the current line. Only used if **'cindent'** is on and **'indentexpr'** is empty.

For the format of this option see [cinkeys-format](#) .

See [C-indenting](#) .

'cinoptions' **'cino'** string (default "")
local to buffer

The **'cinoptions'** affect the way **'cindent'** reindents lines in a C program. See [cinoptions-values](#) for the values of this option, and [C-indenting](#) for info on C indenting in general.

'cinscopedecls' **'cinsd'** string (default "public,protected,private")
local to buffer

Keywords that are interpreted as a C++ scope declaration by [cino-g](#) . Useful e.g. for working with the Qt framework that defines additional scope declarations "signals", "public slots" and "private slots":

`set cinscopedecls+=signals,public\ slots,private\ slots`

'cinwords' **'cinw'** string (default "if,else,while,do,for,switch")
local to buffer

These keywords start an extra indent in the next line when **'smartindent'** or **'cindent'** is set. For **'cindent'** this is only done at an appropriate place (inside {}).

Note that **'ignorecase'** isn't used for **'cinwords'**. If case doesn't matter, include the keyword both the uppercase and lowercase:

"if,If,IF".

'clipboard' **'cb'** string (default "autoselect,exclude:cons\|linux"
for X-windows, "" otherwise)
global
{only in GUI versions or when the `+xterm_clipboard`
feature is included}

This option is a list of comma-separated names.

Note: if one of the items is "exclude:", then you can't add an item after that. Therefore do not append an item with += but use ^= to prepend, e.g.:

`set clipboard^=unnamed`

When using the GUI see ['go-A'](#) .

These names are recognized:

clipboard-unnamed
unnamed When included, Vim will use the clipboard register '*' for all yank, delete, change and put operations which would normally go to the unnamed register. When a

register is explicitly specified, it will always be used regardless of whether "unnamed" is in **'clipboard'** or not. The clipboard register can always be explicitly accessed using the "*" notation. Also see [gui-clipboard](#) .

- clipboard-unnamedplus**
- unnamedplus A variant of the "unnamed" flag which uses the clipboard register '+' ([quoteplus](#)) instead of register '*' for all yank, delete, change and put operations which would normally go to the unnamed register. When "unnamed" is also included to the option, yank operations (but not delete, change or put) will additionally copy the text into register '*'.
Only available with the [+X11](#) feature.
Availability can be checked with:
`if has('unnamedplus')`
- clipboard-autoselect**
- autoselect Works like the 'a' flag in **'guioptions'**: If present, then whenever Visual mode is started, or the Visual area extended, Vim tries to become the owner of the windowing system's global selection or put the selected text on the clipboard used by the selection register "*". See ['go-a'](#) and [quotestar](#) for details. When the GUI is active, the 'a' flag in **'guioptions'** is used, when the GUI is not active, this "autoselect" flag is used.
Also applies to the modeless selection.
- clipboard-autoselectplus**
- autoselectplus Like "autoselect" but using the + register instead of the * register. Compare to the 'P' flag in **'guioptions'**.
- clipboard-autoselectml**
- autoselectml Like "autoselect", but for the modeless selection only. Compare to the 'A' flag in **'guioptions'**.
- clipboard-html**
- html When the clipboard contains HTML, use this when pasting. When putting text on the clipboard, mark it as HTML. This works to copy rendered HTML from Firefox, paste it as raw HTML in Vim, select the HTML in Vim and paste it in a rich edit box in Firefox. You probably want to add this only temporarily, possibly use BufEnter autocommands.
Only supported for GTK version 2 and later.
- clipboard-exclude**
- exclude:{pattern}
Defines a pattern that is matched against the name of the terminal **'term'**. If there is a match, no

connection will be made to the X server. This is useful in this situation:

- Running Vim in a console.
- \$DISPLAY is set to start applications on another display.
- You do not want to connect to the X server in the console, but do want this in a terminal emulator.

To never connect to the X server use:

```
exclude:.*
```

This has the same effect as using the `-X` argument.

Note that when there is no connection to the X server the window title won't be restored and the clipboard cannot be accessed.

The value of `'magic'` is ignored, `{pattern}` is interpreted as if `'magic'` was on.

The rest of the option value will be used for `{pattern}`, this must be the last entry.

```
'cmdheight' 'ch'          number (default 1)          'cmdheight' 'ch'
                          global or local to tab page
Number of screen lines to use for the command-line. A larger value
helps avoiding hit-enter prompts.
The value of this option is stored with the tab page, so that each tab
page can have a different value.
```

```
'cmdwinheight' 'cwh'     number (default 7)          'cmdwinheight' 'cwh'
                          global
Number of screen lines to use for the command-line window. cmdwin
```

```
'colorcolumn' 'cc'      string (default "")          'colorcolumn' 'cc'
                          local to window
                          {not available when compiled without the +syntax
                          feature}
'colorcolumn' is a comma-separated list of screen columns that are
highlighted with ColorColumn hl-ColorColumn. Useful to align
text. Will make screen redrawing slower.
The screen column can be an absolute number, or a number preceded with
'+' or '-', which is added to or subtracted from 'textwidth'.
```

```
:set cc=+1          " highlight column after 'textwidth'
:set cc=+1,+2,+3    " highlight three columns after 'textwidth'
:hi ColorColumn     ctermbg=lightgrey guibg=lightgrey
```

When **'textwidth'** is zero then the items with '-' and '+' are not used. A maximum of 256 columns are highlighted.

```
'columns' 'co'          number (default 80 or terminal width) 'columns' 'co' E594
                          global
Number of columns of the screen. Normally this is set by the terminal
initialization and does not have to be set by hand. Also see
```

[posix-screen-size](#) .

When Vim is running in the GUI or in a resizable window, setting this option will cause the window size to be changed. When you only want to use the size for the GUI, put the command in your [gvimrc](#) file. When you set this option and Vim is unable to change the physical number of columns of the display, the display may be messed up. For the GUI it is always possible and Vim limits the number of columns to what fits on the screen. You can use this command to get the widest window possible:

```
:set columns=9999
```

Minimum value is 12, maximum value is 10000.

'comments' **'com'** [string](#) (default ['comments'](#) ['com'](#) [E524](#) [E525](#)
"s1:/*,mb:*,ex:*/,://,b:#,:%,:XCOMM,n:>,fb:-")
local to buffer

A comma-separated list of strings that can start a comment line. See [format-comments](#) . See [option-backslash](#) about using backslashes to insert a space.

'commentstring' **'cms'** [string](#) (default ['commentstring'](#) ['cms'](#) [E537](#)
"/*%s*/")
local to buffer
{not available when compiled without the [+folding](#)
feature}

A template for a comment. The "%s" in the value is replaced with the comment text. Currently only used to add markers for folding, see [fold-marker](#) .

'compatible' **'cp'** [boolean](#) (default on, off when a [vimrc](#) or [gvimrc](#)
['compatible'](#) ['cp'](#) ['nocompatible'](#) ['nosp'](#)
file is found, reset in [defaults.vim](#))
global

This option has the effect of making Vim either more Vi-compatible, or make Vim behave in a more useful way.

This is a special kind of option, because when it's set or reset, other options are also changed as a side effect.

NOTE: Setting or resetting this option can have a lot of unexpected effects: Mappings are interpreted in another way, undo behaves differently, etc. If you set this option in your vimrc file, you should probably put it at the very start.

By default this option is on and the Vi defaults are used for the options. This default was chosen for those people who want to use Vim just like Vi, and don't even (want to) know about the **'compatible'** option.

When a [vimrc](#) or [gvimrc](#) file is found while Vim is starting up, this option is switched off, and all options that have not been modified will be set to the Vim defaults. Effectively, this means that when a [vimrc](#) or [gvimrc](#) file exists, Vim will use the Vim defaults, otherwise it will use the Vi defaults. (Note: This doesn't happen for the system-wide vimrc or gvimrc file, nor for a file given with the [-u](#) argument). Also see [compatible-default](#) and

[posix-compliance](#) .

You can also set this option with the "-C" argument, and reset it with "-N". See [-C](#) and [-N](#) .

See ['coptions'](#) for more fine tuning of Vi compatibility.

When this option is set, numerous other options are set to make Vim as Vi-compatible as possible. When this option is unset, various options are set to make Vim more useful. The table below lists all the options affected.

The {?} column indicates when the options are affected:

+ Means that the option is set to the value given in {set value} when ['compatible'](#) is set.

& Means that the option is set to the value given in {set value} when ['compatible'](#) is set AND is set to its Vim default value when ['compatible'](#) is unset.

- Means the option is NOT changed when setting ['compatible'](#) but IS set to its Vim default when ['compatible'](#) is unset.

The {effect} column summarises the change when ['compatible'](#) is set.

option	? set value	effect
'allowrevins'	+ off	no CTRL-_ command
'antialias'	+ off	don't use antialiased fonts
'arabic'	+ off	reset arabic-related options
'arabicshape'	+ on	correct character shapes
'backspace'	+ ""	normal backspace
'backup'	+ off	no backup file
'backupcopy'	& Unix: "yes" else: "auto"	backup file is a copy copy or rename backup file
'balloonexpr'	+ ""	text to show in evaluation balloon
'breakindent'	+ off	don't indent when wrapping lines
'cedit'	- {unchanged}	{set vim default only on resetting 'cp'}
'cdhome'	+ off	":cd" don't chdir to home on non-Unix
'cindent'	+ off	no C code indentation
'compatible'	- {unchanged}	{set vim default only on resetting 'cp'}
'copyindent'	+ off	don't copy indent structure
'coptions'	& (all flags)	Vi-compatible flags
'cscopepathcomp'	+ 0	don't show directories in tags list
'cscoperelative'	+ off	don't use basename of path as prefix
'cscopetag'	+ off	don't use cscope for ":tag"
'cscopetagorder'	+ 0	see cscopetagorder
'cscopeverbose'	+ off	see cscopeverbose
'delcombine'	+ off	unicode: delete whole char combination
'digraph'	+ off	no digraphs
'esckeys'	& off	no <Esc>-keys in Insert mode this also disables modifyOtherKeys and xterm-bracketed-paste
'expandtab'	+ off	tabs not expanded to spaces
'fileformats'	& "" "dos,unix"	no automatic file format detection, except for MS-Windows
'formatexpr'	+ ""	use 'formatprg' for auto-formatting
'formatoptions'	& "vt"	Vi compatible formatting
'gdefault'	+ off	no default 'g' flag for ":s"
'history'	& 0	no commandline history

'hmap'	+ off	no Hebrew keyboard mapping
'hkmap'	+ off	no phonetic Hebrew keyboard mapping
'hlsearch'	+ off	no highlighting of search matches
'incsearch'	+ off	no incremental searching
'indentexpr'	+ ""	no indenting by expression
'insertmode'	+ off	do not start in Insert mode
'iskeyword'	& "@,48-57,_"	keywords contain alphanumeric characters and '_'
'joinspaces'	+ on	insert 2 spaces after period
'modeline'	& off	no modelines
'more'	& off	no pauses in listings
'mzquantum'	- {unchanged}	{set vim default only on resetting 'cp'}
'numberwidth'	& 8	min number of columns for line number
'preserveindent'	+ off	don't preserve current indent structure when changing it
'revins'	+ off	no reverse insert
'ruler'	+ off	no ruler
'scrolljump'	+ 1	no jump scroll
'scrolloff'	+ 0	no scroll offset
'shelltemp'	- {unchanged}	{set vim default only on resetting 'cp'}
'shiftround'	+ off	indent not rounded to shiftwidth
'shortmess'	& "S"	no shortening of messages
'showcmd'	& off	command characters not shown
'showmode'	& off	current mode not shown
'sidescrolloff'	+ 0	cursor moves to edge of screen in scroll
'smartcase'	+ off	no automatic ignore case switch
'smartindent'	+ off	no smart indentation
'smarttab'	+ off	no smart tab size
'softtabstop'	+ 0	tabs are always 'tabstop' positions
'startofline'	+ on	goto startofline with some commands
'tagcase'	& "followic"	'ignorecase' when searching tags file
'tagrelative'	& off	tag file names are not relative
'termguicolors'	+ off	don't use highlight-(guifg guibg)
'textauto'	& off	no automatic textmode detection
'textwidth'	+ 0	no automatic line wrap
'tildeop'	+ off	tilde is not an operator
'ttimeout'	+ off	no terminal timeout
'undofile'	+ off	don't use an undo file
'viminfo'	- {unchanged}	{set Vim default only on resetting 'cp'}
'virtualedit'	+ ""	cursor can only be placed on characters
'whichwrap'	& ""	left-right movements don't wrap
'wildchar'	& CTRL-E	only when the current value is <Tab> use CTRL-E for cmdline completion
'writebackup'	+ on or off	depends on the +writebackup feature

'complete' 'cpt' string (default: ".,w,b,u,t,i") local to buffer 'complete' 'cpt' E535

This option specifies how keyword completion `ins-completion` works when **CTRL-P** or **CTRL-N** are used. It is also used for whole-line completion `i_CTRL-X_CTRL-L`. It indicates the type of completion and the places to scan. It is a comma-separated list of flags:

- .
 - w
- scan the current buffer ('wrapscan' is ignored)
scan buffers from other windows

```

b      scan other loaded buffers that are in the buffer list
u      scan the unloaded buffers that are in the buffer list
U      scan the buffers that are not in the buffer list
k      scan the files given with the 'dictionary' option
kspell use the currently active spell checking spell
k{dict} scan the file {dict}. Several "k" flags can be given,
      patterns are valid too. For example:
      :set cpt=k/usr/dict/*,k~/spanish
s      scan the files given with the 'thesaurus' option
s{tsr} scan the file {tsr}. Several "s" flags can be given, patterns
      are valid too.
i      scan current and included files
d      scan current and included files for defined name or macro
      i_CTRL-X_CTRL-D
]      tag completion
t      same as "]"

```

Unloaded buffers are not loaded, thus their autocmds `:autocmd` are not executed, this may lead to unexpected completions from some files (gzipped files for example). Unloaded buffers are not scanned for whole-line completion.

The default is `".,w,b,u,t,i"`, which means to scan:

1. the current buffer
2. buffers in other windows
3. other loaded buffers
4. unloaded buffers
5. tags
6. included files

As you can see, **CTRL-N** and **CTRL-P** can be used to do any **'iskeyword'**-based expansion (e.g., dictionary `i_CTRL-X_CTRL-K`, included patterns `i_CTRL-X_CTRL-I`, tags `i_CTRL-X_CTRL-]` and normal expansions).

```

'completefunc' 'cfu'      string (default: empty)
                             local to buffer
                             {not available when compiled without the +eval
                             feature}

```

This option specifies a function to be used for Insert mode completion with **CTRL-X CTRL-U**. `i_CTRL-X_CTRL-U`

See [complete-functions](#) for an explanation of how the function is invoked and what it should return. The value can be the name of a function, a `lambda` or a `Funcref`. See [option-value-function](#) for more information.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

```

'completeopt' 'cot'      string (default: "menu,preview")
                             global

```

A comma-separated list of options for Insert mode completion [ins-completion](#). The supported values are:

menu Use a popup menu to show the possible completions. The menu is only shown when there is more than one match and sufficient colors are available. `ins-completion-menu`

menuone Use the popup menu also when there is only one match. Useful when there is additional information about the match, e.g., what file it comes from.

longest Only insert the longest common text of the matches. If the menu is displayed you can use **CTRL-L** to add more characters. Whether case is ignored depends on the kind of completion. For buffer text the `'ignorecase'` option is used.

preview Show extra information about the currently selected completion in the preview window. Only works in combination with "menu" or "menuone".

popup Show extra information about the currently selected completion in a popup window. Only works in combination with "menu" or "menuone". Overrides "preview". See `'completepopup'` for specifying properties. `{only works when compiled with the |+textprop| feature}`

popuphidden Just like "popup" but initially hide the popup. Use a `CompleteChanged` autocommand to fetch the info and call `popup_show()` once the popup has been filled. See the example at `complete-popuphidden` . `{only works when compiled with the |+textprop| feature}`

noinset Do not insert any text for a match until the user selects a match from the menu. Only works in combination with "menu" or "menuone". No effect if "longest" is present.

noselect Do not select a match in the menu, force the user to select one from the menu. Only works in combination with "menu" or "menuone".

`'completepopup'` `'cpp'` `'completepopup'` `'cpp'`
 string (default empty)
 global
 {not available when compiled without the `+textprop`
 or `+quickfix` feature}

When `'completeopt'` contains "popup" then this option is used for the properties of the info popup when it is created. If an info popup window already exists it is closed, so that the option value is applied when it is created again.

You can also use `popup_findinfo()` and then set properties for an existing info popup with `popup_setoptions()` . See `complete-popup` .

`'completeslash'` `'csl'` `'completeslash'` `'csl'`
 string (default: "")
 local to buffer

{only for MS-Windows}

When this option is set it overrules **'shellslash'** for completion:

- When this option is set to "slash", a forward slash is used for path completion in insert mode. This is useful when editing HTML tag, or Makefile with **'noshellslash'** on MS-Windows.
- When this option is set to "backslash", backslash is used. This is useful when editing a batch file with **'shellslash'** set on MS-Windows.
- When this option is empty, same character is used as for **'shellslash'**.

For Insert mode completion the buffer-local value is used. For command line completion the global value is used.

'concealcursor' **'cocu'** string (default: "")
local to window
{not available when compiled without the **+conceal** feature}

Sets the modes in which text in the cursor line can also be concealed. When the current mode is listed then concealing happens just like in other lines.

n	Normal mode
v	Visual mode
i	Insert mode
c	Command line editing, for 'incsearch'

'v' applies to all lines in the Visual area, not only the cursor. A useful value is "nc". This is used in help files. So long as you are moving around text is concealed, but when starting to insert text or selecting a Visual area the concealed text is displayed, so that you can see what you are doing.

Keep in mind that the cursor position is not always where it's displayed. E.g., when moving vertically it may change column.

'conceallevel' **'cole'** number (default 0)
local to window
{not available when compiled without the **+conceal** feature}

Determine how text with the "conceal" syntax attribute **:syn-conceal** is shown:

Value	Effect
0	Text is shown normally
1	Each block of concealed text is replaced with one character. If the syntax item does not have a custom replacement character defined (see :syn-cchar) the character defined in 'listchars' is used (default is a space).
2	It is highlighted with the "Conceal" highlight group. Concealed text is completely hidden unless it has a custom replacement character defined (see :syn-cchar).
3	Concealed text is completely hidden.

Note: in the cursor line concealed text is not hidden, so that you can edit and copy the text. This can be changed with the **'concealcursor'** option.

'confirm' **'cf'** **'confirm'** **'cf'** **'noconfirm'** **'nocf'**
boolean (default off)
global

When **'confirm'** is on, certain operations that would normally fail because of unsaved changes to a buffer, e.g. `":q"` and `":e"`, instead raise a **dialog** asking if you wish to save the current file(s). You can still use a `!` to unconditionally **abandon** a buffer. If **'confirm'** is off you can still activate confirmation for one command only (this is most useful in mappings) with the `:confirm` command.

Also see the `confirm()` function and the `'v'` flag in **'guioptions'**.

'conskey' **'consk'** **'conskey'** **'consk'** **'noconskey'** **'noconsk'**
boolean (default off)
global

This was for MS-DOS and is no longer supported.

'copyindent' **'ci'** **'copyindent'** **'ci'** **'nocopyindent'** **'noci'**
boolean (default off)
local to buffer

Copy the structure of the existing lines indent when autoindenting a new line. Normally the new indent is reconstructed by a series of tabs followed by spaces as required (unless **'expandtab'** is enabled, in which case only spaces are used). Enabling this option makes the new line copy whatever characters were used for indenting on the existing line. **'expandtab'** has no effect on these characters, a Tab remains a Tab. If the new indent is greater than on the existing line, the remaining space is filled in the normal manner.

NOTE: This option is reset when **'compatible'** is set.

Also see **'preserveindent'**.

'coptions' **'cpo'** **'coptions'** **'cpo'** **cpo**
string (Vim default: "aABceFs",
Vi default: all flags)
global

A sequence of single character flags. When a character is present this indicates Vi-compatible behavior. This is used for things where not being Vi-compatible is mostly or sometimes preferred.

'coptions' stands for "compatible-options".

Commas can be added for readability.

To avoid problems with flags that are added in the future, use the `"+="` and `"-="` feature of `":set"` **add-option-flags**.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

NOTE: In a **Vim9** script, when ``vim9script`` is encountered, the value is saved, **'coptions'** is set to the Vim default, and the saved value is restored at the end of the script. Changes to the value of **'coptions'** will be applied to the saved value, but keep in mind that

removing a flag that is not present when **'coptions'** is changed has no effect. In the `.vimrc` file the value is not restored, thus using ``vim9script`` in the `.vimrc` file results in using the Vim default.

NOTE: This option is set to the POSIX default value at startup when the Vi default value would be used and the `$VIM_POSIX` environment variable exists `posix`. This means Vim tries to behave like the POSIX specification.

contains	behavior
	cpo-a
a	When included, a <code>":read"</code> command with a file name argument will set the alternate file name for the current window.
	cpo-A
A	When included, a <code>":write"</code> command with a file name argument will set the alternate file name for the current window.
	cpo-b
b	<code>"\ "</code> in a <code>":map"</code> command is recognized as the end of the map command. The <code>\ </code> is included in the mapping, the text after the <code> </code> is interpreted as the next command. Use a CTRL-V instead of a backslash to include the <code> </code> in the mapping. Applies to all mapping, abbreviation, menu and autocmd commands. See also <code>map_bar</code> .
	cpo-B
B	A backslash has no special meaning in mappings, abbreviations, user commands and the "to" part of the menu commands. Remove this flag to be able to use a backslash like a CTRL-V . For example, the command <code>":map X \<esc>"< code=""> results in X being mapped to: 'B' included: <code>"\^["</code> (<code>^[</code> is a real <code><Esc></code>) 'B' excluded: <code>"<Esc>"</code> (5 characters) (<code>'<</code> excluded in both cases)</esc>"<></code>
	cpo-c
c	Searching continues at the end of any match at the cursor position, but not further than the start of the next line. When not present searching continues one character from the cursor position. With <code>'c'</code> <code>"ababababab"</code> only gets three matches when repeating <code>"/abab"</code> , without <code>'c'</code> there are five matches.
	cpo-C
C	Do not concatenate sourced lines that start with a backslash. See <code>line-continuation</code> .
	cpo-d
d	Using <code>"/."</code> in the 'tags' option doesn't mean to use the tags file relative to the current file, but the tags file in the current directory.
	cpo-D
D	Can't use CTRL-K to enter a digraph after Normal mode commands with a character argument, like <code>r</code> , <code>f</code> and <code>t</code> .
	cpo-e

- e When executing a register with ":@", always add a `<CR>` to the last line, also when the register is not linewise. If this flag is not present, the register is not linewise and the last line does not end in a `<CR>`, then the last line is put on the command-line and can be edited before hitting `<CR>`. cpo-E
- E It is an error when using "y", "d", "c", "g~", "gu" or "gU" on an Empty region. The operators only work when at least one character is to be operated on. Example: This makes "y0" fail in the first column. cpo-f
- f When included, a ":read" command with a file name argument will set the file name for the current buffer, if the current buffer doesn't have a file name yet. cpo-F
- F When included, a ":write" command with a file name argument will set the file name for the current buffer, if the current buffer doesn't have a file name yet. Also see `cpo-P`. cpo-g
- g Goto line 1 when using ":edit" without argument. cpo-H
- H When using "I" on a line with only blanks, insert before the last blank. Without this flag insert after the last blank. cpo-i
- i When included, interrupting the reading of a file will leave it modified. cpo-I
- I When moving the cursor up or down just after inserting indent for `'autoindent'`, do not delete the indent. cpo-j
- j When joining lines, only add two spaces after a '.', not after '!' or '?'. Also see `'joinspaces'`. cpo-J
- J A `sentence` has to be followed by two spaces after the '.', '!' or '?'. A `<Tab>` is not recognized as white space. cpo-k
- k Disable the recognition of raw key codes in mappings, abbreviations, and the "to" part of menu commands. For example, if `<Key>` sends `^[OA` (where `^` is `<Esc>`), the command `:"map X ^[OA` results in X being mapped to:
- | | | |
|---------------|--------------------------|----------------|
| 'k' included: | <code>^[OA</code> | (3 characters) |
| 'k' excluded: | <code><Key></code> | (one key code) |
- Also see the '`<`' flag below. cpo-K
- K Don't wait for a key code to complete when it is halfway a mapping. This breaks mapping `<F1><F1>` when only part of the second `<F1>` has been read. It enables cancelling the mapping by typing `<F1><Esc>`. cpo-l

- l Backslash in a [] range in a search pattern is taken literally, only "\]", "\^", "\-" and "\\\" are special. See /[]
 'l' included: "/[\t]" finds <Space>, '\ ' and 't'
 'l' excluded: "/[\t]" finds <Space> and <Tab>
 Also see `cpo-\` .
- L When the **'list'** option is set, **'wrapmargin'**, **'textwidth'**, **'softtabstop'** and Virtual Replace mode (see `gR`) count a <Tab> as two characters, instead of the normal behavior of a <Tab>.
- m When included, a showmatch will always wait half a second. When not included, a showmatch will wait half a second or until a character is typed. `'showmatch'`
- M When excluded, "%" matching will take backslashes into account. Thus in "(\(" and "\((\)" the outer parenthesis match. When included "%" ignores backslashes, which is Vi compatible.
- n When included, the column used for **'number'** and **'relativenumber'** will also be used for text of wrapped lines.
- o Line offset to search command is not remembered for next search.
- O Don't complain if a file is being overwritten, even when it didn't exist when editing it. This is a protection against a file unexpectedly created by someone else. Vi didn't complain about this.
- p Vi compatible Lisp indenting. When not present, a slightly better algorithm is used.
- P When included, a ":write" command that appends to a file will set the file name for the current buffer, if the current buffer doesn't have a file name yet and the 'F' flag is also included `cpo-F` .
- q When joining multiple lines leave the cursor at the position where it would be when joining two lines.
- r Redo (". " command) uses "/" to repeat a search command, instead of the actually used search string.
- R Remove marks from filtered lines. Without this flag marks are kept like `:keepmarks` was used.
- s Set buffer options when entering the buffer for the first time. This is like it is in Vim version 3.0. And it is the default. If not present the options are set when the buffer is created.

- cpo-S
- S Set buffer options always when entering a buffer (except **'readonly'**, **'fileformat'**, **'filetype'** and **'syntax'**). This is the (most) Vi compatible setting. The options are set to the values in the current buffer. When you change an option and go to another buffer, the value is copied. Effectively makes the buffer options global to all buffers.
- | | | |
|-----|-----|--|
| 's' | 'S' | copy buffer options |
| no | no | when buffer created |
| yes | no | when buffer first entered (default) |
| X | yes | each time when buffer entered (vi comp.) |
- cpo-t
- t Search pattern for the tag command is remembered for "n" command. Otherwise Vim only puts the pattern in the history for search pattern, but doesn't change the last used search pattern.
- cpo-u
- u Undo is Vi compatible. See [undo-two-ways](#) .
- cpo-v
- v Backspaced characters remain visible on the screen in Insert mode. Without this flag the characters are erased from the screen right away. With this flag the screen newly typed text overwrites backspaced characters.
- cpo-w
- w When using "cw" on a blank character, only change one character and not all blanks until the start of the next word.
- cpo-W
- W Don't overwrite a readonly file. When omitted, ":w!" overwrites a readonly file, if possible.
- cpo-x
- x **<Esc>** on the command-line executes the command-line. The default in Vim is to abandon the command-line, because **<Esc>** normally aborts a command. [c_<Esc>](#)
- cpo-X
- X When using a count with "R" the replaced text is deleted only once. Also when repeating "R" with "." and a count.
- cpo-y
- y A yank command can be redone with ".". Think twice if you really want to use this, it may break some plugins, since most people expect "." to only repeat a change.
- cpo-Z
- Z When using "w!" while the **'readonly'** option is set, don't reset **'readonly'**.
- cpo-!
- ! When redoing a filter command, use the last used external command, whatever it was. Otherwise the last used `-filter-` command is used.
- cpo-\$

\$ When making a change to one line, don't redisplay the line, but put a '\$' at the end of the changed text. The changed text will be overwritten when you type the new text. The line is redisplayed if you type any command that moves the cursor from the insertion point.

cpo-%

% Vi-compatible matching is done for the "%" command. Does not recognize "#if", "#endif", etc. Does not recognize "/*" and "*/". Parens inside single and double quotes are also counted, causing a string that contains a paren to disturb the matching. For example, in a line like "if (strcmp("foo(", s))" the first paren does not match the last one. When this flag is not included, parens inside single and double quotes are treated specially. When matching a paren outside of quotes, everything inside quotes is ignored. When matching a paren inside quotes, it will find the matching one (if there is one). This works very well for C programs. This flag is also used for other features, such as C-indenting.

cpo--

- When included, a vertical movement command fails when it would go above the first line or below the last line. Without it the cursor moves to the first or last line, unless it already was in that line. Applies to the commands "-", "k", **CTRL-P**, "+", "j", **CTRL-N**, **CTRL-J** and ":1234".

cpo++

+ When included, a ":write file" command will reset the **'modified'** flag of the buffer, even though the buffer itself may still be different from its file.

cpo-star

* Use ":*" in the same way as ":@"". When not included, ":*" is an alias for ":'<,'>", select the Visual area.

cpo-<

< Disable the recognition of special key codes in |<>| form in mappings, abbreviations, and the "to" part of menu commands. For example, the command ":map X <Tab>" results in X being mapped to:
 '<' included: "<Tab>" (5 characters)
 '<' excluded: "^I" (^I is a real <Tab>)
 Also see the 'k' flag above.

cpo->

> When appending to a register, put a line break before the appended text.

cpo-;

; When using , or ; to repeat the last t search and the cursor is right in front of the searched character, the cursor won't move. When not included, the cursor would skip over it and jump to the following occurrence.

POSIX flags. These are not included in the Vi default value, except when \$VIM_POSIX was set on startup. [posix](#)

contains	behavior
	cpo-#
#	A count before "D", "o" and "O" has no effect.
	cpo-&
&	When ":preserve" was used keep the swap file when exiting normally while this buffer is still loaded. This flag is tested when exiting.
	cpo-\
\	Backslash in a [] range in a search pattern is taken literally, only "\]" is special See /[[] '\ ' included: "/[\-]" finds <Space> , '\ ' and '-' '\ ' excluded: "/[\-]" finds <Space> and '-' Also see cpo-l .
	cpo-/
/	When "%" is used as the replacement string in a :s command, use the previous replacement string. :s%
	cpo-{
{	The { and } commands also stop at a "{" character at the start of a line.
	cpo-.
.	The ":chdir" and ":cd" commands fail if the current buffer is modified, unless ! is used. Vim doesn't need this, since it remembers the full path of an opened file.
	cpo-bar
	The value of the \$LINES and \$COLUMNS environment variables overrule the terminal size values obtained with system specific functions.
	'cryptmethod' 'cm'
'cryptmethod' 'cm'	string (default "blowfish2") global or local to buffer global-local
Method used for	encryption when the buffer is written to a file:
	pkzip
zip	PkZip compatible method. A weak kind of encryption. Backwards compatible with Vim 7.2 and older. Only use if you need to be backwards compatible.
	blowfish
blowfish	Blowfish method. Medium strong encryption but it has an implementation flaw. Requires Vim 7.3 or later, files can NOT be read by Vim 7.2 and older. This adds a "seed" to the file, every time you write the file the encrypted bytes will be different. Obsolete, please do no longer use.
	blowfish2
blowfish2	Blowfish method. Medium strong encryption. Requires Vim 7.4.401 or later, files can NOT be read by Vim 7.3 and older. This adds a "seed" to the file, every time you write the file the encrypted bytes will be different. The whole undo file is encrypted, not just the pieces of text.

E1193 E1194 E1195 E1196 E1230
E1197 E1198 E1199 E1200 E1201

xchacha20 XChaCha20 Cipher with Poly1305 Message Authentication Code. Medium strong till strong encryption. Encryption is provided by the libsodium library, it requires Vim to be built with `+sodium`. It adds a seed and a message authentication code (MAC) to the file. This needs at least a Vim 8.2.3022 to read the encrypted file. Encryption of swap files is not supported, therefore no swap file will be used when xchacha20 encryption is enabled. Encryption of undo files is not yet supported, therefore no undo file will currently be written. CAREFUL: Files written with this method might have to be read back with the same version of Vim if the binary format changes later. Obsolete, please do no longer use.

xchacha20v2 Same algorithm as with "xchacha20" that correctly stores the key derivation parameters together with the encrypted file. Should work better in case the parameters in the libsodium library ever change. STILL EXPERIMENTAL: Files written with this method might have to be read back with the same version of Vim if the binary format changes later.

You should use "blowfish2", also to re-encrypt older files. The "xchacha20" method provides better encryption, but it does not work with all versions of Vim.

When reading an encrypted file `'cryptmethod'` will be set automatically to the detected method of the file being read. Thus if you write it without changing `'cryptmethod'` the same method will be used. Changing `'cryptmethod'` does not mark the file as modified, you have to explicitly write it, you don't get a warning unless there are other modifications. Also see `:X`.

When setting the global value to an empty string, it will end up with the value "blowfish2". When setting the local value to an empty string the buffer will use the global value.

When a new encryption method is added in a later version of Vim, and the current version does not recognize it, you will get `E821`. You need to edit this file with the later version of Vim.

`'cscopepathcomp'` `'cspc'` number (default 0) `'cscopepathcomp'` `'cspc'`
global
{not available when compiled without the `+cscope`
feature}

Determines how many components of the path to show in a list of tags. See `cscopepathcomp`.

NOTE: This option is set to 0 when `'compatible'` is set.

'cscopeprg' 'csprg' string (default "cscope")
 global
 {not available when compiled without the `+cscope` feature}
 Specifies the command to execute `cscope`. See `cscopeprg`.
 This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'cscopequickfix' 'csqf' string (default "")
 global
 {not available when compiled without the `+cscope` or `+quickfix` features}
 Specifies whether to use quickfix window to show `cscope` results.
 See `cscopequickfix`.

'cscoperelative' 'csre' boolean (default off)
 global
 {not available when compiled without the `+cscope` feature}
 In the absence of a prefix (-P) for `cscope`, setting this option enables to use the basename of `cscope.out` path as the prefix.
 See `cscoperelative`.
NOTE: This option is reset when `'compatible'` is set.

'cscopetag' 'cst' boolean (default off)
 global
 {not available when compiled without the `+cscope` feature}
 Use `cscope` for tag commands. See `cscope-options`.
NOTE: This option is reset when `'compatible'` is set.

'cscopetagorder' 'csto' number (default 0)
 global
 {not available when compiled without the `+cscope` feature}
 Determines the order in which `":cstag"` performs a search. See `cscopetagorder`.
NOTE: This option is set to 0 when `'compatible'` is set.

'cscopeverbose' 'csverb' boolean (default off)
 global
 {not available when compiled without the `+cscope` feature}
 Give messages when adding a `cscope` database. See `cscopeverbose`.
NOTE: This option is reset when `'compatible'` is set.

'cursorbind' 'crb' boolean (default off)
 global
 {not available when compiled without the `+cscope` feature}

'cursorbind' 'crb' boolean (default off)
 local to window

When this option is set, as the cursor in the current window moves other cursorbound windows (windows that also have this option set) move their cursors to the corresponding line and column. This option is useful for viewing the differences between two versions of a file (see **'diff'**); in diff mode, inserted and deleted lines (though not characters within a line) are taken into account.

'cursorcolumn' 'cuc' **'cursorcolumn' 'cuc'** **'nocursorcolumn' 'nocuc'**
 boolean (default off)
 local to window
 {not available when compiled without the **+syntax**
 feature}

Highlight the screen column of the cursor with CursorColumn **hl-CursorColumn** . Useful to align text. Will make screen redrawing slower.

If you only want the highlighting in the current window you can use these autocommands:

```
au WinLeave * set nocursorline nocursorcolumn
au WinEnter * set cursorline cursorcolumn
```

'cursorline' 'cul' **'cursorline' 'cul'** **'nocursorline' 'nocul'**
 boolean (default off)
 local to window
 {not available when compiled without the **+syntax**
 feature}

Highlight the text line of the cursor with CursorLine **hl-CursorLine** . Useful to easily spot the cursor. Will make screen redrawing slower. When Visual mode is active the highlighting isn't used to make it easier to see the selected text.

'cursorlineopt' 'culopt' **'cursorlineopt' 'culopt'**
 string (default: "number,line")
 local to window
 {not available when compiled without the **+syntax**
 feature}

Comma-separated list of settings for how **'cursorline'** is displayed. Valid values:

"line" Highlight the text line of the cursor with
 CursorLine **hl-CursorLine** .
"screenline" Highlight only the screen line of the cursor with
 CursorLine **hl-CursorLine** .
"number" Highlight the line number of the cursor with
 CursorLineNr **hl-CursorLineNr** .

Special value:

"both" Alias for the values "line,number".

"line" and "screenline" cannot be used together.

'debug'

'debug' string (default "")
global

These values can be used:

msg Error messages that would otherwise be omitted will be given anyway.
throw Error messages that would otherwise be omitted will be given anyway and also throw an exception and set `v:errmsg`.
beep A message will be given when otherwise only a beep would be produced.

The values can be combined, separated by a comma.

"msg" and "throw" are useful for debugging **'foldexpr'**, **'formatexpr'** or **'indentexpr'**.

'define' **'def'** string (default `^\s*#\s*define`)
global or local to buffer `global-local`

Pattern to be used to find a macro definition. It is a search pattern, just like for the "/" command. This option is used for the commands like "[i" and "[d" `include-search`. The **'isident'** option is used to recognize the defined name after the match:

```
{match with 'define'}{non-ID chars}{defined name}{non-ID char}
```

See `option-backslash` about inserting backslashes to include a space or backslash.

The default value is for C programs. For C++ this value would be useful, to include const type declarations:

```
^\(#\s*define\|[a-z]*\s*const\s*[a-z]*\)
```

You can also use "\ze" just before the name and continue the pattern to check what is following. E.g. for Javascript, if a function is defined with "func_name = function(args)":

```
^\s*\ze|i+\s*=\s*function(
```

If the function is defined with "func_name : function() {...":

```
^\s*\ze|i+\s*[:]\s*(function\s*(
```

When using the ":set" command, you need to double the backslashes!

To avoid that use `:let` with a single quote string:`

```
let &l:define = '^\s*\ze|k+\s*=\s*function('
```

'delcombine' **'deco'** **'delcombine'** **'deco'** **'nodelcombine'** **'nodeco'**
'delcombine' **'deco'** boolean (default off)
global

If editing Unicode and this option is set, backspace and Normal mode "x" delete each combining character on its own. When it is off (the default) the character along with its combining characters are deleted.

Note: When **'delcombine'** is set "xx" may work differently from "2x"!

This is useful for Arabic, Hebrew and many other languages where one may have combining characters overtop of base characters, and want to remove only the combining ones.

NOTE: This option is reset when **'compatible'** is set.

'dictionary' **'dict'** string (default "")
global or local to buffer `global-local`

List of file names, separated by commas, that are used to lookup words for keyword completion commands `i_CTRL-X_CTRL-K`. Each file should contain a list of words. This can be one word per line, or several words per line, separated by non-keyword characters (white space is preferred). Maximum line length is 510 bytes.

When this option is empty or an entry "spell" is present, and spell checking is enabled, words in the word lists for the currently active **'spellang'** are used. See `spell`.

To include a comma in a file name precede it with a backslash. Spaces after a comma are ignored, otherwise spaces are included in the file name. See `option-backslash` about using backslashes.

This has nothing to do with the `Dictionary` variable type.

Where to find a list of words?

- On FreeBSD, there is the file `/usr/share/dict/words`.
- In the Simtel archive, look in the `"msdos/linguist"` directory.
- In `"miscfiles"` of the GNU collection.

The use of `:set+=` and `:set-=` is preferred when adding or removing directories from the list. This avoids problems when a future version uses another default.

Backticks cannot be used in this option for security reasons.

'diff' 'diff' 'nodiff'
boolean (default off)
local to window
{not available when compiled without the `+diff` feature}
Join the current window in the group of windows that shows differences between files. See `vimdiff`.

'diffexpr' 'dex' 'dex' 'diffexpr'
string (default "")
global
{not available when compiled without the `+diff` feature}
Expression which is evaluated to obtain a diff file (either ed-style or unified-style) from two versions of a file. See `diff-diffexpr`. This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'diffopt' 'dip' 'dip' 'diffopt'
string (default "internal,filler,closeoff")
global
{not available when compiled without the `+diff` feature}
Option settings for diff mode. It can consist of the following items. All are optional. Items must be separated by a comma.

filler Show filler lines, to keep the text synchronized with a window that has inserted lines at the same position. Mostly useful when windows are side-by-side and **'scrollbind'** is set.

context:{n}	Use a context of {n} lines between a change and a fold that contains unchanged lines. When omitted a context of six lines is used. When using zero the context is actually one, since folds require a line in between, also for a deleted line. Set it to a very large value (999999) to disable folding completely. See <code>fold-diff</code> .
iblack	Ignore changes where lines are all blank. Adds the "-B" flag to the "diff" command if <code>'diffexpr'</code> is empty. Check the documentation of the "diff" command for what this does exactly. NOTE: the diff windows will get out of sync, because no differences between blank lines are taken into account.
icase	Ignore changes in case of text. "a" and "A" are considered the same. Adds the "-i" flag to the "diff" command if <code>'diffexpr'</code> is empty.
iwhite	Ignore changes in amount of white space. Adds the "-b" flag to the "diff" command if <code>'diffexpr'</code> is empty. Check the documentation of the "diff" command for what this does exactly. It should ignore adding trailing white space, but not leading white space.
iwhiteall	Ignore all white space changes. Adds the "-w" flag to the "diff" command if <code>'diffexpr'</code> is empty. Check the documentation of the "diff" command for what this does exactly.
iwhiteeol	Ignore white space changes at end of line. Adds the "-Z" flag to the "diff" command if <code>'diffexpr'</code> is empty. Check the documentation of the "diff" command for what this does exactly.
horizontal	Start diff mode with horizontal splits (unless explicitly specified otherwise).
vertical	Start diff mode with vertical splits (unless explicitly specified otherwise).
closeoff	When a window is closed where <code>'diff'</code> is set and there is only one window remaining in the same tab page with <code>'diff'</code> set, execute <code>`:diffoff`</code> in that window. This undoes a <code>`:diffsplit`</code> command.

hiddenoff Do not use diff mode for a buffer when it becomes hidden.

foldcolumn:{n} Set the **'foldcolumn'** option to {n} when starting diff mode. Without this 2 is used.

followwrap Follow the **'wrap'** option and leave as it is.

internal Use the internal diff library. This is ignored when **'diffexpr'** is set. E960
When running out of memory when writing a buffer this item will be ignored for diffs involving that buffer. Set the **'verbose'** option to see when this happens.

indent-heuristic Use the indent heuristic for the internal diff library.

algorithm:{text} Use the specified diff algorithm with the internal diff engine. Currently supported algorithms are:

myers	the default algorithm
minimal	spend extra time to generate the smallest possible diff
patience	patience diff algorithm
histogram	histogram diff algorithm

Examples:

```
:set diffopt=internal,filler,context:4
:set diffopt=
:set diffopt=internal,filler,foldcolumn:3
:set diffopt-=internal " do NOT use the internal diff parser
```

'digraph' **'dg'** **'digraph'** **'dg'** **'nodigraph'** **'nodg'**
boolean (default off)
global
{not available when compiled without the **+digraphs** feature}

Enable the entering of digraphs in Insert mode with {char1} <BS> {char2}. See **digraphs** .

NOTE: This option is reset when **'compatible'** is set.

'directory' **'dir'** **'directory'** **'dir'**
string (default for Amiga: ".,t:",
for Win32: ".,\$TEMP,c:\tmp,c:\temp"
for Unix: ".,~/tmp,/var/tmp,/tmp")
global

List of directory names for the swap file, separated with commas.
Recommended value: ".,~/vimswap/" - this will put the swap file next to the edited file if possible, and in your personal swap directory otherwise. Make sure "~/vimswap/" is only readable for you.

Possible items:

- The swap file will be created in the first directory where this is possible.
- Empty means that no swap file will be used (recovery is impossible!) and no `E303` error will be given.
- A directory "." means to put the swap file in the same directory as the edited file. On Unix, a dot is prepended to the file name, so it doesn't show in a directory listing. On MS-Windows the "hidden" attribute is set and a dot prepended if possible.
- A directory starting with "./" (or ".\" for MS-Windows) means to put the swap file relative to where the edited file is. The leading "." is replaced with the path name of the edited file.
- For Unix and Win32, if a directory ends in two path separators "//", the swap file name will be built from the complete path to the file with all path separators replaced by percent '%' signs (including the colon following the drive letter on Win32). This will ensure file name uniqueness in the preserve directory. On Win32, it is also possible to end with "\\". However, when a separating comma is following, you must use "/", since "\\" will include the comma in the file name. Therefore it is recommended to use '/', instead of '\\'.
 - Spaces after the comma are ignored, other spaces are considered part of the directory name. To have a space at the start of a directory name, precede it with a backslash.
 - To include a comma in a directory name precede it with a backslash.
 - A directory name may end in an ':' or '/'.
 - Environment variables are expanded `:set_env`.
 - Careful with '\' characters, type one before a space, type two to get one in the option (see `option-backslash`), for example:


```
:set dir=c:\\tmp,\\ dir\\,with\\,commas,\\ dir\\ with\\ spaces
```
- For backwards compatibility with `Vim version 3.0` a '>' at the start of the option is removed.

Using "." first in the list is recommended. This means that editing the same file twice will result in a warning. Using "/tmp" on Unix is discouraged: When the system crashes you lose the swap file.

"/var/tmp" is often not cleared when rebooting, thus is a better choice than "/tmp". But others on the computer may be able to see the files, and it can contain a lot of files, your swap files get lost in the crowd. That is why a "tmp" directory in your home directory is tried first.

The use of `:set+=` and `:set-=` is preferred when adding or removing directories from the list. This avoids problems when a future version uses another default.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'display' 'dy' string (default "", set to "truncate" in `defaults.vim`)
global

Change the way text is displayed. This is a comma-separated list of flags:

lastline When included, as much as possible of the last line in a window will be displayed. "@@" is put in the last columns of the last screen line to indicate the

truncate rest of the line is not displayed.
Like "lastline", but "^^^" is displayed in the first column of the last screen line. Overrides "lastline".

uhex Show unprintable characters hexadecimal as <xx> instead of using ^C and ~C.

When neither "lastline" nor "truncate" is included, a last line that doesn't fit is replaced with "@" lines.

The "@" character can be changed by setting the "lastline" item in **'fillchars'**. The character is highlighted with [hl-NonText](#) .

'eadirection' **'ead'** string (default "both")
 global
 ['eadirection'](#) ['ead'](#)
Tells when the **'equalalways'** option applies:
 ver vertically, width of windows is not affected
 hor horizontally, height of windows is not affected
 both width and height of windows is affected

'edcompatible' **'ed'** boolean (default off)
 global
 ['ed'](#) ['edcompatible'](#) ['noed'](#) ['noedcompatible'](#)
Makes the 'g' and 'c' flags of the ":substitute" command to be toggled each time the flag is given. See [complex-change](#) . See also **'gdefault'** option.
Switching this option on may break plugins!
This option is not used in [Vim9](#) script.

'emoji' **'emo'** boolean (default: on)
 global
 ['emoji'](#) ['emo'](#) ['noemoji'](#) ['noemo'](#)
When on all Unicode emoji characters are considered to be full width. This excludes "text emoji" characters, which are normally displayed as single width. Unfortunately there is no good specification for this and it has been determined on trial-and-error basis. Use the [setcellwidths\(\)](#) function to change the behavior.

'encoding' **'enc'** string (default for MS-Windows: "utf-8",
 otherwise: value from \$LANG or "latin1")
 global
 ['encoding'](#) ['enc'](#) E543
Sets the character encoding used inside Vim. It applies to text in the buffers, registers, Strings in expressions, text stored in the viminfo file, etc. It sets the kind of characters which Vim can work with. See [encoding-names](#) for the possible values.

NOTE: Changing this option will not change the encoding of the existing text in Vim. It may cause non-ASCII text to become invalid. It should normally be kept at its default value, or set when Vim starts up. See [multibyte](#) . To reload the menus see [:menutrans](#) .

This option cannot be set from a [modeline](#) . It would most likely corrupt the text.

NOTE: For GTK+ 2 or later, it is highly recommended to set **'encoding'** to "utf-8". Although care has been taken to allow different values of **'encoding'**, "utf-8" is the natural choice for the environment and avoids unnecessary conversion overhead. "utf-8" has not been made the default to prevent different behavior of the GUI and terminal versions, and to avoid changing the encoding of newly created files without your knowledge (in case **'fileencodings'** is empty).

The character encoding of files can be different from **'encoding'**. This is specified with **'fileencoding'**. The conversion is done with `iconv()` or as specified with **'charconvert'**.

If you need to know whether **'encoding'** is a multibyte encoding, you can use:

```
if has("multi_byte_encoding")
```

Normally **'encoding'** will be equal to your current locale. This will be the default if Vim recognizes your environment settings. If **'encoding'** is not set to the current locale, **'termencoding'** must be set to convert typed and displayed text. See [encoding-table](#).

When you set this option, it fires the `EncodingChanged` autocommand event so that you can set up fonts if necessary.

When the option is set, the value is converted to lowercase. Thus you can set it with uppercase values too. Underscores are translated to '-' signs.

When the encoding is recognized, it is changed to the standard name. For example "Latin-1" becomes "latin1", "ISO_88592" becomes "iso-8859-2" and "utf8" becomes "utf-8".

Note: "latin1" is also used when the encoding could not be detected. This only works when editing files in the same encoding! When the actual character set is not latin1, make sure **'fileencoding'** and **'fileencodings'** are empty. When conversion is needed, switch to using utf-8.

When "unicode", "ucs-2" or "ucs-4" is used, Vim internally uses utf-8. You don't notice this while editing, but it does matter for the `viminfo-file`. And Vim expects the terminal to use utf-8 too. Thus setting **'encoding'** to one of these values instead of utf-8 only has effect for encoding used for files when **'fileencoding'** is empty.

When **'encoding'** is set to a Unicode encoding, and **'fileencodings'** was not set yet, the default for **'fileencodings'** is changed.

'endoffile'	'eof'	'endoffile'	'eof'	'noendoffile'	'noeof'
'endoffile'	'eof'	boolean (default off)			
		local to buffer			

Indicates that a **CTRL-Z** character was found at the end of the file when reading it. Normally only happens when **'fileformat'** is "dos". When writing a file and this option is off and the **'binary'** option is on, or **'fixeol'** option is off, no **CTRL-Z** will be written at the

end of the file.
See [eol-and-eof](#) for example settings.

'endofline' 'eol' **'endofline'** **'eol'** **'noendofline'** **'noeol'**
boolean (default on)
local to buffer

When writing a file and this option is off and the **'binary'** option is on, or **'fixeol'** option is off, no `<EOL>` will be written for the last line in the file. This option is automatically set or reset when starting to edit a new file, depending on whether file has an `<EOL>` for the last line in the file. Normally you don't have to set or reset this option.

When **'binary'** is off and **'fixeol'** is on the value is not used when writing the file. When **'binary'** is on or **'fixeol'** is off it is used to remember the presence of a `<EOL>` for the last line in the file, so that when you write the file the situation from the original file can be kept. But you can change it if you want to.

See [eol-and-eof](#) for example settings.

'equalalways' 'ea' **'equalalways'** **'ea'** **'noequalalways'** **'noea'**
boolean (default on)
global

When on, all the windows are automatically made the same size after splitting or closing a window. This also happens the moment the option is switched on. When off, splitting a window will reduce the size of the current window and leave the other windows the same. When closing a window the extra lines are given to the window next to it (depending on **'splitbelow'** and **'splitright'**).

When mixing vertically and horizontally split windows, a minimal size is computed and some windows may be larger if there is room. The **'eadirection'** option tells in which direction the size is affected. Changing the height and width of a window can be avoided by setting **'winfixheight'** and **'winfixwidth'**, respectively.

If a window size is specified when creating a new window sizes are currently not equalized (it's complicated, but may be implemented in the future).

'equalprg' 'ep' **'equalprg'** **'ep'**
string (default "")
global or local to buffer [global-local](#)

External program to use for "=" command. When this option is empty the internal formatting functions are used; either **'lisp'**, **'cindent'** or **'indentexpr'**. When Vim was compiled without internal formatting, the "indent" program is used.

Environment variables are expanded `:set_env` . See [option-backslash](#) about including spaces and backslashes.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'errorbells' 'eb' **'errorbells'** **'eb'** **'noerrorbells'** **'noeb'**
boolean (default off)
global

Ring the bell (beep or screen flash) for error messages. This only makes a difference for error messages, the bell will be used always

for a lot of errors without a message (e.g., hitting `<Esc>` in Normal mode). See `'visualbell'` on how to make the bell behave like a beep, screen flash or do nothing. See `'belloff'` to finetune when to ring the bell.

`'errorfile' 'ef'` `'errorfile' 'ef'`
string (Amiga default: "AztecC.Err",
others: "errors.err")
global
{not available when compiled without the `+quickfix`
feature}
Name of the errorfile for the QuickFix mode (see `:cf`).
When the "-q" command-line argument is used, `'errorfile'` is set to the
following argument. See `-q` .
NOT used for the ":make" command. See `'makeef'` for that.
Environment variables are expanded `:set_env` .
See `option-backslash` about including spaces and backslashes.
This option cannot be set from a `modeline` or in the `sandbox` , for
security reasons.

`'errorformat' 'efm'` `'errorformat' 'efm'`
string (default is very long)
global or local to buffer `global-local`
{not available when compiled without the `+quickfix`
feature}
Scanf-like description of the format for the lines in the error file
(see `errorformat`).

`'esckey' 'ek'` `'esckey' 'ek'` `'noesckey' 'noek'`
boolean (Vim default: on, Vi default: off)
global
Function keys that start with an `<Esc>` are recognized in Insert
mode. When this option is off, the cursor and function keys cannot be
used in Insert mode if they start with an `<Esc>`. The advantage of
this is that the single `<Esc>` is recognized immediately, instead of
after one second. Instead of resetting this option, you might want to
try changing the values for `'timeoutlen'` and `'ttimeoutlen'`. Note that
when `'esckey'` is off, you can still map anything, but the cursor keys
won't work by default.
NOTE: This option is set to the Vi default value when `'compatible'` is
set and to the Vim default value when `'compatible'` is reset.
NOTE: when this option is off then the `modifyOtherKeys` and
`xterm-bracketed-paste` functionality is disabled while in Insert mode
to avoid ending Insert mode with any key that has a modifier.

`'eventignore' 'ei'` `'eventignore' 'ei'`
string (default "")
global
A list of autocommand event names, which are to be ignored.
When set to "all" or when "all" is one of the items, all autocommand
events are ignored, autocommands will not be executed.
Otherwise this is a comma-separated list of event names. Example:
`:set ei=WinEnter,WinLeave`

'expandtab' **'et'** **'expandtab'** **'et'** **'noexpandtab'** **'noet'**
boolean (default off)
local to buffer

In Insert mode: Use the appropriate number of spaces to insert a <Tab>. Spaces are used in indents with the '>' and '<' commands and when **'autoindent'** is on. To insert a real tab when **'expandtab'** is on, use **CTRL-V<Tab>**. See also **:retab** and **ins-expandtab**. This option is reset when the **'paste'** option is set and restored when the **'paste'** option is reset.

NOTE: This option is reset when **'compatible'** is set.

'execr' **'ex'** **'execr'** **'ex'** **'noexecr'** **'noex'**
boolean (default off)
global

Enables the reading of **.vimrc**, **.execr** and **.gvimrc** in the current directory.

Setting this option is a potential security leak. E.g., consider unpacking a package or fetching files from github, a **.vimrc** in there might be a trojan horse. **BETTER NOT SET THIS OPTION!**

Instead, define an autocommand in your **.vimrc** to set options for a matching directory.

If you do switch this option on you should also consider setting the **'secure'** option (see **initialization**).

Also see **.vimrc** and **gui-init**.

This option cannot be set from a **modeline** or in the **sandbox**, for security reasons.

'fileencoding' **'fenc'** **'fileencoding'** **'fenc'** **E213**
string (default: "")
local to buffer

Sets the character encoding for the file of this buffer.

When **'fileencoding'** is different from **'encoding'**, conversion will be done when writing the file. For reading see below.

When **'fileencoding'** is empty, the same value as **'encoding'** will be used (no conversion when reading or writing a file).

No error will be given when the value is set, only when it is used, only when writing a file.

Conversion will also be done when **'encoding'** and **'fileencoding'** are both a Unicode encoding and **'fileencoding'** is not utf-8. That's because internally Unicode is always stored as utf-8.

WARNING: Conversion can cause loss of information! When **'encoding'** is "utf-8" or another Unicode encoding, conversion is most likely done in a way that the reverse conversion results in the same text. When **'encoding'** is not "utf-8" some characters may be lost!

See **'encoding'** for the possible values. Additionally, values may be specified that can be handled by the converter, see **mbyte-conversion**.

When reading a file **'fileencoding'** will be set from **'fileencodings'**.

To read a file in a certain encoding it won't work by setting **'fileencoding'**, use the `++enc` argument. One exception: when **'fileencodings'** is empty the value of **'fileencoding'** is used. For a new file the global value of **'fileencoding'** is used.

Prepending "8bit-" and "2byte-" has no meaning here, they are ignored. When the option is set, the value is converted to lowercase. Thus you can set it with uppercase values too. '_' characters are replaced with '-'. If a name is recognized from the list for **'encoding'**, it is replaced by the standard name. For example "ISO8859-2" becomes "iso-8859-2".

When this option is set, after starting to edit a file, the **'modified'** option is set, because the file would be different when written.

Keep in mind that changing **'fenc'** from a modeline happens AFTER the text has been read, thus it applies to when the file will be written. If you do set **'fenc'** in a modeline, you might want to set **'nomodified'** to avoid not being able to `:"q"`.

This option can not be changed when **'modifiable'** is off.

'fe'
NOTE: Before version 6.0 this option specified the encoding for the whole of Vim, this was a mistake. Now use **'encoding'** instead. The old short name was **'fe'**, which is no longer used.

'fileencodings' **'fencs'** string (default: "ucs-bom",
"ucs-bom,utf-8,default,latin1" when
'encoding' is set to a Unicode value)
global

This is a list of character encodings considered when starting to edit an existing file. When a file is read, Vim tries to use the first mentioned character encoding. If an error is detected, the next one in the list is tried. When an encoding is found that works, **'fileencoding'** is set to it. If all fail, **'fileencoding'** is set to an empty string, which means the value of **'encoding'** is used.

WARNING: Conversion can cause loss of information! When **'encoding'** is "utf-8" (or one of the other Unicode variants) conversion is most likely done in a way that the reverse conversion results in the same text. When **'encoding'** is not "utf-8" some non-ASCII characters may be lost! You can use the `++bad` argument to specify what is done with characters that can't be converted.

For an empty file or a file with only ASCII characters most encodings will work and the first entry of **'fileencodings'** will be used (except "ucs-bom", which requires the BOM to be present). If you prefer another encoding use an BufReadPost autocommand event to test if your preferred encoding is to be used. Example:

```
au BufReadPost * if search('\S', 'w') == 0 |  
  \ set fenc=iso-2022-jp | endif
```

This sets **'fileencoding'** to "iso-2022-jp" if the file does not contain non-blank characters.

When the `++enc` argument is used then the value of `'fileencodings'` is not used.

Note that `'fileencodings'` is not used for a new file, the global value of `'fileencoding'` is used instead. You can set it with:

```
:setglobal fenc=iso-8859-2
```

This means that a non-existing file may get a different encoding than an empty file.

The special value `"ucs-bom"` can be used to check for a Unicode BOM (Byte Order Mark) at the start of the file. It must not be preceded by `"utf-8"` or another Unicode encoding for this to work properly. An entry for an 8-bit encoding (e.g., `"latin1"`) should be the last, because Vim cannot detect an error, thus the encoding is always accepted.

The special value `"default"` can be used for the encoding from the environment. On MS-Windows this is the system encoding. Otherwise this is the default value for `'encoding'`. It is useful when `'encoding'` is set to `"utf-8"` and your environment uses a non-`latin1` encoding, such as Russian.

When `'encoding'` is `"utf-8"` and a file contains an illegal byte sequence it won't be recognized as UTF-8. You can use the `8g8` command to find the illegal byte sequence.

WRONG VALUES:

```
latin1,utf-8
utf-8,ucs-bom,latin1
cp1250,latin1
```

WHAT'S WRONG:

```
"latin1" will always be used
BOM won't be recognized in an utf-8
file
"cp1250" will always be used
```

If `'fileencodings'` is empty, `'fileencoding'` is not modified.

See `'fileencoding'` for the possible values.

Setting this option does not have an effect until the next time a file is read.

```
'fileformat' 'ff'          'fileformat' 'ff'
                           string (MS-Windows default: "dos",
                               Unix, macOS default: "unix")
                           local to buffer
```

This gives the `<EOL>` of the current buffer, which is used for reading/writing the buffer from/to a file:

```
dos      <CR><NL>
unix     <NL>
mac      <CR>
```

When `"dos"` is used, **CTRL-Z** at the end of a file is ignored.

See `file-formats` and `file-read`.

For the character encoding of the file see `'fileencoding'`.

When `'binary'` is set, the value of `'fileformat'` is ignored, file I/O works like it was set to `"unix"`.

This option is set automatically when starting to edit a file and `'fileformats'` is not empty and `'binary'` is off.

When this option is set, after starting to edit a file, the `'modified'` option is set, because the file would be different when written.

This option can not be changed when `'modifiable'` is off.

For backwards compatibility: When this option is set to `"dos"`, `'textmode'` is set, otherwise `'textmode'` is reset.

```
'fileformats' 'ffs'
```

```
'fileformats' 'ffs'      string (default:
                        Vim+Vi  MS-Windows: "dos,unix",
                        Vim      Unix, macOS: "unix,dos",
                        Vi       Cygwin: "unix,dos",
                        Vi       others: "")
                        global
```

This gives the end-of-line (<EOL>) formats that will be tried when starting to edit a new buffer and when reading a file into an existing buffer:

- When empty, the format defined with **'fileformat'** will be used always. It is not set automatically.
- When set to one name, that format will be used whenever a new buffer is opened. **'fileformat'** is set accordingly for that buffer. The **'fileformats'** name will be used when a file is read into an existing buffer, no matter what **'fileformat'** for that buffer is set to.
- When more than one name is present, separated by commas, automatic <EOL> detection will be done when reading a file. When starting to edit a file, a check is done for the <EOL>:
 1. If all lines end in <CR><NL>, and **'fileformats'** includes "dos", **'fileformat'** is set to "dos".
 2. If a <NL> is found and **'fileformats'** includes "unix", **'fileformat'** is set to "unix". **Note** that when a <NL> is found without a preceding <CR>, "unix" is preferred over "dos".
 3. If **'fileformat'** has not yet been set, and if a <CR> is found, and if **'fileformats'** includes "mac", **'fileformat'** is set to "mac". This means that "mac" is only chosen when:
 - "unix" is not present or no <NL> is found in the file, and
 - "dos" is not present or no <CR><NL> is found in the file.
 Except: if "unix" was chosen, but there is a <CR> before the first <NL>, and there appear to be more <CR>s than <NL>s in the first few lines, "mac" is used.
 4. If **'fileformat'** is still not set, the first name from **'fileformats'** is used.

When reading a file into an existing buffer, the same is done, but this happens like **'fileformat'** has been set appropriately for that file only, the option is not changed.

When **'binary'** is set, the value of **'fileformats'** is not used.

When Vim starts up with an empty buffer the first item is used. You can overrule this by setting **'fileformat'** in your .vimrc.

For systems with a Dos-like <EOL> (<CR><NL>), when reading files that are ":source"ed and for vimrc files, automatic <EOL> detection may be done:

- When **'fileformats'** is empty, there is no automatic detection. Dos format will be used.
- When **'fileformats'** is set to one or more names, automatic detection is done. This is based on the first <NL> in the file: If there is a <CR> in front of it, Dos format is used, otherwise Unix format is used.

Also see [file-formats](#) .

For backwards compatibility: When this option is set to an empty string or one format (no comma is included), **'textauto'** is reset, otherwise **'textauto'** is set.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'fileignorecase' 'fic' **'noignorecase' 'nofic'**
boolean (default on for systems where case in file names is normally ignored)
global

When set case is ignored when using file names and directories.
See **'wildignorecase'** for only ignoring case when doing completion.

'filetype' 'ft' **'filetype' 'ft'**
string (default: "")
local to buffer **local-noglobal**

When this option is set, the FileType autocommand event is triggered. All autocommands that match with the value of this option will be executed. Thus the value of **'filetype'** is used in place of the file name.

Otherwise this option does not always reflect the current file type. This option is normally set when the file type is detected. To enable this use the **":filetype on"** command. **:filetype**

Setting this option to a different value is most useful in a modeline, for a file for which the file type is not automatically recognized.

Example, for in an IDL file:

```
/* vim: set filetype=idl : */
```

```
FileType filetypes
```

When a dot appears in the value then this separates two filetype names. Example:

```
/* vim: set filetype=c.doxygen : */
```

This will use the "c" filetype first, then the "doxygen" filetype.

This works both for filetype plugins and for syntax files. More than one dot may appear.

This option is not copied to another buffer, independent of the 's' or 'S' flag in **'coptions'**.

Only normal file name characters can be used, **"/*?[]<>"** are illegal.

'fillchars' 'fcs' **'fillchars' 'fcs'**
string (default "vert:|,fold:-,eob:~")
global or local to window **global-local**

Characters to fill the statuslines, vertical separators and special lines in the window.

It is a comma-separated list of items. Each item has a name, a colon and the value of that item: **E1511**

item name	default	Used for
stl	' '	statusline of the current window
stlnc	' '	statusline of the non-current windows
vert	' '	vertical separators :vsplit
fold	'-'	filling 'foldtext'
foldopen	'-'	mark the beginning of a fold
foldclose	'+'	show a closed fold
foldsep	' '	open fold middle character
diff	'-'	deleted lines of the 'diff' option
eob	'~'	empty lines below the end of a buffer
lastline	'@'	'display' contains lastline/truncate

Any one that is omitted will fall back to the default.

Example:

```
:set fillchars=stl:\ ,stlnc:\ ,vert:|,fold:-,diff:-
```

For the "stl", "stlnc", "foldopen", "foldclose" and "foldsep" items single-byte and multibyte characters are supported. But double-width characters are not supported. [E1512](#)

The highlighting used for these items:

item name	highlight group	
stl	StatusLine	hl-StatusLine
stlnc	StatusLineNC	hl-StatusLineNC
vert	VertSplit	hl-VertSplit
fold	Folded	hl-Folded
diff	DiffDelete	hl-DiffDelete
eob	EndOfBuffer	hl-EndOfBuffer
lastline	NonText	hl-NonText

```
'fixendofline' 'fixeol' 'nofixendofline' 'nofixeol'  
'fixendofline' 'fixeol' boolean (default on)  
local to buffer
```

When writing a file and this option is on, `<EOL>` at the end of file will be restored if missing. Turn this option off if you want to preserve the situation from the original file.

When the `'binary'` option is set the value of this option doesn't matter.

See the `'endofline'` option.

See [eol-and-eof](#) for example settings.

```
'fkmap' 'fk' 'fkmap' 'fk' 'nofkmap' 'nofk'  
'fkmap' 'fk' boolean (default off)  
global  
{only available when compiled with the +rightleft  
feature}
```

This option was for using Farsi, which has been removed. See [farsi.txt](#).

```
'foldclose' 'fcl' 'foldclose' 'fcl'  
'foldclose' 'fcl' string (default "")  
global  
{not available when compiled without the +folding  
feature}
```

When set to "all", a fold is closed when the cursor isn't in it and its level is higher than `'foldlevel'`. Useful if you want folds to automatically close when moving out of them.

```
'foldcolumn' 'fdc' 'foldcolumn' 'fdc'  
'foldcolumn' 'fdc' number (default 0)  
local to window  
{not available when compiled without the +folding  
feature}
```

When non-zero, a column with the specified width is shown at the side

of the window which indicates open and closed folds. The maximum value is 12.
See [folding](#) .

'foldenable' **'fen'** **'foldenable'** **'fen'** **'nofoldenable'** **'nofen'**
boolean (default on)
local to window
{not available when compiled without the [+folding](#) feature}

When off, all folds are open. This option can be used to quickly switch between showing all text unfolded and viewing the text with folds (including manually opened or closed folds). It can be toggled with the [zi](#) command. The **'foldcolumn'** will remain blank when **'foldenable'** is off.

This option is set by commands that create a new fold or close a fold.
See [folding](#) .

'foldexpr' **'fde'** **'foldexpr'** **'fde'**
string (default: "0")
local to window
{not available when compiled without the [+folding](#) or [+eval](#) features}

The expression used for when **'foldmethod'** is "expr". It is evaluated for each line to obtain its fold level. The context is set to the script where **'foldexpr'** was set, script-local items can be accessed.
See [fold-expr](#) for the usage.

The expression will be evaluated in the [sandbox](#) if set from a modeline, see [sandbox-option](#) .

This option can't be set from a [modeline](#) when the **'diff'** option is on or the **'modelineexpr'** option is off.

It is not allowed to change text or jump to another window while evaluating **'foldexpr'** [textlock](#) .

'foldignore' **'fdi'** **'foldignore'** **'fdi'**
string (default: "#")
local to window
{not available when compiled without the [+folding](#) feature}

Used only when **'foldmethod'** is "indent". Lines starting with characters in **'foldignore'** will get their fold level from surrounding lines. White space is skipped before checking for this character. The default "#" works well for C programs. See [fold-indent](#) .

'foldlevel' **'fdl'** **'foldlevel'** **'fdl'**
number (default: 0)
local to window
{not available when compiled without the [+folding](#) feature}

Sets the fold level: Folds with a higher level will be closed. Setting this option to zero will close all folds. Higher numbers will close fewer folds.

This option is set by commands like [zm](#) , [zM](#) and [zR](#) .

See [fold-foldlevel](#) .

'foldlevelstart' **'fdls'** number (default: -1) **'foldlevelstart'** **'fdls'**
global
{not available when compiled without the [+folding](#)
feature}

Sets **'foldlevel'** when starting to edit another buffer in a window. Useful to always start editing with all folds closed (value zero), some folds closed (one) or no folds closed (99).

This is done before reading any modeline, thus a setting in a modeline overrules this option. Starting to edit a file for [diff-mode](#) also ignores this option and closes all folds.

It is also done before BufReadPre autocommands, to allow an autocmd to overrule the **'foldlevel'** value for specific files.

When the value is negative, it is not used.

'foldmarker' **'fmr'** string (default: "{{{,}}}") **'foldmarker'** **'fmr'** E536
local to window
{not available when compiled without the [+folding](#)
feature}

The start and end marker used when **'foldmethod'** is "marker". There must be one comma, which separates the start and end marker. The marker is a literal string (a regular expression would be too slow). See [fold-marker](#) .

'foldmethod' **'fdm'** string (default: "manual") **'foldmethod'** **'fdm'**
local to window
{not available when compiled without the [+folding](#)
feature}

The kind of folding used for the current window. Possible values:

fold-manual	manual	Folds are created manually.
fold-indent	indent	Lines with equal indent form a fold.
fold-expr	expr	'foldexpr' gives the fold level of a line.
fold-marker	marker	Markers are used to specify folds.
fold-syntax	syntax	Syntax highlighting items specify folds.
fold-diff	diff	Fold text that is not changed.

'foldminlines' **'fml'** number (default: 1) **'foldminlines'** **'fml'**
local to window
{not available when compiled without the [+folding](#)
feature}

Sets the number of screen lines above which a fold can be displayed closed. Also for manually closed folds. With the default value of one a fold can only be closed if it takes up two or more screen lines. Set to zero to be able to close folds of just one screen line.

Note that this only has an effect on what is displayed. After using "zc" to close a fold, which is displayed open because it's smaller than **'foldminlines'**, a following "zc" may close a containing fold.

'foldnestmax' **'fdn'**

'foldnestmax' 'fdn' number (default: 20)
 local to window
 {not available when compiled without the **+folding**
 feature}

Sets the maximum nesting of folds for the "indent" and "syntax" methods. This avoids that too many folds will be created. Using more than 20 doesn't work, because the internal limit is 20.

'foldopen' 'fdo' **'foldopen' 'fdo'**
 string (default: "block,hor,mark,percent,quickfix,
 search,tag,undo")
 global
 {not available when compiled without the **+folding**
 feature}

Specifies for which type of commands folds will be opened, if the command moves the cursor into a closed fold. It is a comma-separated list of items.

NOTE: When the command is part of a mapping this option is not used. Add the **zv** command to the mapping to get the same effect. (rationale: the mapping may want to control opening folds itself)

item	commands
all	any
block	"(", "{", "[[", "[{", etc.
hor	horizontal movements: "l", "w", "fx", etc.
insert	any command in Insert mode
jump	far jumps: "G", "gg", etc.
mark	jumping to a mark: "'m", CTRL-O , etc.
percent	"%"
quickfix	":cn", ":crew", ":make", etc.
search	search for a pattern: "/", "n", "*", "gd", etc. (not for a search pattern in a ":" command) Also for [s and]s .
tag	jumping to a tag: ":ta", CTRL-T , etc.
undo	undo or redo: "u" and CTRL-R

When a movement command is used for an operator (e.g., "dl" or "y%") this option is not used. This means the operator will include the whole closed fold.

Note that vertical movements are not here, because it would make it very difficult to move onto a closed fold.

In insert mode the folds containing the cursor will always be open when text is inserted.

To close folds you can re-apply **'foldlevel'** with the **zx** command or set the **'foldclose'** option to "all".

'foldtext' 'fdt' **'foldtext' 'fdt'**
 string (default: "foldtext()")
 local to window
 {not available when compiled without the **+folding**
 feature}

An expression which is used to specify the text displayed for a closed fold. The context is set to the script where **'foldexpr'** was set, script-local items can be accessed. See **fold-foldtext** for the usage.

The expression will be evaluated in the `sandbox` if set from a modeline, see `sandbox-option` . This option cannot be set in a modeline when `'modelineexpr'` is off.

It is not allowed to change text or jump to another window while evaluating `'foldtext'` `textlock` .

`'formatexpr'` `'fex'` string (default `""`)
local to buffer
{not available when compiled without the `+eval` feature}

Expression which is evaluated to format a range of lines for the `gq` operator or automatic formatting (see `'formatoptions'`). When this option is empty `'formatprg'` is used.

The `v:lnum` variable holds the first line to be formatted.
The `v:count` variable holds the number of lines to be formatted.
The `v:char` variable holds the character that is going to be inserted if the expression is being evaluated due to automatic formatting. This can be empty. Don't insert it yet!

Example:

```
:set formatexpr=mylang#Format()
```

This will invoke the `mylang#Format()` function in the `autoload/mylang.vim` file in `'runtimepath'`. `autoload`

The advantage of using a function call without arguments is that it is faster, see `expr-option-function` .

The expression is also evaluated when `'textwidth'` is set and adding text beyond that limit. This happens under the same conditions as when internal formatting is used. Make sure the cursor is kept in the same spot relative to the text then! The `mode()` function will return `"i"` or `"R"` in this situation.

When the expression evaluates to non-zero Vim will fall back to using the internal format mechanism.

If the expression starts with `s:` or `<SID>` , then it is replaced with the script ID (`local-function`). Example:

```
set formatexpr=s:MyFormatExpr()  
set formatexpr=<SID>SomeFormatExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

The expression will be evaluated in the `sandbox` when set from a modeline, see `sandbox-option` . That stops the option from working, since changing the buffer text is not allowed.

This option cannot be set in a modeline when `'modelineexpr'` is off.

NOTE: This option is set to `""` when `'compatible'` is set.

'formatlistpat' **'flp'** string (default: `^\s*\d\+([\]:.])\t]\s*`)
local to buffer

A pattern that is used to recognize a list header. This is used for the "n" flag in **'formatoptions'**.

The pattern must match exactly the text that will be the indent for the line below it. You can use `/\ze` to mark the end of the match while still checking more characters. There must be a character following the pattern, when it matches the whole line it is handled like there is no match.

The default recognizes a number, followed by an optional punctuation character and white space.

'formatoptions' **'fo'** string (Vim default: `"tcq"`, Vi default: `"vt"`)
local to buffer

This is a sequence of letters which describes how automatic formatting is to be done.

See [fo-table](#) for possible values and [gq](#) for how to format text. When the **'paste'** option is on, no formatting is done (like **'formatoptions'** is empty). Commas can be inserted for readability. To avoid problems with flags that are added in the future, use the `"+="` and `"-="` feature of `:"set"` [add-option-flags](#) .

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'formatprg' **'fp'** string (default `""`)
global or local to buffer [global-local](#)

The name of an external program that will be used to format the lines selected with the [gq](#) operator. The program must take the input on stdin and produce the output on stdout. The Unix program `"fmt"` is such a program.

If the **'formatexpr'** option is not empty it will be used instead. Otherwise, if **'formatprg'** option is an empty string, the internal format function will be used [C-indenting](#) .

Environment variables are expanded [:set_env](#) . See [option-backslash](#) about including spaces and backslashes.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'fsync' **'fs'** boolean (default on)
global

When on, the library function `fsync()` will be called after writing a file. This will flush a file to disk, ensuring that it is safely written even on filesystems which do metadata-only journaling. This will force the harddrive to spin up on Linux systems running in laptop mode, so it may be undesirable in some situations. Be warned that turning this off increases the chances of data loss after a crash. On systems without an `fsync()` implementation, this variable is always off.

Also see **'swapsync'** for controlling `fsync()` on swap files.

'fsync' also applies to [writefile\(\)](#) (unless a flag is used to

override it) and when writing undo files (see [undo-persistence](#)). This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'gdefault' 'gd' 'gdefault' 'gd' 'nogdefault' 'nogd'
boolean (default off)
global

When on, the ":substitute" flag 'g' is default on. This means that all matches in a line are substituted instead of one. When a 'g' flag is given to a ":substitute" command, this will toggle the substitution of all or one match. See [complex-change](#) .

command	'gdefault' on	'gdefault' off
:s///	subst. all	subst. one
:s///g	subst. one	subst. all
:s///gg	subst. all	subst. one

NOTE: This option is reset when **'compatible'** is set. Setting this option may break plugins that rely on the default behavior of the 'g' flag. This will also make the 'g' flag have the opposite effect of that documented in [:s_g](#) . This option is not used in [Vim9](#) script.

'grepformat' 'gfm' 'grepformat' 'gfm'
string (default "%f:%l:%m,%f:%l%m,%f %l%m")
global

Format to recognize for the ":grep" command output. This is a scanf-like string that uses the same format as the **'errorformat'** option: see [errorformat](#) .

'grepprg' 'gp' 'grepprg' 'gp'
string (default "grep -n ",
Unix: "grep -n \$* /dev/null",
Win32: "findstr /n" or "grep -n",
VMS: "SEARCH/NUMBERS ")
global or local to buffer [global-local](#)

Program to use for the [:grep](#) command. This option may contain '%' and '#' characters, which are expanded like when used in a command-line. The placeholder "\$*" is allowed to specify where the arguments will be included. Environment variables are expanded [:set_env](#) . See [option-backslash](#) about including spaces and backslashes.

When your "grep" accepts the "-H" argument, use this to make ":grep" also work well with a single file:

```
:set grepprg=grep\ -nH
```

Special value: When **'grepprg'** is set to "internal" the [:grep](#) command works like [:vimgrep](#) , [:lgrep](#) like [:lvimgrep](#) , [:grepadd](#) like [:vimgrepadd](#) and [:lgrepadd](#) like [:lvimgrepadd](#) .

See also the section [:make_makeprg](#) , since most of the comments there apply equally to **'grepprg'** .

For Win32, the default is "findstr /n" if "findstr.exe" can be found, otherwise it's "grep -n".

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

```

'guicursor' 'gcr'          'guicursor' 'gcr'  E545  E546  E548  E549
string (default "n-v-c:block-Cursor/lCursor,
                    ve:ver35-Cursor,
                    o:hor50-Cursor,
                    i-ci:ver25-Cursor/lCursor,
                    r-cr:hor20-Cursor/lCursor,
                    sm:block-Cursor
                    -blinkwait175-blinkoff150-blinkon175",
for Win32 console:
                    "n-v-c:block,o:hor50,i-ci:hor15,
                    r-cr:hor30,sm:block")

global
{only available when compiled with GUI enabled, and
for Win32 console}

```

This option tells Vim what the cursor should look like in different modes. It fully works in the GUI. In a Win32 console, only the height of the cursor can be changed. This can be done by specifying a block cursor, or a percentage for a vertical or horizontal cursor. For a console the `'t_SI'`, `'t_SR'`, and `'t_EI'` escape sequences are used.

The option is a comma-separated list of parts. Each part consist of a mode-list and an argument-list:

```
mode-list:argument-list,mode-list:argument-list,..
```

The mode-list is a dash separated list of these modes:

```

n      Normal mode
v      Visual mode
ve     Visual mode with 'selection' "exclusive" (same as 'v',
      if not specified)
o      Operator-pending mode
i      Insert mode
r      Replace mode
c      Command-line Normal (append) mode
ci     Command-line Insert mode
cr     Command-line Replace mode
sm     showmatch in Insert mode
a      all modes

```

The argument-list is a dash separated list of these arguments:

```

hor{N} horizontal bar, {N} percent of the character height
ver{N} vertical bar, {N} percent of the character width
block  block cursor, fills the whole character
      [only one of the above three should be present]
blinkwait{N}
blinkon{N}
blinkoff{N}

```

blink times for cursor: blinkwait is the delay before the cursor starts blinking, blinkon is the time that the cursor is shown and blinkoff is the time that the cursor is not shown. The times are in msec. When one of the numbers is zero, there is no blinking. The default is: "blinkwait700-blinkon400-blinkoff250". These numbers are used for a missing entry. This means that blinking is enabled by default. To switch blinking off you can use "blinkon0". The cursor only

blinks when Vim is waiting for input, not while executing a command.

To make the cursor blink in an xterm, see [xterm-blink](#) .

{group-name}

a highlight group name, that sets the color and font for the cursor

{group-name}/{group-name}

Two highlight group names, the first is used when no language mappings are used, the other when they are. [language-mapping](#)

Examples of parts:

`n-c-v:block-nCursor` in Normal, Command-line and Visual mode, use a block cursor with colors from the "nCursor" highlight group

`i-ci:ver30-iCursor-blinkwait300-blinkon200-blinkoff150`

In Insert and Command-line Insert mode, use a 30% vertical bar cursor with colors from the "iCursor" highlight group. Blink a bit faster.

The 'a' mode is different. It will set the given argument-list for all modes. It does not reset anything to defaults. This can be used to do a common setting for all modes. For example, to switch off blinking: `"a:blinkon0"`

Examples of cursor highlighting:

```
:highlight Cursor gui=reverse guifg=NONE guibg=NONE
```

```
:highlight Cursor gui=NONE guifg=bg guibg=fg
```

'guifont' **'gfn'**
E235 E596

'guifont' **'gfn'** string (default "")
global
{only available when compiled with GUI enabled}

This is a list of fonts which will be used for the GUI version of Vim. In its simplest form the value is just one font name.

See [gui-font](#) for the details.

'guifontset' **'gfs'**
E250 E252 E234 E597 E598

'guifontset' **'gfs'** string (default "")
global
{only available when compiled with GUI enabled and with the `+xfontset` feature}
{not available in the GTK+ GUI}

When not empty, specifies two (or more) fonts to be used. The first one for normal English, the second one for your special language. See [xfontset](#) .

'guifontwide' **'gfw'** **'guifontwide'** **'gfw'** E231 E533 E534
string (default "")
global

`{only available when compiled with GUI enabled}`
When not empty, specifies a comma-separated list of fonts to be used for double-width characters. The first font that can be loaded is used. See `gui-fontwide` .

'guiheadroom' 'ghr' `number` (default 50) `'guiheadroom'` `'ghr'`
global
`{only for GTK and X11 GUI}`

The number of pixels subtracted from the screen height when fitting the GUI window on the screen. Set this before the GUI is started, e.g., in your `gvimrc` file. When zero, the whole screen height will be used by the window. When positive, the specified number of pixel lines will be left for window decorations and other items on the screen. Set it to a negative value to allow windows taller than the screen.

'guiligatures' 'gli' `string` (default "") `'guiligatures'` `'gli'` E1243
global
`{only for GTK and Win32 GUI}`

List of ASCII characters that, when combined together, can create more complex shapes. Each character must be a printable ASCII character with a value in the 32–127 range.

Example:

```
:set guiligatures=!\"#$%&()*+,-./:<=>@[ ]^_ { | ~
```

Changing this option updates screen output immediately. Set it to an empty string to disable ligatures.

'guioptions' 'go' `string` (default "egmrLt" (MS-Windows, "t" is removed in `defaults.vim`), "aegimrLt" (GTK and Motif),) `'guioptions'` `'go'`
global
`{only available when compiled with GUI enabled}`

This option only has an effect in the GUI version of Vim. It is a sequence of letters which describes what components and options of the GUI should be used.

To avoid problems with flags that are added in the future, use the "+=" and "-=" feature of `:set` `add-option-flags` .

Valid characters are as follows:

'!' `'go-!'`
External commands are executed in a terminal window. Without this flag the MS-Windows GUI will open a console window to execute the command. The Unix GUI will simulate a dumb terminal to list the command output. The terminal window will be positioned at the bottom, and grow upwards as needed.

'a' `'go-a'`
Autoselect: If present, then whenever VISUAL mode is started, or the Visual area extended, Vim tries to become the owner of the windowing system's global selection. This means that the

Visually highlighted text is available for pasting into other applications as well as into Vim itself. When the Visual mode ends, possibly due to an operation on the text, or when an application wants to paste the selection, the highlighted text is automatically yanked into the "*" selection register. Thus the selection is still available for pasting into other applications after the VISUAL mode has ended.

If not present, then Vim won't become the owner of the windowing system's global selection unless explicitly told to by a yank or delete operation for the "*" register. The same applies to the modeless selection.

'P' Like autoselect but using the "+" register instead of the "*" register. 'go-P'

'A' Autoselect for the modeless selection. Like 'a', but only applies to the modeless selection. 'go-A'

'guioptions'	autoselect Visual	autoselect modeless
""	-	-
"a"	yes	yes
"A"	-	yes
"aA"	yes	yes

When using a terminal see the **'clipboard'** option.

'c' Use console dialogs instead of popup dialogs for simple choices. 'go-c'

'd' Use dark theme variant if available. Currently only works for GTK+ GUI. 'go-d'

'e' Add tab pages when indicated with **'showtabline'**. **'guitablabel'** can be used to change the text in the labels. When 'e' is missing a non-GUI tab pages line may be used. The GUI tabs are only supported on some systems, currently GTK, Motif, Mac OS/X, Haiku, and MS-Windows. 'go-e'

'f' Foreground: Don't use fork() to detach the GUI from the shell where it was started. Use this for programs that wait for the editor to finish (e.g., an e-mail program). Alternatively you can use "gvim -f" or ":gui -f" to start the GUI in the foreground. [gui-fork](#) 'go-f'
Note: Set this option in the vimrc file. The forking may have happened already when the [gvimrc](#) file is read.

'i' Use a Vim icon. For GTK with KDE it is used in the left-upper corner of the window. It's black&white on non-GTK, because of limitations of X11. For a color icon, see [X11-icon](#). 'go-i'

'm' Menu bar is present. 'go-m'

'M' The system menu "\$VIMRUNTIME/menu.vim" is not sourced. **Note** 'go-M'

that this flag must be added in the `.vimrc` file, before switching on syntax or filetype recognition (when the `gvimrc` file is sourced the system menu has already been loaded; the ``:syntax on`` and ``:filetype on`` commands load the menu too).

- 'g' Grey menu items: Make menu items that are not active grey. If 'g' is not included inactive menu items are not shown at all. 'go-g'
- 't' Include tearoff menu items. Currently only works for Win32, GTK+, and Motif 1.2 GUI. 'go-t'
- 'T' Include Toolbar. Currently only in Win32, GTK+, Motif and Photon GUIs. 'go-T'
- 'r' Right-hand scrollbar is always present. 'go-r'
- 'R' Right-hand scrollbar is present when there is a vertically split window. 'go-R'
- 'l' Left-hand scrollbar is always present. 'go-l'
- 'L' Left-hand scrollbar is present when there is a vertically split window. 'go-L'
- 'b' Bottom (horizontal) scrollbar is present. Its size depends on the longest visible line, or on the cursor line if the 'h' flag is included. `gui-horiz-scroll` 'go-b'
- 'h' Limit horizontal scrollbar size to the length of the cursor line. Reduces computations. `gui-horiz-scroll` 'go-h'

And yes, you may even have scrollbars on the left AND the right if you really want to :-). See `gui-scrollbars` for more information.

- 'v' Use a vertical button layout for dialogs. When not included, a horizontal layout is preferred, but when it doesn't fit a vertical layout is used anyway. Not supported in GTK 3. 'go-v'
- 'p' Use Pointer callbacks for X11 GUI. This is required for some window managers. If the cursor is not blinking or hollow at the right moment, try adding this flag. This must be done before starting the GUI. Set it in your `gvimrc`. Adding or removing it after the GUI has started has no effect. 'go-p'
- 'F' Add a footer. Only for Motif. See `gui-footer`. 'go-F'
- 'k' Keep the GUI window size when adding/removing a scrollbar, or toolbar, tabline, etc. Instead, the behavior is similar to when the window is maximized and will adjust `'lines'` and `'columns'` to fit to the window. Without the 'k' flag Vim will try to keep `'lines'` and `'columns'` the same when adding and removing GUI components. 'go-k'

'guipty' 'guipty' 'noguipty'
 boolean (default on)
 global
 {only available when compiled with GUI enabled}
 Only in the GUI: If on, an attempt is made to open a pseudo-tty for I/O to/from shell commands. See [gui-pty](#) .

'guitablabel' 'gtl' 'guitablabel' 'gtl'
 string (default empty)
 global
 {only available when compiled with GUI enabled}
 When non-empty describes the text to use in a label of the GUI tab pages line. When empty and when the result is empty Vim will use a default label. See [setting-guitablabel](#) for more info.

The format of this option is like that of **'statusline'**.
'guitabtooltip' is used for the tooltip, see below.
 The expression will be evaluated in the [sandbox](#) when set from a modeline, see [sandbox-option](#) .
 This option cannot be set in a modeline when **'modelineexpr'** is off.
 Only used when the GUI tab pages line is displayed. 'e' must be present in **'guioptions'**. For the non-GUI tab pages line **'tabline'** is used.

'guitabtooltip' 'gtt' 'guitabtooltip' 'gtt'
 string (default empty)
 global
 {only available when compiled with GUI enabled}
 When non-empty describes the text to use in a tooltip for the GUI tab pages line. When empty Vim will use a default tooltip.
 This option is otherwise just like **'guitablabel'** above.
 You can include a line break. Simplest method is to use `:let` :
`:let &guitabtooltip = "line one\nline two"`

'helpfile' 'hf' 'helpfile' 'hf'
 string (default (MS-Windows) "\$VIMRUNTIME\doc\help.txt"
 (others) "\$VIMRUNTIME/doc/help.txt")
 global
 Name of the main help file. All distributed help files should be placed together in one directory. Additionally, all "doc" directories in **'runtimepath'** will be used.
 Environment variables are expanded `:set_env` . For example: "\$VIMRUNTIME/doc/help.txt". If \$VIMRUNTIME is not set, \$VIM is also tried. Also see [\\$VIMRUNTIME](#) and [option-backslash](#) about including spaces and backslashes.
 This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'helpheight' 'hh' 'helpheight' 'hh'
 number (default 20)
 global
 Minimal initial height of the help window when it is opened with the

":help" command. The initial height of the help window is half of the current window, or (when the 'ea' option is on) the same as other windows. When the height is less than 'helpheight', the height is set to 'helpheight'. Set to zero to disable.

'helplang' **'hlg'** 'helplang' 'hlg'
string (default: messages language or empty)
global
{only available when compiled with the +multi_lang feature}

Comma-separated list of languages. Vim will use the first language for which the desired help can be found. The English help will always be used as a last resort. You can add "en" to prefer English over another language, but that will only find tags that exist in that language and not in the English help.

Example:

```
:set helplang=de,it
```

This will first search German, then Italian and finally English help files.

When using CTRL-] and ":help!" in a non-English help file Vim will try to find the tag in the current language before using this option. See [help-translated](#) .

'hidden' **'hid'** 'hidden' 'hid' 'nohidden' 'nohid'
boolean (default off)
global

When off a buffer is unloaded when it is abandoned. When on a buffer becomes hidden when it is abandoned. If the buffer is still displayed in another window, it does not become hidden, of course.

The commands that move through the buffer list sometimes make a buffer hidden even if the 'hidden' option is off when these three are true:

- the buffer is modified
- 'autowrite' is off or writing is not possible
- the '!' flag was used

Also see [windows.txt](#) .

To only make one buffer hidden use the 'bufhidden' option.

This option is set for one command with ":hide {command}" :hide .

WARNING: It's easy to forget that you have changes in hidden buffers. Think twice when using ":q!" or ":qa!".

'highlight' **'hl'** 'highlight' 'hl'
string (default (as a single string):
"8:SpecialKey,~:EndOfBuffer,@:NonText,
d:Directory,e:ErrorMsg,i:IncSearch,
l:Search,m:MoreMsg,M:ModeMsg,n:LineNr,
a:LineNrAbove,b:LineNrBelow,
N:CursorLineNr,r:Question,s:StatusLine,
S:StatusLineNC,c:VertSplit,t:Title,
v:Visual,V:VisualNOS,w:WarningMsg,
W:WildMenu,f:Folded,F:FoldColumn,
A:DiffAdd,C:DiffChange,D:DiffDelete,
T:DiffText,>:SignColumn,-:Conceal,

B:SpellBad,P:SpellCap,R:SpellRare,
L:SpellLocal,+:Pmenu,=:PmenuSel,
[:PmenuKind,]:PmenuKindSel,
{:PmenuExtra,}:PmenuExtraSel,
x:PmenuSbar,X:PmenuThumb,*:TabLine,
#:TabLineSel,_:TabLineFill,!:CursorColumn,
.:CursorLine,o:ColorColumn,q:QuickFixLine,
z:StatusLineTerm,Z:StatusLineTermNC")

global

This option can be used to set highlighting mode for various occasions. It is a comma-separated list of character pairs. The first character in a pair gives the occasion, the second the mode to use for that occasion. The occasions are:

hl-SpecialKey	8	Meta and special keys listed with ":map"
hl-EndOfBuffer	~	lines after the last line in the buffer
hl-NonText	@	'@' at the end of the window and characters from 'showbreak'
hl-Directory	d	directories in CTRL-D listing and other special things in listings
hl-ErrorMsg	e	error messages
	h	(obsolete, ignored)
hl-IncSearch	i	'incsearch' highlighting
hl-CurSearch	y	current instance of last search pattern
hl-Search	l	last search pattern highlighting (see 'hlsearch')
hl-MoreMsg	m	more-prompt
hl-ModeMsg	M	Mode (e.g., "-- INSERT --")
hl-LineNr	n	line number for ":number" and ":#" commands, and when 'number' or 'relativenumber' option is set.
hl-LineNrAbove	a	line number above the cursor for when the 'relativenumber' option is set.
hl-LineNrBelow	b	line number below the cursor for when the 'relativenumber' option is set.
hl-CursorLineNr	N	like n for when 'cursorline' or 'relativenumber' is set.
hl-Question	r	hit-enter prompt and yes/no questions
hl-StatusLine	s	status line of current window status-line
hl-StatusLineNC	S	status lines of not-current windows
hl-Title	t	Titles for output from ":set all", ":autocmd" etc.
hl-VertSplit	c	column used to separate vertically split windows
hl-Visual	v	Visual mode
hl-VisualNOS	V	Visual mode when Vim does is "Not Owing the Selection" Only X11 Gui's gui-x11 and xterm-clipboard .
hl-WarningMsg	w	warning messages
hl-WildMenu	W	wildcard matches displayed for 'wildmenu'
hl-Folded	f	line used for closed folds
hl-FoldColumn	F	'foldcolumn'
hl-DiffAdd	A	added line in diff mode
hl-DiffChange	C	changed line in diff mode
hl-DiffDelete	D	deleted line in diff mode
hl-DiffText	T	inserted text in diff mode
hl-SignColumn	>	column used for signs
hl-Conceal	-	the placeholders used for concealed characters (see 'conceallevel')

```

hl-SpellBad      B  misspelled word  spell
hl-SpellCap      P  word that should start with capital  spell
hl-SpellRare     R  rare word  spell
hl-SpellLocal    L  word from other region  spell
hl-Pmenu         +  popup menu normal line
hl-PmenuSel      =  popup menu selected line
hl-PmenuKind     [  popup menu "kind" normal line
hl-PmenuKindSel  ]  popup menu "kind" selected line
hl-PmenuExtra    {  popup menu "extra" normal line
hl-PmenuExtraSel }  popup menu "extra" selected line
hl-PmenuSbar     x  popup menu scrollbar
hl-PmenuThumb    X  popup menu scrollbar thumb

```

The display modes are:

```

r      reverse          (termcap entry "mr" and "me")
i      italic           (termcap entry "ZH" and "ZR")
b      bold             (termcap entry "md" and "me")
s      standout         (termcap entry "so" and "se")
u      underline        (termcap entry "us" and "ue")
c      undercurl        (termcap entry "Us" and "Ce")
2      double underline (termcap entry "Ds" and "Ce")
d      dotted underline (termcap entry "ds" and "Ce")
=      dashed underline (termcap entry "Ds" and "Ce")
t      strikethrough    (termcap entry "Ts" and "Te")
n      no highlighting
-      no highlighting
:      use a highlight group

```

The default is used for occasions that are not included.

If you want to change what the display modes do, see [dos-colors](#) for an example.

When using the ':' display mode, this must be followed by the name of a highlight group. A highlight group can be used to define any type of highlighting, including using color. See [:highlight](#) on how to define one. The default uses a different group for each occasion. See [highlight-default](#) for the default highlight groups.

```

'history' 'hi'          number (Vim default: 50, Vi default: 0,
                                set to 200 in defaults.vim )
                                global

```

A history of ":" commands, and a history of previous search patterns is remembered. This option decides how many entries may be stored in each of these histories (see [cmdline-editing](#)).

The maximum value is 10000.

NOTE: This option is set to the Vi default value when '**compatible**' is set and to the Vim default value when '**compatible**' is reset.

```

'hkmap' 'hk'          boolean (default off)
                                global
                                {only available when compiled with the +rightleft
                                feature}

```

When on, the keyboard is mapped for the Hebrew character set.

Normally you would set '**allowrevins**' and use CTRL- in insert mode to

toggle this option. See [rileft.txt](#) .

NOTE: This option is reset when **'compatible'** is set.

'hkmapp' 'hkp' **'hkmapp'** **'hkp'** **'nohkmapp'** **'nohkp'**
boolean (default off)
global
{only available when compiled with the [+rightleft](#)
feature}

When on, phonetic keyboard mapping is used. **'hkmap'** must also be on. This is useful if you have a non-Hebrew keyboard.

See [rileft.txt](#) .

NOTE: This option is reset when **'compatible'** is set.

'hlsearch' 'hls' **'hlsearch'** **'hls'** **'nohlsearch'** **'nohls'**
boolean (default off)
global
{not available when compiled without the
[+extra_search](#) feature}

When there is a previous search pattern, highlight all its matches. The type of highlighting used can be set with the 'l' option in the **'highlight'** option. This uses the "Search" highlight group by default. **Note** that only the matching text is highlighted, any offsets are not applied. If the "CurSearch" highlight group is set then the current match is highlighted with that.

See also: **'incsearch'** and [:match](#) .

When you get bored looking at the highlighted matches, you can turn it off with [:nohlsearch](#) . This does not change the option value, as soon as you use a search command, the highlighting comes back.

'redrawtime' specifies the maximum time spent on finding matches.

When the search pattern can match an end-of-line, Vim will try to highlight all of the matched text. However, this depends on where the search starts. This will be the first line in the window or the first line below a closed fold. A match in a previous line which is not drawn may not continue in a newly drawn line.

You can specify whether the highlight status is restored on startup with the 'h' flag in **'viminfo'** [viminfo-h](#) .

NOTE: This option is reset when **'compatible'** is set.

'icon' **'icon'** **'noicon'**
boolean (default off, on when title can be restored)
global

When on, the icon text of the window will be set to the value of **'iconstring'** (if it is not empty), or to the name of the file currently being edited. Only the last part of the name is used. Overridden by the **'iconstring'** option.

Only works if the terminal supports setting window icons (currently only X11 GUI and terminals with a non-empty **'t_IS'** option - these are Unix xterm and iris-ansi by default, where **'t_IS'** is taken from the builtin termcap).

When Vim was compiled with HAVE_X11 defined, the original icon will be restored if possible [X11](#) . See [X11-icon](#) for changing the icon on X11.

For MS-Windows the icon can be changed, see [windows-icon](#) .

'iconstring' 'iconstring'
 string (default "")
 global

When this option is not empty, it will be used for the icon text of the window. This happens only when the **'icon'** option is on. Only works if the terminal supports setting window icon text (currently only X11 GUI and terminals with a non-empty **'t_IS'** option). Does not work for MS-Windows.

When Vim was compiled with HAVE_X11 defined, the original icon will be restored if possible [X11](#) .

When this option contains printf-style '%' items, they will be expanded according to the rules used for **'statusline'**. See **'titlestring'** for example settings.

This option cannot be set in a modeline when **'modelineexpr'** is off.
 {not available when compiled without the |+statusline| feature}

'ignorecase' **'ic'** 'ignorecase' 'ic' 'noignorecase' 'noic'
 boolean (default off)
 global

Ignore case in search patterns, [cmdline-completion](#) , when searching in the tags file, and non- Vim9 [expr-==](#) .

Also see **'smartcase'** and **'tagcase'**.

Can be overruled by using "\c" or "\C" in the pattern, see [/ignorecase](#) .

'imactivatefunc' **'imaf'** 'imactivatefunc' 'imaf'
 string (default "")
 global

This option specifies a function that will be called to activate or deactivate the Input Method. The value can be the name of a function, a [lambda](#) or a [Funcref](#) . See [option-value-function](#) for more information.

It is not used in the MS-Windows GUI version.

The expression will be evaluated in the [sandbox](#) when set from a modeline, see [sandbox-option](#) .

Example:

```
function ImActivateFunc(active)
  if a:active
    ... do something
  else
    ... do something
  endif
  " return value is not used
endfunction
set imactivatefunc=ImActivateFunc
```

'imactivatekey' **'imak'** 'imactivatekey' 'imak'
 string (default "")
 global
 {only available when compiled with [+xim](#) and [+GUI_GTK](#) } [E599](#)

Specifies the key that your Input Method in X-Windows uses for activation. When this is specified correctly, vim can fully control

IM with `'imcmdline'`, `'iminsert'` and `'imsearch'`.

You can't use this option to change the activation key, the option tells Vim what the key is.

Format:

`[MODIFIER_FLAG-]KEY_STRING`

These characters can be used for MODIFIER_FLAG (case is ignored):

S	Shift key
L	Lock key
C	Control key
1	Mod1 key
2	Mod2 key
3	Mod3 key
4	Mod4 key
5	Mod5 key

Combinations are allowed, for example "S-C-space" or "SC-space" are both shift+ctrl+space.

See `<X11/keysymdef.h>` and `XStringToKeysym` for `KEY_STRING`.

Example:

```
:set imactivatekey=S-space
```

"S-space" means shift+space. This is the activation key for `kinput2` + `canna` (Japanese), and `ami` (Korean).

<code>'imcmdline'</code>	<code>'imc'</code>	<code>'imcmdline'</code>	<code>'imc'</code>	<code>'noimcmdline'</code>	<code>'noimc'</code>
<code>'imcmdline'</code>	<code>'imc'</code>	boolean (default off)			
		global			

When set the Input Method is always on when starting to edit a command line, unless entering a search pattern (see `'imsearch'` for that). Setting this option is useful when your input method allows entering English characters directly, e.g., when it's used to type accented characters with dead keys.

<code>'imdisable'</code>	<code>'imd'</code>	<code>'imdisable'</code>	<code>'imd'</code>	<code>'noimdisable'</code>	<code>'noimd'</code>
<code>'imdisable'</code>	<code>'imd'</code>	boolean (default off, on for some systems (SGI))			
		global			

When set the Input Method is never used. This is useful to disable the IM when it doesn't work properly.

Currently this option is on by default for SGI/IRIX machines. This may change in later releases.

<code>'iminsert'</code>	<code>'imi'</code>	<code>'iminsert'</code>	<code>'imi'</code>
<code>'iminsert'</code>	<code>'imi'</code>	number (default 0)	
		local to buffer	

Specifies whether `:lmap` or an Input Method (IM) is to be used in Insert mode. Valid values:

0	:lmap is off and IM is off
1	:lmap is ON and IM is off
2	:lmap is off and IM is ON

To always reset the option to zero when leaving Insert mode with `<Esc>` this can be used:

```
:inoremap <ESC> <ESC>:set iminsert=0<CR>
```

This makes `:lmap` and IM turn off automatically when leaving Insert mode.

Note that this option changes when using **CTRL-^** in Insert mode `i_CTRL-^` .

The value is set to 1 when setting **'keymap'** to a valid keymap name. It is also used for the argument of commands like "r" and "f". The value 0 may not work correctly with Motif with some XIM methods. Use **'imdisable'** to disable XIM then.

You can set **'imactivatefunc'** and **'imstatusfunc'** to handle IME/XIM via external command if Vim is not compiled with the `+xim` , `+multi_byte_ime` or `global-ime` .

'imsearch' **'ims'** **'imsearch'** **'ims'**
number (default -1)
local to buffer

Specifies whether `:lmap` or an Input Method (IM) is to be used when entering a search pattern. Valid values:

- 1 the value of **'iminsert'** is used, makes it look like **'iminsert'** is also used when typing a search pattern
- 0 `:lmap` is off and IM is off
- 1 `:lmap` is ON and IM is off
- 2 `:lmap` is off and IM is ON

Note that this option changes when using **CTRL-^** in Command-line mode `c_CTRL-^` .

The value is set to 1 when it is not -1 and setting the **'keymap'** option to a valid keymap name.

The value 0 may not work correctly with Motif with some XIM methods. Use **'imdisable'** to disable XIM then.

'imstatusfunc' **'imsf'** **'imstatusfunc'** **'imsf'**
string (default "")
global

This option specifies a function that is called to obtain the status of Input Method. It must return a positive number when IME is active. The value can be the name of a function, a `lambda` or a `Funcref` . See [option-value-function](#) for more information. It is not used in the MS-Windows GUI version.

Example:

```
function ImStatusFunc()  
  let is_active = ...do something  
  return is_active ? 1 : 0  
endfunction  
set imstatusfunc=ImStatusFunc
```

NOTE: This function is invoked very often. Keep it fast. The expression will be evaluated in the `sandbox` when set from a modeline, see [sandbox-option](#) .

'imstyle' **'imst'** **'imstyle'** **'imst'**
number (default 1)
global
{only available when compiled with `+xim` and `+GUI_GTK` }

This option specifies the input style of Input Method:

0 use on-the-spot style
1 over-the-spot style
See: [xim-input-style](#)

For a long time on-the-spot style had been used in the GTK version of vim, however, it is known that it causes troubles when using mappings, [single-repeat](#), etc. Therefore over-the-spot style becomes the default now. This should work fine for most people, however if you have any problem with it, try using on-the-spot style. The expression will be evaluated in the [sandbox](#) when set from a modeline, see [sandbox-option](#).

'include' 'inc' ['include'](#) ['inc'](#)
string (default "`^\s*#\s*include`")
global or local to buffer [global-local](#)
{not available when compiled without the
[+find_in_path](#) feature}

Pattern to be used to find an include command. It is a search pattern, just like for the "/" command (See [pattern](#)). The default value is for C programs. This option is used for the commands "[i", "]I", "[d", etc.

Normally the **'isfname'** option is used to recognize the file name that comes after the matched pattern. But if "\zs" appears in the pattern then the text matched from "\zs" to the end, or until "\ze" if it appears, is used as the file name. Use this to include characters that are not in **'isfname'**, such as a space. You can then use **'includeexpr'** to process the matched text.

See [option-backslash](#) about including spaces and backslashes.

'includeexpr' 'inex' ['includeexpr'](#) ['inex'](#)
string (default "")
local to buffer
{not available when compiled without the
[+find_in_path](#) or [+eval](#) features}

Expression to be used to transform the string found with the **'include'** option to a file name. Mostly useful to change "." to "/" for Java:

```
:setlocal includeexpr=substitute(v:fname,'\\.', '/', 'g')
```

The "v:fname" variable will be set to the file name that was detected.

Note the double backslash: the `:set` command first halves them, then one remains it the value, where "\." matches a dot literally. For simple character replacements `tr()` avoids the need for escaping:

```
:setlocal includeexpr=tr(v:fname,'.','/')
```

Also used for the [gf](#) command if an unmodified file name can't be found. Allows doing "gf" on the name after an **'include'** statement. Also used for [<cfile>](#).

If the expression starts with s: or [<SID>](#), then it is replaced with the script ID ([local-function](#)). Example:

```
setlocal includeexpr=s:MyIncludeExpr()  
setlocal includeexpr=<SID>SomeIncludeExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

It is more efficient if the value is just a function call without arguments, see `expr-option-function` .

The expression will be evaluated in the `sandbox` when set from a modeline, see `sandbox-option` .
This option cannot be set in a modeline when `'modelineexpr'` is off.

It is not allowed to change text or jump to another window while evaluating `'includeexpr'` `textlock` .

```
'incsearch' 'is'          'incsearch' 'is' 'noincsearch' 'nois'
                          boolean (default off, set in defaults.vim if the
                              +reltime feature is supported)
                          global
                          {not available when compiled without the
                              +extra_search features}
```

While typing a search command, show where the pattern, as it was typed so far, matches. The matched string is highlighted. If the pattern is invalid or not found, nothing is shown. The screen will be updated often, this is only useful on fast terminals.

Also applies to the pattern in commands:

```
:global
:lvimgrep
:lvimgrepadd
:smagic
:snomagic
:sort
:substitute
:vglobal
:vimgrep
:vimgrepadd
```

Note that the match will be shown, but the cursor will return to its original position when no match is found and when pressing `<Esc>`. You still need to finish the search command with `<Enter>` to move the cursor to the match.

You can use the **CTRL-G** and **CTRL-T** keys to move to the next and previous match. `c_CTRL-G` `c_CTRL-T`

When compiled with the `+reltime` feature Vim only searches for about half a second. With a complicated pattern and/or a lot of text the match may not be found. This is to avoid that Vim hangs while you are typing the pattern.

The highlighting can be set with the `'i'` flag in `'highlight'`.

When `'hlsearch'` is on, all matched strings are highlighted too while typing a search command. See also: `'hlsearch'`.

If you don't want to turn `'hlsearch'` on, but want to highlight all matches while searching, you can turn on and off `'hlsearch'` with `autocmd`. Example:

```
augroup vimrc-incsearch-highlight
autocmd!
autocmd CmdlineEnter /,\? :set hlsearch
autocmd CmdlineLeave /,\? :set nohlsearch
augroup END
```

CTRL-L can be used to add one character from after the current match

to the command line. If `'ignorecase'` and `'smartcase'` are set and the command line has no uppercase characters, the added character is converted to lowercase.

CTRL-R CTRL-W can be used to add the word at the end of the current match, excluding the characters that were already typed.

NOTE: This option is reset when `'compatible'` is set.

```
'indentexpr' 'inde'      string (default "")
                          local to buffer
                          {not available when compiled without the +eval
                          feature}
```

Expression which is evaluated to obtain the proper indent for a line. It is used when a new line is created, for the `=` operator and in Insert mode as specified with the `'indentkeys'` option.

When this option is not empty, it overrules the `'cindent'` and `'smartindent'` indenting. When `'lisp'` is set, this option is only used when `'lispoptions'` contains `"expr:1"`.

When `'paste'` is set this option is not used for indenting.

The expression is evaluated with `v:lnum` set to the line number for which the indent is to be computed. The cursor is also in this line when the expression is evaluated (but it may be moved around).

If the expression starts with `s:` or `<SID>`, then it is replaced with the script ID (`local-function`). Example:

```
set indentexpr=s:MyIndentExpr()
set indentexpr=<SID>SomeIndentExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

The advantage of using a function call without arguments is that it is faster, see `expr-option-function`.

The expression must return the number of spaces worth of indent. It can return `"-1"` to keep the current indent (this means `'autoindent'` is used for the indent).

Functions useful for computing the indent are `indent()`, `cindent()` and `lispindent()`.

The evaluation of the expression must not have side effects! It must not change the text, jump to another window, etc. Afterwards the cursor position is always restored, thus the cursor may be moved.

Normally this option would be set to call a function:

```
:set indentexpr=GetMyIndent()
```

Error messages will be suppressed, unless the `'debug'` option contains `"msg"`.

See `indent-expression`.

NOTE: This option is set to `""` when `'compatible'` is set.

The expression will be evaluated in the `sandbox` when set from a modeline, see `sandbox-option`.

This option cannot be set in a modeline when `'modelineexpr'` is off.

It is not allowed to change text or jump to another window while evaluating `'indentexpr'` `textlock`.

'indentkeys' **'indk'** string (default "0{,0},0),0],:,0#,!^F,o,0,e")
local to buffer

A list of keys that, when typed in Insert mode, cause reindenting of the current line. Only happens if **'indentexpr'** isn't empty. The format is identical to **'cinkeys'**, see [indentkeys-format](#). See [C-indenting](#) and [indent-expression](#).

'infercase' **'inf'** boolean (default off)
local to buffer

When doing keyword completion in insert mode [ins-completion](#), and **'ignorecase'** is also on, the case of the match is adjusted depending on the typed text. If the typed text contains a lowercase letter where the match has an upper case letter, the completed part is made lowercase. If the typed text has no lowercase letters and the match has a lowercase letter where the typed text has an uppercase letter, and there is a letter before it, the completed part is made uppercase. With **'noinfercase'** the match is used as-is.

'insertmode' **'im'** boolean (default off)
global

Makes Vim work in a way that Insert mode is the default mode. Useful if you want to use Vim as a modeless editor. Used for [evim](#).

These Insert mode commands will be useful:

- Use the cursor keys to move around.
- Use **CTRL-O** to execute one Normal mode command [i_CTRL-O](#). When this is a mapping, it is executed as if **'insertmode'** was off. Normal mode remains active until the mapping is finished.
- Use **CTRL-L** to execute a number of Normal mode commands, then use [<Esc>](#) to get back to Insert mode. Note that **CTRL-L** moves the cursor left, like [<Esc>](#) does when **'insertmode'** isn't set. [i_CTRL-L](#)

These items change when **'insertmode'** is set:

- when starting to edit of a file, Vim goes to Insert mode.
- [<Esc>](#) in Insert mode is a no-op and beeps.
- [<Esc>](#) in Normal mode makes Vim go to Insert mode.
- **CTRL-L** in Insert mode is a command, it is not inserted.
- **CTRL-Z** in Insert mode suspends Vim, see [CTRL-Z](#). [i_CTRL-Z](#)

However, when [<Esc>](#) is used inside a mapping, it behaves like **'insertmode'** was not set. This was done to be able to use the same mappings with **'insertmode'** set or not set.

When executing commands with [:normal](#) **'insertmode'** is not used.

NOTE: This option is reset when **'compatible'** is set.

'isfname' **'isf'** string (default for Win32:
"@,48-57,/,\,.,-,_+,,#,\$,%,{,},[,],:,@-@,!,~,=""
for AMIGA: "@,48-57,/.,-,_+,,\$,:""
for VMS: "@,48-57,/.,-,_+,,#,\$,%,<,>[,],:,;~,~"
for OS/390: "@,240-249,/.,-,_+,,#,\$,%,~,=""

```
otherwise: "@,48-57,/,. - ,+ , , # , $ , % , ~ , =" )
global
```

The characters specified by this option are included in file names and path names. Filenames are used for commands like "gf", "[i" and in the tags file. It is also used for "\f" in a [pattern](#). Multi-byte characters 256 and above are always included, only the characters up to 255 are specified with this option. For UTF-8 the characters 0xa0 to 0xff are included as well. Think twice before adding white space to this option. Although a space may appear inside a file name, the effect will be that Vim doesn't know where a file name starts or ends when doing completion. It most likely works better without a space in **'isfname'**.

Note that on systems using a backslash as path separator, Vim tries to do its best to make it work as you would expect. That is a bit tricky, since Vi originally used the backslash to escape special characters. Vim will not remove a backslash in front of a normal file name character on these systems, but it will on Unix and alikes. The '&' and '^' are not included by default, because these are special for cmd.exe.

The format of this option is a list of parts, separated with commas. Each part can be a single character number or a range. A range is two character numbers with '-' in between. A character number can be a decimal number between 0 and 255 or the ASCII character itself (does not work for digits). Example:

```
"_,-,128-140,#-43" (include '_' and '-' and the range
                    128 to 140 and '#' to 43)
```

If a part starts with '^', the following character number or range will be excluded from the option. The option is interpreted from left to right. Put the excluded character after the range where it is included. To include '^' itself use it as the last character of the option or the end of a range. Example:

```
"^a-z,#,^" (exclude 'a' to 'z', include '#' and '^')
```

If the character is '@', all characters where isalpha() returns TRUE are included. Normally these are the characters a to z and A to Z, plus accented characters. To include '@' itself use "@-@". Examples:

```
"@,^a-z" All alphabetic characters, excluding lower
          case ASCII letters.
```

```
"a-z,A-Z,@-@" All letters plus the '@' character.
```

A comma can be included by using it where a character number is expected. Example:

```
"48-57,,,_ " Digits, comma and underscore.
```

A comma can be excluded by prepending a '^'. Example:

```
" ~,^,9" All characters from space to '~', excluding
          comma, plus <Tab>.
```

See [option-backslash](#) about including spaces and backslashes.

```
'isident' 'isi' string (default for Win32:
                    'isident' 'isi'
                    "@,48-57,_,128-167,224-235"
                    otherwise: "@,48-57,_,192-255")
                    global
```

The characters given by this option are included in identifiers.

Identifiers are used in recognizing environment variables and after a match of the **'define'** option. It is also used for "\i" in a [pattern](#) . See **'isfname'** for a description of the format of this option. For '@' only characters up to 255 are used. Careful: If you change this option, it might break expanding environment variables. E.g., when '/' is included and Vim tries to expand "\$HOME/.viminfo". Maybe you should change **'iskeyword'** instead.

```
'iskeyword' 'isk'          'iskeyword' 'isk'
                        string (Vim default for Win32:
                            "@,48-57,_,128-167,224-235"
                            otherwise: "@,48-57,_,192-255"
                            Vi default: "@,48-57,_")
                        local to buffer
```

Keywords are used in searching and recognizing with many commands: "w", "*", "[i", etc. It is also used for "\k" in a [pattern](#) . See **'isfname'** for a description of the format of this option. For '@' characters above 255 check the "word" character class (any character that is not white space or punctuation).

For C programs you could use "a-z,A-Z,48-57,_,.,-,>".

For a help file it is set to all non-blank printable characters except '*', '"' and '|' (so that **CTRL-]** on a command finds the help for that command).

When the **'lisp'** option is on the '-' character is always included. This option also influences syntax highlighting, unless the syntax uses **:syn-iskeyword** .

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

```
'isprint' 'isp' string (default for Win32 and macOS:
                        'isprint' 'isp'
                        "@,~-255"; otherwise: "@,161-255")
                        global
```

The characters given by this option are displayed directly on the screen. It is also used for "\p" in a [pattern](#) . The characters from space (ASCII 32) to '~' (ASCII 126) are always displayed directly, even when they are not included in **'isprint'** or excluded. See **'isfname'** for a description of the format of this option.

Non-printable characters are displayed with two characters:

0 - 31	"^@" - "^_"
32 - 126	always single characters
127	"^?"
128 - 159	"~@" - "~_"
160 - 254	" " - " ~"
255	"~?"

When **'encoding'** is a Unicode one, illegal bytes from 128 to 255 are displayed as <xx>, with the hexadecimal value of the byte.

When **'display'** contains "uhex" all unprintable characters are displayed as <xx>.

The SpecialKey highlighting will be used for unprintable characters. [hl-SpecialKey](#)

Multi-byte characters 256 and above are always included, only the

characters up to 255 are specified with this option. When a character is printable but it is not available in the current font, a replacement character will be shown. Unprintable and zero-width Unicode characters are displayed as `<xxxx>`. There is no option to specify these characters.

'joinspaces' 'js' **'joinspaces'** **'js'** **'nojoinspaces'** **'nojs'**
boolean (default on)
global

Insert two spaces after a '.', '?' and '!' with a join command. When **'cptions'** includes the 'j' flag, only do this after a '.'. Otherwise only one space is inserted.

NOTE: This option is set when **'compatible'** is set.

'jumpoptions' 'jop' **'jumpoptions'** **'jop'**
string (default "")
global

List of words that change the behavior of the `jumplist`.
stack Make the jumplist behave like the tagstack.
Relative location of entries in the jumplist is preserved at the cost of discarding subsequent entries when navigating backwards in the jumplist and then jumping to a location. `jumplist-stack`

'key' **'key'**
string (default "")
local to buffer
{only available when compiled with the `+cryptv` feature}

The key that is used for encrypting and decrypting the current buffer. See `encryption` and **'cryptmethod'**.

Careful: Do not set the key value by hand, someone might see the typed key. Use the `:X` command. But you can make **'key'** empty:

```
:set key=
```

It is not possible to get the value of this option with `":set key"` or `"echo &key"`. This is to avoid showing it to someone who shouldn't know. It also means you cannot see it yourself once you have set it, be careful not to make a typing error!

You also cannot use `:set-=`, `:set+=`, `:set^=` on this option to prevent an attacker from guessing substrings in your key.

You can use `"&key"` in an expression to detect whether encryption is enabled. When **'key'** is set it returns `"*****"` (five stars).

'keymap' 'kmp' **'keymap'** **'kmp'** E544
string (default "")
local to buffer
{only available when compiled with the `+keymap` feature}

Name of a keyboard mapping. See `mbyte-keymap`.

Setting this option to a valid keymap name has the side effect of setting **'iminsert'** to one, so that the keymap becomes effective.

'imsearch' is also set to one, unless it was `-1`

Only normal file name characters can be used, `"/*?[]<>"` are illegal.

'keymodel' 'km' string (default "")
global

List of comma-separated words, which enable special things that keys can do. These values can be used:

startsel Using a shifted special key starts selection (either Select mode or Visual mode, depending on "key" being present in **'selectmode'**).

stopsel Using a not-shifted special key stops selection.

Special keys in this context are the cursor keys, <End>, <Home>, <PageUp> and <PageDown>.

The **'keymodel'** option is set by the `:behave` command.

'keyprotocol' 'kpc' string (default: see below)
global

Specifies what keyboard protocol to use depending on the value of **'term'**. The supported keyboard protocols names are:

none whatever the terminal uses

mok2 modifyOtherKeys level 2, as supported by xterm

kitty Kitty keyboard protocol, as supported by Kitty

The option value is a list of comma separated items. Each item has a pattern that is matched against the **'term'** option, a colon and the protocol name to be used. To illustrate this, the default value would be set with:

```
set keyprotocol=kitty:kitty,foot:kitty,wezterm:kitty,xterm:mok2
```

This means that when **'term'** contains "kitty", "foot" or "wezterm" somewhere then the "kitty" protocol is used. When **'term'** contains "xterm" somewhere, then the "mok2" protocol is used.

The first match is used, thus if you want to have "kitty" use the kitty protocol, but "badkitty" not, then you should match "badkitty" first and use the "none" value:

```
set keyprotocol=badkitty:none,kitty:kitty
```

The option is used after **'term'** has been changed. First the termcap entries are set, possibly using the builtin list, see [builtin-terms](#). Then this option is inspected and if there is a match and a protocol is specified the following happens:

none Nothing, the regular t_TE and t_TI values remain

mok2 The t_TE value is changed to:

```
CSI >4;m disables modifyOtherKeys
```

The t_TI value is changed to:

```
CSI >4;2m enables modifyOtherKeys
```

```
CSI ?4m request the modifyOtherKeys state
```

kitty The t_TE value is changed to:

```
CSI >4;m disables modifyOtherKeys
```

```
CSI =0;1u disables the kitty keyboard protocol
```

The t_TI value is changed to:

```
CSI =1;1u enables the kitty keyboard protocol
```

```
CSI ?u      request kitty keyboard protocol state
CSI >c      request the termresponse
```

If you notice problems, such as characters being displayed that disappear after ``CTRL-L``, you might want to try making this option empty. Then set the `'term'` option to have it take effect:

```
set keyprotocol=
let &term = &term
```

'keywordprg' 'kp' 'keywordprg' 'kp'
string (default "man" or "man -s", DOS: ":help",
VMS: "help")

global or local to buffer `global-local`

Program to use for the `K` command. Environment variables are expanded `:set_env`. `:help` may be used to access the Vim internal help. (Note that previously setting the global option to the empty value did this, which is now deprecated.)

When the first character is `:"`, the command is invoked as a Vim Ex command with `[count]` added as an argument if it is not zero.

When "man", "man -s" or an Ex command is used, Vim will automatically translate a count for the "K" command and pass it as the first argument. For "man -s" the "-s" is removed when there is no count. See `option-backslash` about including spaces and backslashes.

Example:

```
:set keywordprg=man\ -s
```

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'langmap' 'lmap' 'langmap' 'lmap' E357 E358
string (default "")
global
{only available when compiled with the `+langmap`
feature}

This option allows switching your keyboard into a special language mode. When you are typing text in Insert mode the characters are inserted directly. When in Normal mode the `'langmap'` option takes care of translating these special characters to the original meaning of the key. This means you don't have to change the keyboard mode to be able to execute Normal mode commands.

This is the opposite of the `'keymap'` option, where characters are mapped in Insert mode.

Also consider setting `'langremap'` to off, to prevent `'langmap'` from applying to characters resulting from a mapping.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

Example (for Greek, in UTF-8):

```
:set langmap=␣A,␣B,␣C,␣D,␣E,␣F,␣G,␣H,␣I,␣J,␣K,␣L,␣M,␣N,␣O,␣P,␣Q,␣R,␣S,␣T,␣U,␣V
```

Example (exchanges meaning of z and y for commands):

```
:set langmap=zy,yz,ZY,YZ
```

The `'langmap'` option is a list of parts, separated with commas. Each part can be in one of two forms:

1. A list of pairs. Each pair is a "from" character immediately followed by the "to" character. Examples: "aA", "aAbBcC".
2. A list of "from" characters, a semi-colon and a list of "to" characters. Example: "abc;ABC"

Example: "aA, fgh;FGH, cCdDeE"

Special characters need to be preceded with a backslash. These are ";", ',', "'", '|' and backslash itself.

This will allow you to activate vim actions without having to switch back and forth between the languages. Your language characters will be understood as normal vim English characters (according to the langmap mappings) in the following cases:

- o Normal/Visual mode (commands, buffer/register names, user mappings)
- o Insert/Replace Mode: Register names after **CTRL-R**
- o Insert/Replace Mode: Mappings

Characters entered in Command-line mode will NOT be affected by this option. **Note** that this option can be changed at any time allowing to switch between mappings for different languages/encodings. Use a mapping to avoid having to type it each time!

```

                                'langmenu' 'lm'
'langmenu' 'lm'                string (default "")
                                global
                                {only available when compiled with the +menu and
                                +multi_lang features}

```

Language to use for menu translation. Tells which file is loaded from the "lang" directory in **'runtimepath'**:

```
"lang/menu_" .. &langmenu .. ".vim"
```

(without the spaces). For example, to always use the Dutch menus, no matter what \$LANG is set to:

```
:set langmenu=nl_NL.ISO_8859-1
```

When **'langmenu'** is empty, **v:lang** is used.

Only normal file name characters can be used, "/*?[]<>" are illegal. If your \$LANG is set to a non-English language but you do want to use the English menus:

```
:set langmenu=none
```

This option must be set before loading menus, switching on filetype detection or syntax highlighting. Once the menus are defined setting this option has no effect. But you could do this:

```
:source $VIMRUNTIME/delmenu.vim
:set langmenu=de_DE.ISO_8859-1
:source $VIMRUNTIME/menu.vim
```

Warning: This deletes all menus that you defined yourself!

```

                                'langnoremap' 'lnr' 'nolangnoremap' 'nolnr'
'langnoremap' 'lnr'           boolean (default off, set in defaults.vim )
                                global
                                {only available when compiled with the +langmap
                                feature}

```

This is just like **'langremap'** but with the value inverted. It only exists for backwards compatibility. When setting **'langremap'** then **'langnoremap'** is set to the inverted value, and the other way around.

```

                                'langremap' 'lrm' 'nolangremap' 'nolrm'

```

'langremap' 'lrm' boolean (default on, set to off in [defaults.vim](#))
 global
 {only available when compiled with the [+langmap](#)
 feature}

When off, setting **'langmap'** does not apply to characters resulting from a mapping. This basically means, if you noticed that setting **'langmap'** disables some of your mappings, try resetting this option. This option defaults to on for backwards compatibility. Set it off if that works for you to avoid mappings to break.

'laststatus' 'ls' number (default 1)
 global

The value of this option influences when the last window will have a status line:

- 0: never
- 1: only if there are at least two windows
- 2: always

The screen looks nicer with a status line if you have several windows, but it takes another screen line. [status-line](#)

'lazyredraw' 'lz' boolean (default off)
 global

When this option is set, the screen will not be redrawn while executing macros, registers and other commands that have not been typed. Also, updating the window title is postponed. To force an update use [:redraw](#) .

This may occasionally cause display errors. It is only meant to be set temporarily when performing an operation where redrawing may cause flickering or cause a slow down.

'linebreak' 'lbr' boolean (default off)
 local to window
 {not available when compiled without the [+linebreak](#)
 feature}

If on, Vim will wrap long lines at a character in **'breakat'** rather than at the last character that fits on the screen. Unlike **'wrapmargin'** and **'textwidth'**, this does not insert [<EOL>](#)s in the file, it only affects the way the file is displayed, not its contents.

If **'breakindent'** is set, line is visually indented. Then, the value of **'showbreak'** is used to put in front of wrapped lines. This option is not used when the **'wrap'** option is off.

Note that [<Tab>](#) characters after an [<EOL>](#) are mostly not displayed with the right amount of white space.

'lines' number (default 24 or terminal height)
 global

Number of lines of the Vim window.

Normally you don't need to set this. It is done automatically by the terminal initialization code. Also see [posix-screen-size](#) .

When Vim is running in the GUI or in a resizable window, setting this

option will cause the window size to be changed. When you only want to use the size for the GUI, put the command in your `gvimrc` file. Vim limits the number of lines to what fits on the screen. You can use this command to get the tallest window possible:

```
:set lines=999
```

Minimum value is 2, maximum value is 1000.

If you get fewer lines than expected, check the `'guiheadroom'` option. When you set this option and Vim is unable to change the physical number of lines of the display, the display may be messed up.

'linespace' 'lsp' 'linespace' 'lsp'
 number (default 0, 1 for Win32 GUI)
 global
 {only in the GUI}

Number of pixel lines inserted between characters. Useful if the font uses the full character cell height, making lines touch each other. When non-zero there is room for underlining.

With some fonts there can be too much room between lines (to have space for ascents and descents). Then it makes sense to set `'linespace'` to a negative value. This may cause display problems though!

'lisp' 'lisp' 'nolisp'
 boolean (default off)
 local to buffer

Lisp mode: When `<Enter>` is typed in insert mode set the indent for the next line to Lisp standards (well, sort of). Also happens with `"cc"` or `"S"`. `'autoindent'` must also be on for this to work. The `'p'` flag in `'coptions'` changes the method of indenting: Vi compatible or better. Also see `'lispwords'`.

The `'-'` character is included in keyword characters. Redefines the `"="` operator to use this same indentation algorithm rather than calling an external program if `'equalprg'` is empty. This option is not used when `'paste'` is set.

'lispoptions' 'lop' 'lispoptions' 'lop'
 string (default "")
 local to buffer

Comma-separated list of items that influence the Lisp indenting when enabled with the `'lisp'` option. Currently only one item is supported:

```
expr:1 use 'indentexpr' for Lisp indenting when it is set
expr:0 do not use 'indentexpr' for Lisp indenting (default)
```

Note that when using `'indentexpr'` the ``='` operator indents all the lines, otherwise the first line is not indented (Vi-compatible).

'lispwords' 'lw' 'lispwords' 'lw'
 string (default is very long)
 global or local to buffer `global-local`

Comma-separated list of words that influence the Lisp indenting when enabled with the `'lisp'` option.

'list' 'list' 'nolist'
 boolean (default off)

local to window

List mode: By default show tabs as **CTRL-I** is displayed, display \$ after end of line. Useful to see the difference between tabs and spaces and for trailing blanks. Further changed by the **'listchars'** option.

The cursor is displayed at the start of the space a Tab character occupies, not at the end as usual in Normal mode. To get this cursor position while displaying Tabs with spaces, use:

```
:set list lcs=tab:\ \
```

Note that list mode will also affect formatting (set with **'textwidth'** or **'wrapmargin'**) when **'coptions'** includes 'L'. See **'listchars'** for changing the way tabs are displayed.

		'listchars'	'lcs'
'listchars'	'lcs'	string (default "eol:\$")	
		global or local to window	global-local
		Strings to use in 'list' mode and for the :list command. It is a comma-separated list of string settings.	E1511
eol:c		Character to show at the end of each line. When omitted, there is no extra character at the end of the line.	lcs-eol
tab:xy[z]		Two or three characters to be used to show a tab. The third character is optional.	lcs-tab
tab:xy		The 'x' is always used, then 'y' as many times as will fit. Thus "tab:>-" displays: > >- >-- etc.	
tab:xyz		The 'z' is always used, then 'x' is prepended, and then 'y' is used as many times as will fit. Thus "tab:<->" displays: > <> <-> <--> etc.	
		When "tab:" is omitted, a tab is shown as ^I.	
space:c		Character to show for a space. When omitted, spaces are left blank.	lcs-space
multispace:c...		One or more characters to use cyclically to show for multiple consecutive spaces. Overrides the "space" setting, except for single spaces. When omitted, the "space" setting is used. For example,	lcs-multispace

```
`:set listchars=multispace:---+` shows ten consecutive spaces as:
```

```
---+---+---
```

`lcs-lead`

`lead:c` Character to show for leading spaces. When omitted, leading spaces are blank. Overrides the "space" and "multispace" settings for leading spaces. You can combine it with "tab:", for example:

```
:set listchars+=tab:>-,lead:.
```

`lcs-leadmultispace`

`leadmultispace:c...`

Like the `lcs-multispace` value, but for leading spaces only. Also overrides `lcs-lead` for leading multiple spaces.

```
`:set listchars=leadmultispace:---+` shows ten consecutive leading spaces as:
```

```
---+---+---XXX
```

Where "XXX" denotes the first non-blank characters in the line.

`lcs-trail`

`trail:c` Character to show for trailing spaces. When omitted, trailing spaces are blank. Overrides the "space" and "multispace" settings for trailing spaces.

`lcs-extends`

`extends:c` Character to show in the last column, when `'wrap'` is off and the line continues beyond the right of the screen.

`lcs-precedes`

`precedes:c` Character to show in the first visible column of the physical line, when there is text preceding the character visible in the first column.

`lcs-conceal`

`conceal:c` Character to show in place of concealed text, when `'conceallevel'` is set to 1.

`lcs-nbsp`

`nbsp:c` Character to show for a non-breakable space character (0xA0 (160 decimal) and U+202F). Left blank when omitted.

The characters ':' and ',' should not be used. UTF-8 characters can be used when `'encoding'` is "utf-8", otherwise only printable characters are allowed. All characters must be single width. [E1512](#)

Each character can be specified as hex:

```
set listchars=eol:\x24
```

```
set listchars=eol:\u21b5
```

```
set listchars=eol:\U000021b5
```

Note that a double backslash is used. The number of hex characters must be exactly 2 for `\x`, 4 for `\u` and 8 for `\U`.

Examples:

```
:set lcs=tab:>-,trail:-
```

```
:set lcs=tab:>-,eol:<,nbsp:%
```

```
:set lcs=extends:>,precedes:<
```

The "NonText" highlighting will be used for "eol", "extends" and "precedes". "SpecialKey" will be used for "tab", "nbsp", "space", "multispace", "lead" and "trail".

hl-NonText hl-SpecialKey

'loadplugins' **'lpl'** **'nolpl'** **'loadplugins'** **'noloadplugins'**
boolean (default on)
global

When on the plugin scripts are loaded when starting up `load-plugins`. This option can be reset in your `vimrc` file to disable the loading of plugins.

Note that using the "-u NONE", "-u DEFAULTS" and "--noplugin" command line arguments reset this option. See `-u` and `--noplugin`.

'luadll' **'luadll'**
string (default depends on the build)
global
{only available when compiled with the `+lua/dyn` feature}

Specifies the name of the Lua shared library. The default is DYNAMIC_LUA_DLL, which was specified at compile time.

Environment variables are expanded `:set_env`.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'macatsui' **'macatsui'** **'nomacatsui'**
boolean (default on)
global
{not supported}

No longer supported, as the Mac OS X GUI code was removed.

'magic' **'magic'** **'nomagic'**
boolean (default on)
global

Changes the special characters that can be used in search patterns. See `pattern`.

WARNING: Switching this option off most likely breaks plugins! That is because many patterns assume it's on and will fail when it's off. Only switch it off when working with old Vi scripts. In any other situation write patterns that work when **'magic'** is on. Include `"\M"` when you want to `/\M`.

In `Vim9` script the value of **'magic'** is ignored, patterns behave like it is always set.

'makeef' **'mef'** **'makeef'** **'mef'**
string (default: "")
global
{not available when compiled without the `+quickfix` feature}

Name of the errorfile for the `:make` command (see `:make_makeprg`) and the `:grep` command.

When it is empty, an internally generated temp file will be used.

When `"##"` is included, it is replaced by a number to make the name unique. This makes sure that the `:make` command doesn't overwrite an

existing file.
 NOT used for the ":cf" command. See **'errorfile'** for that.
 Environment variables are expanded `:set_env` .
 See `option-backslash` about including spaces and backslashes.
 This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'makeencoding' 'menc' `string` (default "")
 global or local to buffer `global-local`
 Encoding used for reading the output of external commands. When empty, encoding is not converted.
 This is used for `:make`, `:lmake`, `:grep`, `:lgrep`, `:grepadd`, `:lgrepadd`, `:cfile`, `:cgetfile`, `:caddfile`, `:lfile`, `:lgetfile`, and `:laddfile`.

This would be mostly useful when you use MS-Windows and set **'encoding'** to "utf-8". If `+iconv` is enabled and GNU libiconv is used, setting **'makeencoding'** to "char" has the same effect as setting to the system locale encoding. Example:

```
:set encoding=utf-8
:set makeencoding=char " system locale is used
```

'makeprg' 'mp' `string` (default "make", VMS: "MMS")
 global or local to buffer `global-local`
 Program to use for the ":make" command. See `:make_makeprg` .
 This option may contain '%' and '#' characters (see `:_%` and `:_#`), which are expanded to the current and alternate file name. Use `::S` to escape file names in case they contain special characters.
 Environment variables are expanded `:set_env` . See `option-backslash` about including spaces and backslashes.
Note that a '|' must be escaped twice: once for ":set" and once for the interpretation of a command. When you use a filter called "myfilter" do it like this:

```
:set makeprg=gmake\ \\|\ myfilter
```

The placeholder "\$*" can be given (even multiple times) to specify where the arguments will be included, for example:

```
:set makeprg=latex\ \\|\nonstopmode\ \\|\input\{\$*}
```

This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'matchpairs' 'mps' `string` (default "(:),{:},[:]")
 local to buffer
 Characters that form pairs. The `%` command jumps from one to the other.
 Only character pairs are allowed that are different, thus you cannot jump between two double quotes.
 The characters must be separated by a colon.
 The pairs must be separated by a comma. Example for including '<' and '>' (for HTML):

```
:set mps+=<:>
```

A more exotic example, to jump between the '=' and ';' in an assignment, useful for languages like C and Java:

```
:au FileType c,cpp,java set mps+==;
```

For a more advanced way of using "%", see the matchit.vim plugin in the \$VIMRUNTIME/pack/dist/opt/matchit directory. [add-local-help](#)

'matchtime' **'mat'** 'matchtime' 'mat'
number (default 5)
global
Tenths of a second to show the matching paren, when **'showmatch'** is set. **Note** that this is not in milliseconds, like other options that set a time. This is to be compatible with Nvi.

'maxcombine' **'mco'** 'maxcombine' 'mco'
number (default 2)
global
The maximum number of combining characters supported for displaying. Only used when **'encoding'** is "utf-8". The default is OK for most languages. Hebrew may require 4. Maximum value is 6. Even when this option is set to 2 you can still edit text with more combining characters, you just can't see them. Use `g8` or `ga`. See [mbyte-combining](#).

'maxfuncdepth' **'mfd'** 'maxfuncdepth' 'mfd'
number (default 100)
global
{not available when compiled without the `+eval` feature}
Maximum depth of function calls for user functions. This normally catches endless recursion. When using a recursive function with more depth, set **'maxfuncdepth'** to a bigger number. But this will use more memory, there is the danger of failing when memory is exhausted. Increasing this limit above 200 also changes the maximum for Ex command recursion, see [E169](#). See also `:function`. Also used for maximum depth of callback functions.

'maxmapdepth' **'mmd'** 'maxmapdepth' 'mmd' E223
number (default 1000)
global
Maximum number of times a mapping is done without resulting in a character to be used. This normally catches endless mappings, like `":map x y"` with `":map y x"`. It still does not catch `":map g wg"`, because the 'w' is used before the next mapping is done. See also [key-mapping](#).

'maxmem' **'mm'** 'maxmem' 'mm'
number (default between 256 to 5120 (system dependent) or half the amount of memory available)
global
Maximum amount of memory (in Kbyte) to use for one buffer. When this

limit is reached allocating extra memory for a buffer will cause other memory to be freed.
The maximum usable value is about 2000000. Use this to work without a limit.
The value is ignored when **'swapfile'** is off.
Also see **'maxmemtot'**.

'maxmempattern' **'mmp'** **'maxmempattern'** **'mmp'**
number (default 1000)
global
Maximum amount of memory (in Kbyte) to use for pattern matching.
The maximum value is about 2000000. Use this to work without a limit.
E363
When Vim runs into the limit it gives an error message and mostly behaves like **CTRL-C** was typed.
Running into the limit often means that the pattern is very inefficient or too complex. This may already happen with the pattern `"\(.\)*" on a very long line. ".*" works much better.`
Might also happen on redraw, when syntax rules try to match a complex text structure.
Vim may run out of memory before hitting the **'maxmempattern'** limit, in which case you get an "Out of memory" error instead.

'maxmemtot' **'mmt'** **'maxmemtot'** **'mmt'**
number (default between 2048 and 10240 (system dependent) or half the amount of memory available)
global
Maximum amount of memory in Kbyte to use for all buffers together.
The maximum usable value is about 2000000 (2 Gbyte). Use this to work without a limit.
On 64 bit machines higher values might work. But hey, do you really need more than 2 Gbyte for text editing? Keep in mind that text is stored in the swap file, one can edit files > 2 Gbyte anyway. We do need the memory to store undo info.
Buffers with **'swapfile'** off still count to the total amount of memory used.
Also see **'maxmem'**.

'menuitems' **'mis'** **'menuitems'** **'mis'**
number (default 25)
global
{not available when compiled without the **+menu** feature}
Maximum number of items to use in a menu. Used for menus that are generated from a list of items, e.g., the Buffers menu. Changing this option has no direct effect, the menu must be refreshed first.

'mkspellmem' **'msm'** **'mkspellmem'** **'msm'**
string (default "460000,2000,500")
global
{not available when compiled without the **+syntax** feature}
Parameters for **:mkspell** . This tunes when to start compressing the

word tree. Compression can be slow when there are many words, but it's needed to avoid running out of memory. The amount of memory used per word depends very much on how similar the words are, that's why this tuning is complicated.

There are three numbers, separated by commas:

```
{start},{inc},{added}
```

For most languages the uncompressed word tree fits in memory. `{start}` gives the amount of memory in Kbyte that can be used before any compression is done. It should be a bit smaller than the amount of memory that is available to Vim.

When going over the `{start}` limit the `{inc}` number specifies the amount of memory in Kbyte that can be allocated before another compression is done. A low number means compression is done after less words are added, which is slow. A high number means more memory will be allocated.

After doing compression, `{added}` times 1024 words can be added before the `{inc}` limit is ignored and compression is done when any extra amount of memory is needed. A low number means there is a smaller chance of hitting the `{inc}` limit, less memory is used but it's slower.

The languages for which these numbers are important are Italian and Hungarian. The default works for when you have about 512 Mbyte. If you have 1 Gbyte you could use:

```
:set mkspellmem=900000,3000,800
```

If you have less than 512 Mbyte `:mkspell` may fail for some languages, no matter what you set `'mkspellmem'` to.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

```
'modeline' 'ml'          'modeline' 'ml' 'nomodeline' 'noml'  
boolean (Vim default: on (off for root),  
Vi default: off)  
local to buffer
```

If `'modeline'` is on `'modelines'` gives the number of lines that is checked for set commands. If `'modeline'` is off or `'modelines'` is zero no lines are checked. See `modeline`.

```
'modelineexpr' 'mle'     'modelineexpr' 'mle' 'nomodelineexpr' 'nomle'  
boolean (default: off)  
global
```

When on allow some options that are an expression to be set in the `modeline`. Check the option for whether it is affected by `'modelineexpr'`. Also see `modeline`.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

```
'modelines' 'mls'       'modelines' 'mls'  
number (default 5)
```

global

If **'modeline'** is on **'modelines'** gives the number of lines that is checked for set commands. If **'modeline'** is off or **'modelines'** is zero no lines are checked. See [modeline](#) .

NOTE: **'modeline'** is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'modifiable' **'ma'** **'modifiable'** **'ma'** **'nommodifiable'** **'noma'**
E21
boolean (default on)
local to buffer

When off the buffer contents cannot be changed. The **'fileformat'** and **'fileencoding'** options also can't be changed.

Can be reset on startup with the **-M** command line argument.

'modified' **'mod'** **'modified'** **'mod'** **'nommodified'** **'nomod'**
boolean (default off)
local to buffer **local-noglobal**

When on, the buffer is considered to be modified. This option is set when:

1. A change was made to the text since it was last written. Using the [undo](#) command to go back to the original text will reset the option. But undoing changes that were made before writing the buffer will set the option again, since the text is different from when it was written.
2. **'fileformat'** or **'fileencoding'** is different from its original value. The original value is set when the buffer is read or written. A `":set nommodified"` command also resets the original values to the current values and the **'modified'** option will be reset.

Similarly for **'eol'** and **'bomb'**.

This option is not set when a change is made to the buffer as the result of a `BufNewFile`, `BufRead/BufReadPost`, `BufWritePost`, `FileAppendPost` or `VimLeave` autocommand event. See [gzip-example](#) for an explanation.

When **'buftype'** is "nowrite" or "nofile" this option may be set, but will be ignored.

Note that the text may actually be the same, e.g. **'modified'** is set when using "rA" on an "A".

'more' **'more'** **'nomore'**
boolean (Vim default: on, Vi default: off)
global

When on, listings pause when the whole screen is filled. You will get the [more-prompt](#) . When this option is off there are no pauses, the listing continues until finished.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'mouse' **'mouse'**
string (default "", "a" for GUI and Win32,
set to "a" or "nvi" in [defaults.vim](#))
global

Enable the use of the mouse. Works for most terminals (xterm, Win32

`win32-mouse` , QNX pterm, *BSD console with `sysmouse` and Linux console with `gpm`). For using the mouse in the GUI, see `gui-mouse` . The mouse can be enabled for different modes:

```
n      Normal mode and Terminal modes
v      Visual mode
i      Insert mode
c      Command-line mode
h      all previous modes when editing a help file
a      all previous modes
r      for hit-enter and more-prompt prompt
```

Normally you would enable the mouse in all five modes with:

```
:set mouse=a
```

If your terminal can't overrule the mouse events going to the application, use:

```
:set mouse=nvi
```

Then you can press ":", select text for the system, and press Esc to go back to Vim using the mouse events.

In `defaults.vim` "nvi" is used if the `'term'` option is not matching "xterm".

When the mouse is not enabled, the GUI will still use the mouse for modeless selection. This doesn't move the text cursor.

See `mouse-using` . Also see `'clipboard'` .

Note: When enabling the mouse in a terminal, copy/paste will use the "*" register if there is access to an X-server. The xterm handling of the mouse buttons can still be used by keeping the shift key pressed. Also see the `'clipboard'` option.

```
'mousefocus' 'mousef'      'mousefocus'  'mousef'  'nomousefocus'  'nomousef'
boolean (default off)
global
{only works in the GUI}
```

The window that the mouse pointer is on is automatically activated. When changing the window layout or window focus in another way, the mouse pointer is moved to the window with keyboard focus. Off is the default because it makes using the pull down menus a little goofy, as a pointer transit may activate a window unintentionally.

MS-Windows: Also see `'scrollfocus'` for what window is scrolled when using the mouse scroll wheel.

```
'mousehide' 'mh'           'mousehide'  'mh'  'nomousehide'  'nomh'
boolean (default on)
global
{only works in the GUI}
```

When on, the mouse pointer is hidden when characters are typed. The mouse pointer is restored when the mouse is moved.

```
'mousemodel' 'mousem'      'mousemodel'  'mousem'
string (default "extend", "popup" for Win32)
global
```

Sets the model to use for the mouse. The name mostly specifies what the right mouse button is used for:

extend Right mouse button extends a selection. This works like in an xterm.
 popup Right mouse button pops up a menu. The shifted left mouse button extends a selection. This works like with Microsoft Windows.
 popup_setpos Like "popup", but the cursor will be moved to the position where the mouse was clicked, and thus the selected operation will act upon the clicked object. If clicking inside a selection, that selection will be acted upon, i.e. no cursor move. This implies of course, that right clicking outside a selection will end Visual mode.

Overview of what button does what for each model:

mouse	extend	popup(_setpos)
left click	place cursor	place cursor
left drag	start selection	start selection
shift-left	search word	extend selection
right click	extend selection	popup menu (place cursor)
right drag	extend selection	-
middle click	paste	paste

In the "popup" model the right mouse button produces a pop-up menu. You need to define this first, see [popup-menu](#) .

Note that you can further refine the meaning of buttons with mappings. See [gui-mouse-mapping](#) . But mappings are NOT used for modeless selection (because that's handled in the GUI code directly).

The **'mousemodel'** option is set by the `:behave` command.

```

'mousemoveevent' 'mousemev' 'nomousemoveevent' 'nomousemev'
'mousemoveevent' 'mousemev' boolean (default off)
global
{only works in the GUI}
  
```

When on, mouse move events are delivered to the input queue and are available for mapping. The default, off, avoids the mouse movement overhead except when needed. See [gui-mouse-mapping](#) .

Warning: Setting this option can make pending mappings to be aborted when the mouse is moved.

Currently only works in the GUI, may be made to work in a terminal later.

```

'mouseshape' 'mouses' string (default "i-r:beam,s:updown,sd:udsizing,
                                vs:left:right,vd:lrsizing,m:no,
                                ml:up-arrow,v:rightup-arrow")
global
{only available when compiled with the +mouseshape
feature}
  
```

This option tells Vim what the mouse pointer should look like in different modes. The option is a comma-separated list of parts, much like used for **'guicursor'**. Each part consist of a mode/location-list and an argument-list:

```

mode-list:shape,mode-list:shape,..
  
```

The mode-list is a dash separated list of these modes/locations:

In a normal window:

n	Normal mode
v	Visual mode
ve	Visual mode with ' selection ' "exclusive" (same as 'v', if not specified)
o	Operator-pending mode
i	Insert mode
r	Replace mode

Others:

c	appending to the command-line
ci	inserting in the command-line
cr	replacing in the command-line
m	at the 'Hit ENTER' or 'More' prompts
ml	idem, but cursor in the last line
e	any mode, pointer below last window
s	any mode, pointer on a status line
sd	any mode, while dragging a status line
vs	any mode, pointer on a vertical separator line
vd	any mode, while dragging a vertical separator line
a	everywhere

The shape is one of the following:

avail	name	looks like
w x	arrow	Normal mouse pointer
w x	blank	no pointer at all (use with care!)
w x	beam	I-beam
w x	updown	up-down sizing arrows
w x	leftright	left-right sizing arrows
w x	busy	The system's usual busy pointer
w x	no	The system's usual 'no input' pointer
x	ud sizing	indicates up-down resizing
x	lrsizing	indicates left-right resizing
x	crosshair	like a big thin +
x	hand1	black hand
x	hand2	white hand
x	pencil	what you write with
x	question	big ?
x	rightup-arrow	arrow pointing right-up
w x	up-arrow	arrow pointing up
x	<number>	any X11 pointer number (see X11/cursorfont.h)

The "avail" column contains a 'w' if the shape is available for Win32, x for X11.

Any modes not specified or shapes not available use the normal mouse pointer.

Example:

```
:set mouseshape=s:ud sizing,m:no
```

will make the mouse turn to a sizing arrow over the status lines and indicate no input when the hit-enter prompt is displayed (since clicking the mouse has no effect in this state.)

hex If included, numbers starting with "0x" or "0X" will be considered to be hexadecimal. Example: Using **CTRL-X** on "0x100" results in "0x0ff".

bin If included, numbers starting with "0b" or "0B" will be considered to be binary. Example: Using **CTRL-X** on "0b1000" subtracts one, resulting in "0b0111".

unsigned If included, numbers are recognized as unsigned. Thus a leading dash or negative sign won't be considered as part of the number. Examples:
 Using **CTRL-X** on "2020" in "9-2020" results in "9-2019" (without "unsigned" it would become "9-2021").
 Using **CTRL-A** on "2020" in "9-2020" results in "9-2021" (without "unsigned" it would become "9-2019").
 Using **CTRL-X** on "0" or **CTRL-A** on "18446744073709551615" ($2^{64} - 1$) has no effect, overflow is prevented.

Numbers which simply begin with a digit in the range 1-9 are always considered decimal. This also happens for numbers that are not recognized as octal or hex.

'number' **'nu'** **'nonumber'** **'nonu'**
 boolean (default off)
 local to window

Print the line number in front of each line. When the 'n' option is excluded from **'coptions'** a wrapped line will not use the column of line numbers (this is the default when **'compatible'** isn't set). The **'numberwidth'** option can be used to set the room used for the line number.

When a long, wrapped line doesn't start with the first character, '-' characters are put before the number.

For highlighting see [hl-LineNr](#) , and [hl-CursorLineNr](#) , and the [:sign-define](#) "numhl" argument.

The **'relativenumber'** option changes the displayed number to be relative to the cursor. Together with **'number'** there are these four combinations (cursor in line 3):

	'nonu' 'nornu'	'nu' 'nornu'	'nonu' 'rnu'	'nu' 'rnu'
apple	1 apple	2 apple	2 apple	
pear	2 pear	1 pear	1 pear	
nobody	3 nobody	0 nobody	3 nobody	
there	4 there	1 there	1 there	

'numberwidth' **'nuw'** **'numberwidth'** **'nuw'**
 number (Vim default: 4 Vi default: 8)
 local to window
 {only available when compiled with the [+linebreak](#) feature}

Minimal number of columns to use for the line number. Only relevant when the **'number'** or **'relativenumber'** option is set or printing lines with a line number. Since one space is always between the number and the text, there is one less character for the number itself. The value is the minimum width. A bigger width is used when needed to

fit the highest line number in the buffer respectively the number of rows in the window, depending on whether `'number'` or `'relativenumber'` is set. Thus with the Vim default of 4 there is room for a line number up to 999. When the buffer has 1000 lines five columns will be used. The minimum value is 1, the maximum value is 20.

NOTE: This option is set to the Vi default value when `'compatible'` is set and to the Vim default value when `'compatible'` is reset.

`'omnifunc'` `'ofu'` string (default: empty)
local to buffer
{not available when compiled without the `+eval` feature}

This option specifies a function to be used for Insert mode omni completion with `CTRL-X CTRL-O`. `i_CTRL-X_CTRL-O`
See [complete-functions](#) for an explanation of how the function is invoked and what it should return. The value can be the name of a function, a `lambda` or a `Funcref`. See [option-value-function](#) for more information.

This option is usually set by a filetype plugin:

```
:filetype-plugin-on
```

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'opendevic`e' `'odev'` boolean (default off)
global
{only for MS-Windows} [E796](#)

Enable reading and writing from devices. This may get Vim stuck on a device that can be opened but doesn't actually do the I/O. Therefore it is off by default.

Note that on MS-Windows editing "aux.h", "lpt1.txt" and the like also result in editing a device.

`'operatorfunc'` `'opfunc'` string (default: empty)
global

This option specifies a function to be called by the `g@` operator. See [:map-operator](#) for more info and an example. The value can be the name of a function, a `lambda` or a `Funcref`. See [option-value-function](#) for more information.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'osfiletype'` `'oft'` string (default: "")
local to buffer

This option was supported on RISC OS, which has been removed.

`'packpath'` `'pp'` string (default: see `'runtimepath'`)

Directories used to find packages. See [packages](#).

This option cannot be set from a `modeline` or in the `sandbox`, for

security reasons.

'paragraphs' **'para'** string (default "IPLPPPQPP TPHPLIPpLpItpplpipbp")
global

Specifies the nroff macros that separate paragraphs. These are pairs of two letters (see [object-motions](#)).

'paste' boolean (default off)
global

Put Vim in Paste mode. This is useful if you want to cut or copy some text from one window and paste it in Vim. This will avoid unexpected effects.

Setting this option is useful when using Vim in a terminal, where Vim cannot distinguish between typed text and pasted text. In the GUI, Vim knows about pasting and will mostly do the right thing without **'paste'** being set. The same is true for a terminal where Vim handles the mouse clicks itself.

This option is reset when starting the GUI. Thus if you set it in your `.vimrc` it will work in a terminal, but not in the GUI. Setting **'paste'** in the GUI has side effects: e.g., the Paste toolbar button will no longer work in Insert mode, because it uses a mapping.

When the **'paste'** option is switched on (also when it was already on):

- mapping in Insert mode and Command-line mode is disabled
- abbreviations are disabled
- **'autoindent'** is reset
- **'expandtab'** is reset
- **'hkmap'** is reset
- **'revins'** is reset
- **'ruler'** is reset
- **'showmatch'** is reset
- **'smarttab'** is reset
- **'softtabstop'** is set to 0
- **'textwidth'** is set to 0
- **'wrapmargin'** is set to 0
- **'varsofttabstop'** is made empty

These options keep their value, but their effect is disabled:

- **'cindent'**
- **'formatoptions'** is used like it is empty
- **'indentexpr'**
- **'lisp'**
- **'smartindent'**

NOTE: When you start editing another file while the **'paste'** option is on, settings from the modelines or autocommands may change the settings again, causing trouble when pasting text. You might want to set the **'paste'** option again.

When the **'paste'** option is reset the mentioned options are restored to the value before the moment **'paste'** was switched from off to on.

Resetting **'paste'** before ever setting it does not have any effect.

Since mapping doesn't work while **'paste'** is active, you need to use the **'pastetoggle'** option to toggle the **'paste'** option with some key.

'pastetoggle' **'pt'**

'pastetoggle' 'pt' string (default "")
global

When non-empty, specifies the key sequence that toggles the **'paste'** option. This is like specifying a mapping:

```
:map {keys} :set invpaste<CR>
```

Where {keys} is the value of **'pastetoggle'**.

The difference is that it will work even when **'paste'** is set.

'pastetoggle' works in Insert mode and Normal mode, but not in Command-line mode.

Mappings are checked first, thus overrule **'pastetoggle'**. However, when **'paste'** is on mappings are ignored in Insert mode, thus you can do this:

```
:map <F10> :set paste<CR>
:map <F11> :set nopaste<CR>
:imap <F10> <C-O>:set paste<CR>
:imap <F11> <nop>
:set pastetoggle=<F11>
```

This will make <F10> start paste mode and <F11> stop paste mode.

Note that typing <F10> in paste mode inserts "<F10>", since in paste mode everything is inserted literally, except the **'pastetoggle'** key sequence.

When the value has several bytes **'timeoutlen'** applies.

'patchexpr' 'pex' string (default "")
global
{not available when compiled without the **+diff** feature}

Expression which is evaluated to apply a patch to a file and generate the resulting new version of the file. See [diff-patchexpr](#).

This option cannot be set from a [modeline](#) or in the [sandbox](#), for security reasons.

'patchmode' 'pm' string (default "")
global

When non-empty the oldest version of a file is kept. This can be used to keep the original version of a file if you are changing files in a source distribution. Only the first time that a file is written a copy of the original file will be kept. The name of the copy is the name of the original file with the string in the **'patchmode'** option appended. This option should start with a dot. Use a string like ".orig" or ".org". **'backupdir'** must not be empty for this to work (Detail: The backup file is renamed to the patchmode file after the new file has been successfully written, that's why it must be possible to write a backup file). If there was no file to be backed up, an empty file is created.

When the **'backupskip'** pattern matches, a patchmode file is not made. Using **'patchmode'** for compressed files appends the extension at the end (e.g., "file.gz.orig"), thus the resulting name isn't always recognized as a compressed file.

Only normal file name characters can be used, "*?[]<>" are illegal.

'path' 'pa' E343 E345 E347 E854

'path' **'pa'** string (default on Unix: `"/usr/include,,"`,
other systems: `"/,,"`)
global or local to buffer `global-local`

This is a list of directories which will be searched when using the `gf`, `[f,]f`, `^Wf`, `:find`, `:sfind`, `:tabfind` and other commands, provided that the file being searched for has a relative path (not starting with `"/`, `"/.` or `"/..`). The directories in the **'path'** option may be relative or absolute.

- Use commas to separate directory names:
`:set path=.,/usr/local/include,/usr/include`
- Spaces can also be used to separate directory names (for backwards compatibility with version 3.0). To have a space in a directory name, precede it with an extra backslash, and escape the space:
`:set path=.,/dir/with\\ space`
- To include a comma in a directory name precede it with an extra backslash:
`:set path=.,/dir/with\\,comma`
- To search relative to the directory of the current file, use:
`:set path=.`
- To search in the current directory use an empty string between two commas:
`:set path=,,`
- A directory name may end in a `:` or `/`.
 - Environment variables are expanded `:set_env`.
 - When using `netrw.vim` URLs can be used. For example, adding `"http://www.vim.org"` will make `:find index.html` work.
 - Search upwards and downwards in a directory tree using `"*`, `**` and `;"`. See `file-searching` for info and syntax.
 - Careful with `\` characters, type two to get one in the option:
`:set path=.,c:\\include`
Or just use `/` instead:
`:set path=.,c:/include`

Don't forget `."` or files won't even be found in the same directory as the file!

The maximum length is limited. How much depends on the system, mostly it is something like 256 or 1024 characters.

You can check if all the include files are found, using the value of **'path'**, see `:checkpath`.

The use of `:set+=` and `:set-=` is preferred when adding or removing directories from the list. This avoids problems when a future version uses another default. To remove the current directory use:

```
:set path-=
```

To add the current directory use:

```
:set path+=
```

To use an environment variable, you probably need to replace the separator. Here is an example to append `$INCL`, in which directory names are separated with a semi-colon:

```
:let &path = &path .. "," .. substitute($INCL, ';', ',', 'g')
```

Replace the `;` with a `:` or whatever separator is used. Note that this doesn't work when `$INCL` contains a comma or white space.

'perldll' string (default depends on the build)
global `'perldll'`

{only available when compiled with the `+perl/dyn` feature}

Specifies the name of the Perl shared library. The default is `DYNAMIC_PERL_DLL`, which was specified at compile time.

Environment variables are expanded `:set_env`.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'preserveindent' **'pi'** **'preserveindent'** **'pi'** **'nopreserveindent'** **'nopi'**
boolean (default off)
local to buffer

When changing the indent of the current line, preserve as much of the indent structure as possible. Normally the indent is replaced by a series of tabs followed by spaces as required (unless `'expandtab'` is enabled, in which case only spaces are used). Enabling this option means the indent will preserve as many existing characters as possible for indenting, and only add additional tabs or spaces as required.

'expandtab' does not apply to the preserved white space, a Tab remains a Tab.

NOTE: When using `">>"` multiple times the resulting indent is a mix of tabs and spaces. You might not like this.

NOTE: This option is reset when **'compatible'** is set.

Also see **'copyindent'**.

Use `:retab` to clean up white space.

'previewheight' **'pvh'** **'previewheight'** **'pvh'**
number (default 12)
global
{not available when compiled without the `+quickfix` feature}

Default height for a preview window. Used for `:ptag` and associated commands. Used for `CTRL-W_` when no count is given. Not used when **'previewpopup'** is set.

'previewpopup' **'pvp'** **'previewpopup'** **'pvp'**
string (default empty)
global
{not available when compiled without the `+textprop` or `+quickfix` feature}

When not empty a popup window is used for commands that would open a preview window. See `preview-popup`.

Not used for the insert completion info, add "popup" to **'completeopt'** for that.

'previewwindow' **'pvw'** **'previewwindow'** **'nopreviewwindow'**
'pvw' **'nopvw'** E590
boolean (default off)
local to window `local-noglobal`
{not available when compiled without the `+quickfix` feature}

Identifies the preview window. Only one window can have this option set. It's normally not set directly, but by using one of the commands

`:ptag`, `:pedit`, etc.

'printdevice' **'pdev'** string (default empty) 'printdevice' 'pdev'
global
{only available when compiled with the `+printer`
feature}
The name of the printer to be used for `:hardcopy` .
See `pdev-option` .
This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'printencoding' **'penc'** string (default empty, except for some systems) 'printencoding' 'penc'
global
{only available when compiled with the `+printer`
and `+postscript` features}
Sets the character encoding used when printing.
See `penc-option` .

'printexpr' **'pexpr'** string (default: see below) 'printexpr' 'pexpr'
global
{only available when compiled with the `+printer`
and `+postscript` features}
Expression used to print the PostScript produced with `:hardcopy` .
See `pexpr-option` .
This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'printfont' **'pfn'** string (default "courier") 'printfont' 'pfn'
global
{only available when compiled with the `+printer`
feature}
The name of the font that will be used for `:hardcopy` .
See `pfn-option` .

'printhead' **'pheader'** string (default "%<%f%h%m%=Page %N") 'printhead' 'pheader'
global
{only available when compiled with the `+printer`
feature}
The format of the header produced in `:hardcopy` output.
See `pheader-option` .

'printmbcharset' **'pmbcs'** string (default "") 'printmbcharset' 'pmbcs'
global
{only available when compiled with the `+printer`
and `+postscript` features}
The CJK character set to be used for CJK output from `:hardcopy` .
See `pmbcs-option` .

'printmbfont' **'pmbfn'** string (default "") 'printmbfont' 'pmbfn'

global
 {only available when compiled with the `+printer`
 and `+postscript` features}

List of font names to be used for CJK output from `:hardcopy` .
 See `pmbfn-option` .

'printoptions' **'popt'** string (default "") 'printoptions' 'popt'
 global
 {only available when compiled with `|+printer|` feature}

List of items that control the format of the output of `:hardcopy` .
 See `popt-option` .

'prompt' boolean (default on) 'prompt' 'noprompt'
 global
 When on a ":" prompt is used in Ex mode.

'pumheight' **'ph'** number (default 0) 'pumheight' 'ph'
 global
 Determines the maximum number of items to show in the popup menu for
 Insert mode completion. When zero as much space as available is used.
[ins-completion-menu](#) .

'pumwidth' **'pw'** number (default 15) 'pumwidth' 'pw'
 global
 Determines the minimum width to use for the popup menu for Insert mode
 completion. [ins-completion-menu](#) .

'pythondll' string (default depends on the build) 'pythondll'
 global
 {only available when compiled with the `+python/dyn`
 feature}

Specifies the name of the Python 2.x shared library. The default is
 DYNAMIC_PYTHON_DLL, which was specified at compile time.
 Environment variables are expanded `:set_env` .
 This option cannot be set from a `modeline` or in the `sandbox` , for
 security reasons.

'pythonhome' string (default "") 'pythonhome'
 global
 {only available when compiled with the `+python/dyn`
 feature}

Specifies the name of the Python 2.x home directory. When **'pythonhome'**
 and the PYTHONHOME environment variable are not set, PYTHON_HOME,
 which was specified at compile time, will be used for the Python 2.x
 home directory.
 Environment variables are expanded `:set_env` .
 This option cannot be set from a `modeline` or in the `sandbox` , for
 security reasons.

'pythonthreedll' 'pythonthreedll'
 string (default depends on the build)
 global
 {only available when compiled with the `+python3/dyn`
 feature}

Specifies the name of the Python 3 shared library. The default is `DYNAMIC_PYTHON3_DLL`, which was specified at compile time. Environment variables are expanded `:set_env`. This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'pythonthreehome' 'pythonthreehome'
 string (default "")
 global
 {only available when compiled with the `+python3/dyn`
 feature}

Specifies the name of the Python 3 home directory. When **'pythonthreehome'** and the `PYTHONHOME` environment variable are not set, `PYTHON3_HOME`, which was specified at compile time, will be used for the Python 3 home directory. Environment variables are expanded `:set_env`. This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'pyxversion' **'pyx'** 'pyxversion' 'pyx'
 number (default depends on the build)
 global
 {only available when compiled with the `+python` or
 the `+python3` feature}

Specifies the python version used for `pyx*` functions and commands `python_x`. The default value is as follows:

Compiled with	Default
<code>+python</code> and <code>+python3</code>	0
only <code>+python</code>	2
only <code>+python3</code>	3

Available values are 0, 2 and 3.

If **'pyxversion'** is 0, it is set to 2 or 3 after the first execution of any `python2/3` commands or functions. E.g. `:py` sets to 2, and `:py3` sets to 3. `:pyx` sets it to 3 if Python 3 is available, otherwise sets to 2 if Python 2 is available.

See also: [has-pythonx](#)

If Vim is compiled with only `+python` or `+python3` setting **'pyxversion'** has no effect. The `pyx*` functions and commands are always the same as the compiled version.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'quickfixtextfunc' **'qftf'** 'quickfixtextfunc' 'qftf'
 string (default "")

global
{only available when compiled with the `+quickfix`
feature}

This option specifies a function to be used to get the text to display in the quickfix and location list windows. This can be used to customize the information displayed in the quickfix or location window for each entry in the corresponding quickfix or location list. See [quickfix-window-function](#) for an explanation of how to write the function and an example. The value can be the name of a function, a `lambda` or a `Funcref`. See [option-value-function](#) for more information.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'quotescape' 'qe' string (default "\\")
local to buffer 'quotescape' 'qe'

The characters that are used to escape quotes in a string. Used for objects like `a'`, `a"` and `a` a'`.

When one of the characters in this option is found inside a string, the following character will be skipped. The default value makes the text `"foo\"bar\""` considered to be one string.

'readonly' 'ro' boolean (default off)
local to buffer `local-noglobal` 'readonly' 'ro' 'noreadonly' 'noro'

If on, writes fail unless you use a `!!`. Protects you from accidentally overwriting a file. Default on when Vim is started in read-only mode (`"vim -R"`) or when the executable is called `"view"`. When using `":w!"` the **'readonly'** option is reset for the current buffer, unless the `'Z'` flag is in **'coptions'**.

When using the `":view"` command the **'readonly'** option is set for the newly edited buffer.

See **'modifiable'** for disallowing changes to the buffer.

'redrawtime' 'rdt' number (default 2000)
global 'redrawtime' 'rdt'
{only available when compiled with the `+reltime`
feature}

The time in milliseconds for redrawing the display. This applies to searching for patterns for **'hlsearch'**, `:match` highlighting and syntax highlighting.

When redrawing takes more than this many milliseconds no further matches will be highlighted.

For syntax highlighting the time applies per window. When over the limit syntax highlighting is disabled until `CTRL-L` is used.

This is used to avoid that Vim hangs when using a very complicated pattern.

'regexpengine' 're' number (default 0)
global 'regexpengine' 're'

This selects the default regexp engine. [two-engines](#)
The possible values are:

0	automatic selection
1	old engine
2	NFA engine

Note that when using the NFA engine and the pattern contains something that is not supported the pattern will not match. This is only useful for debugging the regexp engine.

Using automatic selection enables Vim to switch the engine, if the default engine becomes too costly. E.g., when the NFA engine uses too many states. This should prevent Vim from hanging on a combination of a complex pattern with long text.

'relativenumber' **'rnu'** **'norelativenumber'** **'nornu'**
'relativenumber' **'rnu'** boolean (default off)
local to window

Show the line number relative to the line with the cursor in front of each line. Relative line numbers help you use the [count](#) you can precede some vertical motion commands (e.g. j k + -) with, without having to calculate it yourself. Especially useful in combination with other commands (e.g. y d c < > gq gw =).

When the 'n' option is excluded from **'coptions'** a wrapped line will not use the column of line numbers (this is the default when **'compatible'** isn't set).

The **'numberwidth'** option can be used to set the room used for the line number.

When a long, wrapped line doesn't start with the first character, '-' characters are put before the number.

See [hl-LineNr](#) and [hl-CursorLineNr](#) for the highlighting used for the number.

The number in front of the cursor line also depends on the value of **'number'**, see [number_relativenumber](#) for all combinations of the two options.

'remap' **'noremap'**
boolean (default on)
global

Allows for mappings to work recursively. If you do not want this for a single entry, use the `:noremap[!]` command.

NOTE: To avoid portability problems with Vim scripts, always keep this option at the default "on". Only switch it off when working with old Vi scripts.

'renderoptions' **'rop'** **'renderoptions'** **'rop'**
string (default: empty)
global
{only available when compiled with GUI and DIRECTX on MS-Windows}

Select a text renderer and set its options. The options depend on the renderer.

Syntax:

`set rop=type:{renderer}(,{name}:{value})*`

Currently, only one optional renderer is available.

render behavior

directx Vim will draw text using DirectX (DirectWrite). It makes drawn glyphs more beautiful than default GDI. It requires **'encoding'** is "utf-8", and only works on MS-Windows Vista or newer version.

Options:

name	meaning	type	value
gamma	gamma	float	1.0 - 2.2 (maybe)
contrast	enhancedContrast	float	(unknown)
level	clearTypeLevel	float	(unknown)
geom	pixelGeometry	int	0 - 2 (see below)
renmode	renderingMode	int	0 - 6 (see below)
taamode	textAntialiasMode	int	0 - 3 (see below)
sctrlines	Scroll Lines	int	(deprecated)

See this URL for detail (except for sctrlines):

<https://msdn.microsoft.com/en-us/library/dd368190.aspx>

For geom: structure of a device pixel.

- 0 - DWRITE_PIXEL_GEOMETRY_FLAT
- 1 - DWRITE_PIXEL_GEOMETRY_RGB
- 2 - DWRITE_PIXEL_GEOMETRY_BGR

See this URL for detail:

<https://msdn.microsoft.com/en-us/library/dd368114.aspx>

For renmode: method of rendering glyphs.

- 0 - DWRITE_RENDERING_MODE_DEFAULT
- 1 - DWRITE_RENDERING_MODE_ALIASED
- 2 - DWRITE_RENDERING_MODE_GDI_CLASSIC
- 3 - DWRITE_RENDERING_MODE_GDI_NATURAL
- 4 - DWRITE_RENDERING_MODE_NATURAL
- 5 - DWRITE_RENDERING_MODE_NATURAL_SYMMETRIC
- 6 - DWRITE_RENDERING_MODE_OUTLINE

See this URL for detail:

<https://msdn.microsoft.com/en-us/library/dd368118.aspx>

For taamode: antialiasing mode used for drawing text.

- 0 - D2D1_TEXT_ANTIALIAS_MODE_DEFAULT
- 1 - D2D1_TEXT_ANTIALIAS_MODE_CLEARTYPE
- 2 - D2D1_TEXT_ANTIALIAS_MODE_GRAYSCALE
- 3 - D2D1_TEXT_ANTIALIAS_MODE_ALIASED

See this URL for detail:

<https://msdn.microsoft.com/en-us/library/dd368170.aspx>

For sctrlines:

This was used for optimizing scrolling behavior, however this is now deprecated. If specified, it is simply ignored.

(Where `^[` is an `<Esc>`, type **CTRL-V** `<Esc>` to insert it)

'revins' **'ri'** `'revins'` `'ri'` `'norevins'` `'nori'`
boolean (default off)
global
{only available when compiled with the `+rightleft` feature}

Inserting characters in Insert mode will work backwards. See "typing backwards" `ins-reverse`. This option can be toggled with the `CTRL-_` command in Insert mode, when `'allowrevins'` is set.

NOTE: This option is reset when `'compatible'` is set.

This option is reset when `'paste'` is set and restored when `'paste'` is reset.

'rightleft' **'rl'** `'rightleft'` `'rl'` `'norightleft'` `'norl'`
boolean (default off)
local to window
{only available when compiled with the `+rightleft` feature}

When on, display orientation becomes right-to-left, i.e., characters that are stored in the file appear from the right to the left.

Using this option, it is possible to edit files for languages that are written from the right to the left such as Hebrew and Arabic.

This option is per window, so it is possible to edit mixed files simultaneously, or to view the same file in both ways (this is useful whenever you have a mixed text file with both right-to-left and left-to-right strings so that both sets are displayed properly in different windows). Also see `rileft.txt`.

'rightleftcmd' **'rlc'** `'rightleftcmd'` `'rlc'`
string (default "search")
local to window
{only available when compiled with the `+rightleft` feature}

Each word in this option enables the command line editing to work in right-to-left mode for a group of commands:

search `"/` and `"?"` commands

This is useful for languages such as Hebrew, Arabic and Farsi.

The `'rightleft'` option must be set for `'rightleftcmd'` to take effect.

'rubydll' `'rubydll'`
string (default: depends on the build)
global
{only available when compiled with the `+ruby/dyn` feature}

Specifies the name of the Ruby shared library. The default is `DYNAMIC_RUBY_DLL`, which was specified at compile time.

Environment variables are expanded `:set_env`.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'ruler'` `'ru'` `'noruler'` `'noru'`

'ruler' 'ru' boolean (default off, set in `defaults.vim`)
 global

Show the line and column number of the cursor position, separated by a comma. When there is room, the relative position of the displayed text in the file is shown on the far right:

```

Top      first line is visible
Bot      last line is visible
All      first and last line are visible
45%     relative position in the file

```

If **'rulerformat'** is set, it will determine the contents of the ruler. Each window has its own ruler. If a window has a status line, the ruler is shown there. Otherwise it is shown in the last line of the screen. If the statusline is given by **'statusline'** (i.e. not empty), this option takes precedence over **'ruler'** and **'rulerformat'**. If the number of characters displayed is different from the number of bytes in the text (e.g., for a TAB or a multibyte character), both the text column (byte number) and the screen column are shown, separated with a dash. For an empty line "0-1" is shown. For an empty buffer the line number will also be zero: "0,0-1". This option is reset when **'paste'** is set and restored when **'paste'** is reset. If you don't want to see the ruler all the time but want to know where you are, use "g **CTRL-G**" `g_CTRL-G` .

NOTE: This option is reset when **'compatible'** is set.

'rulerformat' 'ruf' string (default empty)
 global
 {not available when compiled without the `+statusline` feature}

When this option is not empty, it determines the content of the ruler string, as displayed for the **'ruler'** option. The format of this option is like that of **'statusline'**. This option cannot be set in a modeline when **'modelineexpr'** is off.

The default ruler width is 17 characters. To make the ruler 15 characters wide, put "%15(" at the start and "%)" at the end.

Example:

```
:set rulerformat=%15(%c%V\ %p%%%)
```

'runtimepath' 'rtp' string (default:
 'runtimepath' **'rtp'** **vimfiles**
 Unix: "\$HOME/.vim,
 \$VIM/vimfiles,
 \$VIMRUNTIME,
 \$VIM/vimfiles/after,
 \$HOME/.vim/after"
 Amiga: "home:vimfiles,
 \$VIM/vimfiles,
 \$VIMRUNTIME,
 \$VIM/vimfiles/after,
 home:vimfiles/after"
 MS-Windows: "\$HOME/vimfiles,


```

$VIM/vimfiles,
$VIMRUNTIME,
$VIM/vimfiles/after,
$HOME/vimfiles/after"
macOS: "$VIM:vimfiles,
$VIMRUNTIME,
$VIM:vimfiles:after"
Haiku: "$BE_USER_SETTINGS/vim,
$VIM/vimfiles,
$VIMRUNTIME,
$VIM/vimfiles/after,
$BE_USER_SETTINGS/vim/after"
VMS: "sys$login:vimfiles,
$VIM/vimfiles,
$VIMRUNTIME,
$VIM/vimfiles/after,
sys$login:vimfiles/after")

```

global

This is a list of directories which will be searched for runtime files:

filetype.vim	filetypes by file name	<code>new-filetype</code>
scripts.vim	filetypes by file contents	<code>new-filetype-scripts</code>
autoload/	automatically loaded scripts	<code>autoload-functions</code>
colors/	color scheme files	<code>:colorscheme</code>
compiler/	compiler files	<code>:compiler</code>
doc/	documentation	<code>write-local-help</code>
ftplugin/	filetype plugins	<code>write-filetype-plugin</code>
import/	files that are found by <code>`:import`</code>	
indent/	indent scripts	<code>indent-expression</code>
keymap/	key mapping files	<code>mbyte-keymap</code>
lang/	menu translations	<code>:menutrans</code>
menu.vim	GUI menus	<code>menu.vim</code>
pack/	packages	<code>:packadd</code>
plugin/	plugin scripts	<code>write-plugin</code>
print/	files for printing	<code>postscript-print-encoding</code>
spell/	spell checking files	<code>spell</code>
syntax/	syntax files	<code>mysyntaxfile</code>
tutor/	files for vimtutor	<code>tutor</code>

And any other file searched for with the `:runtime` command.

The defaults for most systems are setup to search five locations:

1. In your home directory, for your personal preferences.
2. In a system-wide Vim directory, for preferences from the system administrator.
3. In `$VIMRUNTIME`, for files distributed with Vim.
4. In the "after" directory in the system-wide Vim directory. This is for the system administrator to overrule or add to the distributed defaults (rarely needed) `after-directory`
5. In the "after" directory in your home directory. This is for personal preferences to overrule or add to the distributed defaults or system-wide settings (rarely needed).

More entries are added when using `packages` . If it gets very long then `:set rtp` will be truncated, use `:echo &rtp` to see the full string.

Note that, unlike `'path'`, no wildcards like `"**"` are allowed. Normal wildcards are allowed, but can significantly slow down searching for runtime files. For speed, use as few items as possible and avoid wildcards.

See `:runtime` .

Example:

```
:set runtimepath=~/.vimruntime,/mygroup/vim,$VIMRUNTIME
```

This will use the directory `"~/.vimruntime"` first (containing your personal Vim runtime files), then `"/mygroup/vim"` (shared between a group of people) and finally `"$VIMRUNTIME"` (the distributed runtime files).

You probably should always include `$VIMRUNTIME` somewhere, to use the distributed runtime files. You can put a directory before `$VIMRUNTIME` to find files which replace a distributed runtime files. You can put a directory after `$VIMRUNTIME` to find files which add to distributed runtime files.

When Vim is started with `--clean` the home directory entries are not included.

This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'scroll' **'scr'** `'scroll'` `'scr'`
 number (default: half the window height)
 local to window `local-noglobal`

Number of lines to scroll with `CTRL-U` and `CTRL-D` commands. Will be set to half the number of lines in the window when the window size changes. This may happen when enabling the `status-line` or `'tabline'` option after setting the `'scroll'` option.

If you give a count to the `CTRL-U` or `CTRL-D` command it will be used as the new value for `'scroll'`. Reset to half the window height with `":set scroll=0"`.

'scrollbind' **'scb'** `'scrollbind'` `'scb'` `'noscrollbind'` `'noscb'`
 boolean (default off)
 local to window

See also `scroll-binding` . When this option is set, scrolling the current window also scrolls other scrollbind windows (windows that also have this option set). This option is useful for viewing the differences between two versions of a file, see `'diff'`.

See `'scrollopt'` for options that determine how this option should be interpreted.

This option is mostly reset when splitting a window to edit another file. This means that `":split | edit file"` results in two windows with scroll-binding, but `":split file"` does not.

'scrollfocus' **'scf'** `'scrollfocus'` `'scf'` `'noscrollfocus'` `'noscf'`
 boolean (default off)
 global
 {only for MS-Windows GUI}

When using the scroll wheel and this option is set, the window under

the mouse pointer is scrolled. With this option off the current window is scrolled.
Systems other than MS-Windows always behave like this option is on.

'scrolljump' 'sj' number (default 1)
 global
Minimal number of lines to scroll when the cursor gets off the screen (e.g., with "j"). Not used for scroll commands (e.g., **CTRL-E**, **CTRL-D**). Useful if your terminal scrolls very slowly.
When set to a negative number from -1 to -100 this is used as the percentage of the window height. Thus -50 scrolls half the window height.
NOTE: This option is set to 1 when **'compatible'** is set.

'scrolloff' 'so' number (default 0, set to 5 in `defaults.vim`)
 global or local to window `global-local`
Minimal number of screen lines to keep above and below the cursor. This will make some context visible around where you are working. If you set it to a very large value (999) the cursor line will always be in the middle of the window (except at the start or end of the file or when long lines wrap).
After using the local value, go back the global value with one of these two:
 `setlocal scrolloff<`
 `setlocal scrolloff=-1`
For scrolling horizontally see **'sidescrolloff'**.
NOTE: This option is set to 0 when **'compatible'** is set.

'scrollopt' 'sbo' string (default "ver,jump")
 global
This is a comma-separated list of words that specifies how **'scrollbind'** windows should behave. **'sbo'** stands for ScrollBind Options.
The following words are available:
 ver Bind vertical scrolling for **'scrollbind'** windows
 hor Bind horizontal scrolling for **'scrollbind'** windows
 jump Applies to the offset between two windows for vertical scrolling. This offset is the difference in the first displayed line of the bound windows. When moving around in a window, another **'scrollbind'** window may reach a position before the start or after the end of the buffer. The offset is not changed though, when moving back the **'scrollbind'** window will try to scroll to the desired position when possible.
When now making that window the current one, two things can be done with the relative offset:
1. When "jump" is not included, the relative offset is adjusted for the scroll position in the new current window. When going back to the other window, the new relative offset will be used.
2. When "jump" is included, the other windows are

scrolled to keep the same relative offset. When going back to the other window, it still uses the same relative offset.

Also see [scroll-binding](#) .

When **'diff'** mode is active there always is vertical scroll binding, even when "ver" isn't there.

'sections' 'sect' string (default "SHNH HUUnsh")
global

Specifies the nroff macros that separate sections. These are pairs of two letters (See [object-motions](#)). The default makes a section start at the nroff macros ".SH", ".NH", ".H", ".HU", ".nh" and ".sh".

'secure' boolean (default off)
global

When on, ":autocmd", shell and write commands are not allowed in ".vimrc" and ".exrc" in the current directory and map commands are displayed. Switch it off only if you know that you will not run into problems, or when the **'exrc'** option is off. On Unix this option is only used if the ".vimrc" or ".exrc" is not owned by you. This can be dangerous if the systems allows users to do a "chown". You better set **'secure'** at the end of your ~/.vimrc then.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'selection' 'sel' string (default "inclusive")
global

This option defines the behavior of the selection. It is only used in Visual and Select mode.

Possible values:

value	past line	inclusive
old	no	yes
inclusive	yes	yes
exclusive	yes	no

"past line" means that the cursor is allowed to be positioned one character past the line.

"inclusive" means that the last character of the selection is included in an operation. For example, when "x" is used to delete the selection.

When "old" is used and **'virtualedit'** allows the cursor to move past the end of line the line break still isn't included.

Note that when "exclusive" is used and selecting from the end backwards, you cannot include the last character of a line, when starting in Normal mode and **'virtualedit'** empty.

The **'selection'** option is set by the `:behave` command.

'selectmode' 'slm' string (default "")
global

This is a comma-separated list of words, which specifies when to start

Select mode instead of Visual mode, when a selection is started.

Possible values:

mouse	when using the mouse
key	when using shifted special keys
cmd	when using "v", "V" or CTRL-V

See [Select-mode](#) .

The **'selectmode'** option is set by the `:behave` command.

'sessionoptions' **'ssop'** string (default: "blank,buffers,curdir,folds,
help,options,tabpages,winsize,terminal")
global
{not available when compiled without the `+mksession`
feature}

Changes the effect of the `:mksession` command. It is a comma separated list of words. Each word enables saving and restoring something:

word	save and restore
blank	empty windows
buffers	hidden and unloaded buffers, not just those in windows
curdir	the current directory
fold	manually created folds, opened/closed folds and local fold options
globals	global variables that start with an uppercase letter and contain at least one lowercase letter. Only String and Number types are stored.
help	the help window
localoptions	options and mappings local to a window or buffer (not global values for local options)
options	all options and mappings (also global values for local options)
skiprtp	exclude 'runtimepath' and 'packpath' from the options
resize	size of the Vim window: 'lines' and 'columns'
sesdir	the directory in which the session file is located will become the current directory (useful with projects accessed over a network from different systems)
slash	backslashes in file names replaced with forward slashes
tabpages	all tab pages; without this only the current tab page is restored, so that you can make a session for each tab page separately
terminal	include terminal windows where the command can be restored
unix	with Unix end-of-line format (single <code><NL></code>), even when on Windows or DOS
winpos	position of the whole Vim window
winsize	window sizes

Don't include both "curdir" and "sesdir".

When neither "curdir" nor "sesdir" is included, file names are stored with absolute paths.

If you leave out "options" many things won't work well after restoring the session.

"slash" and "unix" are useful on Windows when sharing session files with Unix. The Unix version of Vim cannot source dos format scripts, but the Windows version of Vim can source unix format scripts.

'shell' **'sh'** string (default \$SHELL or "sh", Win32: "cmd.exe")
global

Name of the shell to use for ! and :! commands. When changing the value also check these options: **'shelltype'**, **'shellpipe'**, **'shellslash'**, **'shellredir'**, **'shellquote'**, **'shellxquote'** and **'shellcmdflag'**.

It is allowed to give an argument to the command, e.g. "csh -f".

See [option-backslash](#) about including spaces and backslashes.

Environment variables are expanded `:set_env` .

In [restricted-mode](#) shell commands will not be possible. This mode is used if the value of \$SHELL ends in "false" or "nologin".

If the name of the shell contains a space, you need to enclose it in quotes and escape the space. Example with quotes:

```
:set shell=\"c:\program\ files\unix\sh.exe\" -f
```

Note the backslash before each quote (to avoid starting a comment) and each space (to avoid ending the option value). Also **note** that the "-f" is not inside the quotes, because it is not part of the command name. Vim automatically recognizes the backslashes that are path separators.

Example with escaped space (Vim will do this when initializing the option from \$SHELL):

```
:set shell=/bin/with\\ space/sh
```

The resulting value of **'shell'** is "/bin/with\ space/sh", two backslashes are consumed by `:set` .

Under MS-Windows, when the executable ends in ".com" it must be included. Thus setting the shell to "command.com" or "4dos.com" works, but "command" and "4dos" do not work for all commands (e.g., filtering).

For unknown reasons, when using "4dos.com" the current directory is changed to "C:\". To avoid this set **'shell'** like this:

```
:set shell=command.com\ /c\ 4dos
```

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'shellcmdflag' **'shcf'** string (default: "-c";
Win32, when **'shell'** contains "powershell":
"-Command", or when it does not contain "sh"
somewhere: "/c")

global

Flag passed to the shell to execute "!" and ":!" commands; e.g., "bash.exe -c ls", "powershell.exe -Command dir", or "cmd.exe /c dir". For MS-Windows, the default is set according to the value of **'shell'**, to reduce the need to set this option by the user.

On Unix it can have more than one flag. Each white space separated part is passed as an argument to the shell command.

See [option-backslash](#) about including spaces and backslashes.

Also see [dos-shell](#) and [dos-powershell](#) for MS-Windows. This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

```
'shellpipe' 'sp'          string (default ">", ">%s 2>&1", "| tee", "|& tee"  
                          "2>&1| tee", or  
                          "2>&1 | Out-File -Encoding default")  
global  
{not available when compiled without the +quickfix  
feature}
```

String to be used to put the output of the `:"make"` command in the error file. See also `:make_makeprg` . See [option-backslash](#) about including spaces and backslashes.

The name of the temporary file can be represented by `%s` if necessary (the file name is appended automatically if no `%s` appears in the value of this option).

For the Amiga the default is `>`. For MS-Windows using powershell the default is `2>&1 | Out-File -Encoding default`, otherwise the default is `>%s 2>&1`. The output is directly saved in a file and not echoed to the screen.

For Unix the default is `| tee`. The stdout of the compiler is saved in a file and echoed to the screen. If the `'shell'` option is `csh` or `tcsh` after initializations, the default becomes `|& tee`. If the `'shell'` option is `sh`, `ksh`, `mksh`, `pdksh`, `zsh`, `zsh-beta`, `bash`, `fish`, `ash` or `dash` the default becomes `2>&1| tee`. This means that stderr is also included. Before using the `'shell'` option a path is removed, thus `/bin/sh` uses `sh`.

For Unix and MS-Windows, when the `'shell'` option is `pwsh` the default becomes `>%s 2>&1` and the output is not echoed to the screen.

The initialization of this option is done after reading the `.vimrc` and the other initializations, so that when the `'shell'` option is set there, the `'shellpipe'` option changes automatically, unless it was explicitly set before.

When `'shellpipe'` is set to an empty string, no redirection of the `:"make` output will be done. This is useful if you use a `'makeprg'` that writes to `'makeef'` by itself. If you want no piping, but do want to include the `'makeef'`, set `'shellpipe'` to a single space.

Don't forget to precede the space with a backslash: `:"set sp=\ "`.

In the future pipes may be used for filtering and this option will become obsolete (at least for Unix).

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

```
'shellquote' 'shq'      string (default: "")  
global
```

Quoting character(s), put around the command passed to the shell, for the `!"` and `:"!` commands. The redirection is kept outside of the quoting. See `'shellxquote'` to include the redirection. It's probably not useful to set both options.

This is an empty string by default. Only known to be useful for third-party shells on MS-Windows-like systems, such as the MKS Korn Shell or bash, where it should be `"\"`. See [dos-shell](#) .

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'shellredir' 'srr'` string (default ">", ">&", ">%s 2>&1", or "2>&1 | Out-File -Encoding default")
global

String to be used to put the output of a filter command in a temporary file. See also `:!`. See `option-backslash` about including spaces and backslashes.

The name of the temporary file can be represented by "%s" if necessary (the file name is appended automatically if no %s appears in the value of this option).

The default is ">". For Unix, if the `'shell'` option is "csh" or "tcsh" during initializations, the default becomes ">&". If the `'shell'` option is "sh", "ksh", "mksh", "pdksh", "zsh", "zsh-beta", "bash", "fish", or "pwsh", the default becomes ">%s 2>&1". This means that stderr is also included. For Win32, the Unix checks are done and additionally "cmd" is checked for, which makes the default ">%s 2>&1", and "powershell" is checked for which makes the default "2>&1 | Out-File -Encoding default" (see `dos-powershell`). Also, the same names with ".exe" appended are checked for.

The initialization of this option is done after reading the ".vimrc" and the other initializations, so that when the `'shell'` option is set there, the `'shellredir'` option changes automatically unless it was explicitly set before.

In the future pipes may be used for filtering and this option will become obsolete (at least for Unix).

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'shellslash' 'ssl'` boolean (default off)
global
{only for MS-Windows}

When set, a forward slash is used when expanding file names. This is useful when a Unix-like shell is used instead of cmd.exe, pwsh.exe, or powershell.exe. Backward slashes can still be typed, but they are changed to forward slashes by Vim.

Note that setting or resetting this option has no effect for some existing file names, thus this option needs to be set before opening any file for best results. This might change in the future.

`'shellslash'` only works when a backslash can be used as a path separator. To test if this is so use:

```
if exists('+shellslash')
```

Also see `'completeslash'`.

`'shelltemp' 'stmp'` boolean (Vi default off, Vim default on)
global

When on, use temp files for shell commands. When off use a pipe. When using a pipe is not possible temp files are used anyway.

Currently a pipe is only supported on Unix and MS-Windows 2K and later. You can check it with:


```
:if has("filterpipe")
```

The advantage of using a pipe is that nobody can read the temp file and the **'shell'** command does not need to support redirection.

The advantage of using a temp file is that the file type and encoding can be detected.

The `FilterReadPre` , `FilterReadPost` and `FilterWritePre` , `FilterWritePost` autocommands event are not triggered when **'shelltemp'** is off.

The `system()` function does not respect this option and always uses temp files.

NOTE: This option is set to the Vim default value when **'compatible'** is reset.

```
'shelltype' 'st'          number (default 0)          'shelltype' 'st'  
                          global  
                          {only for the Amiga}
```

On the Amiga this option influences the way how the commands work which use a shell.

0 and 1: always use the shell

2 and 3: use the shell only to filter lines

4 and 5: use shell only for **'sh'** command

When not using the shell, the command is executed directly.

0 and 2: use `"shell 'shellcmdflag' cmd"` to start external commands

1 and 3: use `"shell cmd"` to start external commands

```
'shellxescape' 'sxe'     string (default: "");          'shellxescape' 'sxe'  
                          for MS-Windows: "\"&|<>()@^")  
                          global
```

When **'shellxquote'** is set to `"("` then the characters listed in this option will be escaped with a `^` character. This makes it possible to execute most external commands with `cmd.exe`.

This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

```
'shellxquote' 'sxq'     string (default: "");          'shellxquote' 'sxq'  
                          for Win32, when 'shell' is cmd.exe: "("  
                          for Win32, when 'shell' is  
                          powershell.exe: "\""  
                          for Win32, when 'shell' contains "sh"  
                          somewhere: "\""  
                          for Unix, when using system(): "\""  
                          global
```

Quoting character(s), put around the command passed to the shell, for the `!"` and `:!"` commands. Includes the redirection. See

'shellxquote' to exclude the redirection. It's probably not useful to set both options.

When the value is `'('` then `')` is appended. When the value is `'('` then `')` is appended.

When the value is `'('` then also see **'shellxescape'**.

This is an empty string by default on most systems, but is known to be

useful for on Win32 version, either for cmd.exe, powershell.exe, or pwsh.exe which automatically strips off the first and last quote on a command, or 3rd-party shells such as the MKS Korn Shell or bash, where it should be "\". The default is adjusted according the value of **'shell'**, to reduce the need to set this option by the user. See [dos-shell](#) .

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'shiftround' 'sr' **'shiftround' 'sr' 'noshiftround' 'nosr'**
 boolean (default off)
 global

Round indent to multiple of **'shiftwidth'**. Applies to > and < commands. **CTRL-T** and **CTRL-D** in Insert mode always round the indent to a multiple of **'shiftwidth'** (this is Vi compatible).

NOTE: This option is reset when **'compatible'** is set.

'shiftwidth' 'sw' **'shiftwidth' 'sw'**
 number (default 8)
 local to buffer

Number of spaces to use for each step of (auto)indent. Used for **'cindent'** , **>>** , **<<** , etc.

When zero the **'tabstop'** value will be used. Use the [shiftwidth\(\)](#) function to get the effective shiftwidth value.

'shortmess' 'shm' **'shortmess' 'shm'**
 string (Vim default "filnxtToOS", Vi default: "S",
 POSIX default: "AS")
 global **E1336**

This option helps to avoid all the [hit-enter](#) prompts caused by file messages, for example with **CTRL-G**, and to avoid some other messages.

It is a list of flags:

flag	meaning when present	
f	use "(3 of 5)" instead of "(file 3 of 5)"	shm-f
i	use "[noeol]" instead of "[Incomplete last line]"	shm-i
l	use "999L, 888B" instead of "999 lines, 888 bytes"	shm-l
m	use "[+]" instead of "[Modified]"	shm-m
n	use "[New]" instead of "[New File]"	shm-n
r	use "[RO]" instead of "[readonly]"	shm-r
w	use "[w]" instead of "written" for file write message and "[a]" instead of "appended" for ':w >> file' command	shm-w
x	use "[dos]" instead of "[dos format]", "[unix]" instead of "[unix format]" and "[mac]" instead of "[mac format]"	shm-x
a	all of the above abbreviations	shm-a
o	overwrite message for writing a file with subsequent message for reading a file (useful for ":wn" or when 'autowrite' on)	shm-o
O	message for reading a file overwrites any previous message; also for quickfix message (e.g., ":cn")	shm-O
s	don't give "search hit BOTTOM, continuing at TOP" or "search hit TOP, continuing at BOTTOM" messages; when using the search count do not show "W" after the count message (see	shm-s

S below)

t	truncate file message at the start if it is too long to fit on the command-line, "<" will appear in the left most column; ignored in Ex mode	shm-t
T	truncate other messages in the middle if they are too long to fit on the command line; "... " will appear in the middle; ignored in Ex mode	shm-T
W	don't give "written" or "[w]" when writing a file	shm-W
A	don't give the "ATTENTION" message when an existing swap file is found	shm-A
I	don't give the intro message when starting Vim, see <code>:intro</code>	shm-I
c	don't give <code>ins-completion-menu</code> messages; for example, "-- XXX completion (YYY)", "match 1 of 2", "The only match", "Pattern not found", "Back at original", etc.	shm-c
C	don't give messages while scanning for ins-completion items, for instance "scanning tags"	shm-C
q	use "recording" instead of "recording @a"	shm-q
F	don't give the file info when editing a file, like <code>:silent`</code> was used for the command; note that this also affects messages from autocommands and <code>'autoread'</code> reloading	shm-F
S	do not show search count message when searching, e.g. "[1/5]"	shm-S

This gives you the opportunity to avoid that a change between buffers requires you to hit `<Enter>`, but still gives as useful a message as possible for the space available. To get the whole message that you would have got with `'shm'` empty, use `:file!`

Useful values:

shm=	No abbreviation of message.
shm=a	Abbreviation, but no loss of information.
shm=at	Abbreviation, and truncate message when necessary.

NOTE: This option is set to the Vi default value when `'compatible'` is set and to the Vim default value when `'compatible'` is reset.

		<code>'shortname'</code>	<code>'sn'</code>	<code>'noshortname'</code>	<code>'nosn'</code>
<code>'shortname'</code>	<code>'sn'</code>	boolean (default off)			
		local to buffer			

Filenames are assumed to be 8 characters plus one extension of 3 characters. Multiple dots in file names are not allowed. When this option is on, dots in file names are replaced with underscores when adding an extension ("`~`" or "`.swp`"). This option is useful when editing files on an MS-DOS compatible filesystem, e.g., `messydos` or `crossdos`.

		<code>'showbreak'</code>	<code>'sbr'</code>	E595
<code>'showbreak'</code>	<code>'sbr'</code>	string (default "")		
		global or local to window <code>global-local</code>		
		{not available when compiled without the <code>+linebreak</code> feature}		

String to put at the start of lines that have been wrapped. Useful values are `>` or `+++`:

```
:set showbreak=>
```

Note the backslash to escape the trailing space. It's easier like this:

```
:let &showbreak = '+++ '
```

Only printable single-cell characters are allowed, excluding <Tab> and comma (in a future version the comma might be used to separate the part that is shown at the end and at the start of a line).

The characters are highlighted according to the '@' flag in **'highlight'**.

Note that tabs after the showbreak will be displayed differently.

If you want the **'showbreak'** to appear in between line numbers, add the "n" flag to **'coptions'**.

A window-local value overrules a global value. If the global value is set and you want no value in the current window use NONE:

```
:setlocal showbreak=NONE
```

```
'showcmd' 'sc'          'showcmd' 'sc' 'noshowcmd' 'nosc'  
boolean (Vim default: on, off for Unix,  
Vi default: off, set in defaults.vim )  
global
```

Show (partial) command in the last line of the screen. Set this option off if your terminal is slow.

In Visual mode the size of the selected area is shown:

- When selecting characters within a line, the number of characters.

If the number of bytes is different it is also displayed: "2-6" means two characters and six bytes.

- When selecting more than one line, the number of lines.

- When selecting a block, the size in screen characters:
{lines}x{columns}.

This information can be displayed in an alternative location using the **'showcmdloc'** option.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

```
'showcmdloc' 'sloc'    'showcmdloc' 'sloc'  
string (default "last")  
global
```

This option can be used to display the (partially) entered command in another location. Possible values are:

last Last line of the screen (default).

statusline Status line of the current window.

tabline First line of the screen if **'showtabline'** is enabled.

Setting this option to "statusline" or "tabline" means that these will be redrawn whenever the command changes, which can be on every key pressed.

The %S **'statusline'** item can be used in **'statusline'** or **'tabline'** to place the text. Without a custom **'statusline'** or **'tabline'** it will be displayed in a convenient location.

```
'showfulltag' 'sft'    'showfulltag' 'sft' 'noshowfulltag' 'nosft'  
boolean (default off)  
global
```

When completing a word in insert mode (see [ins-completion](#)) from the tags file, show both the tag name and a tidied-up form of the search pattern (if there is one) as possible matches. Thus, if you have

matched a C function, you can see a template for what arguments are required (coding style permitting).

Note that this doesn't work well together with having "longest" in '**completeopt**', because the completion from the search pattern may not match the typed text.

'showmatch' 'sm' **'showmatch'** **'sm'** **'noshowmatch'** **'nosm'**
boolean (default off)
global

When a bracket is inserted, briefly jump to the matching one. The jump is only done if the match can be seen on the screen. The time to show the match can be set with '**matchtime**'.

A Beep is given if there is no match (no matter if the match can be seen or not).

This option is reset when '**paste**' is set and restored when '**paste**' is reset.

When the 'm' flag is not included in '**coptions**', typing a character will immediately move the cursor back to where it belongs.

See the "sm" field in '**guicursor**' for setting the cursor shape and blinking when showing the match.

The '**matchpairs**' option can be used to specify the characters to show matches for. '**rightleft**' and '**revins**' are used to look for opposite matches.

Also see the matchparen plugin for highlighting the match when moving around [pi_paren.txt](#) .

Note: Use of the short form is rated PG.

'showmode' 'smd' **'showmode'** **'smd'** **'noshowmode'** **'nosmd'**
boolean (Vim default: on, Vi default: off)
global

If in Insert, Replace or Visual mode put a message on the last line. Use the 'M' flag in '**highlight**' to set the type of highlighting for this message.

When **XIM** may be used the message will include "XIM". But this doesn't mean XIM is really active, especially when '**imactivatekey**' is not set.

NOTE: This option is set to the Vi default value when '**compatible**' is set and to the Vim default value when '**compatible**' is reset.

'showtabline' 'stal' **'showtabline'** **'stal'**
number (default 1)
global

The value of this option specifies when the line with tab page labels will be displayed:

- 0: never
- 1: only if there are at least two tab pages
- 2: always

This is both for the GUI and non-GUI implementation of the tab pages line.

See [tab-page](#) for more information about tab pages.

'sidescroll' 'ss' **'sidescroll'** **'ss'**
number (default 0)
global

'smartindent' **'si'** **'smartindent'** **'si'** **'nosmartindent'** **'nosi'**
boolean (default off)
local to buffer

Do smart autoindenting when starting a new line. Works for C-like programs, but can also be used for other languages. **'cindent'** does something like this, works better in most cases, but is more strict, see [C-indenting](#). When **'cindent'** is on or **'indentexpr'** is set, setting **'si'** has no effect. **'indentexpr'** is a more advanced alternative.

Normally **'autoindent'** should also be on when using **'smartindent'**.

An indent is automatically inserted:

- After a line ending in '{'.
- After a line starting with a keyword from **'cinwords'**.
- Before a line starting with '}' (only with the "O" command).

When typing '}' as the first character in a new line, that line is given the same indent as the matching '{'.

When typing '#' as the first character in a new line, the indent for that line is removed, the '#' is put in the first column. The indent is restored for the next line. If you don't want this, use this mapping: `":inoremap # X^H#"`, where ^H is entered with **CTRL-V CTRL-H**. When using the ">>" command, lines starting with '#' are not shifted right.

NOTE: This option is reset when **'compatible'** is set.

This option is reset when **'paste'** is set and restored when **'paste'** is reset.

'smarttab' **'sta'** **'smarttab'** **'sta'** **'nosmarttab'** **'nosta'**
boolean (default off)
global

When on, a <Tab> in front of a line inserts blanks according to **'shiftwidth'**. **'tabstop'** or **'softtabstop'** is used in other places. A <BS> will delete a **'shiftwidth'** worth of space at the start of the line.

When off, a <Tab> always inserts blanks according to **'tabstop'** or **'softtabstop'**. **'shiftwidth'** is only used for shifting text left or right [shift-left-right](#).

What gets inserted (a <Tab> or spaces) depends on the **'expandtab'** option. Also see [ins-expandtab](#). When **'expandtab'** is not set, the number of spaces is minimized by using <Tab>s.

This option is reset when **'paste'** is set and restored when **'paste'** is reset.

NOTE: This option is reset when **'compatible'** is set.

'smoothscroll' **'sms'** **'smoothscroll'** **'sms'** **'nosmoothscroll'** **'nosms'**
boolean (default off)
local to window

Scrolling works with screen lines. When **'wrap'** is set and the first line in the window wraps part of it may not be visible, as if it is above the window. "<<<" is displayed at the start of the first line, highlighted with [hl-NonText](#).

You may also want to add "lastline" to the **'display'** option to show as much of the last line as possible.

NOTE: only partly implemented, currently works with **CTRL-E**, **CTRL-Y** and scrolling with the mouse.

'softtabstop' **'sts'** **'softtabstop'** **'sts'**
 number (default 0)
 local to buffer

Number of spaces that a <Tab> counts for while performing editing operations, like inserting a <Tab> or using <BS>. It "feels" like <Tab>s are being inserted, while in fact a mix of spaces and <Tab>s is used. This is useful to keep the **'ts'** setting at its standard value of 8, while being able to edit like it is set to **'sts'**. However, commands like "x" still work on the actual characters. When **'sts'** is zero, this feature is off. When **'sts'** is negative, the value of **'shiftwidth'** is used. **'softtabstop'** is set to 0 when the **'paste'** option is set and restored when **'paste'** is reset. See also [ins-expandtab](#) . When **'expandtab'** is not set, the number of spaces is minimized by using <Tab>s. The 'L' flag in **'coptions'** changes how tabs are used when **'list'** is set.

NOTE: This option is set to 0 when **'compatible'** is set.

If Vim is compiled with the [+vartabs](#) feature then the value of **'softtabstop'** will be ignored if **'varsofttabstop'** is set to anything other than an empty string.

'spell' **'spell'** **'nospell'**
 boolean (default off)
 local to window
 {not available when compiled without the [+syntax](#)
 feature}

When on spell checking will be done. See [spell](#) . The languages are specified with **'spelllang'**.

'spellcapcheck' **'spc'** **'spellcapcheck'** **'spc'**
 string (default "[.?!]_[\])'" \t]\+")
 local to buffer
 {not available when compiled without the [+syntax](#)
 feature}

Pattern to locate the end of a sentence. The following word will be checked to start with a capital letter. If not then it is highlighted with SpellCap [hl-SpellCap](#) (unless the word is also badly spelled). When this check is not wanted make this option empty. Only used when **'spell'** is set. Be careful with special characters, see [option-backslash](#) about including spaces and backslashes. To set this option automatically depending on the language, see [set-spc-auto](#) .

'spellfile' **'spf'** **'spellfile'** **'spf'**
 string (default empty)
 local to buffer
 {not available when compiled without the [+syntax](#)
 feature}

Name of the word list file where words are added for the [zg](#) and [zw](#) commands. It must end in ".{encoding}.add". You need to include the

path, otherwise the file is placed in the current directory. The path may include characters from **'isfname'**, space, comma and '@'.

E765

It may also be a comma-separated list of names. A count before the **zg** and **zw** commands can be used to access each. This allows using a personal word list file and a project word list file.

When a word is added while this option is empty Vim will set it for you: Using the first directory in **'runtimepath'** that is writable. If there is no "spell" directory yet it will be created. For the file name the first language name that appears in **'spelllang'** is used, ignoring the region.

The resulting ".spl" file will be used for spell checking, it does not have to appear in **'spelllang'**.

Normally one file is used for all regions, but you can add the region name if you want to. However, it will then only be used when **'spellfile'** is set to it, for entries in **'spelllang'** only files without region name will be found.

This option cannot be set from a **modeline** or in the **sandbox**, for security reasons.

'spelllang'	'spl'	
		'spelllang'
		'spl'
		string (default "en")
		local to buffer
		{not available when compiled without the +syntax
		feature}

A comma-separated list of word list names. When the **'spell'** option is on spellchecking will be done for these languages. Example:

```
set spelllang=en_us,nl,medical
```

This means US English, Dutch and medical words are recognized. Words that are not recognized will be highlighted.

The word list name must consist of alphanumeric characters, a dash or an underscore. It should not include a comma or dot. Using a dash is recommended to separate the two letter language name from a specification. Thus "en-rare" is used for rare English words.

A region name must come last and have the form "_xx", where "xx" is the two-letter, lower case region name. You can use more than one region by listing them: "en_us,en_ca" supports both US and Canadian English, but not words specific for Australia, New Zealand or Great Britain. (Note: currently en_au and en_nz dictionaries are older than en_ca, en_gb and en_us).

If the name "cjk" is included East Asian characters are excluded from spell checking. This is useful when editing text that also has Asian words.

Note that the "medical" dictionary does not exist, it is just an example of a longer name.

E757

As a special case the name of a .spl file can be given as-is. The first "_xx" in the name is removed and used as the region name (_xx is an underscore, two letters and followed by a non-letter). This is mainly for testing purposes. You must make sure the correct encoding is used, Vim doesn't check it.

When **'encoding'** is set the word lists are reloaded. Thus it's a good idea to set **'spelllang'** after setting **'encoding'** to avoid loading the files twice.

How the related spell files are found is explained here: [spell-load](#) .

If the [spellfile.vim](#) plugin is active and you use a language name for which Vim cannot find the .spl file in '[runtimepath](#)' the plugin will ask you if you want to download the file.

After this option has been set successfully, Vim will source the files "spell/LANG.vim" in '[runtimepath](#)'. "LANG" is the value of '[spelllang](#)' up to the first character that is not an ASCII letter or number and not a dash. Also see [set-spc-auto](#) .

'spelloptions' '**spo**' ['spelloptions'](#) ['spo'](#)
string (default "")
local to buffer
{not available when compiled without the [+syntax](#) feature}

A comma-separated list of options for spell checking:

camel When a word is CamelCased, assume "Cased" is a separate word: every upper-case character in a word that comes after a lower case character indicates the start of a new word.

'spellsuggest' '**sps**' ['spellsuggest'](#) ['sps'](#)
string (default "best")
global
{not available when compiled without the [+syntax](#) feature}

Methods used for spelling suggestions. Both for the [z=](#) command and the [spellsuggest\(\)](#) function. This is a comma-separated list of items:

best Internal method that works best for English. Finds changes like "fast" and uses a bit of sound-a-like scoring to improve the ordering.

double Internal method that uses two methods and mixes the results. The first method is "fast", the other method computes how much the suggestion sounds like the bad word. That only works when the language specifies sound folding. Can be slow and doesn't always give better results.

fast Internal method that only checks for simple changes: character inserts/deletes/swaps. Works well for simple typing mistakes.

{number} The maximum number of suggestions listed for [z=](#) . Not used for [spellsuggest\(\)](#) . The number of suggestions is never more than the value of '[lines](#)' minus two.

timeout:{millisec} Limit the time searching for suggestions to {millisec} milli seconds. Applies to the following methods. When omitted the limit is 5000. When

negative there is no limit. {only works when built with the `+reltime` feature}

`file:{filename}` Read file `{filename}`, which must have two columns, separated by a slash. The first column contains the bad word, the second column the suggested good word. Example:

`theribal/terrible`

Use this for common mistakes that do not appear at the top of the suggestion list with the internal methods. Lines without a slash are ignored, use this for comments.

The word in the second column must be correct, otherwise it will not be used. Add the word to an ".add" file if it is currently flagged as a spelling mistake.

The file is used for all languages.

`expr:{expr}` Evaluate expression `{expr}`. Use a function to avoid trouble with spaces. Best is to call a function without arguments, see `expr-option-function`. `v:val` holds the badly spelled word. The expression must evaluate to a List of Lists, each with a suggestion and a score.

Example:

`[['the', 33], ['that', 44]]`

Set `'verbose'` and use `z=` to see the scores that the internal methods use. A lower score is better.

This may invoke `spellsuggest()` if you temporarily set `'spellsuggest'` to exclude the "expr:" part.

Errors are silently ignored, unless you set the `'verbose'` option to a non-zero value.

Only one of "best", "double" or "fast" may be used. The others may appear several times in any order. Example:

```
:set sps=file:~/vim/sugg,best,expr:MySuggest()
```

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

`'splitbelow' 'sb'` `'splitbelow'` `'sb'` `'nosplitbelow'` `'nosb'`
boolean (default off)
global

When on, splitting a window will put the new window below the current one. `:split`

`'splitkeep' 'spk'` `'splitkeep'` `'spk'`
string (default "cursor")
global

The value of this option determines the scroll behavior when opening, closing or resizing horizontal splits.

Possible values are:

cursor Keep the same relative cursor position.

screen Keep the text on the same screen line.
topline Keep the topline the same.

For the "screen" and "topline" values, the cursor position will be changed when necessary. In this case, the jumplist will be populated with the previous cursor position. For "screen", the text cannot always be kept on the same screen line when **'wrap'** is enabled.

'splitright' 'spr' **'splitright' 'spr' 'nosplitright' 'nospr'**
 boolean (default off)
 global

When on, splitting a window will put the new window right of the current one. **:vsplit**

'startofline' 'sol' **'startofline' 'sol' 'nostartofline' 'nosol'**
 boolean (default on)
 global

When "on" the commands listed below move the cursor to the first non-blank of the line. When off the cursor is kept in the same column (if possible). This applies to the commands:

- **CTRL-D**, **CTRL-U**, **CTRL-B**, **CTRL-F**, "G", "H", "M", "L", "gg"
- "d", "<<" and ">>" with a linewise operator
- "%" with a count
- buffer changing commands (CTRL-^, :bnext, :bNext, etc.)
- Ex commands that only has a line number, e.g., ":25" or ":+".

In case of buffer changing commands the cursor is placed at the column where it was the last time the buffer was edited.

NOTE: This option is set when **'compatible'** is set.

'statusline' 'stl' **'statusline' 'stl' E540 E542**
 string (default empty)
 global or local to window **global-local**
 {not available when compiled without the **+statusline**
 feature}

When non-empty, this option determines the content of the status line. Also see **status-line** .

The option consists of printf style '%' items interspersed with normal text. Each status line item is of the form:

`%-0{minwid}.{maxwid}{item}`

All fields except the **{item}** are optional. A single percent sign can be given as "%%".

When the option starts with "%!" then it is used as an expression, evaluated and the result is used as the option value. Example:

:set statusline=%!MyStatusLine()

The **g:statusline_winid** variable will be set to the **window-ID** of the window that the status line belongs to.

The result can contain **%{}** items that will be evaluated too.

Note that the "%!" expression is evaluated in the context of the current window and buffer, while **%{}** items are evaluated in the context of the window that the statusline belongs to.

When there is error while evaluating the option then it will be made

empty to avoid further errors. Otherwise screen updating would loop. When the result contains unprintable characters the result is unpredictable.

Note that the only effect of **'ruler'** when this option is set (and **'laststatus'** is 2) is controlling the output of **CTRL-G** .

field	meaning	
-	Left justify the item. The default is right justified when minwid is larger than the length of the item.	
0	Leading zeroes in numeric items. Overridden by '-'. minwid	Minimum width of the item, padding as set by '-' & '0'. Value must be 50 or less.
maxwid	Maximum width of the item. Truncation occurs with a '<' on the left for text items. Numeric items will be shifted down to maxwid-2 digits followed by '>'number where number is the amount of missing digits, much like an exponential notation.	
item	A one letter code as described below.	

Following is a description of the possible statusline items. The second character in "item" is the type:

- N for number
- S for string
- F for flags as described below
- not applicable

item	meaning
f S	Path to the file in the buffer, as typed or relative to current directory.
F S	Full path to the file in the buffer.
t S	File name (tail) of file in the buffer.
m F	Modified flag, text is "[+]"; "[-]" if 'modifiable' is off.
M F	Modified flag, text is ",+" or ",-".
r F	Readonly flag, text is "[RO]".
R F	Readonly flag, text is ",RO".
h F	Help buffer flag, text is "[help]".
H F	Help buffer flag, text is ",HLP".
w F	Preview window flag, text is "[Preview]".
W F	Preview window flag, text is ",PRV".
y F	Type of file in the buffer, e.g., "[vim]". See 'filetype' .
Y F	Type of file in the buffer, e.g., ",VIM". See 'filetype' .
q S	"[Quickfix List]", "[Location List]" or empty.
k S	Value of "b:keymap_name" or 'keymap' when :lmap mappings are being used: "<keymap>"
n N	Buffer number.
b N	Value of character under cursor.
B N	As above, in hexadecimal.
o N	Byte number in file of byte under cursor, first byte is 1. Mnemonic: Offset from start of file (with one added) {not available when compiled without +byte_offset feature}
O N	As above, in hexadecimal.
N N	Printer page number. (Only works in the 'printhead' option.)
l N	Line number.

- L N Number of lines in buffer.
- c N Column number (byte index).
- v N Virtual column number (screen column).
- V N Virtual column number as `-{num}`. Not displayed if equal to 'c'.
- p N Percentage through file in lines as in [CTRL-G](#).
- P S Percentage through file of displayed window. This is like the percentage described for '[ruler](#)'. Always 3 in length, unless translated.
- S S '[showcmd](#)' content, see '[showcmdloc](#)'.
- a S Argument list status as in default title. (`{current}` of `{max}`) Empty if the argument file count is zero or one.
- { NF Evaluate expression between `'%{'` and `'}'` and substitute result. **Note** that there is no `'%'` before the closing `'}'`. The expression cannot contain a `'}'` character, call a function to work around that. See `stl-#{` below.
- {% - This is almost same as { except the result of the expression is re-evaluated as a statusline format string. Thus if the return value of `expr` contains % items they will get expanded. The expression can contain the `}` character, the end of expression is denoted by `%}`.
For example:

```
func! Stl_filename() abort
    return "%t"
endfunc
`stl=%{Stl_filename()}` results in `"%t"`
`stl=%{%Stl_filename()}` results in `"Name of current file"`
```
- %} - End of `{%` expression
- (- Start of item group. Can be used for setting the width and alignment of a section. Must be followed by %) somewhere.
-) - End of item group. No width fields allowed.
- T N For '[tabline](#)': start of tab page N label. Use %T after the last label. This information is used for mouse clicks.
- X N For '[tabline](#)': start of close tab N label. Use %X after the label, e.g.: `%3Xclose%X`. Use `%999X` for a "close current tab" mark. This information is used for mouse clicks.
- < - Where to truncate line if too long. Default is at the start. No width fields allowed.
- = - Separation point between alignment sections. Each section will be separated by an equal number of spaces. With one %= what comes after it will be right-aligned. With two %= there is a middle part, with white space left and right of it. No width fields allowed.
- # - Set highlight group. The name must follow and then a # again. Thus use `%#HLname#` for highlight group HLname. The same highlighting is used, also for the statusline of non-current windows.
- * - Set highlight group to User{N}, where {N} is taken from the minwid field, e.g. `%1*`. Restore normal highlight with `%*` or `%0*`. The difference between User{N} and StatusLine will be applied to StatusLineNC for the statusline of non-current windows. The number N must be between 1 and 9. See [hl-User1..9](#)

When displaying a flag, Vim removes the leading comma, if any, when that flag comes right after plaintext. This will make a nice display

when flags are used like in the examples below.

When all items in a group becomes an empty string (i.e. flags that are not set) and a minwid is not set for the group, the whole group will become empty. This will make a group like the following disappear completely from the statusline when none of the flags are set.

```
:set statusline=...%( \ [%M%R%H]%)...
```

Beware that an expression is evaluated each and every time the status line is displayed.

```
stl-%{ g:actual_curbuf g:actual_curwin
```

While evaluating `%{}` the current buffer and current window will be set temporarily to that of the window (and buffer) whose statusline is currently being drawn. The expression will evaluate in this context. The variable `"g:actual_curbuf"` is set to the ``bufnr()`` number of the real current buffer and `"g:actual_curwin"` to the `window-ID` of the real current window. These values are strings.

The `'statusline'` option will be evaluated in the `sandbox` if set from a modeline, see `sandbox-option`.

This option cannot be set in a modeline when `'modelineexpr'` is off.

It is not allowed to change text or jump to another window while evaluating `'statusline'` `textlock`.

If the statusline is not updated when you want it (e.g., after setting a variable that's used in an expression), you can force an update by using `:redrawstatus``.

A result of all digits is regarded a number for display purposes. Otherwise the result is taken as flag text and applied to the rules described above.

Watch out for errors in expressions. They may render Vim unusable! If you are stuck, hold down `':'` or `'Q'` to get a prompt, then quit and edit your `.vimrc` or whatever with `"vim --clean"` to get it right.

Examples:

Emulate standard status line with `'ruler'` set

```
:set statusline=%<%f\ %h%m%r%=-14.(%l,%c%V%)\ %P
```

Similar, but add ASCII value of char under the cursor (like `"ga"`)

```
:set statusline=%<%f%h%m%r%=%b\ 0x%B\ \ %l,%c%V\ %P
```

Display byte count and byte value, modified flag in red.

```
:set statusline=%<%f%=\ [%1*M%*%n%R%H]\ %-19(%3l,%02c%03V%)%0'%02b'
```

```
:hi User1 term=inverse,bold cterm=inverse,bold ctermfg=red
```

Display a `,GZ` flag if a compressed file is loaded

```
:set statusline=...%r%{VarExists('b:gzflag','\ [GZ]')}%h...
```

In the `:autocmd 's:`

```
:let b:gzflag = 1
```

And:

```
:unlet b:gzflag
```

And define this function:

```
:function VarExists(var, val)
```

```
:  if exists(a:var) | return a:val | else | return '' | endif
```

```
:endfunction
```

'suffixes' 'su' string (default ".bak,~, .o,.h,.info,.swp,.obj")
global

Files with these suffixes get a lower priority when multiple files match a wildcard. See [suffixes](#) . Commas can be used to separate the suffixes. Spaces after the comma are ignored. A dot is also seen as the start of a suffix. To avoid a dot or comma being recognized as a separator, precede it with a backslash (see [option-backslash](#) about including spaces and backslashes).

See **'wildignore'** for completely ignoring files.

The use of **:set+=** and **:set-=** is preferred when adding or removing suffixes from the list. This avoids problems when a future version uses another default.

'suffixesadd' 'sua' string (default "")
local to buffer

Comma-separated list of suffixes, which are used when searching for a file for the "gf", "[I", etc. commands. Example:

```
:set suffixesadd=.java
```

'swapfile' 'swf' boolean (default on)
local to buffer

Use a swapfile for the buffer. This option can be reset when a swapfile is not wanted for a specific buffer. For example, with confidential information that even root must not be able to access. Careful: All text will be in memory:

- Don't use this for big files.
- Recovery will be impossible!

A swapfile will only be present when **'updatecount'** is non-zero and **'swapfile'** is set.

When **'swapfile'** is reset, the swap file for the current buffer is immediately deleted. When **'swapfile'** is set, and **'updatecount'** is non-zero, a swap file is immediately created.

Also see [swap-file](#) and **'swapsync'** .

If you want to open a new buffer without creating a swap file for it, use the **:noswapfile** modifier.

See **'directory'** for where the swap file is created.

This option is used together with **'bufhidden'** and **'buftype'** to specify special kinds of buffers. See [special-buffers](#) .

'swapsync' 'sws' string (default "fsync")
global

When this option is not empty a swap file is synced to disk after writing to it. This takes some time, especially on busy unix systems. When this option is empty parts of the swap file may be in memory and not written to disk. When the system crashes you may lose more work. On Unix the system does a sync now and then without Vim asking for it, so the disadvantage of setting this option off is small. On some systems the swap file will not be written at all. For a unix system

setting it to "sync" will use the sync() call instead of the default fsync(), which may work better on some systems. The **'fsync'** option is used for the actual file.

'switchbuf' 'swb' **'switchbuf' 'swb'**
string (default "")
global

This option controls the behavior when switching between buffers. This option is checked, when

- jumping to errors with the **quickfix** commands (**:cc** , **:cn** , **:cp** , etc.).
- jumping to a tag using the **:stag** command.
- opening a file using the **CTRL-W_f** or **CTRL-W_F** command.
- jumping to a buffer using a buffer split command (e.g. **:sbuffer** , **:sbnxt** , or **:sbrewind**).

Possible values (comma-separated list):

useopen	If included, jump to the first open window in the current tab page that contains the specified buffer (if there is one). Otherwise: Do not examine other windows.
usetab	Like "useopen", but also consider windows in other tab pages.
split	If included, split the current window before loading a buffer for a quickfix command that display errors. Otherwise: do not split, use current window (when used in the quickfix window: the previously used window or split if there is no other window).
vsplit	Just like "split" but split vertically.
newtab	Like "split", but open a new tab page. Overrides "split" when both are present.
uselast	If included, jump to the previously used window when jumping to errors with quickfix commands.

If a window has **'winfixbuf'** enabled, **'switchbuf'** is currently not applied to the split window.

'synmaxcol' 'smc' **'synmaxcol' 'smc'**
number (default 3000)
local to buffer
{not available when compiled without the **+syntax** feature}

Maximum column in which to search for syntax items. In long lines the text after this column is not highlighted and following lines may not be highlighted correctly, because the syntax state is cleared. This helps to avoid very slow redrawing for an XML file that is one long line.
Set to zero to remove the limit.

'syntax' 'syn' **'syntax' 'syn'**
string (default empty)
local to buffer **local-noglobal**
{not available when compiled without the **+syntax** feature}

When this option is set, the syntax with this name is loaded, unless syntax highlighting has been switched off with **":syntax off"**.

Otherwise this option does not always reflect the current syntax (the `b:current_syntax` variable does).

This option is most useful in a modeline, for a file which syntax is not automatically recognized. Example, in an IDL file:

```
/* vim: set syntax=idl : */
```

When a dot appears in the value then this separates two filetype names. Example:

```
/* vim: set syntax=c.doxygen : */
```

This will use the "c" syntax first, then the "doxygen" syntax.

Note that the second one must be prepared to be loaded as an addition, otherwise it will be skipped. More than one dot may appear.

To switch off syntax highlighting for the current file, use:

```
:set syntax=OFF
```

To switch syntax highlighting on according to the current value of the **'filetype'** option:

```
:set syntax=ON
```

What actually happens when setting the **'syntax'** option is that the Syntax autocommand event is triggered with the value as argument.

This option is not copied to another buffer, independent of the 's' or 'S' flag in **'coptions'**.

Only normal file name characters can be used, `"/*?*[]<>"` are illegal.

'tabline' 'tal' string (default empty) **'tabline' 'tal'**
global

When non-empty, this option determines the content of the tab pages line at the top of the Vim window. When empty Vim will use a default tab pages line. See [setting-tabline](#) for more info.

The tab pages line only appears as specified with the **'showtabline'** option and only when there is no GUI tab line. When 'e' is in **'guioptions'** and the GUI supports a tab line **'guitablabel'** is used instead. **Note** that the two tab pages lines are very different.

The value is evaluated like with **'statusline'**. You can use `tabpagenr()`, `tabpagewinnr()` and `tabpagebuflist()` to figure out the text to be displayed. Use `"%1T"` for the first label, `"%2T"` for the second one, etc. Use `"%X"` items for closing labels.

When changing something that is used in **'tabline'** that does not trigger it to be updated, use `:redrawtabline`.

This option cannot be set in a modeline when **'modelineexpr'** is off.

Keep in mind that only one of the tab pages is the current one, others are invisible and you can't jump to their windows.

'tabpagemax' 'tpm' number (default 10) **'tabpagemax' 'tpm'**
global

Maximum number of tab pages to be opened by the `-p` command line argument or the `":tab all"` command. [tabpage](#)

'tabstop' 'ts' number (default 8) **'tabstop' 'ts'**

local to buffer

Number of spaces that a <Tab> in the file counts for. Also see the `:retab` command, and the `'softtabstop'` option.

Note: Setting `'tabstop'` to any other value than 8 can make your file appear wrong in many places, e.g., when printing it. The value must be more than 0 and less than 10000.

There are four main ways to use tabs in Vim:

1. Always keep `'tabstop'` at 8, set `'softtabstop'` and `'shiftwidth'` to 4 (or 3 or whatever you prefer) and use `'noexpandtab'`. Then Vim will use a mix of tabs and spaces, but typing <Tab> and <BS> will behave like a tab appears every 4 (or 3) characters. This is the recommended way, the file will look the same with other tools and when listing it in a terminal.
2. Set `'softtabstop'` and `'shiftwidth'` to whatever you prefer and use `'expandtab'`. This way you will always insert spaces. The formatting will never be messed up when `'tabstop'` is changed (leave it at 8 just in case). The file will be a bit larger. You do need to check if no Tabs exist in the file. You can get rid of them by first setting `'expandtab'` and using ``:%retab!``, making sure the value of `'tabstop'` is set correctly.
3. Set `'tabstop'` and `'shiftwidth'` to whatever you prefer and use `'expandtab'`. This way you will always insert spaces. The formatting will never be messed up when `'tabstop'` is changed. You do need to check if no Tabs exist in the file, just like in the item just above.
4. Set `'tabstop'` and `'shiftwidth'` to whatever you prefer and use a `modeline` to set these values when editing the file again. Only works when using Vim to edit the file, other tools assume a tabstop is worth 8 spaces.
5. Always set `'tabstop'` and `'shiftwidth'` to the same value, and `'noexpandtab'`. This should then work (for initial indents only) for any tabstop setting that people use. It might be nice to have tabs after the first non-blank inserted as spaces if you do this though. Otherwise aligned comments will be wrong when `'tabstop'` is changed.

If Vim is compiled with the `+vartabs` feature then the value of `'tabstop'` will be ignored if `'vartabstop'` is set to anything other than an empty string.

<code>'tagbsearch'</code>	<code>'tbs'</code>	<code>'tagbsearch'</code>	<code>'tbs'</code>	<code>'notagbsearch'</code>	<code>'notbs'</code>
		boolean (default on)			
		global			

When searching for a tag (e.g., for the `:ta` command), Vim can either use a binary search or a linear search in a tags file. Binary searching makes searching for a tag a LOT faster, but a linear search will find more tags if the tags file wasn't properly sorted. Vim normally assumes that your tags files are sorted, or indicate that they are not sorted. Only when this is not the case does the `'tagbsearch'` option need to be switched off.

When `'tagbsearch'` is on, binary searching is first used in the tags

files. In certain situations, Vim will do a linear search instead for certain files, or retry all files with a linear search. When **'tagbsearch'** is off, only a linear search is done.

Linear searching is done anyway, for one file, when Vim finds a line at the start of the file indicating that it's not sorted:

```
!_TAG_FILE_SORTED 0 /some comment/
```

[The whitespace before and after the '0' must be a single <Tab>]

When a binary search was done and no match was found in any of the files listed in **'tags'**, and case is ignored or a pattern is used instead of a normal tag name, a retry is done with a linear search. Tags in unsorted tags files, and matches with different case will only be found in the retry.

If a tag file indicates that it is case-fold sorted, the second, linear search can be avoided when case is ignored. Use a value of '2' in the `!_TAG_FILE_SORTED` line for this. A tag file can be case-fold sorted with the `-f` switch to "sort" in most unices, as in the command: `"sort -f -o tags tags"`. For Universal ctags and Exuberant ctags version 5.x or higher (at least 5.5) the `--sort=foldcase` switch can be used for this as well. **Note** that case must be folded to uppercase for this to work.

By default, tag searches are case-sensitive. Case is ignored when **'ignorecase'** is set and **'tagcase'** is "followic", or when **'tagcase'** is "ignore".

Also when **'tagcase'** is "followscs" and **'smartcase'** is set, or **'tagcase'** is "smart", and the pattern contains only lowercase characters.

When **'tagbsearch'** is off, tags searching is slower when a full match exists, but faster when no full match exists. Tags in unsorted tags files may only be found with **'tagbsearch'** off.

When the tags file is not sorted, or sorted in a wrong way (not on ASCII byte value), **'tagbsearch'** should be off, or the line given above must be included in the tags file.

This option doesn't affect commands that find all matching tags (e.g., command-line completion and `:"help"`).

```
'tagcase' 'tc'                string (default "followic") 'tagcase' 'tc'  
                                global or local to buffer global-local
```

This option specifies how case is handled when searching the tags file:

followic	Follow the 'ignorecase' option
followscs	Follow the 'smartcase' and 'ignorecase' options
ignore	Ignore case
match	Match case
smart	Ignore case unless an upper case letter is used

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

```
'tagfunc' 'tfu'
```

'tagfunc' 'tfu' string (default: empty)
local to buffer
{not available when compiled without the `+eval`
feature}

This option specifies a function to be used to perform tag searches. The function gets the tag pattern and should return a List of matching tags. See [tag-function](#) for an explanation of how to write the function and an example. The value can be the name of a function, a `lambda` or a `Funcref`. See [option-value-function](#) for more information.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'taglength' 'tl' number (default 0) **'taglength' 'tl'**
global

If non-zero, tags are significant up to this number of characters.

'tagrelative' 'tr' boolean (Vim default: on, Vi default: off) **'tagrelative' 'tr' 'notagrelative' 'notr'**
global

If on and using a tags file in another directory, file names in that tags file are relative to the directory where the tags file is.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'tags' 'tag' string (default `"/tags,tags"`, when compiled with `+emacs_tags` : `"/tags,./TAGS,tags,TAGS"`) **'tags' 'tag' E433**
global or local to buffer `global-local`

Filenames for the tag command, separated by spaces or commas. To include a space or comma in a file name, precede it with backslashes (see [option-backslash](#) about including spaces/commas and backslashes). When a file name starts with `"/"`, the `"/` is replaced with the path of the current file. But only when the `'d'` flag is not included in **'coptions'**. Environment variables are expanded `:set_env`. Also see [tags-option](#).

`"*`, `**` and other wildcards can be used to search for tags files in a directory tree. See [file-searching](#). E.g., `"/lib/**/tags"` will find all files named `tags` below `"/lib"`. The filename itself cannot contain wildcards, it is used as-is. E.g., `"/lib/**/tags?"` will find files called `tags?`.

The `tagfiles()` function can be used to get a list of the file names actually used.

If Vim was compiled with the `+emacs_tags` feature, Emacs-style tag files are also supported. They are automatically recognized. The default value becomes `"/tags,./TAGS,tags,TAGS"`, unless case differences are ignored (MS-Windows). [emacs-tags](#)

The use of `:set+=` and `:set-=` is preferred when adding or removing file names from the list. This avoids problems when a future version uses another default.

'tagstack' 'tgst' boolean (default on) **'tagstack' 'tgst' 'notagstack' 'notgst'**

global
 When on, the `tagstack` is used normally. When off, a `":tag"` or `":tselect"` command with an argument will not push the tag onto the tagstack. A following `":tag"` without an argument, a `":pop"` command or any other command that uses the tagstack will use the unmodified tagstack, but does change the pointer to the active entry. Resetting this option is useful when using a `":tag"` command in a mapping which should not change the tagstack.

'tcldll' 'tcldll'
 string (default depends on the build)
 global
 {only available when compiled with the `+tcl/dyn` feature}

Specifies the name of the Tcl shared library. The default is `DYNAMIC_TCL_DLL`, which was specified at compile time. Environment variables are expanded `:set_env`. This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'term' 'term' E529 E530 E531
 string (default is `$TERM`, if that fails:
 in the GUI: `"builtin_gui"`
 on Amiga: `"amiga"`
 on Haiku: `"xterm"`
 on Mac: `"mac-ansi"`
 on Unix: `"ansi"`
 on VMS: `"ansi"`
 on Win 32: `"win32"`)

global
 Name of the terminal. Used for choosing the terminal control characters. Environment variables are expanded `:set_env`.

For example:
`:set term=$TERM`

See `termcap`.

'termbidi' 'tbidi' 'termbidi' 'tbidi'
'notermbidi' 'notbidi'
 boolean (default off, on for `"mlterm"`)
 global
 {only available when compiled with the `+arabic` feature}

The terminal is in charge of Bi-directionality of text (as specified by Unicode). The terminal is also expected to do the required shaping that some languages (such as Arabic) require.

Setting this option implies that **'rightleft'** will not be set when **'arabic'** is set and the value of **'arabicshape'** will be ignored.

Note that setting **'termbidi'** has the immediate effect that **'arabicshape'** is ignored, but **'rightleft'** isn't changed automatically.

This option is reset when the GUI is started.

For further details see `arabic.txt`.

'termencoding' 'tenc' 'termencoding' 'tenc'
 string (default `""`; with GTK+ GUI: `"utf-8"`)

global

Encoding used for the terminal. This specifies what character encoding the keyboard produces and the display will understand. For the GUI it only applies to the keyboard ('encoding' is used for the display).

E617 E950

Note: This does not apply to the GTK+ GUI. After the GUI has been successfully initialized, '**termencoding**' is forcibly set to "utf-8". Any attempts to set a different value will be rejected, and an error message is shown.

For the Win32 GUI and console versions '**termencoding**' is not used, because the Win32 system always passes Unicode characters.

When empty, the same encoding is used as for the '**encoding**' option. This is the normal value.

Not all combinations for '**termencoding**' and '**encoding**' are valid. See [encoding-table](#) .

The value for this option must be supported by internal conversions or `iconv()`. When this is not possible no conversion will be done and you will probably experience problems with non-ASCII characters.

Example: You are working with the locale set to `euc-jp` (Japanese) and want to edit a UTF-8 file:

```
:let &termencoding = &encoding
:set encoding=utf-8
```

You need to do this when your system has no locale support for UTF-8.

'termguicolors' **'tgc'** **'notermguicolors'** **'notgc'** E954
boolean (default off)
global
{not available when compiled without the
+**termguicolors** feature}

When on, uses `highlight-guifg` and `highlight-guibg` attributes in the terminal (thus using 24-bit color).

Requires a ISO-8613-3 compatible terminal. If setting this option does not work (produces a colorless UI) reading `xterm-true-color` might help.

For Win32 console, Windows 10 version 1703 (Creators Update) or later is required. Use this check to find out:

```
if has('vcon')
```

This requires Vim to be built with the `+vtp` feature.

Note that the "cterm" attributes are still used, not the "gui" ones.

When using Vim with Windows Terminal, the background of Windows Terminal is normally filled with the Vim background color. Setting '**termguicolors**' and the `guibg` of the Normal highlight group to NONE will make the background transparent:

```
:hi Normal guibg=NONE
```

NOTE: This option is reset when '**compatible**' is set.

'termwinkey' **'twk'** **'termwinkey'** **'twk'**
string (default "")

local to window

The key that starts a **CTRL-W** command in a terminal window. Other keys are sent to the job running in the window.

The <> notation can be used, e.g.:

```
:set termwinkey=<C-L>
```

The string must be one key stroke but can be multiple bytes.

When not set **CTRL-W** is used, so that **CTRL-W** : gets you to the command line. If '**termwinkey**' is set to **CTRL-L** then **CTRL-L** : gets you to the command line.

```
'termwincscroll' 'tws1'      'termwincscroll' 'tws1'
number (default 10000)
local to buffer
{not available when compiled without the
+terminal feature}
Number of scrollback lines to keep. When going over this limit the
first 10% of the scrollback lines are deleted. This is just to reduce
the memory usage. See Terminal-Normal .
Also used as a limit for text sent to the terminal in one write,
multiplied by the number of columns times 3 (average number of bytes
per cell).
```

```
'termwinsize' 'tws'         'termwinsize' 'tws'
string (default "")
local to window
Size used when opening the terminal window. Format:
{rows}x{columns} or {rows}*{columns}.
- When empty the terminal gets the size from the window.
- When set with a "x" (e.g., "24x80") the terminal size is not
adjusted to the window size. If the window is smaller only the
top-left part is displayed.
- When set with a "*" (e.g., "10*50") the terminal size follows the
window size, but will not be smaller than the specified rows and/or
columns.
- When rows is zero then use the height of the window.
- When columns is zero then use the width of the window.
- Using "0x0" or "0*0" is the same as empty.
- Can be overruled in the term\_start\(\) options with "term_rows" and
"term_cols".
```

Examples:

"30x0" uses 30 rows and the current window width.

"20*0" uses at least 20 rows and the current window width.

"0*40" uses the current window height and at least 40 columns.

Note that the command running in the terminal window may still change the size of the terminal. In that case the Vim window will be adjusted to that size, if possible.

```
'termwintype' 'twt'        'termwintype' 'twt'
string (default "")
global
{only available when compiled with the terminal
feature on MS-Windows}
Specify the virtual console (pty) used when opening the terminal
```


window.

Possible values are:

```
""           use ConPTY if it is stable, winpty otherwise
"winpty"    use winpty, fail if not supported
"conpty"    use ConPTY , fail if not supported
```

ConPTY support depends on the platform. Windows 10 October 2018 Update is the first version that supports ConPTY, however it is still considered unstable. ConPTY might become stable in the next release of Windows 10. winpty support needs to be installed. If neither is supported then you cannot open a terminal window.

```
'terse'           'terse'  'noterse'
```

'terse' boolean (default off)
 global

When set: Add 's' flag to **'shortmess'** option (this makes the message for a search that hits the start or end of the file not being displayed). When reset: Remove 's' flag from **'shortmess'** option. {Vi shortens a lot of messages}

```
'textauto' 'ta'   'textauto' 'ta'  'notextauto' 'nota'
```

'textauto' 'ta' boolean (Vim default: on, Vi default: off)
 global

This option is obsolete. Use **'fileformats'**.
For backwards compatibility, when **'textauto'** is set, **'fileformats'** is set to the default value for the current system. When **'textauto'** is reset, **'fileformats'** is made empty.
NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

```
'textmode' 'tx'   'textmode' 'tx'  'notextmode' 'notx'
```

'textmode' 'tx' boolean (Win32: default on,
 others: default off)
 local to buffer

This option is obsolete. Use **'fileformat'**.
For backwards compatibility, when **'textmode'** is set, **'fileformat'** is set to "dos". When **'textmode'** is reset, **'fileformat'** is set to "unix".

```
'textwidth' 'tw'   'textwidth' 'tw'
```

'textwidth' 'tw' number (default 0)
 local to buffer

Maximum width of text that is being inserted. A longer line will be broken after white space to get this width. A zero value disables this.
'textwidth' is set to 0 when the **'paste'** option is set and restored when **'paste'** is reset.
When **'textwidth'** is zero, **'wrapmargin'** may be used. See also **'formatoptions'** and **ins-textwidth**.
When **'formatexpr'** is set it will be used to break the line.
NOTE: This option is set to 0 when **'compatible'** is set.

```
'thesaurus' 'tsr'
```

'thesaurus' 'tsr' string (default "")
 global or local to buffer [global-local](#)
 List of file names, separated by commas, that are used to lookup words for thesaurus completion commands [i_CTRL-X_CTRL-T](#) . See [compl-thesaurus](#) .

This option is not used if **'thesaurusfunc'** is set, either for the buffer or globally.

To include a comma in a file name precede it with a backslash. Spaces after a comma are ignored, otherwise spaces are included in the file name. See [option-backslash](#) about using backslashes. The use of [:set+=](#) and [:set-=](#) is preferred when adding or removing directories from the list. This avoids problems when a future version uses another default. Backticks cannot be used in this option for security reasons.

'thesaurusfunc' 'tsrfu' string (default: empty)
 global or local to buffer [global-local](#)
 {not available when compiled without the [+eval](#) feature}

This option specifies a function to be used for thesaurus completion with **CTRL-X CTRL-T**. [i_CTRL-X_CTRL-T](#) See [compl-thesaurusfunc](#) . The value can be the name of a function, a [lambda](#) or a [Funcref](#) . See [option-value-function](#) for more information.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

'tildeop' 'top' ['tildeop'](#) ['top'](#) ['notildeop'](#) ['notop'](#)
 boolean (default off)
 global

When on: The tilde command "~" behaves like an operator.

NOTE: This option is reset when **'compatible'** is set.

'timeout' 'to' ['timeout'](#) ['to'](#) ['notimeout'](#) ['noto'](#)
 boolean (default on)
 global

'ttimeout' ['ttimeout'](#) ['nottimeout'](#)
 boolean (default off, set in [defaults.vim](#))
 global

These two options together determine the behavior when part of a mapped key sequence or keyboard code has been received:

'timeout'	'ttimeout'	action
off	off	do not time out
on	on or off	time out on :mappings and key codes
off	on	time out on key codes

If both options are off, Vim will wait until either the complete mapping or key sequence has been received, or it is clear that there is no mapping or key sequence for the received characters. For

example: if you have mapped "vl" and Vim has received 'v', the next character is needed to see if the 'v' is followed by an 'l'. When one of the options is on, Vim will wait for about 1 second for the next character to arrive. After that the already received characters are interpreted as single characters. The waiting time can be changed with the **'timeoutlen'** option.

On slow terminals or very busy systems timing out may cause malfunctioning cursor keys. If both options are off, Vim waits forever after an entered <Esc> if there are key codes that start with <Esc>. You will have to type <Esc> twice. If you do not have problems with key codes, but would like to have :mapped key sequences not timing out in 1 second, set the **'ttimeout'** option and reset the **'timeout'** option.

NOTE: **'ttimeout'** is reset when **'compatible'** is set.

			'timeoutlen'	'tm'
'timeoutlen'	'tm'	number (default 1000)		
		global		
			'ttimeoutlen'	'ttm'
'ttimeoutlen'	'ttm'	number (default -1, set to 100 in defaults.vim)		
		global		

The time in milliseconds that is waited for a key code or mapped key sequence to complete. Also used for CTRL-\ CTRL-N and CTRL-\ CTRL-G when part of a command has been typed. Normally only **'timeoutlen'** is used and **'ttimeoutlen'** is -1. When a different timeout value for key codes is desired set **'ttimeoutlen'** to a non-negative number.

ttimeoutlen	mapping delay	key code delay
< 0	'timeoutlen'	'timeoutlen'
>= 0	'ttimeoutlen'	'ttimeoutlen'

The timeout only happens when the **'timeout'** and **'ttimeout'** options tell so. A useful setting would be

```
:set timeout timeoutlen=3000 ttimeoutlen=100
```

(time out on mapping after three seconds, time out on key codes after a tenth of a second).

		'title'	'notitle'
'title'	boolean (default off, on when title can be restored)		
	global		

When on, the title of the window will be set to the value of

'titlestring' (if it is not empty), or to:

```
filename [+--] (path) - VIM
```

Where:

filename	the name of the file being edited
-	indicates the file cannot be modified, 'ma' off
+	indicates the file was modified
=	indicates the file is read-only
=+	indicates the file is read-only and modified
(path)	is the path of the file being edited
- VIM	the server name v:servername or "VIM"

Only works if the terminal supports setting window titles (currently Amiga console, Win32 console, all GUI versions and terminals with a non-empty `'t_ts'` option - these are Unix xterm and iris-ansi by default, where `'t_ts'` is taken from the builtin termcap).

X11

When Vim was compiled with HAVE_X11 defined, the original title will be restored if possible. The output of `":version"` will include `"+X11"` when HAVE_X11 was defined, otherwise it will be `"-X11"`. This also works for the icon name `'icon'`.

But: When Vim was started with the `-X` argument, restoring the title will not work (except in the GUI).

If the title cannot be restored, it is set to the value of `'titleold'`. You might want to restore the title outside of Vim then.

When using an xterm from a remote machine you can use this command:

```
rsh machine_name xterm -display $DISPLAY &
ssh -X machine_name xterm &
```

then the WINDOWID environment variable should be inherited and the title of the window should change back to what it should be after exiting Vim.

'titlelen'

'titlelen' number (default 85)
 global

Gives the percentage of `'columns'` to use for the length of the window title. When the title is longer, only the end of the path name is shown. A `'<'` character before the path name is used to indicate this. Using a percentage makes this adapt to the width of the window. But it won't work perfectly, because the actual number of characters available also depends on the font used and other things in the title bar. When `'titlelen'` is zero the full path is used. Otherwise, values from 1 to 30000 percent can be used.

`'titlelen'` is also used for the `'titlestring'` option.

'titleold'

'titleold' string (default "Thanks for flying Vim")
 global

This option will be used for the window title when exiting Vim if the original title cannot be restored. Only happens if `'title'` is on or `'titlestring'` is not empty.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

'titlestring'

'titlestring' string (default "")
 global

When this option is not empty, it will be used for the title of the window. This happens only when the `'title'` option is on.

Only works if the terminal supports setting window titles (currently Amiga console, Win32 console, all GUI versions and terminals with a non-empty `'t_ts'` option).

When Vim was compiled with HAVE_X11 defined, the original title will be restored if possible, see [X11](#).

When this option contains printf-style '%' items, they will be expanded according to the rules used for **'statusline'**. This option cannot be set in a modeline when **'modelineexpr'** is off.

Example:

```
:auto BufEnter * let &titlestring = hostname() .. "/" .. expand("%:p")
:set title titlestring=%<%F%=%l/%L-%P titlelen=70
```

The value of **'titlelen'** is used to align items in the middle or right of the available space.

Some people prefer to have the file name first:

```
:set titlestring=%t%(\ %M%)%(\ (%{expand("\":~::~:h\"})})%(\ %a%)
```

Note the use of "%{ }" and an expression to get the path of the file, without the file name. The "%(%)" constructs are used to add a separating space only when needed.

NOTE: Use of special characters in **'titlestring'** may cause the display to be garbled (e.g., when it contains a CR or NL character).

{not available when compiled without the |+statusline| feature}

```

'toolbar' 'tb'
                'toolbar' 'tb'
                string (default "icons, tooltips")
                global
                {only for |+GUI_GTK|, |+GUI_Motif| and |+GUI_Photon|}

```

The contents of this option controls various toolbar settings. The possible values are:

icons	Toolbar buttons are shown with icons.
text	Toolbar buttons shown with text.
horiz	Icon and text of a toolbar button are horizontally arranged. {only in GTK+ 2 GUI}
tooltips	Tooltips are active for toolbar buttons.

Tooltips refer to the popup help text which appears after the mouse cursor is placed over a toolbar button for a brief moment.

If you want the toolbar to be shown with icons as well as text, do the following:

```
:set tb=icons, text
```

Motif cannot display icons and text at the same time. They will show icons if both are requested.

If none of the strings specified in **'toolbar'** are valid or if **'toolbar'** is empty, this option is ignored. If you want to disable the toolbar, you need to set the **'guioptions'** option. For example:

```
:set guioptions-=T
```

Also see [gui-toolbar](#) .

```

'toolbariconsize' 'tbis'
                'toolbariconsize' 'tbis'
                string (default "small")
                global
                {only in the GTK+ GUI}

```

Controls the size of toolbar icons. The possible values are:

tiny	Use tiny icons.
small	Use small icons (default).
medium	Use medium-sized icons.
large	Use large icons.
huge	Use even larger icons.

giant Use very big icons.
The exact dimensions in pixels of the various icon sizes depend on the current theme. Common dimensions are giant=48x48, huge=32x32, large=24x24, medium=24x24, small=20x20 and tiny=16x16.

If **'toolbariconsize'** is empty, the global default size as determined by user preferences or the current theme is used.

'ttybuiltin' **'tbi'** **'ttybuiltin'** **'tbi'** **'nottybuiltin'** **'notbi'**
boolean (default on)
global

When on, the builtin termcaps are searched before the external ones. When off the builtin termcaps are searched after the external ones. When this option is changed, you should set the **'term'** option next for the change to take effect, for example:

```
:set notbi term=$TERM
```

See also `termcap`.

Rationale: The default for this option is "on", because the builtin termcap entries are generally better (many systems contain faulty xterm entries...).

'ttyfast' **'tf'** **'ttyfast'** **'tf'** **'nottyfast'** **'notf'**
boolean (default on)
global

Indicates a fast terminal connection. More characters will be sent to the screen for redrawing, instead of using insert/delete line commands. Improves smoothness of redrawing when there are multiple windows and the terminal does not support a scrolling region. Also enables the extra writing of characters at the end of each screen line for lines that wrap. This helps when using copy/paste with the mouse in an xterm and other terminals.

The default used to be set only for some terminal names, but these days nearly all terminals are fast, therefore the default is now "on". If you have a slow connection you may want to set this option off, e.g. depending on the host name:

```
if hostname() =~ 'faraway'  
  set nottyfast  
endif
```

'ttymouse' **'ttym'** **'ttymouse'** **'ttym'**
string (default depends on **'term'**)
global
{only in Unix and VMS, doesn't work in the GUI; not available when compiled without `+mouse`}

Name of the terminal type for which mouse codes are to be recognized. Currently these strings are valid:

xterm **xterm-mouse**
xterm-like mouse handling. The mouse generates "`<Esc>[Mscr`", where "scr" is three bytes:
"s" = button state
"c" = column plus 33
"r" = row plus 33

This only works up to 223 columns! See "dec",

	"urxvt", and "sgr" for solutions.
xterm2	Works like "xterm", but with the xterm reporting the mouse position while the mouse is dragged. This works much faster and more precise. Your xterm must at least at patchlevel 88 / XFree 3.3.3 for this to work. See below for how Vim detects this automatically.
	netterm-mouse
netterm	NetTerm mouse handling. A left mouse click generates "<Esc>}r,c<CR>", where "r,c" are two decimal numbers for the row and column. No other mouse events are supported.
	dec-mouse
dec	DEC terminal mouse handling. The mouse generates a rather complex sequence, starting with "<Esc>[". This is also available for an Xterm, if it was configured with "--enable-dec-locator".
	jsbterm-mouse
jsbterm	JSB term mouse handling.
	pterm-mouse
pterm	QNX pterm mouse handling.
	urxvt-mouse
urxvt	Mouse handling for the urxvt (rxvt-unicode) terminal. The mouse works only if the terminal supports this encoding style, but it does not have 223 columns limit unlike "xterm" or "xterm2".
	sgr-mouse
sgr	Mouse handling for the terminal that emits SGR-styled mouse reporting. The mouse works even in columns beyond 223. This option is backward compatible with "xterm2" because it can also decode "xterm2" style mouse codes.

The mouse handling must be enabled at compile time `+mouse_xterm +mouse_dec +mouse_netterm +mouse_jsbterm +mouse_urxvt +mouse_sgr` .

Only "xterm"(2) is really recognized. NetTerm mouse codes are always recognized, if enabled at compile time. DEC terminal mouse codes are recognized if enabled at compile time, and `'ttymouse'` is not "xterm", "xterm2", "urxvt" or "sgr" (because dec mouse codes conflict with them).

This option is automatically set to "xterm", when the `'term'` option is set to a name that starts with "xterm", "mlterm", "screen", "tmux", "st" (full match only), "st-" or "stterm", and `'ttymouse'` is not set already.

If the terminfo/termcap entry "XM" exists and the first number is "1006" then `'ttymouse'` will be set to "sgr". This works for many modern terminals.

Additionally, if vim is compiled with the `+termresponse` feature and `t_RV` is set to the escape sequence to request the xterm version number, more intelligent detection is done.

The "xterm2" value will be set if the xterm version is reported to be from 95 to 276. The "sgr" value will be set if Vim detects Mac Terminal.app, iTerm2 or mintty, and when the xterm version is 277 or

higher.

If you do not want **'ttymouse'** to be set to "xterm2" or "sgr" automatically, set `t_RV` to an empty string:

```
:set t_RV=
```

'ttyscroll' **'tsl'** number (default 999)
 global
Maximum number of lines to scroll the screen. If there are more lines to scroll the window is redrawn. For terminals where scrolling is very slow and redrawing is not slow this can be set to a small number, e.g., 3, to speed up displaying.

'ttytype' **'tty'** string (default from \$TERM)
 global
Alias for **'term'**, see above.

'undodir' **'udir'** string (default ".")
 global
 {only when compiled with the |+persistent_undo| feature}
List of directory names for undo files, separated with commas. See **'backupdir'** for details of the format. "." means using the directory of the file. The undo file name for "file.txt" is ".file.txt.un~". For other directories the file name is the full path of the edited file, with path separators replaced with "%". When writing: The first directory that exists is used. "." always works, no directories after "." will be used for writing. When reading all entries are tried to find an undo file. The first undo file that exists is used. When it cannot be read an error is given, no further entry is used. See [undo-persistence](#) . This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

'undofile' **'udf'** boolean (default off)
 local to buffer
 {only when compiled with the |+persistent_undo| feature}
When on, Vim automatically saves undo history to an undo file when writing a buffer to a file, and restores undo history from the same file on buffer read. The directory where the undo file is stored is specified by **'undodir'**. For more information about this feature see [undo-persistence](#) . The undo file is not read when **'undoreload'** causes the buffer from before a reload to be saved for undo. When **'undofile'** is turned off the undo file is NOT deleted.
NOTE: This option is reset when **'compatible'** is set.

'undolevels' **'ul'** number (default 100, 1000 for Unix, VMS and Win32)
 global or local to buffer `global-local`

Maximum number of changes that can be undone. Since undo information is kept in memory, higher numbers will cause more memory to be used. Nevertheless, a single change can already use a large amount of memory. Set to 0 for Vi compatibility: One level of undo and "u" undoes itself:

```
set ul=0
```

But you can also get Vi compatibility by including the 'u' flag in '**coptions**', and still be able to use **CTRL-R** to repeat undo.

Also see [undo-two-ways](#) .

Set to -1 for no undo at all. You might want to do this only for the current buffer:

```
setlocal ul=-1
```

This helps when you run out of memory for a single change.

The local value is set to -123456 when the global value is to be used.

Also see [clear-undo](#) .

'undoreload' **'ur'** number (default 10000)
 global
Save the whole buffer for undo when reloading it. This applies to the ":e!" command and reloading for when the buffer changed outside of Vim. [FileChangedShell](#)
The save only happens when this option is negative or when the number of lines is smaller than the value of this option.
Set this option to zero to disable undo for a reload.

When saving undo for a reload, any undo file is not read.

Note that this causes the whole buffer to be stored in memory. Set this option to a lower value if you run out of memory.

'updatecount' **'uc'** number (default: 200)
 global
After typing this many characters the swap file will be written to disk. When zero, no swap file will be created at all (see chapter on recovery [crash-recovery](#)). '**updatecount**' is set to zero by starting Vim with the "-n" option, see [startup](#) . When editing in readonly mode this option will be initialized to 10000.
The swapfile can be disabled per buffer with '[swapfile](#)' .
When '**updatecount**' is set from zero to non-zero, swap files are created for all buffers that have '[swapfile](#)' set. When '**updatecount**' is set to zero, existing swap files are not deleted.
Also see '[swapsync](#)' .
This option has no meaning in buffers where '[buftype](#)' is "nofile" or "nowrite".

'updatetime' **'ut'** number (default 4000)
 global
If this many milliseconds nothing is typed the swap file will be written to disk (see [crash-recovery](#)). Also used for the

`CursorHold` autocommand event.

`'varsofftabstop'` `'vsts'` string (default "")
local to buffer
{only available when compiled with the `+vartabs`
feature}

A list of the number of spaces that a `<Tab>` counts for while editing, such as inserting a `<Tab>` or using `<BS>`. It "feels" like variable-width `<Tab>`s are being inserted, while in fact a mixture of spaces and `<Tab>`s is used. Tab widths are separated with commas, with the final value applying to all subsequent tabs.

For example, when editing assembly language files where statements start in the 9th column and comments in the 41st, it may be useful to use the following:

```
:set varsofftabstop=8,32,8
```

This will set soft tabstops with 8 and 8 + 32 spaces, and 8 more for every column thereafter.

Note that the value of `'softtabstop'` will be ignored while `'varsofftabstop'` is set.

`'vartabstop'` `'vts'` string (default "")
local to buffer
{only available when compiled with the `+vartabs`
feature}

A list of the number of spaces that a `<Tab>` in the file counts for, separated by commas. Each value corresponds to one tab, with the final value applying to all subsequent tabs. For example:

```
:set vartabstop=4,20,10,8
```

This will make the first tab 4 spaces wide, the second 20 spaces, the third 10 spaces, and all following tabs 8 spaces.

Note that the value of `'tabstop'` will be ignored while `'vartabstop'` is set.

`'verbose'` `'vbs'` number (default 0)
global

When bigger than zero, Vim will give messages about what it is doing. Currently, these messages are given:

- >= 1 When the viminfo file is read or written.
- >= 2 When a file is `":source"`ed.
- >= 4 Shell commands.
- >= 5 Every searched tags file and include file.
- >= 8 Files for which a group of autocommands is executed.
- >= 9 Every executed autocommand.
- >= 11 Finding items in a path
- >= 12 Every executed function.
- >= 13 When an exception is thrown, caught, finished, or discarded.
- >= 14 Anything pending in a `":finally"` clause.
- >= 15 Every executed Ex command from a script (truncated at 200)

characters).
>= 16 Every executed Ex command.

This option can also be set with the "-V" argument. See `-V` .
This option is also set by the `:verbose` command.

When the `'verbosefile'` option is set then the verbose messages are not displayed.

`'verbosefile'` `'vfile'` string (default empty)
global
When not empty all messages are written in a file with this name.
When the file exists messages are appended.
Writing to the file ends when Vim exits or when `'verbosefile'` is made empty. Writes are buffered, thus may not show up for some time.
Setting `'verbosefile'` to a new value is like making it empty first.
The difference with `:redir` is that verbose messages are not displayed when `'verbosefile'` is set.
This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

`'viewdir'` `'vdir'` string (default for Amiga: "home:vimfiles/view",
for Win32: "\$HOME/vimfiles/view",
for Unix: "\$HOME/.vim/view",
for macOS: "\$VIM/vimfiles/view",
for VMS: "sys\$login:vimfiles/view")
global
{not available when compiled without the `+mksession`
feature}
Name of the directory where to store files for `:mkview` .
This option cannot be set from a `modeline` or in the `sandbox` , for security reasons.

`'viewoptions'` `'vop'` string (default: "folds,options,cursor,curdir")
global
{not available when compiled without the `+mksession`
feature}
Changes the effect of the `:mkview` command. It is a comma-separated list of words. Each word enables saving and restoring something:
`word` `save and restore`
`cursor` cursor position in file and in window
`folds` manually created folds, opened/closed folds and local fold options
`options` options and mappings local to a window or buffer (not global values for local options)
`localoptions` same as "options"
`slash` backslashes in file names replaced with forward slashes
`unix` with Unix end-of-line format (single `<NL>`), even when on MS-Windows
`curdir` the window-local directory, if set with ``:lcd``

"slash" and "unix" are useful on MS-Windows when sharing view files with Unix. The Unix version of Vim cannot source dos format scripts, but the MS-Windows version of Vim can source unix format scripts.

```

'viminfo' 'vi'          string (Vi default: "", Vim default for
                        MS-Windows: '100,<50,s10,h,rA:,rB:,
                        for Amiga: '100,<50,s10,h,rdf0:,rdf1:,rdf2:
                        for others: '100,<50,s10,h)
                        global
                        {not available when compiled without the +viminfo
                        feature}

```

When non-empty, the viminfo file is read upon startup and written when exiting Vim (see [viminfo-file](#)). Except when **'viminfofile'** is "NONE".

The string should be a comma-separated list of parameters, each consisting of a single character identifying the particular parameter, followed by a number or string which specifies the value of that parameter. If a particular character is left out, then the default value is used for that parameter. The following is a list of the identifying characters and the effect of their value.

CHAR VALUE

- viminfo-!**

! When included, save and restore global variables that start with an uppercase letter, and don't contain a lowercase letter. Thus "KEEPTHIS and "K_L_M" are stored, but "KeepThis" and "_K_L_M" are not. Nested List and Dict items may not be read back correctly, you end up with an empty item.
- viminfo-quote**

" Maximum number of lines saved for each register. Old name of the '<' item, with the disadvantage that you need to put a backslash before the ", otherwise it will be recognized as the start of a comment!
- viminfo-%**

% When included, save and restore the buffer list. If Vim is started with a file name argument, the buffer list is not restored. If Vim is started without a file name argument, the buffer list is restored from the viminfo file. Quickfix ('buftype'), unlisted ('buflisted'), unnamed and buffers on removable media ([viminfo-r](#)) are not saved. When followed by a number, the number specifies the maximum number of buffers that are stored. Without a number all buffers are stored.
- viminfo-'**

' Maximum number of previously edited files for which the marks are remembered. This parameter must always be included when **'viminfo'** is non-empty. Including this item also means that the [jumplist](#) and the [changelist](#) are stored in the viminfo file.
- viminfo-/**

/ Maximum number of items in the search pattern history to be saved. If non-zero, then the previous search and substitute patterns are also saved. When not included, the value of

'history' is used.

- viminfo-:**

: Maximum number of items in the command-line history to be saved. When not included, the value of **'history'** is used.
- viminfo-<**

< Maximum number of lines saved for each register. If zero then registers are not saved. When not included, all lines are saved. **'** is the old name for this item. Also see the **'s'** item below: limit specified in Kbyte.
- viminfo-@**

@ Maximum number of items in the input-line history to be saved. When not included, the value of **'history'** is used.
- viminfo-c**

c When included, convert the text in the viminfo file from the **'encoding'** used when writing the file to the current **'encoding'**. See [viminfo-encoding](#) .
- viminfo-f**

f Whether file marks need to be stored. If zero, file marks ('0 to '9, 'A to 'Z) are not stored. When not present or when non-zero, they are all stored. '0 is used for the current cursor position (when exiting or when doing **":wviminfo"**).
- viminfo-h**

h Disable the effect of **'hlsearch'** when loading the viminfo file. When not included, it depends on whether **":nohlsearch"** has been used since the last search command.
- viminfo-n**

n Name of the viminfo file. The name must immediately follow the **'n'**. Must be at the end of the option! If the **'viminfofile'** option is set, that file name overrides the one given here with **'viminfo'**. Environment variables are expanded when opening the file, not when setting the option.
- viminfo-r**

r Removable media. The argument is a string (up to the next **'**,**'**). This parameter can be given several times. Each specifies the start of a path for which no marks will be stored. This is to avoid removable media. For MS-Windows you could use **"ra:,rb:"**, for Amiga **"rdf0:,rdf1:,rdf2:"**. You can also use it for temp files, e.g., for Unix: **"r/tmp"**. Case is ignored. Maximum length of each **'r'** argument is 50 characters.
- viminfo-s**

s Maximum size of an item in Kbyte. If zero then registers are not saved. Currently only applies to registers. The default **"s10"** will exclude registers with more than 10 Kbyte of text. Also see the **'<'** item above: line count limit.

Example:

```
:set viminfo='50,<1000,s100,:0,n~/vim/viminfo
```

- '50** Marks will be remembered for the last 50 files you edited.
- <1000** Contents of registers (up to 1000 lines each) will be remembered.
- s100** Registers with more than 100 Kbyte text are skipped.

<code>:0</code>	Command-line history will not be saved.
<code>n~/vim/viminfo</code>	The name of the file to use is " <code>~/vim/viminfo</code> ".
<code>no /</code>	Since <code>'/'</code> is not specified, the default will be used, that is, save all of the search history, and also the previous search and substitute patterns.
<code>no %</code>	The buffer list will not be saved nor read back.
<code>no h</code>	<code>'hlsearch'</code> highlighting will be restored.

When setting `'viminfo'` from an empty value you can use `:rviminfo` to load the contents of the file, this is not done automatically.

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

NOTE: This option is set to the Vim default value when `'compatible'` is reset.

	<code>'viminfofile'</code>	<code>'vif'</code>	
<code>'viminfofile'</code>	<code>'vif'</code>	string (default: <code>""</code>)	
		global	
		{not available when compiled without the <code>+viminfo</code> feature}	

When non-empty, overrides the file name used for viminfo.

When equal to "NONE" no viminfo file will be read or written.

This option can be set with the `-i` command line flag. The `--clean` command line flag sets it to "NONE".

This option cannot be set from a `modeline` or in the `sandbox`, for security reasons.

	<code>'virtualedit'</code>	<code>'ve'</code>	
<code>'virtualedit'</code>	<code>'ve'</code>	string (default: <code>""</code>)	
		global or local to window	<code>global-local</code>

A comma-separated list of these words:

<code>block</code>	Allow virtual editing in Visual block mode.
<code>insert</code>	Allow virtual editing in Insert mode.
<code>all</code>	Allow virtual editing in all modes.
<code>onemore</code>	Allow the cursor to move just past the end of the line
<code>none</code>	When used as the local value, do not allow virtual editing even when the global value is set. When used as the global value, "none" is the same as <code>""</code> .
<code>NONE</code>	Alternative spelling of "none".

Virtual editing means that the cursor can be positioned where there is no actual character. This can be halfway into a tab or beyond the end of the line. Useful for selecting a rectangle in Visual mode and editing a table.

"onemore" is not the same, it will only allow moving the cursor just after the last character of the line. This makes some commands more consistent. Previously the cursor was always past the end of the line if the line was empty. But it is far from Vi compatible. It may also break some plugins or Vim scripts. For example because `l` can move the cursor after the last character. Use with care!

Using the ``$`` command will move to the last character in the line, not past it. This may actually move the cursor to the left!

The ``g$`` command will move to the end of the screen line.

It doesn't make sense to combine "all" with "onemore", but you will not get a warning for it.

When combined with other words, "none" is ignored.

NOTE: This option is set to "" when **'compatible'** is set.

'visualbell' 'vb' **'visualbell'** **'vb'** **'novisualbell'** **'novb'** **beep**
boolean (default off)
global

Use a visual bell instead of beeping. The terminal code to display the visual bell is given with **'t_vb'**. When no beep or flash is wanted, use:

```
:set vb t_vb=
```

If you want a short flash, you can use this on many terminals:

```
:set vb t_vb=?[?5h$<100>?[?5l
```

Here $\$<100>$ specifies the time, you can use a smaller or bigger value to get a shorter or longer flash.

Note: Vim will limit the bell to once per half a second. This avoids having to wait for the flashing to finish when there are lots of bells, e.g. on key repeat. This also happens without **'visualbell'** set.

In the GUI, **'t_vb'** defaults to "`<Esc>|f`", which inverts the display for 20 msec. If you want to use a different time, use "`<Esc>|40f`", where 40 is the time in msec.

Note: When the GUI starts, **'t_vb'** is reset to its default value. You might want to set it again in your `gvimrc`.

Does not work on the Amiga, you always get a screen flash.
Also see **'errorbells'**.

'warn' **'warn'** **'warn'** **'nowarn'**
boolean (default on)
global

Give a warning message when a shell command is used while the buffer has been changed.

'weirdinvert' 'wiv' **'weirdinvert'** **'wiv'** **'noweirdinvert'** **'nowiv'**
boolean (default off)
global

This option has the same effect as the **'t_xs'** terminal option. It is provided for backwards compatibility with version 4.x. Setting **'weirdinvert'** has the effect of making **'t_xs'** non-empty, and vice versa. Has no effect when the GUI is running.

'whichwrap' 'ww' **'whichwrap'** **'ww'**
string (Vim default: "b,s", Vi default: "")
global

Allow specified keys that move the cursor left/right to move to the previous/next line when the cursor is on the first/last character in the line. Concatenate characters to allow this for these keys:

char	key	mode
b	<BS>	Normal and Visual

s	<Space>	Normal and Visual
h	"h"	Normal and Visual (not recommended)
l	"l"	Normal and Visual (not recommended)
<	<Left>	Normal and Visual
>	<Right>	Normal and Visual
~	"~"	Normal
[<Left>	Insert and Replace
]	<Right>	Insert and Replace

For example:

```
:set ww=<,>[,]
```

allows wrap only when cursor keys are used.

When the movement keys are used in combination with a delete or change operator, the <EOL> also counts for a character. This makes "3h" different from "3dh" when the cursor crosses the end of a line. This is also true for "x" and "X", because they do the same as "dl" and "dh". If you use this, you may also want to use the mapping `":map <BS> X"` to make backspace delete the character in front of the cursor.

When 'l' is included and it is used after an operator at the end of a line (not an empty line) then it will not move to the next line. This makes "dl", "cl", "yl" etc. work normally.

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'wildchar' **'wc'** '**wildchar**' '**wc**'
number (Vim default: <Tab>, Vi default: **CTRL-E**)
global

Character you have to type to start wildcard expansion in the command-line, as specified with **'wildmode'**.

More info here: [cmdline-completion](#) .

The character is not recognized when used inside a macro. See **'wildcharm'** for that.

Some keys will not work, such as **CTRL-C**, <CR> and Enter.

<Esc> can be used, but hitting it twice in a row will still exit command-line as a failsafe measure.

Although **'wc'** is a number option, you can set it to a special key:

```
:set wc=<Tab>
```

NOTE: This option is set to the Vi default value when **'compatible'** is set and to the Vim default value when **'compatible'** is reset.

'wildcharm' **'wcm'** '**wildcharm**' '**wcm**'
number (default: none (0))
global

'wildcharm' works exactly like **'wildchar'**, except that it is recognized when used inside a macro. You can find "spare" command-line keys suitable for this option by looking at [ex-edit-index](#) . Normally you'll never actually type **'wildcharm'**, just use it in mappings that automatically invoke completion mode, e.g.:

```
:set wcm=<C-Z>
```

```
:cnoremap ss so $vim/sessions/*.vim<C-Z>
```

Then after typing `:ss` you can use **CTRL-P** & **CTRL-N**.

'wildignore' **'wig'** '**wildignore**' '**wig**'
string (default "")

global

A list of file patterns. A file that matches with one of these patterns is ignored when expanding `wildcards`, completing file or directory names, and influences the result of `expand()`, `glob()` and `globpath()` unless a flag is passed to disable this.

The pattern is used like with `:autocmd`, see `autocmd-patterns`.

Also see `'suffixes'`.

Example:

```
:set wildignore=*.o,*.obj
```

The use of `:set+=` and `:set--` is preferred when adding or removing a pattern from the list. This avoids problems when a future version uses another default.

`'wildignorecase'` `'wic'` `'nowildignorecase'` `'nowic'`
`'wildignorecase'` `'wic'` boolean (default off)
global

When set case is ignored when completing file names and directories.

Has no effect when `'fileignorecase'` is set.

Does not apply when the shell is used to expand wildcards, which happens when there are special characters.

`'wildmenu'` `'wmnu'` `'wildmenu'` `'wmnu'` `'nowildmenu'` `'nowmnu'`
`'wildmenu'` `'wmnu'` boolean (default off, set in `defaults.vim`)
global

When `'wildmenu'` is on, command-line completion operates in an enhanced mode. On pressing `'wildchar'` (usually `<Tab>`) to invoke completion, the possible matches are shown.

When `'wildoptions'` contains "pum", then the completion matches are shown in a popup menu. Otherwise they are displayed just above the command line, with the first match highlighted (overwriting the status line, if there is one).

Keys that show the previous/next match, such as `<Tab>` or

CTRL-P/CTRL-N, cause the highlight to move to the appropriate match.

When `'wildmode'` is used, "wildmenu" mode is used where "full" is specified. "longest" and "list" do not start "wildmenu" mode.

You can check the current mode with `wildmenumode()`.

If there are more matches than can fit in the line, a ">" is shown on the right and/or a "<" is shown on the left. The status line scrolls as needed.

The "wildmenu" mode is abandoned when a key is hit that is not used for selecting a completion.

While the "wildmenu" is active, the following keys have special meanings:

- CTRL-P** - go to the previous entry
- CTRL-N** - go to the next entry
- `<CR>` - in menu completion, when the cursor is just after a dot: move into a submenu.
- CTRL-E** - end completion, go back to what was there before selecting a match.
- CTRL-Y** - accept the currently selected match and stop completion.

When not using the popup menu for command line completion, the following keys have special meanings:

- <Left> <Right> - select previous/next match (like **CTRL-P/CTRL-N**)
- <Up> - in filename/menu name completion: move up into parent directory or parent menu.
- <Down> - in filename/menu name completion: move into a subdirectory or submenu.

When using the popup menu for command line completion, the following keys have special meanings:

- <Up> <Down> - select previous/next match (like **CTRL-P/CTRL-N**)
- <PageUp> - select a match several entries back
- <PageDown> - select a match several entries further
- <Left> - in filename/menu name completion: move up into parent directory or parent menu.
- <Right> - in filename/menu name completion: move into a subdirectory or submenu.

This makes the menus accessible from the console `console-menus` .

If you prefer the <Left> and <Right> keys to move the cursor instead of selecting a different match, use this:

```
:cnoremap <Left> <Space><BS><Left>
:cnoremap <Right> <Space><BS><Right>
```

The "WildMenu" highlighting is used for displaying the current match `hl-WildMenu` .

```
'wildmode' 'wim'          string (Vim default: "full")
                           global
```

Completion mode that is used for the character specified with `'wildchar'`. It is a comma-separated list of up to four parts. Each part specifies what to do for each consecutive use of `'wildchar'`. The first part specifies the behavior for the first use of `'wildchar'`, The second part for the second use, etc.

Each part consists of a colon separated list consisting of the following possible values:

- "" Complete only the first match.
- "full" Complete the next full match. After the last match, the original string is used and then the first match again. Will also start `'wildmenu'` if it is enabled.
- "longest" Complete till longest common string. If this doesn't result in a longer string, use the next part.
- "list" When more than one match, list all matches.
- "lastused" When completing buffer names and more than one buffer matches, sort buffers by time last used (other than the current buffer).

When there is only a single match, it is fully completed in all cases.

Examples of useful colon-separated values:

- "longest:full" Like "longest", but also start `'wildmenu'` if it is enabled. Will not complete to the next full match.
- "list:full" When more than one match, list all matches and complete first match.

"list:longest" When more than one match, list all matches and complete till longest common string.

"list:lastused" When more than one buffer matches, list all matches and sort buffers by time last used (other than the current buffer).

Examples:

```

:set wildmode=full
Complete first full match, next match, etc. (the default)
:set wildmode=longest,full
Complete longest common string, then each full match
:set wildmode=list:full
List all matches and complete each full match
:set wildmode=list,full
List all matches without completing, then each full match
:set wildmode=longest,list
Complete longest common string, then list alternatives.
More info here: cmdline-completion .

```

'wildoptions' 'wop' string (default "") 'wildoptions' 'wop'
 global

A list of words that change how [cmdline-completion](#) is done. The following values are supported:

fuzzy	Use fuzzy-matching to find completion matches. When this value is specified, wildcard expansion will not be used for completion. The matches will be sorted by the "best match" rather than alphabetically sorted. This will find more matches than the wildcard expansion. Currently fuzzy matching based completion is not supported for file and directory names and instead wildcard expansion is used.
pum	Display the completion matches using the popup menu in the same style as the ins-completion-menu .
tagfile	When using CTRL-D to list matching tags, the kind of tag and the file of the tag is listed. Only one match is displayed per line. Often used tag kinds are:
	d #define
	f function

'winaltkeys' 'wak' string (default "menu") 'winaltkeys' 'wak'
 global
 {only used in Win32, Motif, GTK and Photon GUI}

Some GUI versions allow the access to menu entries by using the ALT key in combination with a character that appears underlined in the menu. This conflicts with the use of the ALT key for mappings and entering special characters. This option tells what to do:

no	Don't use ALT keys for menus. ALT key combinations can be mapped, but there is no automatic handling. This can then be done with the :simalt command.
yes	ALT key handling is done by the windowing system. ALT key combinations cannot be mapped.
menu	Using ALT in combination with a character that is a menu

shortcut key, will be handled by the windowing system. Other keys can be mapped.

If the menu is disabled by excluding 'm' from **'guioptions'**, the ALT key is never used for the menu.

This option is not used for <F10>; on Win32 and with GTK <F10> will select the menu, unless it has been mapped.

'wincolor' **'wcr'** string (default empty) 'wincolor' 'wcr'
local to window
Highlight group name to use for this window instead of the Normal color `hl-Normal` .

'window' **'wi'** number (default screen height - 1) 'window' 'wi'
global
Window height used for `CTRL-F` and `CTRL-B` when there is only one window and the value is smaller than **'lines'** minus one. The screen will scroll **'window'** minus two lines, with a minimum of one. When **'window'** is equal to **'lines'** minus one `CTRL-F` and `CTRL-B` scroll in a much smarter way, taking care of wrapping lines. When resizing the Vim window, the value is smaller than 1 or more than or equal to **'lines'** it will be set to **'lines'** minus 1.
Note: Do not confuse this with the height of the Vim window, use **'lines'** for that.

'winfixbuf' **'wfb'** boolean (default off) 'winfixbuf'
local to window
If enabled, the window and the buffer it is displaying are paired. For example, attempting to change the buffer with `:edit` will fail. Other commands which change a window's buffer such as `:cnext` will also skip any window with **'winfixbuf'** enabled. However if an Ex command has a `!"` modifier, it can force switching buffers.

'winfixheight' **'wfh'** boolean (default off) 'winfixheight' 'wfh' 'nowinfixheight' 'nowfh'
local to window `local-noglobal`
Keep the window height when windows are opened or closed and **'equalalways'** is set. Also for `CTRL-W_=` . Set by default for the `preview-window` and `quickfix-window` .
The height may be changed anyway when running out of room.

'winfixwidth' **'wfw'** boolean (default off) 'winfixwidth' 'wfw' 'nowinfixwidth' 'nowfw'
local to window `local-noglobal`
Keep the window width when windows are opened or closed and **'equalalways'** is set. Also for `CTRL-W_=` .
The width may be changed anyway when running out of room.

'winheight' **'wh'** number (default 1) 'winheight' 'wh' E591
global
Minimal number of lines for the current window. This is not a hard

security reasons.

'winwidth' **'wiw'** number (default 20) **'winwidth'** **'wiw'** E592
 global

Minimal number of columns for the current window. This is not a hard minimum, Vim will use fewer columns if there is not enough room. If the current window is smaller, its size is increased, at the cost of the width of other windows. Set it to 999 to make the current window always fill the screen. Set it to a small number for normal editing. The width is not adjusted after one of the commands to change the width of the current window.

'winwidth' applies to the current window. Use **'winminwidth'** to set the minimal width for other windows.

'wrap' **'wrap'** **'nowrap'**
 boolean (default on)
 local to window

This option changes how text is displayed. It doesn't change the text in the buffer, see **'textwidth'** for that.

When on, lines longer than the width of the window will wrap and displaying continues on the next line. When off lines will not wrap and only part of long lines will be displayed. When the cursor is moved to a part that is not shown, the screen will scroll horizontally.

The line will be broken in the middle of a word if necessary. See **'linebreak'** to get the break at a word boundary.

To make scrolling horizontally a bit more useful, try this:

```
:set sidescroll=5
:set listchars+=precedes:<,extends:>
```

See **'sidescroll'**, **'listchars'** and `wrap-off` .

This option can't be set from a `modeline` when the **'diff'** option is on.

'wrapmargin' **'wm'** **'wrapmargin'** **'wm'**
 number (default 0)
 local to buffer

Number of characters from the right window border where wrapping starts. When typing text beyond this limit, an `<EOL>` will be inserted and inserting continues on the next line.

Options that add a margin, such as **'number'** and **'foldcolumn'**, cause the text width to be further reduced. This is Vi compatible.

When **'textwidth'** is non-zero, this option is not used.

This option is set to 0 when **'paste'** is set and restored when **'paste'** is reset.

See also **'formatoptions'** and `ins-textwidth` .

'wrapscan' **'ws'** **'wrapscan'** **'ws'** **'nowrapscan'** **'nows'**
 boolean (default on) E384 E385
 global

Searches wrap around the end of the file. Also applies to `]s` and `[s` , searching for spelling mistakes.

'write' **'nowrite'**

'write' boolean (default on)
 global
Allows writing files. When not set, writing a file is not allowed. Can be used for a view-only mode, where modifications to the text are still allowed. Can be reset with the `-m` or `-M` command line argument. Filtering text is still possible, even though this requires writing a temporary file.

'writeany' **'wa'** boolean (default off) **'writeany'** **'wa'** **'nowriteany'** **'nowa'**
 global
Allows writing to any file with no need for "!" override.

'writebackup' **'wb'** boolean (default on with `+writebackup` feature, off otherwise) **'writebackup'** **'wb'** **'nowritebackup'** **'nowb'**
 global
Make a backup before overwriting a file. The backup is removed after the file was successfully written, unless the **'backup'** option is also on.
WARNING: Switching this option off means that when Vim fails to write your buffer correctly and then, for whatever reason, Vim exits, you lose both the original file and what you were writing. Only reset this option if your file system is almost full and it makes the write fail (and make sure not to exit Vim until the write was successful). See [backup-table](#) for another explanation.
When the **'backupskip'** pattern matches, a backup is not made anyway. Depending on **'backupcopy'** the backup is a new file or the original file renamed (and a new file is written).
NOTE: This option is set to the default value when **'compatible'** is set.

'writedelay' **'wd'** number (default 0) **'writedelay'** **'wd'**
 global
The number of milliseconds to wait for each character sent to the screen. When non-zero, characters are sent to the terminal one by one. For debugging purposes.

'xtermcodes' boolean (default on) **'xtermcodes'** **'noxtermcodes'**
 global
When detecting xterm patchlevel 141 or higher with the termresponse mechanism and this option is set, Vim will request the actual terminal key codes and number of colors from the terminal. This takes care of various configuration options of the terminal that cannot be obtained from the termlib/terminfo entry, see [xterm-codes](#) .
A side effect may be that `t_Co` changes and Vim will redraw the display.

Patterns and search commands

pattern-searches

The very basics can be found in section 03.9 of the user manual. A few more explanations are in chapter 27 [usr_27.txt](#) .

- | | |
|--------------------------------|-------------------------------------|
| 1. Search commands | search-commands |
| 2. The definition of a pattern | search-pattern |
| 3. Magic | /magic |
| 4. Overview of pattern items | pattern-overview |
| 5. Multi items | pattern-multi-items |
| 6. Ordinary atoms | pattern-atoms |
| 7. Ignoring case in a pattern | /ignorecase |
| 8. Composing characters | patterns-composing |
| 9. Compare with Perl patterns | perl-patterns |
| 10. Highlighting matches | match-highlight |
| 11. Fuzzy matching | fuzzy-matching |

=====

1. Search commands

search-commands

- [/](#)
- [/{pattern}\[/\]<CR>](#) Search forward for the [\[count\]](#)'th occurrence of [{pattern}](#) [exclusive](#) .
- [/{pattern}/{offset}<CR>](#) Search forward for the [\[count\]](#)'th occurrence of [{pattern}](#) and go [{offset}](#) lines up or down. [linewise](#) .
- [/<CR>](#) Search forward for the [\[count\]](#)'th occurrence of the latest used pattern [last-pattern](#) with latest used [{offset}](#) .
- [//{offset}<CR>](#) Search forward for the [\[count\]](#)'th occurrence of the latest used pattern [last-pattern](#) with new [{offset}](#) . If [{offset}](#) is empty no offset is used.
- [?{pattern}\[?\]<CR>](#) Search backward for the [\[count\]](#)'th previous occurrence of [{pattern}](#) [exclusive](#) .
- [?{pattern}?{offset}<CR>](#) Search backward for the [\[count\]](#)'th previous occurrence of [{pattern}](#) and go [{offset}](#) lines up or down [linewise](#) .
- [?<CR>](#) Search backward for the [\[count\]](#)'th occurrence of the latest used pattern [last-pattern](#) with latest used

`{offset} .`

`??{offset}<CR>` Search backward for the `[count]`'th occurrence of the latest used pattern `last-pattern` with new `{offset}` . If `{offset}` is empty no offset is used.

`n` Repeat the latest `"/` or `"?` `n` `[count]` times. If the cursor doesn't move the search is repeated with `count + 1`.
`last-pattern`

`N` Repeat the latest `"/` or `"?` `N` `[count]` times in opposite direction. `last-pattern`

`*` Search forward for the `[count]`'th occurrence of the word nearest to the cursor. The word used for the search is the first of:

1. the keyword under the cursor `'iskeyword'`
2. the first keyword after the cursor, in the current line
3. the non-blank word under the cursor
4. the first non-blank word after the cursor, in the current line

Only whole keywords are searched for, like with the command `/\<keyword\>`. `exclusive` `'ignorecase'` is used, `'smartcase'` is not.

`#` Same as `"*`, but search backward. The pound sign (character 163) also works. If the `"#` key works as backspace, try using `"stty erase <BS>`" before starting Vim (`<BS>` is **CTRL-H** or a real backspace).

`g*` Like `"*`, but don't put `"\<` and `"\>` around the word. This makes the search also find matches that are not a whole word.

`g#` Like `"#`, but don't put `"\<` and `"\>` around the word. This makes the search also find matches that are not a whole word.

`gd` Goto local Declaration. When the cursor is on a local variable, this command will jump to its declaration. This was made to work for C code, in other languages it may not work well. First Vim searches for the start of the current function, just like `"[`". If it is not found the search stops in line 1. If it is found, Vim goes back

until a blank line is found. From this position Vim searches for the keyword under the cursor, like with "*", but lines that look like a comment are ignored (see '**comments**' option).

Note that this is not guaranteed to work, Vim does not really check the syntax, it only searches for a match with the keyword. If included files also need to be searched use the commands listed in [include-search](#) . After this command **n** searches forward for the next match (not backward).

gD **gD**
Goto global Declaration. When the cursor is on a global variable that is defined in the file, this command will jump to its declaration. This works just like "gd", except that the search for the keyword always starts in line 1.

lgd **lgd**
Like "gd", but ignore matches inside a {} block that ends before the cursor position.

lgD **lgD**
Like "gD", but ignore matches inside a {} block that ends before the cursor position.

CTRL-C **CTRL-C**
Interrupt current (search) command. Use **CTRL-Break** on MS-Windows [dos-CTRL-Break](#) .
In Normal mode, any pending command is aborted. When Vim was started with output redirected and there are no changed buffers **CTRL-C** exits Vim. That is to help users who use "vim file | grep word" and don't know how to get out (blindly typing :qa<CR> would work).

:noh[lsearch] **:noh** **:nohlsearch**
Stop the highlighting for the '**hlsearch**' option. It is automatically turned back on when using a search command, or setting the '**hlsearch**' option. This command doesn't work in an autocommand, because the highlighting state is saved and restored when executing autocommands [autocmd-searchpat](#) . Same thing for when invoking a user function.

While typing the search pattern the current match will be shown if the '**incsearch**' option is on. Remember that you still have to finish the search command with <CR> to actually position the cursor at the displayed match. Or use <Esc> to abandon the search.

All matches for the last used search pattern will be highlighted if you set the '**hlsearch**' option. This can be suspended with the **:nohlsearch** command.

When '**shortmess**' does not include the "S" flag, Vim will automatically show an

index, on which the cursor is. This can look like this:

```
[1/5]           Cursor is on first of 5 matches.
[1/>99]         Cursor is on first of more than 99 matches.
[>99/>99]       Cursor is after 99 match of more than 99 matches.
[?/??]         Unknown how many matches exists, generating the
                statistics was aborted because of search timeout.
```

Note: the count does not take offset into account.

When no match is found you get the error: **E486** Pattern not found

Note that for the `:global` command, when used in legacy script, you get a normal message "Pattern not found", for Vi compatibility.

In **Vim9** script you get **E486** for "pattern not found" or **E538** when the pattern matches in every line with `:vglobal`.

For the `:s` command the "e" flag can be used to avoid the error message `:s_flags`.

search-offset {offset}

These commands search for the specified pattern. With "/" and "?" an additional offset may be given. There are two types of offsets: line offsets and character offsets.

The offset gives the cursor position relative to the found match:

```
[num]          [num] lines downwards, in column 1
+[num]         [num] lines downwards, in column 1
-[num]         [num] lines upwards, in column 1
e+[num]        [num] characters to the right of the end of the match
e[-num]        [num] characters to the left of the end of the match
s+[num]        [num] characters to the right of the start of the match
s[-num]        [num] characters to the left of the start of the match
b+[num]        [num] identical to s+[num] above (mnemonic: begin)
b[-num]        [num] identical to s[-num] above (mnemonic: begin)
;{pattern}     perform another search, see //;
```

If a '-' or '+' is given but [num] is omitted, a count of one will be used. When including an offset with 'e', the search becomes inclusive (the character the cursor lands on is included in operations).

Examples:

pattern	cursor position
/test/+1	one line below "test", in column 1
/test/e	on the last t of "test"
/test/s+2	on the 's' of "test"
/test/b-3	three characters before "test"

If one of these commands is used after an operator, the characters between the cursor position before and after the search is affected. However, if a line offset is given, the whole lines between the two cursor positions are affected.

An example of how to search for matches with a pattern and change the match with another word:

/foo<CR>	find "foo"
c//e<CR>	change until end of match
bar<Esc>	type replacement
//<CR>	go to start of next match
c//e<CR>	change until end of match
beep<Esc>	type another replacement
	etc.

//; E386

A very special offset is ';' followed by another search command. For example:

```
/test 1;/test
/test.*/+1;?ing?
```

The first one first finds the next occurrence of "test 1", and then the first occurrence of "test" after that.

This is like executing two search commands after each other, except that:

- It can be used as a single motion command after an operator.
- The direction for a following "n" or "N" command comes from the first search command.
- When an error occurs the cursor is not moved at all.

last-pattern

The last used pattern and offset are remembered. They can be used to repeat the search, possibly in another direction or with another count. Note that two patterns are remembered: One for "normal" search commands and one for the substitute command ":s". Each time an empty pattern is given, the previously used pattern is used. However, if there is no previous search command, a previous substitute pattern is used, if possible.

The 'magic' option sticks with the last used pattern. If you change 'magic', this will not change how the last used pattern will be interpreted.

The 'ignorecase' option does not do this. When 'ignorecase' is changed, it will result in the pattern to match other text.

All matches for the last used search pattern will be highlighted if you set the 'hlsearch' option.

To clear the last used search pattern:

```
:let @/ = ""
```

This will not set the pattern to an empty string, because that would match everywhere. The pattern is really cleared, like when starting Vim.

The search usually skips matches that don't move the cursor. Whether the next match is found at the next character or after the skipped match depends on the 'c' flag in 'cptions'. See cpo-c .

with 'c' flag: "/"... advances 1 to 3 characters

without 'c' flag: "/"... advances 1 character

The unpredictability with the 'c' flag is caused by starting the search in the first column, skipping matches until one is found past the cursor position.

When searching backwards, searching starts at the start of the line, using the 'c' flag in 'cptions' as described above. Then the last match before the

cursor position is used.

In Vi the `":tag"` command sets the last search pattern when the tag is searched for. In Vim this is not done, the previous search pattern is still remembered, unless the `'t'` flag is present in `'coptions'`. The search pattern is always put in the search history.

If the `'wrapscan'` option is on (which is the default), searches wrap around the end of the buffer. If `'wrapscan'` is not set, the backward search stops at the beginning and the forward search stops at the end of the buffer. If `'wrapscan'` is set and the pattern was not found the error message "pattern not found" is given, and the cursor will not be moved. If `'wrapscan'` is not set the message becomes "search hit BOTTOM without match" when searching forward, or "search hit TOP without match" when searching backward. If `wrapscan` is set and the search wraps around the end of the file the message "search hit TOP, continuing at BOTTOM" or "search hit BOTTOM, continuing at TOP" is given when searching backwards or forwards respectively. This can be switched off by setting the `'s'` flag in the `'shortmess'` option. The highlight method `'w'` is used for this message (default: `standout`).

search-range

You can limit the search command `"/` to a certain range of lines by including `\%>l` items. For example, to match the word "limit" below line 199 and above line 300:

```
/\%>199l\%<300llimit
```

Also see `/\%>l` .

Another way is to use the `":substitute"` command with the `'c'` flag. Example:

```
.,300s/Pattern//gc
```

This command will search from the cursor position until line 300 for "Pattern". At the match, you will be asked to type a character. Type `'q'` to stop at this match, type `'n'` to find the next match.

The `"*"`, `"#"`, `"g*"` and `"g#"` commands look for a word near the cursor in this order, the first one that is found is used:

- The keyword currently under the cursor.
- The first keyword to the right of the cursor, in the same line.
- The WORD currently under the cursor.
- The first WORD to the right of the cursor, in the same line.

The keyword may only contain letters and characters in `'iskeyword'`.

The WORD may contain any non-blanks (`<Tab>`s and/or `<Space>`s).

Note that if you type with ten fingers, the characters are easy to remember: the `"#"` is under your left hand middle finger (search to the left and up) and the `"*"` is under your right hand middle finger (search to the right and down). (this depends on your keyboard layout though).

E956

In very rare cases a regular expression is used recursively. This can happen when executing a pattern takes a long time and when checking for messages on channels a callback is invoked that also uses a pattern or an autocommand is triggered. In most cases this should be fine, but if a pattern is in use when it's used again it fails. Usually this means there is something wrong with the pattern.

2. The definition of a pattern

```
search-pattern  pattern  [pattern]
regular-expression  regexp  Pattern
E383  E476
```

For starters, read chapter 27 of the user manual [usr_27.txt](#) .

1. A pattern is one or more branches, separated by `"|"`. It matches anything that matches one of the branches. Example: `"foo|beep"` matches `"foo"` and matches `"beep"`. If more than one branch matches, the first one is used.

```
pattern ::=      branch
           or branch | branch
           or branch | branch | branch
           etc.
```

2. A branch is one or more concats, separated by `"&"`. It matches the last concat, but only if all the preceding concats also match at the same position. Examples:

```
"/branch  /\&
"foobeep&..." matches "foo" in "foobeep".
".*Peter&.*Bob" matches in a line containing both "Peter" and "Bob"
```

```
branch ::=      concat
              or concat & concat
              or concat & concat & concat
              etc.
```

3. A concat is one or more pieces, concatenated. It matches a match for the first piece, followed by a match for the second piece, etc. Example: `"f[0-9]b"`, first matches `"f"`, then a digit and then `"b"`.

```
/concat
concat ::=      piece
              or piece piece
              or piece piece piece
              etc.
```

4. A piece is an atom, possibly followed by a multi, an indication of how many times the atom can be matched. Example: `"a*"` matches any sequence of `"a"` characters: `"", "a", "aa",` etc. See [/multi](#) .

```
/piece
piece ::=      atom
              or atom multi
```

5. An atom can be one of a long list of items. Many atoms match one character in the text. It is often an ordinary character or a character class. Parentheses can be used to make a pattern into an atom. The `"z(\)"` construct is only for syntax highlighting.

```
/atom
atom ::=      ordinary-atom  /ordinary-atom
```

```

or \( pattern \)           /\(
or \% ( pattern \)         /\%(
or \z( pattern \)         /\z(

```

/\%#= two-engines NFA

Vim includes two regexp engines:

1. An old, backtracking engine that supports everything.
2. A new, NFA engine that works much faster on some patterns, possibly slower on some patterns.

E1281

Vim will automatically select the right engine for you. However, if you run into a problem or want to specifically select one engine or the other, you can prepend one of the following to the pattern:

```

\%#=0 Force automatic selection. Only has an effect when
      'regengine' has been set to a non-zero value.
\%#=1 Force using the old engine.
\%#=2 Force using the NFA engine.

```

You can also use the **'regengine'** option to change the default.

E864 E868 E874 E875 E876 E877 E878

If selecting the NFA engine and it runs into something that is not implemented the pattern will not match. This is only useful when debugging Vim.

3. Magic /magic

Some characters in the pattern, such as letters, are taken literally. They match exactly the same character in the text. When preceded with a backslash however, these characters may get a special meaning. For example, "a" matches the letter "a", while "\a" matches any alphabetic character.

Other characters have a special meaning without a backslash. They need to be preceded with a backslash to match literally. For example "." matches any character while "\." matches a dot.

If a character is taken literally or not depends on the **'magic'** option and the items in the pattern mentioned next. The **'magic'** option should always be set, but it can be switched off for Vi compatibility. We mention the effect of **'nomagic'** here for completeness, but we recommend against using that.

Use of "\m" makes the pattern after it be interpreted as if **'magic'** is set, ignoring the actual value of the **'magic'** option.

Use of "\M" makes the pattern after it be interpreted as if **'nomagic'** is used.

Use of "\v" means that after it, all ASCII characters except '0'-'9', 'a'-'z', 'A'-'Z' and '_' have special meaning: "very magic"

Use of "\V" means that after it, only a backslash and the terminating character (usually / or ?) have special meaning: "very nomagic"

Examples:

after:	<code>\v</code>	<code>\m</code>	<code>\M</code>	<code>\V</code>	matches
		'magic'	'nomagic'		
	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	literal 'a'
	<code>\a</code>	<code>\a</code>	<code>\a</code>	<code>\a</code>	any alphabetic character
	<code>.</code>	<code>.</code>	<code>\.</code>	<code>\.</code>	any character
	<code>\.</code>	<code>\.</code>	<code>.</code>	<code>.</code>	literal dot
	<code>\$</code>	<code>\$</code>	<code>\$</code>	<code>\\$</code>	end-of-line
	<code>*</code>	<code>*</code>	<code>*</code>	<code>*</code>	any number of the previous atom
	<code>~</code>	<code>~</code>	<code>\~</code>	<code>\~</code>	latest substitute string
	<code>()</code>	<code>\(\)</code>	<code>\(\)</code>	<code>\(\)</code>	group as an atom
	<code> </code>	<code>\ </code>	<code>\ </code>	<code>\ </code>	nothing: separates alternatives
	<code>\\</code>	<code>\\</code>	<code>\\</code>	<code>\\</code>	literal backslash
	<code>\{</code>	<code>{</code>	<code>{</code>	<code>{</code>	literal curly brace

{only Vim supports `\m`, `\M`, `\v` and `\V`}

If you want to you can make a pattern immune to the 'magic' option being set or not by putting `"\m`" or `"\M`" at the start of the pattern.

4. Overview of pattern items

[pattern-overview](#)
[E865](#) [E866](#) [E867](#) [E869](#)

Overview of multi items.

[/multi](#) [E61](#) [E62](#)
[E64](#) [E871](#)

More explanation and examples below, follow the links.

	multi		matches of the preceding atom	
	'magic'	'nomagic'		
<code>/star</code>	<code>*</code>	<code>*</code>	0 or more	as many as possible
<code>/\+</code>	<code>\+</code>	<code>\+</code>	1 or more	as many as possible
<code>/\=</code>	<code>\=</code>	<code>\=</code>	0 or 1	as many as possible
<code>/\?</code>	<code>\?</code>	<code>\?</code>	0 or 1	as many as possible
<code>/\{</code>	<code>\{n,m}</code>	<code>\{n,m}</code>	n to m	as many as possible
	<code>\{n}</code>	<code>\{n}</code>	n	exactly
	<code>\{n,}</code>	<code>\{n,}</code>	at least n	as many as possible
	<code>\{,m}</code>	<code>\{,m}</code>	0 to m	as many as possible
	<code>\{}</code>	<code>\{}</code>	0 or more	as many as possible (same as *)
<code>/\{-</code>	<code>\{-n,m}</code>	<code>\{-n,m}</code>	n to m	as few as possible
	<code>\{-n}</code>	<code>\{-n}</code>	n	exactly
	<code>\{-n,}</code>	<code>\{-n,}</code>	at least n	as few as possible
	<code>\{-,m}</code>	<code>\{-,m}</code>	0 to m	as few as possible
	<code>\{-}</code>	<code>\{-}</code>	0 or more	as few as possible

[E59](#)

<code>/\@></code>	<code>\@></code>	<code>\@></code>	1, like matching a whole pattern
<code>/\@=</code>	<code>\@=</code>	<code>\@=</code>	nothing, requires a match /zero-width
<code>/\@!</code>	<code>\@!</code>	<code>\@!</code>	nothing, requires NO match /zero-width
<code>/\@<=</code>	<code>\@<=</code>	<code>\@<=</code>	nothing, requires a match behind /zero-width
<code>/\@<!</code>	<code>\@<!</code>	<code>\@<!</code>	nothing, requires NO match behind /zero-width

Overview of ordinary atoms.

[/ordinary-atom](#)

More explanation and examples below, follow the links.

	ordinary	atom		
	magic	nomagic	matches	
/^	^	^	start-of-line (at start of pattern)	/zero-width
/\^	\^	\^	literal '^'	
/_^	_^	_^	start-of-line (used anywhere)	/zero-width
/\\$	\\$	\\$	end-of-line (at end of pattern)	/zero-width
/\\\$	\\\$	\\$	literal '\$'	
/_\\$	_\\$	_\\$	end-of-line (used anywhere)	/zero-width
/.	.	.	any single character (not an end-of-line)	
/_.	_.	_.	any single character or end-of-line	
/\<	\<	\<	beginning of a word	/zero-width
/\>	\>	\>	end of a word	/zero-width
/\zs	\zs	\zs	anything, sets start of match	
/\ze	\ze	\ze	anything, sets end of match	
/\%^	\%^	\%^	beginning of file	/zero-width E71
/\%\\$	\%\\$	\%\\$	end of file	/zero-width
/\%V	\%V	\%V	inside Visual area	/zero-width
/\%#	\%#	\%#	cursor position	/zero-width
/\%'m	\%'m	\%'m	mark m position	/zero-width
/\%l	\%23l	\%23l	in line 23	/zero-width
/\%c	\%23c	\%23c	in column 23	/zero-width
/\%v	\%23v	\%23v	in virtual column 23	/zero-width

Character classes:

/character-classes

	magic	nomagic	matches	
/\i	\i	\i	identifier character (see ' isident ' option)	
/\I	\I	\I	like "\i", but excluding digits	
/\k	\k	\k	keyword character (see ' iskeyword ' option)	
/\K	\K	\K	like "\k", but excluding digits	
/\f	\f	\f	file name character (see ' isfname ' option)	
/\F	\F	\F	like "\f", but excluding digits	
/\p	\p	\p	printable character (see ' isprint ' option)	
/\P	\P	\P	like "\p", but excluding digits	
/\s	\s	\s	whitespace character: <Space> and <Tab>	
/\S	\S	\S	non-whitespace character; opposite of \s	
/\d	\d	\d	digit:	[0-9]
/\D	\D	\D	non-digit:	[^0-9]
/\x	\x	\x	hex digit:	[0-9A-Fa-f]
/\X	\X	\X	non-hex digit:	[^0-9A-Fa-f]
/\o	\o	\o	octal digit:	[0-7]
/\O	\O	\O	non-octal digit:	[^0-7]
/\w	\w	\w	word character:	[0-9A-Za-z_]
/\W	\W	\W	non-word character:	[^0-9A-Za-z_]
/\h	\h	\h	head of word character:	[A-Za-z_]
/\H	\H	\H	non-head of word character:	[^A-Za-z_]
/\a	\a	\a	alphabetic character:	[A-Za-z]
/\A	\A	\A	non-alphabetic character:	[^A-Za-z]
/\l	\l	\l	lowercase character:	[a-z]
/\L	\L	\L	non-lowercase character:	[^a-z]
/\u	\u	\u	uppercase character:	[A-Z]
/\U	\U	\U	non-uppercase character	[^A-Z]
/_x	_x	_x	where x is any of the characters above: character	

class with end-of-line included
(end of character classes)

	magic	nomagic	matches
<code>/\e</code>	<code>\e</code>	<code>\e</code>	<Esc>
<code>/\t</code>	<code>\t</code>	<code>\t</code>	<Tab>
<code>/\r</code>	<code>\r</code>	<code>\r</code>	<CR>
<code>/\b</code>	<code>\b</code>	<code>\b</code>	<BS>
<code>/\n</code>	<code>\n</code>	<code>\n</code>	end-of-line
<code>/~</code>	<code>~</code>	<code>~</code>	last given substitute string
<code>/\1</code>	<code>\1</code>	<code>\1</code>	same string as matched by first <code>\(\)</code>
<code>/\2</code>	<code>\2</code>	<code>\2</code>	Like <code>"\1"</code> , but uses second <code>\(\)</code>
<code>/\9</code>	<code>\9</code>	<code>\9</code>	Like <code>"\1"</code> , but uses ninth <code>\(\)</code>
<code>/\z1</code>	<code>\z1</code>	<code>\z1</code>	only for syntax highlighting, see :syn-ext-match E68
<code>/\z1</code>	<code>\z9</code>	<code>\z9</code>	only for syntax highlighting, see :syn-ext-match
	<code>x</code>	<code>x</code>	a character with no special meaning matches itself
<code>/[]</code>	<code>[]</code>	<code>\[]</code>	any character specified inside the <code>[]</code>
<code>/\%[]</code>	<code>\%[]</code>	<code>\%[]</code>	a sequence of optionally matched atoms
<code>/\c</code>	<code>\c</code>	<code>\c</code>	ignore case, do not use the 'ignorecase' option
<code>/\C</code>	<code>\C</code>	<code>\C</code>	match case, do not use the 'ignorecase' option
<code>/\Z</code>	<code>\Z</code>	<code>\Z</code>	ignore differences in Unicode "combining characters". Useful when searching voweled Hebrew or Arabic text.

	magic	nomagic	matches
<code>/\m</code>	<code>\m</code>	<code>\m</code>	'magic' on for the following chars in the pattern
<code>/\M</code>	<code>\M</code>	<code>\M</code>	'magic' off for the following chars in the pattern
<code>/\v</code>	<code>\v</code>	<code>\v</code>	the following chars in the pattern are "very magic"
<code>/\V</code>	<code>\V</code>	<code>\V</code>	the following chars in the pattern are "very nomagic"
<code>/\%#=</code>	<code>\%=1</code>	<code>\%=1</code>	select regexp engine /zero-width
<code>/\%d</code>	<code>\%d</code>	<code>\%d</code>	match specified decimal character (eg <code>\%d123</code>)
<code>/\%x</code>	<code>\%x</code>	<code>\%x</code>	match specified hex character (eg <code>\%x2a</code>)
<code>/\%o</code>	<code>\%o</code>	<code>\%o</code>	match specified octal character (eg <code>\%o040</code>)
<code>/\%u</code>	<code>\%u</code>	<code>\%u</code>	match specified multibyte character (eg <code>\%u20ac</code>)
<code>/\%U</code>	<code>\%U</code>	<code>\%U</code>	match specified large multibyte character (eg <code>\%U12345678</code>)
<code>/\%C</code>	<code>\%C</code>	<code>\%C</code>	match any composing characters

Example

```
\<[Ii]*           or
\<[h\w]*
\<[a-zA-Z_][a-zA-Z0-9_]*
```

matches

An identifier (e.g., in a C program).

```
\(\. $\ | \. \)
```

A period followed by <EOL> or a space.

```
[.!?][()'"']*\\($|[ ]\\)
```

A search pattern that finds the end of a sentence, with almost the same definition as the `")"` command.

cat\Z Both "cat" and "càt" ("a" followed by 0x0300)
Does not match "càt" (character 0x00e0), even
though it may look the same.

5. Multi items

pattern-multi-items

An atom can be followed by an indication of how many times the atom can be matched and in what way. This is called a multi. See [/multi](#) for an overview.

/star \star

* (use * when 'magic' is not set)
Matches 0 or more of the preceding atom, as many as possible.
Example 'nomagic' matches
a* a* "", "a", "aa", "aaa", etc.
. * \.* anything, also an empty string, no end-of-line
_.* _.* everything up to the end of the buffer
_.*END _.*END everything up to and including the last "END"
in the buffer

Exception: When "*" is used at the start of the pattern or just after "^" it matches the star character.

Be aware that repeating "_." can match a lot of text and take a long time. For example, "_.*END" matches all text from the current position to the last occurrence of "END" in the file. Since the "*" will match as many as possible, this first skips over all lines until the end of the file and then tries matching "END", backing up one character at a time.

/\+

\+ Matches 1 or more of the preceding atom, as many as possible.
Example matches
^\.+ \$ any non-empty line
\s\+ white space of at least one character

/\=

\= Matches 0 or 1 of the preceding atom, as many as possible.
Example matches
foo\= "fo" and "foo"

/\?

\? Just like \=. Cannot be used when searching backwards with the "?" command.

/\{ E60 E554 E870

\{n,m} Matches n to m of the preceding atom, as many as possible
\{n} Matches n of the preceding atom
\{n,} Matches at least n of the preceding atom, as many as possible
\{,m} Matches 0 to m of the preceding atom, as many as possible
\{} Matches 0 or more of the preceding atom, as many as possible (like *)

/\{-

\{-n,m} matches n to m of the preceding atom, as few as possible
 \{-n} matches n of the preceding atom
 \{-n,} matches at least n of the preceding atom, as few as possible
 \{-,m} matches 0 to m of the preceding atom, as few as possible
 \{-} matches 0 or more of the preceding atom, as few as possible

n and m are positive decimal numbers or zero

non-greedy

If a "-" appears immediately after the "{", then a shortest match first algorithm is used (see example below). In particular, "\{-}" is the same as "*" but uses the shortest match first algorithm. BUT: A match that starts earlier is preferred over a shorter match: "a\{-}b" matches "aaab" in "xaaab".

Example	matches
ab\{2,3}c	"abbc" or "abbbc"
a\{5}	"aaaaa"
ab\{2,}c	"abbc", "abbbc", "abbbbc", etc.
ab\{,3}c	"ac", "abc", "abbc" or "abbbc"
a[bc]\{3}d	"abbbd", "abbcd", "acbcd", "accdd", etc.
a(bc)\{1,2}d	"abcd" or "abcbcd"
a[bc]\{-}[cd]	"abc" in "abcd"
a[bc]*[cd]	"abcd" in "abcd"

The } may optionally be preceded with a backslash: \{n,m\}.

/\@=

\@= Matches the preceding atom with zero width.
 Like "(?=pattern)" in Perl.

Example	matches
foo(bar)\@=	"foo" in "foobar"
foo(bar)\@=foo	nothing

/zero-width

When using "\@=" (or "^", "\$", "<", ">") no characters are included in the match. These items are only used to check if a match can be made. This can be tricky, because a match with following items will be done in the same position. The last example above will not match "foobarfoo", because it tries match "foo" in the same position where "bar" matched.

Note that using "\&" works the same as using "\@=": "foo&.." is the same as "\(foo)\@=..". But using "\&" is easier, you don't need the parentheses.

/\@!

\@! Matches with zero width if the preceding atom does NOT match at the current position. /zero-width
 Like "(?!pattern)" in Perl.

Example	matches
foo(bar)\@!	any "foo" not followed by "bar"
a.\{-}p\@!	"a", "ap", "app", "apppp", etc. not immediately followed by a "p"

if `\(\\(then\\)\@!.\\)*$` "if " not followed by "then"

Using "`@!`" is tricky, because there are many places where a pattern does not match. "`a.*p@!`" will match from an "a" to the end of the line, because "`.*`" can match all characters in the line and the "p" doesn't match at the end of the line. "`a.\{-}p@!`" will match any "a", "ap", "app", etc. that isn't followed by a "p", because the "." can match a "p" and "`p@!`" doesn't match after that.

You can't use "`@!`" to look for a non-match before the matching position: "`(foo\\)\@!bar`" will match "bar" in "foobar", because at the position where "bar" matches, "foo" does not match. To avoid matching "foobar" you could use "`(foo\\)\@!...bar`", but that doesn't match a bar at the start of a line. Use "`(foo\\)\@<!bar`".

Useful example: to find "foo" in a line that does not contain "bar":
`/^\%(.*bar\\)\@!.*\zsfoo`

This pattern first checks that there is not a single position in the line where "bar" matches. If "`*bar`" matches somewhere the `@!` will reject the pattern. When there is no match any "foo" will be found. The "`\zs`" is to have the match start just before "foo".

`\@<=` `/\@<=`
Matches with zero width if the preceding atom matches just before what follows. `/zero-width`
Like "`(?<=pattern)`" in Perl, but Vim allows non-fixed-width patterns.
Example matches
`\(an_s\+\)\@<=file` "file" after "an" and white space or an end-of-line

For speed it's often much better to avoid this multi. Try using "`\zs`" instead `/\zs`. To match the same as the above example:
`an_s\+\zsfile`

At least set a limit for the look-behind, see below.

"`\@<=`" and "`\@<!`" check for matches just before what follows. Theoretically these matches could start anywhere before this position. But to limit the time needed, only the line where what follows matches is searched, and one line before that (if there is one). This should be sufficient to match most things and not be too slow.

In the old regexp engine the part of the pattern after "`\@<=`" and "`\@<!`" are checked for a match first, thus things like "`\1`" don't work to reference `\(\)` inside the preceding atom. It does work the other way around:

Bad example matches
`\%#=1\1\@<=,\([a-z]\+\)` ",abc" in "abc,abc"

However, the new regexp engine works differently, it is better to not rely on this behavior, do not use `\@<=` if it can be avoided:

Example matches
`\([a-z]\+\)\zs,\1` ",abc" in "abc,abc"

`\@123<=`
Like "`\@<=`" but only look back 123 bytes. This avoids trying lots

of matches that are known to fail and make executing the pattern very slow. Example, check if there is a "<" just before "span":

```
/< \@1<=span
```

This will try matching "<" only one byte before "span", which is the only place that works anyway.

After crossing a line boundary, the limit is relative to the end of the line. Thus the characters at the start of the line with the match are not counted (this is just to keep it simple).

The number zero is the same as no limit.

`\@<!` /`\@<!`
Matches with zero width if the preceding atom does NOT match just before what follows. Thus this matches if there is no position in the current or previous line where the atom matches such that it ends just before what follows. /`zero-width`

Like "(?<!pattern)" in Perl, but Vim allows non-fixed-width patterns. The match with the preceding atom is made to end just before the match with what follows, thus an atom that ends in ".*" will work.

Warning: This can be slow (because many positions need to be checked for a match). Use a limit if you can, see below.

Example	matches
<code>\(foo\) \@<!bar</code>	any "bar" that's not in "foobar"
<code>\(\/\./.*\) \@<!in</code>	"in" which is not after "///"

`\@123<!`
Like "@<!" but only look back 123 bytes. This avoids trying lots of matches that are known to fail and make executing the pattern very slow.

`\@>` /`\@>`
Matches the preceding atom like matching a whole pattern. Like "(?>pattern)" in Perl.

Example	matches
<code>\(a*\) \@>a</code>	nothing (the "a*" takes all the "a"'s, there can't be another one following)

This matches the preceding atom as if it was a pattern by itself. If it doesn't match, there is no retry with shorter sub-matches or anything. Observe this difference: "a*b" and "a*ab" both match "aaab", but in the second case the "a*" matches only the first two "a"s. "`\(a*\) \@>ab`" will not match "aaab", because the "a*" matches the "aaa" (as many "a"s as possible), thus the "ab" can't match.

6. Ordinary atoms pattern-atoms

An ordinary atom can be:

`^` /`^`
At beginning of pattern or after "\|", "\(", "%(" or "\n": matches start-of-line; at other positions, matches literal '^'. /`zero-width`

Example	matches
<code>^beep(</code>	the start of the C function "beep" (probably).

`\^` Matches literal '^'. Can be used at any position in the pattern, but not inside []. /^

`_` Matches start-of-line. /^_ Can be used at any position in the pattern, but not inside []. ^
/zero-width
Example matches
`_s*_^foo` white space and blank lines and then "foo" at start-of-line

`$` At end of pattern or in front of "\\|", "\\)" or "\\n" ('magic' on): matches end-of-line /\$ <EOL>; at other positions, matches literal '\$'. /zero-width

`\$` Matches literal '\$'. Can be used at any position in the pattern, but not inside []. /^\$

`_` Matches end-of-line. /^_ Can be used at any position in the pattern, but not inside []. \$ <EOL> **Note** that "a_b" never matches, since "b" cannot match an end-of-line. Use "a\nb" instead /^_ \$ <EOL>.
/zero-width
Example matches
`foo_s*` "foo" at end-of-line and following white space and blank lines

`.` (with 'nomagic': \.) /^. /^. Matches any single character, but not an end-of-line.

`_.` /^_. Matches any single character or end-of-line.
Careful: "_.*" matches all text to the end of the buffer!

`\<` /^< Matches the beginning of a word: The next char is the first char of a word. The 'iskeyword' option specifies what is a word character. /^<
/zero-width

`\>` /^> Matches the end of a word: The previous char is the last char of a word. The 'iskeyword' option specifies what is a word character. /^>
/zero-width

`\zs` /^zs Matches at any position, but not inside [], and sets the start of the match there: The next char is the first char of the whole match. /^zs
/zero-width
Example:
/^s*\zsif
/zero-width
matches an "if" at the start of a line, ignoring white space.
Can be used multiple times, the last one encountered in a matching

branch is used. Example:
`/\(. \{-}\zsFab\)\{3}`
Finds the third occurrence of "Fab".
This cannot be followed by a multi. E888
{not available when compiled without the |+syntax| feature}

`\ze` /`\ze`
Matches at any position, but not inside [], and sets the end of the match there: The previous char is the last char of the whole match.
/`zero-width`
Can be used multiple times, the last one encountered in a matching branch is used.
Example: "end\ze(if|for)" matches the "end" in "endif" and "endfor".
This cannot be followed by a multi. E888
{not available when compiled without the |+syntax| feature}

`\%^` /`\%^` start-of-file
Matches start of the file. When matching with a string, matches the start of the string.
For example, to find the first "VIM" in a file:
`/\%^ _ \{-}\zsVIM`

`\%$` /`\%$` end-of-file
Matches end of the file. When matching with a string, matches the end of the string.
Note that this does NOT find the last "VIM" in a file:
`/VIM _ \{-}\%$`
It will find the next VIM, because the part after it will always match. This one will find the last "VIM" in the file:
`/VIM\ze\(\(VIM\)\@!_ \)*\%$`
This uses `/\@!` to ascertain that "VIM" does NOT match in any position after the first "VIM".
Searching from the end of the file backwards is easier!

`\%V` /`\%V`
Match inside the Visual area. When Visual mode has already been stopped match in the area that `gv` would reselect.
This is a `/zero-width` match. To make sure the whole pattern is inside the Visual area put it at the start and just before the end of the pattern, e.g.:
`/\%Vfoo.*ba\%Vr`
This also works if only "foo bar" was Visually selected. This:
`/\%Vfoo.*bar\%V`
would match "foo bar" if the Visual selection continues after the "r".
Only works for the current buffer.

`\%#` /`\%#` cursor-position
Matches with the cursor position. Only works when matching in a buffer displayed in a window.
WARNING: When the cursor is moved after the pattern was used, the result becomes invalid. Vim doesn't automatically update the matches. This is especially relevant for syntax highlighting and `'hlsearch'`.
In other words: When the cursor moves the display isn't updated for this change. An update is done for lines which are changed (the whole

line is updated) or when using the `CTRL-L` command (the whole screen is updated). Example, to highlight the word under the cursor:

```
/\k*\%#\k*
```

When `'hlsearch'` is set and you move the cursor around and make changes this will clearly show when the match is updated or not.

```

\%'m      Matches with the position of mark m.
\%<'m     Matches before the position of mark m.
\%>'m     Matches after the position of mark m.
```

Example, to highlight the text from mark 's to 'e:

```
/.%>'s.*\%<'e..
```

Note that two dots are required to include mark 'e in the match. That is because `"\%<'e"` matches at the character before the 'e mark, and since it's a `/zero-width` match it doesn't include that character.

WARNING: When the mark is moved after the pattern was used, the result becomes invalid. Vim doesn't automatically update the matches.

Similar to moving the cursor for `"\%#" /%\%#`.

```

\%23l     Matches in a specific line.
\%<23l    Matches above a specific line (lower line number).
\%>23l    Matches below a specific line (higher line number).
\%.l      Matches at the cursor line.
\%<.l     Matches above the cursor line.
\%>.l     Matches below the cursor line.
```

These six can be used to match specific lines in a buffer. The "23" can be any line number. The first line is 1.

WARNING: When inserting or deleting lines Vim does not automatically update the matches. This means Syntax highlighting quickly becomes wrong. Also when referring to the cursor position (".") and the cursor moves the display isn't updated for this change. An update is done when using the `CTRL-L` command (the whole screen is updated).

Example, to highlight the line where the cursor currently is:

```
:exe '\%' . line(".") . 'l'
```

Alternatively use:

```
/\%.l
```

When `'hlsearch'` is set and you move the cursor around and make changes this will clearly show when the match is updated or not.

```

\%23c     Matches in a specific column.
\%<23c    Matches before a specific column.
\%>23c    Matches after a specific column.
\%.c      Matches at the cursor column.
\%<.c     Matches before the cursor column.
\%>.c     Matches after the cursor column.
```

These six can be used to match specific columns in a buffer or string. The "23" can be any column number. The first column is 1. Actually, the column is the byte number (thus it's not exactly right for multibyte characters).

WARNING: When inserting or deleting text Vim does not automatically update the matches. This means Syntax highlighting quickly becomes wrong. Also when referring to the cursor position (".") and

the cursor moves the display isn't updated for this change. An update is done when using the `CTRL-L` command (the whole screen is updated). Example, to highlight the column where the cursor currently is:

```
:exe '/\%' .. col(".") .. 'c'
```

Alternatively use:

```
/\%.c
```

When **'hlsearch'** is set and you move the cursor around and make changes this will clearly show when the match is updated or not.

Example for matching a single byte in column 44:

```
/\%>43c.\%<46c
```

Note that "\%<46c" matches in column 45 when the "." matches a byte in column 44.

`/\%v` `/\%>v` `/\%<v`

`\/%23v` Matches in a specific virtual column.
`\/%<23v` Matches before a specific virtual column.
`\/%>23v` Matches after a specific virtual column.
`\/%.v` Matches at the current virtual column.
`\/%<.v` Matches before the current virtual column.
`\/%>.v` Matches after the current virtual column.

These six can be used to match specific virtual columns in a buffer or string. When not matching with a buffer in a window, the option values of the current window are used (e.g., **'tabstop'**).

The "23" can be any column number. The first column is 1.

Note that some virtual column positions will never match, because they are halfway through a tab or other character that occupies more than one screen character.

WARNING: When inserting or deleting text Vim does not automatically update highlighted matches. This means Syntax highlighting quickly becomes wrong. Also when referring to the cursor position (".") and the cursor moves the display isn't updated for this change. An update is done when using the `CTRL-L` command (the whole screen is updated). Example, to highlight all the characters after virtual column 72:

```
/\%>72v.*
```

When **'hlsearch'** is set and you move the cursor around and make changes this will clearly show when the match is updated or not.

To match the text up to column 17:

```
/^.*\%17v
```

To match all characters after the current virtual column (where the cursor is):

```
/\%>.v.*
```

Column 17 is not included, because this is a `/zero-width` match. To include the column use:

```
/^.*\%17v.
```

This command does the same thing, but also matches when there is no character in column 17:

```
/^.*\%<18v.
```

Note that without the "^" to anchor the match in the first column, this will also highlight column 17:

```
/.*\%17v
```

Column 17 is highlighted by **'hlsearch'** because there is another match where ".*" matches zero characters.

Character classes:

<code>\i</code>	identifier character (see 'isident' option)	<code>\/i</code>
<code>\I</code>	like <code>"\i"</code> , but excluding digits	<code>\/I</code>
<code>\k</code>	keyword character (see 'iskeyword' option)	<code>\/k</code>
<code>\K</code>	like <code>"\k"</code> , but excluding digits	<code>\/K</code>
<code>\f</code>	file name character (see 'isfname' option)	<code>\/f</code>
<code>\F</code>	like <code>"\f"</code> , but excluding digits	<code>\/F</code>
<code>\p</code>	printable character (see 'isprint' option)	<code>\/p</code>
<code>\P</code>	like <code>"\p"</code> , but excluding digits	<code>\/P</code>

NOTE: the above also work for multibyte characters. The ones below only match ASCII characters, as indicated by the range.

		whitespace	white-space
<code>\s</code>	whitespace character: <code><Space></code> and <code><Tab></code>	<code>\/s</code>	
<code>\S</code>	non-whitespace character; opposite of <code>\s</code>	<code>\/S</code>	
<code>\d</code>	digit: <code>[0-9]</code>	<code>\/d</code>	
<code>\D</code>	non-digit: <code>[^0-9]</code>	<code>\/D</code>	
<code>\x</code>	hex digit: <code>[0-9A-Fa-f]</code>	<code>\/x</code>	
<code>\X</code>	non-hex digit: <code>[^0-9A-Fa-f]</code>	<code>\/X</code>	
<code>\o</code>	octal digit: <code>[0-7]</code>	<code>\/o</code>	
<code>\O</code>	non-octal digit: <code>[^0-7]</code>	<code>\/O</code>	
<code>\w</code>	word character: <code>[0-9A-Za-z_]</code>	<code>\/w</code>	
<code>\W</code>	non-word character: <code>[^0-9A-Za-z_]</code>	<code>\/W</code>	
<code>\h</code>	head of word character: <code>[A-Za-z_]</code>	<code>\/h</code>	
<code>\H</code>	non-head of word character: <code>[^A-Za-z_]</code>	<code>\/H</code>	
<code>\a</code>	alphabetic character: <code>[A-Za-z]</code>	<code>\/a</code>	
<code>\A</code>	non-alphabetic character: <code>[^A-Za-z]</code>	<code>\/A</code>	
<code>\l</code>	lowercase character: <code>[a-z]</code>	<code>\/l</code>	
<code>\L</code>	non-lowercase character: <code>[^a-z]</code>	<code>\/L</code>	
<code>\u</code>	uppercase character: <code>[A-Z]</code>	<code>\/u</code>	
<code>\U</code>	non-uppercase character: <code>[^A-Z]</code>	<code>\/U</code>	

NOTE: Using the atom is faster than the `[]` form.

NOTE: **'ignorecase'**, `"\c"` and `"\C"` are not used by character classes.

	<code>\/_</code>	E63	<code>\/_i</code>	<code>\/_I</code>	<code>\/_k</code>	<code>\/_K</code>	<code>\/_f</code>	<code>\/_F</code>
	<code>\/_p</code>	<code>\/_P</code>	<code>\/_s</code>	<code>\/_S</code>	<code>\/_d</code>	<code>\/_D</code>	<code>\/_x</code>	<code>\/_X</code>
	<code>\/_o</code>	<code>\/_O</code>	<code>\/_w</code>	<code>\/_W</code>	<code>\/_h</code>	<code>\/_H</code>	<code>\/_a</code>	<code>\/_A</code>
	<code>\/_l</code>	<code>\/_L</code>	<code>\/_u</code>	<code>\/_U</code>				

`_x` Where "x" is any of the characters above: The character class with end-of-line added
(end of character classes)

<code>\e</code>	matches <code><Esc></code>	<code>\/e</code>
<code>\t</code>	matches <code><Tab></code>	<code>\/t</code>
<code>\r</code>	matches <code><CR></code>	<code>\/r</code>
<code>\b</code>	matches <code><BS></code>	<code>\/b</code>
<code>\n</code>	matches an end-of-line	<code>\/n</code>

When matching in a string instead of buffer text a literal newline character is matched.

<code>~</code>	matches the last given substitute string	<code>\/~</code>	<code>\/~</code>
----------------	--	------------------	------------------

- If two characters in the sequence are separated by '-', this is shorthand for the full list of ASCII characters between them. E.g., "[0-9]" matches any decimal digit. If the starting character exceeds the ending character, e.g. [c-a], E944 occurs. Non-ASCII characters can be used, but the character values must not be more than 256 apart in the old regexp engine. For example, searching by [\u3000-\u4000] after setting re=1 emits a E945 error. Prepending \##=2 will fix it.
- A character class expression is evaluated to the set of characters belonging to that character class. The following character classes are supported:

	Name	Func	Contents
<code>[:alnum:]</code>	<code>[:alnum:]</code>	isalnum	ASCII letters and digits
<code>[:alpha:]</code>	<code>[:alpha:]</code>	isalpha	ASCII letters
<code>[:blank:]</code>	<code>[:blank:]</code>		space and tab
<code>[:cntrl:]</code>	<code>[:cntrl:]</code>	iscntrl	ASCII control characters
<code>[:digit:]</code>	<code>[:digit:]</code>		decimal digits '0' to '9'
<code>[:graph:]</code>	<code>[:graph:]</code>	isgraph	ASCII printable characters excluding space
<code>[:lower:]</code>	<code>[:lower:]</code>	(1)	lowercase letters (all letters when 'ignorecase' is used)
<code>[:print:]</code>	<code>[:print:]</code>	(2)	printable characters including space
<code>[:punct:]</code>	<code>[:punct:]</code>	ispunct	ASCII punctuation characters
<code>[:space:]</code>	<code>[:space:]</code>		whitespace characters: space, tab, CR, NL, vertical tab, form feed
<code>[:upper:]</code>	<code>[:upper:]</code>	(3)	uppercase letters (all letters when 'ignorecase' is used)
<code>[:xdigit:]</code>	<code>[:xdigit:]</code>		hexadecimal digits: 0-9, a-f, A-F
<code>[:return:]</code>	<code>[:return:]</code>		the <CR> character
<code>[:tab:]</code>	<code>[:tab:]</code>		the <Tab> character
<code>[:escape:]</code>	<code>[:escape:]</code>		the <Esc> character
<code>[:backspace:]</code>	<code>[:backspace:]</code>		the <BS> character
<code>[:ident:]</code>	<code>[:ident:]</code>		identifier character (same as "\i")
<code>[:keyword:]</code>	<code>[:keyword:]</code>		keyword character (same as "\k")
<code>[:fname:]</code>	<code>[:fname:]</code>		file name character (same as "\f")

The square brackets in character class expressions are additional to the square brackets delimiting a collection. For example, the following is a plausible pattern for a UNIX filename:

`"[-./[:alnum:]_~]\+"`. That is, a list of at least one character, each of which is either '-', '.', '/', alphabetic, numeric, '_' or '~'.

These items only work for 8-bit characters, except `[:lower:]` and `[:upper:]` also work for multibyte characters when using the new regexp engine. See [two-engines](#). In the future these items may work for multibyte characters. For now, to get all "alpha" characters you can use: `[:lower:][:upper:]`.

The "Func" column shows what library function is used. The implementation depends on the system. Otherwise:

- (1) Uses `islower()` for ASCII and Vim builtin rules for other characters.
- (2) Uses Vim builtin rules
- (3) As with (1) but using `isupper()`

`/[[= [=]`

- An equivalence class. This means that characters are matched that

have almost the same meaning, e.g., when ignoring accents. This only works for Unicode, latin1 and latin9. The form is:

```
[=a=]
```

```
/[[. [..]
```

- A collation element. This currently simply accepts a single character in the form:

```
[.a.]
```

```
/\]
```

- To include a literal ']', '^', '-' or '\' in the collection, put a backslash before it: "[xyz\]", "[\^xyz]", "[xy\-z]" and "[xyz\\]". (Note: POSIX does not support the use of a backslash this way). For ']' you can also make it the first character (following a possible "^"): "[]xyz]" or "[^]xyz]".

For '-' you can also make it the first or last character: "[-xyz]", "[^~xyz]" or "[xyz~]". For '\' you can also let it be followed by any character that's not in "^~\bdertnoUux". "[\xyz]" matches '\', 'x', 'y' and 'z'. It's better to use "\\\" though, future expansions may use other characters after '\\'.

- Omitting the trailing] is not considered an error. "[]" works like "[]", it matches the ']' character.
- The following translations are accepted when the 'l' flag is not included in 'coptions':

```
\e      <Esc>
\t      <Tab>
\r      <CR>      (NOT end-of-line!)
\b      <BS>
\n      line break, see above /[\n]
\d123   decimal number of character
\o40    octal number of character up to 0o377
\x20    hexadecimal number of character up to 0xff
\u20AC  hex. number of multibyte character up to 0xffff
\U1234  hex. number of multibyte character up to 0xffffffff
```

NOTE: The other backslash codes mentioned above do not work inside []!

- Matching with a collection can be slow, because each character in the text has to be compared with each character in the collection. Use one of the other atoms above when possible. Example: "\d" is much faster than "[0-9]" and matches the same characters. However, the new **NFA** regexp engine deals with this better than the old one.

```
/\%[] E69 E70 E369
```

```
\%[]
```

A sequence of optionally matched atoms. This always matches. It matches as much of the list of atoms it contains as possible. Thus it stops at the first atom that doesn't match. For example:

```
/r\%[ead]
```

matches "r", "re", "rea" or "read". The longest that matches is used.

To match the Ex command "function", where "fu" is required and "nction" is optional, this would work:

```
/\<fu\%[nction]\>
```

The end-of-word atom "\>" is used to avoid matching "fu" in "full".

It gets more complicated when the atoms are not ordinary characters.

You don't often have to use it, but it is possible. Example:

```
/\<r\%[[eo]ad]\>
```

Matches the words "r", "re", "ro", "rea", "roa", "read" and "road".

There can be no `\(\)`, `\%(\\)` or `\z(\)` items inside the `[]` and `\%[]` does not nest.

To include a `"["` use `"[[]"` and for `"]"` use `[]]"`, e.g.,:

```
/index\%[[[]0[]]]
```

matches `"index"` `"index["`, `"index[0"` and `"index[0]"`.

{not available when compiled without the `|+syntax|` feature}

```
/\%d   /\%x   /\%o   /\%u   /\%U   E678
```

- `\%d123` Matches the character specified with a decimal number. Must be followed by a non-digit.
- `\%o40` Matches the character specified with an octal number up to `0o377`. Numbers below `0o40` must be followed by a non-octal digit or a non-digit.
- `\%x2a` Matches the character specified with up to two hexadecimal characters.
- `\%u20AC` Matches the character specified with up to four hexadecimal characters.
- `\%U1234abcd` Matches the character specified with up to eight hexadecimal characters, up to `0x7fffffff`

7. Ignoring case in a pattern

`/ignorecase`

If the `'ignorecase'` option is on, the case of normal letters is ignored. `'smartcase'` can be set to ignore case when the pattern contains lowercase letters only.

```
/\c   /\C
```

When `"\c"` appears anywhere in the pattern, the whole pattern is handled like `'ignorecase'` is on. The actual value of `'ignorecase'` and `'smartcase'` is ignored. `"\C"` does the opposite: Force matching case for the whole pattern. {only Vim supports `\c` and `\C`}

Note that `'ignorecase'`, `"\c"` and `"\C"` are not used for the character classes.

Examples:

pattern	'ignorecase'	'smartcase'	matches
foo	off	-	foo
foo	on	-	foo Foo F00
Foo	on	off	foo Foo F00
Foo	on	on	Foo
\cfoo	-	-	foo Foo F00
foo\C	-	-	foo

Technical detail:

`NL-used-for-Nu\l`

`<Nu\l>` characters in the file are stored as `<NL>` in memory. In the display they are shown as `"^@"`. The translation is done when reading and writing files. To match a `<Nu\l>` with a search pattern you can just enter `CTRL-@` or `"CTRL-V 000"`. This is probably just what you expect. Internally the character is replaced with a `<NL>` in the search pattern. What is unusual is that typing `CTRL-V CTRL-J` also inserts a `<NL>`, thus also searches for a `<Nu\l>` in the file.

`CR-used-for-NL`

When `'fileformat'` is `"mac"`, `<NL>` characters in the file are stored as `<CR>` characters internally. In the text they are shown as `"^J"`. Otherwise this

works similar to the usage of `<NL>` for a `<Nul>`.

When working with expression evaluation, a `<NL>` character in the pattern matches a `<NL>` in the string. The use of `"\n"` (backslash n) to match a `<NL>` doesn't work there, it only works to match text in the buffer.

pattern-multi-byte pattern-multibyte

Patterns will also work with multibyte characters, mostly as you would expect. But invalid bytes may cause trouble, a pattern with an invalid byte will probably never match.

8. Composing characters

patterns-composing

/\Z

When `"\Z"` appears anywhere in the pattern, all composing characters are ignored. Thus only the base characters need to match, the composing characters may be different and the number of composing characters may differ. Only relevant when `'encoding'` is `"utf-8"`.

Exception: If the pattern starts with one or more composing characters, these must match.

/\%C

Use `"\%C"` to skip any composing characters. For example, the pattern `"a"` does not match in `"càt"` (where the `a` has the composing character `0x0300`), but `"a\%C"` does. **Note** that this does not match `"cát"` (where the `á` is character `0xe1`, it does not have a composing character). It does match `"cat"` (where the `a` is just an `a`).

When a composing character appears at the start of the pattern or after an item that doesn't include the composing character, a match is found at any character that includes this composing character.

When using a dot and a composing character, this works the same as the composing character by itself, except that it doesn't matter what comes before this.

The order of composing characters does not matter. Also, the text may have more composing characters than the pattern, it still matches. But all composing characters in the pattern must be found in the text.

Suppose `B` is a base character and `x` and `y` are composing characters:

pattern	text	match
<code>Bxy</code>	<code>Bxy</code>	yes (perfect match)
<code>Bxy</code>	<code>Byx</code>	yes (order ignored)
<code>Bxy</code>	<code>By</code>	no (x missing)
<code>Bxy</code>	<code>Bx</code>	no (y missing)
<code>Bx</code>	<code>Bx</code>	yes (perfect match)
<code>Bx</code>	<code>By</code>	no (x missing)
<code>Bx</code>	<code>Bxy</code>	yes (extra y ignored)
<code>Bx</code>	<code>Byx</code>	yes (extra y ignored)

9. Compare with Perl patterns

perl-patterns

Vim's regexes are most similar to Perl's, in terms of what you can do. The difference between them is mostly just notation; here's a summary of where they differ:

Capability	in Vimspeak	in Perlspeak
force case insensitivity	\c	(?i)
force case sensitivity	\C	(?-i)
backref-less grouping	\%(atom\)	(?:atom)
conservative quantifiers	\{-n,m}	*?, +?, ??, {}?
0-width match	atom\@=	(?=atom)
0-width non-match	atom\@!	(?!atom)
0-width preceding match	atom\@<=	(?<=atom)
0-width preceding non-match	atom\@<!	(?<!atom)
match without retry	atom\@>	(?>atom)

Vim and Perl handle newline characters inside a string a bit differently:

In Perl, ^ and \$ only match at the very beginning and end of the text, by default, but you can set the 'm' flag, which lets them match at embedded newlines as well. You can also set the 's' flag, which causes a . to match newlines as well. (Both these flags can be changed inside a pattern using the same syntax used for the i flag above, BTW.)

On the other hand, Vim's ^ and \$ always match at embedded newlines, and you get two separate atoms, \%^ and \%\$, which only match at the very start and end of the text, respectively. Vim solves the second problem by giving you the _ "modifier": put it in front of a . or a character class, and they will match newlines as well.

Finally, these constructs are unique to Perl:

- execution of arbitrary code in the regex: (?{perl code})
- conditional expressions: (? (condition) true-expr | false-expr)

...and these are unique to Vim:

- changing the magic-ness of a pattern: \v \V \m \M (very useful for avoiding backslashitis)
- sequence of optionally matching atoms: \%[atoms]
- \& (which is to \| what "and" is to "or"; it forces several branches to match at one spot)
- matching lines/columns by number: \%5l \%5c \%5v
- setting the start and end of the match: \zs \ze

10. Highlighting matches

match-highlight

:mat :match

:mat[ch] {group} /{pattern}/

Define a pattern to highlight in the current window. It will be highlighted with {group}. Example:

```
:highlight MyGroup ctermbg=green guibg=green
:match MyGroup /TODO/
```

Instead of // any character can be used to mark the start and end of the {pattern}. Watch out for using special characters,

such as `'"` and `'|'`.

`{group}` must exist at the moment this command is executed.

The `{group}` highlighting still applies when a character is to be highlighted for `'hlsearch'`, as the highlighting for matches is given higher priority than that of `'hlsearch'`. Syntax highlighting (see `'syntax'`) is also overruled by matches.

Note that highlighting the last used search pattern with `'hlsearch'` is used in all windows, while the pattern defined with `":match"` only exists in the current window. It is kept when switching to another buffer.

`'ignorecase'` does not apply, use `/\c` in the pattern to ignore case. Otherwise case is not ignored.

`'redrawtime'` defines the maximum time searched for pattern matches.

When matching end-of-line and Vim redraws only part of the display you may get unexpected results. That is because Vim looks for a match in the line where redrawing starts.

Also see `matcharg()` and `getmatches()`. The former returns the highlight group and pattern of a previous `:match` command. The latter returns a list with highlight groups and patterns defined by both `matchadd()` and `:match`.

Highlighting matches using `:match` are limited to three matches (aside from `:match`, `:2match` and `:3match` are available). `matchadd()` does not have this limitation and in addition makes it possible to prioritize matches.

Another example, which highlights all characters in virtual column 72 and more:

```
:highlight rightMargin term=bold ctermfg=blue guifg=blue
:match rightMargin /\%>72v/
```

To highlight all character that are in virtual column 7:

```
:highlight col8 ctermbg=grey guibg=grey
:match col8 /\%<8v.\%>7v/
```

Note the use of two items to also match a character that occupies more than one virtual column, such as a TAB.

```
:mat[ch]
:mat[ch] none
```

Clear a previously defined match pattern.

```
:2mat[ch] {group} /{pattern}/           :2match
:2mat[ch]
:2mat[ch] none
:3mat[ch] {group} /{pattern}/           :3match
```

```
:3mat[ch]
:3mat[ch] none
```

Just like `:match` above, but set a separate match. Thus there can be three matches active at the same time. The match with the lowest number has priority if several match at the same position. It uses the match id 3.

The `":3match"` command is used by (Vim < 9.0.2054) `matchparen` plugin. You are suggested to use `":match"` for manual matching and `":2match"` for another plugin or even better make use of the more flexible `matchadd()` (and similar) functions instead.

11. Fuzzy matching

fuzzy-matching

Fuzzy matching refers to matching strings using a non-exact search string. Fuzzy matching will match a string, if all the characters in the search string are present anywhere in the string in the same order. Case is ignored. In a matched string, other characters can be present between two consecutive characters in the search string. If the search string has multiple words, then each word is matched separately. So the words in the search string can be present in any order in a string.

Fuzzy matching assigns a score for each matched string based on the following criteria:

- The number of sequentially matching characters.
- The number of characters (distance) between two consecutive matching characters.
- Matches at the beginning of a word
- Matches at a camel case character (e.g. Case in CamelCase)
- Matches after a path separator or a hyphen.
- The number of unmatched characters in a string.

The matching string with the highest score is returned first.

For example, when you search for the "get pat" string using fuzzy matching, it will match the strings "GetPattern", "PatternGet", "getPattern", "patGetter", "getSomePattern", "MatchpatternGet" etc.

The functions `matchfuzzy()` and `matchfuzzypos()` can be used to fuzzy search a string in a List of strings. The `matchfuzzy()` function returns a List of matching strings. The `matchfuzzypos()` functions returns the List of matches, the matching positions and the fuzzy match scores.

The "f" flag of `:vimgrep`` enables fuzzy matching.

VIM REFERENCE MANUAL by Bram Moolenaar

Key mapping, abbreviations and user-defined commands.

This subject is introduced in sections 05.4 , 24.7 and 40.1 of the user manual.

1. Key mapping	key-mapping	
1.1 MAP COMMANDS		:map-commands
1.2 Special arguments		:map-arguments
1.3 Mapping and modes		:map-modes
1.4 Listing mappings		map-listing
1.5 Mapping special keys		:map-special-keys
1.6 Special characters		:map-special-chars
1.7 What keys to map		map-which-keys
1.8 Examples		map-examples
1.9 Using mappings		map-typing
1.10 Mapping alt-keys		:map-alt-keys
1.11 Mapping meta-keys		:map-meta-keys
1.12 Mapping super-keys or command keys		:map-super-keys
1.13 Mapping in modifyOtherKeys mode		modifyOtherKeys
1.14 Mapping with Kitty keyboard protocol		kitty-keyboard-protocol
1.15 Mapping an operator		:map-operator
2. Abbreviations	abbreviations	
3. Local mappings and functions	script-local	
4. User-defined commands	user-commands	

=====

1. Key mapping key-mapping mapping macro

Key mapping is used to change the meaning of typed keys. The most common use is to define a sequence of commands for a function key. Example:

```
:map <F2> a<C-R>=strftime("%c")<CR><Esc>
```

This appends the current date and time after the cursor (in <> notation <>).

1.1 MAP COMMANDS :map-commands

There are commands to enter new mappings, remove mappings and list mappings. See [map-overview](#) for the various forms of "map" and their relationships with modes.

{lhs} means left-hand-side {lhs}
{rhs} means right-hand-side {rhs}

:map	{lhs} {rhs}	mapmode-nvo	:map
:nm[ap]	{lhs} {rhs}	mapmode-n	:nm :nmap
:vm[ap]	{lhs} {rhs}	mapmode-v	:vm :vmap

<code>:xm[ap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-x</code>	<code>:xm</code>	<code>:xmap</code>
<code>:smap</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-s</code>		<code>:smap</code>
<code>:om[ap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-o</code>	<code>:om</code>	<code>:omap</code>
<code>:map!</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-ic</code>	<code>:map!</code>	
<code>:im[ap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-i</code>	<code>:im</code>	<code>:imap</code>
<code>:lm[ap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-l</code>	<code>:lm</code>	<code>:lma</code> <code>:lmap</code>
<code>:cm[ap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-c</code>	<code>:cm</code>	<code>:cmap</code>
<code>:tma[p]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-t</code>	<code>:tma</code>	<code>:tmap</code>

Map the key sequence `{lhs}` to `{rhs}` for the modes where the map command applies. The result, including `{rhs}`, is then further scanned for mappings. This allows for nested and recursive use of mappings.

Note: Trailing spaces are included in the `{rhs}`, because space is a valid Normal mode command.

See [map-trailing-white](#) .

<code>:no[remap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-nvo</code>	<code>:nore</code>	<code>:noremap</code>	<code>:nor</code>
<code>:nn[oremap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-n</code>	<code>:nn</code>	<code>:nnoremap</code>	
<code>:vn[oremap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-v</code>	<code>:vn</code>	<code>:vnoremap</code>	
<code>:xn[oremap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-x</code>	<code>:xn</code>	<code>:xnoremap</code>	
<code>:snor[emap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-s</code>	<code>:snor</code>	<code>:snore</code>	<code>:snoremap</code>
<code>:ono[remap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-o</code>	<code>:ono</code>	<code>:onoremap</code>	
<code>:no[remap]!</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-ic</code>	<code>:no!</code>	<code>:noremap!</code>	
<code>:ino[remap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-i</code>	<code>:ino</code>	<code>:inor</code>	<code>:inoremap</code>
<code>:ln[oremap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-l</code>	<code>:ln</code>	<code>:lnoremap</code>	
<code>:cno[remap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-c</code>	<code>:cno</code>	<code>:cnor</code>	<code>:cnoremap</code>
<code>:tno[remap]</code>	<code>{lhs}</code>	<code>{rhs}</code>	<code>mapmode-t</code>	<code>:tno</code>	<code>:tnoremap</code>	

Map the key sequence `{lhs}` to `{rhs}` for the modes where the map command applies. Disallow mapping of `{rhs}`, to avoid nested and recursive mappings. Often used to redefine a command.

Note: When `<Plug>` appears in the `{rhs}` this part is always applied even if remapping is disallowed.

<code>:unm[ap]</code>	<code>{lhs}</code>		<code>mapmode-nvo</code>	<code>:unm</code>	<code>:unmap</code>
<code>:nun[map]</code>	<code>{lhs}</code>		<code>mapmode-n</code>	<code>:nun</code>	<code>:nunmap</code>
<code>:vu[nmap]</code>	<code>{lhs}</code>		<code>mapmode-v</code>	<code>:vu</code>	<code>:vunmap</code>
<code>:xu[nmap]</code>	<code>{lhs}</code>		<code>mapmode-x</code>	<code>:xu</code>	<code>:xunmap</code>
<code>:sunm[ap]</code>	<code>{lhs}</code>		<code>mapmode-s</code>	<code>:sunm</code>	<code>:sunmap</code>
<code>:ou[nmap]</code>	<code>{lhs}</code>		<code>mapmode-o</code>	<code>:ou</code>	<code>:ounmap</code>
<code>:unm[ap]!</code>	<code>{lhs}</code>		<code>mapmode-ic</code>	<code>:unm!</code>	<code>:unmap!</code>
<code>:iu[nmap]</code>	<code>{lhs}</code>		<code>mapmode-i</code>	<code>:iu</code>	<code>:iunmap</code>
<code>:lu[nmap]</code>	<code>{lhs}</code>		<code>mapmode-l</code>	<code>:lu</code>	<code>:lunmap</code>
<code>:cu[nmap]</code>	<code>{lhs}</code>		<code>mapmode-c</code>	<code>:cu</code>	<code>:cun</code> <code>:cunmap</code>
<code>:tunma[p]</code>	<code>{lhs}</code>		<code>mapmode-t</code>	<code>:tunma</code>	<code>:tunmap</code>

Remove the mapping of `{lhs}` for the modes where the map command applies. The mapping may remain defined for other modes where it applies.

It also works when `{lhs}` matches the `{rhs}` of a mapping. This is for when an abbreviation applied.

Note: Trailing spaces are included in the `{lhs}`.

See [map-trailing-white](#) .

<code>:mapc[lear]</code>	<code>mapmode-nvo</code>	<code>:mapc</code>	<code>:mapclear</code>
<code>:nmapc[lear]</code>	<code>mapmode-n</code>	<code>:nmapc</code>	<code>:nmapclear</code>
<code>:vmapc[lear]</code>	<code>mapmode-v</code>	<code>:vmapc</code>	<code>:vmapclear</code>
<code>:xmapc[lear]</code>	<code>mapmode-x</code>	<code>:xmapc</code>	<code>:xmapclear</code>
<code>:smapc[lear]</code>	<code>mapmode-s</code>	<code>:smapc</code>	<code>:smapclear</code>
<code>:omapc[lear]</code>	<code>mapmode-o</code>	<code>:omapc</code>	<code>:omapclear</code>
<code>:mapc[lear]!</code>	<code>mapmode-ic</code>	<code>:mapc!</code>	<code>:mapclear!</code>
<code>:imapc[lear]</code>	<code>mapmode-i</code>	<code>:imapc</code>	<code>:imapclear</code>
<code>:lmapc[lear]</code>	<code>mapmode-l</code>	<code>:lmapc</code>	<code>:lmapclear</code>
<code>:cmapc[lear]</code>	<code>mapmode-c</code>	<code>:cmapc</code>	<code>:cmapclear</code>
<code>:tmapc[lear]</code>	<code>mapmode-t</code>	<code>:tmapc</code>	<code>:tmapclear</code>

Remove ALL mappings for the modes where the map command applies.

Use the `<buffer>` argument to remove buffer-local mappings `:map-<buffer>`

Warning: This also removes the `mac-standard-mappings` and the `dos-standard-mappings` .

<code>:map</code>	<code>mapmode-nvo</code>
<code>:nm[ap]</code>	<code>mapmode-n</code>
<code>:vm[ap]</code>	<code>mapmode-v</code>
<code>:xm[ap]</code>	<code>mapmode-x</code>
<code>:sm[ap]</code>	<code>mapmode-s</code>
<code>:om[ap]</code>	<code>mapmode-o</code>
<code>:map!</code>	<code>mapmode-ic</code>
<code>:im[ap]</code>	<code>mapmode-i</code>
<code>:lm[ap]</code>	<code>mapmode-l</code>
<code>:cm[ap]</code>	<code>mapmode-c</code>
<code>:tma[p]</code>	<code>mapmode-t</code>

List all key mappings for the modes where the map command applies. Note that `:map` and `:map!` are used most often, because they include the other modes.

<code>:map {lhs}</code>	<code>mapmode-nvo</code>	<code>:map_l</code>
<code>:nm[ap] {lhs}</code>	<code>mapmode-n</code>	<code>:nmap_l</code>
<code>:vm[ap] {lhs}</code>	<code>mapmode-v</code>	<code>:vmap_l</code>
<code>:xm[ap] {lhs}</code>	<code>mapmode-x</code>	<code>:xmap_l</code>
<code>:sm[ap] {lhs}</code>	<code>mapmode-s</code>	<code>:smap_l</code>
<code>:om[ap] {lhs}</code>	<code>mapmode-o</code>	<code>:omap_l</code>
<code>:map! {lhs}</code>	<code>mapmode-ic</code>	<code>:map_l!</code>
<code>:im[ap] {lhs}</code>	<code>mapmode-i</code>	<code>:imap_l</code>
<code>:lm[ap] {lhs}</code>	<code>mapmode-l</code>	<code>:lmap_l</code>
<code>:cm[ap] {lhs}</code>	<code>mapmode-c</code>	<code>:cmap_l</code>
<code>:tma[p] {lhs}</code>	<code>mapmode-t</code>	<code>:tmap_l</code>

List the key mappings for the key sequences starting with `{lhs}` in the modes where the map command applies.

These commands are used to map a key or key sequence to a string of characters. You can use this to put command sequences under function keys, translate one key into another, etc. See `:mkexrc` for how to save and restore the current mappings.

`map-ambiguous`

When two mappings start with the same sequence of characters, they are ambiguous. Example:

```
:imap aa foo
:imap aaa bar
```

When Vim has read "aa", it will need to get another character to be able to decide if "aa" or "aaa" should be mapped. This means that after typing "aa" that mapping won't get expanded yet, Vim is waiting for another character. If you type a space, then "foo" will get inserted, plus the space. If you type "a", then "bar" will get inserted.

Trailing white space

[map-trailing-white](#)

This unmap command does NOT work:

```
:map @@ foo
:unmap @@ | print
```

Because it tries to unmap "@@ ", including the white space before the command separator "|". Other examples with trailing white space:

```
unmap @@
unmap @@      # Vim9 script comment
unmap @@      " legacy script comment
```

An error will be issued, which is very hard to identify, because the ending whitespace character in `unmap @@ ` is not visible.

A generic solution is to put the command separator "|" right after the mapped keys. After that white space and a comment may follow:

```
unmap @@|      # Vim9 script comment
unmap @@|      " legacy script comment
```

1.2 SPECIAL ARGUMENTS

[:map-arguments](#)

"<buffer>", "<nowait>", "<silent>", "<special>", "<script>", "<expr>" and "<unique>" can be used in any order. They must appear right after the command, before any other arguments.

```
:map-local    :map-<buffer>    :map-buffer
E224          E225
```

If the first argument to one of these commands is "<buffer>" the mapping will be effective in the current buffer only. Example:

```
:map <buffer> ,w /[,;]<CR>
```

Then you can map ",w" to something else in another buffer:

```
:map <buffer> ,w /[#&!]<CR>
```

The local buffer mappings are used before the global ones. See <nowait> below to make a short local mapping not taking effect when a longer global one exists.

The "<buffer>" argument can also be used to clear mappings:

```
:unmap <buffer> ,w
:mapclear <buffer>
```

Local mappings are also cleared when a buffer is deleted, but not when it is unloaded. Just like local option values.

Also see [map-precedence](#) .

`:map-<nowait>` `:map-nowait`

When defining a buffer-local mapping for ",", there may be a global mapping that starts with ",". Then you need to type another character for Vim to know whether to use the "," mapping or the longer one. To avoid this add the `<nowait>` argument. Then the mapping will be used when it matches, Vim does not wait for more characters to be typed. However, if the characters were already typed they are used.

Note that this works when the `<nowait>` mapping fully matches and is found before any partial matches. This works when:

- There is only one matching buffer-local mapping, since these are always found before global mappings.
- There is another buffer-local mapping that partly matches, but it is defined earlier (last defined mapping is found first).

`:map-<silent>` `:map-silent`

To define a mapping which will not be echoed on the command line, add "`<silent>`" as the first argument. Example:

```
:map <silent> ,h /Header<CR>
```

The search string will not be echoed when using this mapping. Messages from the executed command are still given though. To shut them up too, add a `":silent"` in the executed command:

```
:map <silent> ,h :exe ":silent normal /Header\r"<CR>
```

Note that the effect of a command might also be silenced, e.g., when the mapping selects another entry for command line completion it won't be displayed.

Prompts will still be given, e.g., for `inputdialog()`.

Using "`<silent>`" for an abbreviation is possible, but will cause redrawing of the command line to fail.

`:map-<special>` `:map-special`

Define a mapping with `<>` notation for special keys, even though the "`<`" flag may appear in '`coptions`'. This is useful if the side effect of setting '`coptions`' is not desired. Example:

```
:map <special> <F12> /Header<CR>
```

`:map-<script>` `:map-script`

If the first argument to one of these commands is "`<script>`" and it is used to define a new mapping or abbreviation, the mapping will only remap characters in the `{rhs}` using mappings that were defined local to a script, starting with "`<SID>`". This can be used to avoid that mappings from outside a script interfere (e.g., when **CTRL-V** is remapped in `mswin.vim`), but do use other mappings defined in the script.

Note: `":map <script>`" and `":noremap <script>`" do the same thing. The "`<script>`" overrules the command name. Using `":noremap <script>`" is preferred, because it's clearer that remapping is (mostly) disabled.

`:map-<unique>` `:map-unique` E226 E227

If the first argument to one of these commands is "`<unique>`" and it is used to define a new mapping or abbreviation, the command will fail if the mapping or abbreviation already exists. Example:

```
:map <unique> ,w /[#&!]<CR>
```

When defining a local mapping, there will also be a check if a global map already exists which is equal.

Example of what will fail:

```
:map ,w /[#&!]<CR>
:map <buffer> <unique> ,w /[,.;]<CR>
```

If you want to map a key and then have it do what it was originally mapped to, have a look at `maparg()` .

`:map-<expr>` `:map-expression`

If the first argument to one of these commands is "`<expr>`" and it is used to define a new mapping or abbreviation, the argument is an expression. The expression is evaluated to obtain the `{rhs}` that is used. Example:

```
:inoremap <expr> . <SID>InsertDot()
```

The result of the `s:InsertDot()` function will be inserted. It could check the text before the cursor and start omni completion when some condition is met. Using a script-local function is preferred, to avoid polluting the global namespace. Use `<SID>` in the RHS so that the script that the mapping was defined in can be found.

For abbreviations `v:char` is set to the character that was typed to trigger the abbreviation. You can use this to decide how to expand the `{lhs}`. You should not either insert or change the `v:char`.

In case you want the mapping to not do anything, you can have the expression evaluate to an empty string. If something changed that requires Vim to go through the main loop (e.g. to update the display), return "`\<Ignore>`". This is similar to "nothing" but makes Vim return from the loop that waits for input. Example:

```
func s:OpenPopup()
  call popup_create(... arguments ...)
  return "\<Ignore>"
endfunc
nnoremap <expr> <F3> <SID>OpenPopup()
```

Keep in mind that the expression may be evaluated when looking for typeahead, before the previous command has been executed. For example:

```
func StoreColumn()
  let g:column = col('.')
  return 'x'
endfunc
nnoremap <expr> x StoreColumn()
nmap ! f!x
```

You will notice that `g:column` has the value from before executing "`f!`", because "`x`" is evaluated before "`f!`" is executed.

This can be solved by inserting `<Ignore>` before the character that is expression-mapped:

```
nmap ! f!<Ignore>x
```

When defining a mapping in a `Vim9` script, the expression will be evaluated in the context of that script. This means that script-local items can be accessed in the expression.

Be very careful about side effects! The expression is evaluated while obtaining characters, you may very well make the command dysfunctional. For this reason the following is blocked:

- Changing the buffer text `textlock` .

- Editing another buffer.
- The `:normal` command.
- Moving the cursor is allowed, but it is restored afterwards.

If you want the mapping to do any of these let the returned characters do that, or use a `<Cmd>` mapping instead.

You can use `getchar()`, it consumes typeahead if there is any. E.g., if you have these mappings:

```
inoremap <expr> <C-L> nr2char(getchar())
inoremap <expr> <C-L>x "foo"
```

If you now type **CTRL-L** nothing happens yet, Vim needs the next character to decide what mapping to use. If you type 'x' the second mapping is used and "foo" is inserted. If you type any other key the first mapping is used, `getchar()` gets the typed key and returns it.

Here is an example that inserts a list number that increases:

```
let counter = 0
inoremap <expr> <C-L> ListItem()
inoremap <expr> <C-R> ListReset()

func ListItem()
  let g:counter += 1
  return g:counter .. '. '
endfunc

func ListReset()
  let g:counter = 0
  return ''
endfunc
```

CTRL-L inserts the next number, **CTRL-R** resets the count. **CTRL-R** returns an empty string, so that nothing is inserted.

Note that using `0x80` as a single byte before other text does not work, it will be seen as a special key.

`<Cmd>` `:map-cmd`

The special text `<Cmd>` begins a "command mapping", it executes the command directly without changing modes. Where you might use `":...<CR>"` in the `{rhs}` of a mapping, you can instead use `"<Cmd>...<CR>"`.

Example:

```
noremap x <Cmd>echo mode(1)<CR>
```

This is more flexible than ``:<C-U>`` in Visual and Operator-pending mode, or ``<C-O>:`` in Insert mode, because the commands are executed directly in the current mode, instead of always going to Normal mode. Visual mode is preserved, so tricks with `gv` are not needed. Commands can be invoked directly in Command-line mode (which would otherwise require timer hacks).

Example of using `<Cmd>` halfway Insert mode:

```
nnoremap <F3> aText <Cmd>echo mode(1)<CR> Added<Esc>
```

Unlike `<expr>` mappings, there are no special restrictions on the `<Cmd>` command: it is executed as if an (unrestricted) `autocommand` was invoked.

[<ScriptCmd>](#)

`<ScriptCmd>` is like `<Cmd>` but sets the context to the script the mapping was defined in, for the duration of the command execution. This is especially useful for [Vim9](#) script. It also works to access an import, which is useful in a plugin using a, possibly autoloaded, script:

```
vim9script
import autoload 'implementation.vim' as impl
nnoremap <F4> <ScriptCmd>impl.DoTheWork()<CR>
```

No matter where `<F4>` is typed, the "impl" import will be found in the script context of where the mapping was defined. When it's an autoload import, as in the example, the "implementation.vim" script will only be loaded once `<F4>` is typed, not when the mapping is defined.

Without `<ScriptCmd>` using "s:impl" would result in "E121: Undefined variable".

Note:

- Because `<Cmd>` and `<ScriptCmd>` avoid mode-changes it does not trigger `CmdlineEnter` and `CmdlineLeave` events, because no user interaction is expected.
- For the same reason, `keycodes` like `<C-R><C-W>` are interpreted as plain, unmapped keys.
- The command is not echo'ed, no need for `<silent>`.
- The `{rhs}` is not subject to abbreviations nor to other mappings, even if the mapping is recursive.
- In Visual mode you can use ``line('v')`` and ``col('v')`` to get one end of the Visual area, the cursor is at the other end.

[E1255](#) [E1136](#)

`<Cmd>` and `<ScriptCmd>` commands must terminate, that is, they must be followed by `<CR>` in the `{rhs}` of the mapping definition. `Command-line` mode is never entered. To use a literal `<CR>` in the `{rhs}`, use `<lt>` .

1.3 MAPPING AND MODES

[:map-modes](#)

[mapmode-nvo](#) [mapmode-n](#) [mapmode-v](#) [mapmode-o](#)

There are seven sets of mappings

- For Normal mode: When typing commands.
- For Visual mode: When typing commands while the Visual area is highlighted.
- For Select mode: like Visual mode but typing text replaces the selection.
- For Operator-pending mode: When an operator is pending (after "d", "y", "c", etc.). See below: [omap-info](#) .
- For Insert mode. These are also used in Replace mode.
- For Command-line mode: When entering a ":" or "/" command.
- For Terminal mode: When typing in a `:terminal` buffer.

Special case: While typing a count for a command in Normal mode, mapping zero is disabled. This makes it possible to map zero without making it impossible to type a count with a zero.

[map-overview](#) [map-modes](#)

Overview of which map command works in which mode. More details below.

[COMMANDS](#)

[MODES](#)

```

:map  :noremap :unmap  Normal, Visual, Select, Operator-pending
:nmap :nnoremap :nunmap Normal
:vmap :vnoremap :vunmap Visual and Select
:smap :snoremap :sunmap Select
:xmap :xnoremap :xunmap Visual
:omap :onoremap :ounmap Operator-pending
:map! :noremap! :unmap! Insert and Command-line
:imap :inoremap :iunmap Insert
:lmap :lnoremap :lunmap Insert, Command-line, Lang-Arg
:cmap :cnoremap :cunmap Command-line
:tmap :tnoremap :tunmap Terminal-Job

```

Same information in a table:

Command	Mode	map-table							
		Norm	Ins	Cmd	Vis	Sel	Opr	Term	Lang
[nore]map		yes	-	-	yes	yes	yes	-	-
n[nore]map		yes	-	-	-	-	-	-	-
[nore]map!		-	yes	yes	-	-	-	-	-
i[nore]map		-	yes	-	-	-	-	-	-
c[nore]map		-	-	yes	-	-	-	-	-
v[nore]map		-	-	-	yes	yes	-	-	-
x[nore]map		-	-	-	yes	-	-	-	-
s[nore]map		-	-	-	-	yes	-	-	-
o[nore]map		-	-	-	-	-	yes	-	-
t[nore]map		-	-	-	-	-	-	yes	-
l[nore]map		-	yes	yes	-	-	-	-	yes

COMMANDS

```

:map  :noremap :unmap  :mapclear
:nmap :nnoremap :nunmap :nmapclear
:vmap :vnoremap :vunmap :vmapclear
:omap :onoremap :ounmap :omapclear

```

MODES

	Normal	Visual+Select	Operator-pending
yes	yes	yes	yes
yes	yes	-	-
-	-	yes	-
-	-	-	yes

:nunmap can also be used outside of a monastery.

mapmode-x mapmode-s

Some commands work both in Visual and Select mode, some in only one. **Note** that quite often "Visual" is mentioned where both Visual and Select mode apply. [Select-mode-mapping](#)

NOTE: Mapping a printable character in Select mode may confuse the user. It's better to explicitly use :xmap and :smap for printable characters. Or use :sunmap after defining the mapping.

COMMANDS

```

:vmap :vnoremap :vunmap :vmapclear
:xmap :xnoremap :xunmap :xmapclear
:smap :snoremap :sunmap :smapclear

```

MODES

	Visual	Select
yes	yes	yes
yes	yes	-
-	-	yes

mapmode-ic mapmode-i mapmode-c mapmode-l

Some commands work both in Insert mode and Command-line mode, some not:

COMMANDS

<code>:map!</code>	<code>:noremap!</code>	<code>:unmap!</code>	<code>:mapclear!</code>
<code>:imap</code>	<code>:inoremap</code>	<code>:iunmap</code>	<code>:imapclear</code>
<code>:cmap</code>	<code>:cnoremap</code>	<code>:cunmap</code>	<code>:cmapclear</code>
<code>:lmap</code>	<code>:lnoremap</code>	<code>:lunmap</code>	<code>:lmapclear</code>

MODES

Insert	Command-line	Lang-Arg
yes	yes	-
yes	-	-
-	yes	-
yes*	yes*	yes*

* If `'iminsert'` is 1, see [language-mapping](#) below.

The original Vi did not have separate mappings for Normal/Visual/Operator-pending mode and for Insert/Command-line mode. Therefore the `":map"` and `":map!"` commands enter and display mappings for several modes. In Vim you can use the `":nmap"`, `":vmap"`, `":omap"`, `":cmap"` and `":imap"` commands to enter mappings for each mode separately.

mapmode-t

The terminal mappings are used in a terminal window, when typing keys for the job running in the terminal. See [terminal-typing](#).

omap-info

Operator-pending mappings can be used to define a movement command that can be used with any operator. Simple example:

```
:omap { w
```

makes `"y{"` work like `"yw"` and `"d{"` like `"dw"`.

To ignore the starting cursor position and select different text, you can have the `omap` start Visual mode to select the text to be operated upon. Example that operates on a function name in the current line:

```
onoremap <silent> F :<C-U>normal! @f(hv<CR>
```

The **CTRL-U** (`<C-U>`) is used to remove the range that Vim may insert. The Normal mode commands find the first `'(` character and select the first word before it. That usually is the function name.

To enter a mapping for Normal and Visual mode, but not Operator-pending mode, first define it for all three modes, then unmap it for Operator-pending mode:

```
:map xx something-difficult
:ounmap xx
```

Likewise for a mapping for Visual and Operator-pending mode or Normal and Operator-pending mode.

language-mapping

`":lmap"` defines a mapping that applies to:

- Insert mode
- Command-line mode
- when entering a search pattern
- the argument of the commands that accept a text character, such as `"r"` and `"f"`
- for the `input()` line

Generally: Whenever a character is to be typed that is part of the text in the buffer, not a Vim command character. "Lang-Arg" isn't really another mode, it's just used here for this situation.

The simplest way to load a set of related language mappings is by using the

'keymap' option. See 45.5 .

In Insert mode and in Command-line mode the mappings can be disabled with the **CTRL-^** command `i_CTRL-^ c_CTRL-^` . These commands change the value of the 'iminsert' option. When starting to enter a normal command line (not a search pattern) the mappings are disabled until a **CTRL-^** is typed. The state last used is remembered for Insert mode and Search patterns separately. The state for Insert mode is also used when typing a character as an argument to command like "f" or "t".

Language mappings will never be applied to already mapped characters. They are only used for typed characters. This assumes that the language mapping was already done when typing the mapping.

1.4 LISTING MAPPINGS

map-listing

When listing mappings the characters in the first two columns are:

CHAR	MODE
<Space>	Normal, Visual, Select and Operator-pending
n	Normal
v	Visual and Select
s	Select
x	Visual
o	Operator-pending
!	Insert and Command-line
i	Insert
l	":lmap" mappings for Insert, Command-line and Lang-Arg
c	Command-line
t	Terminal-Job

Just before the {rhs} a special character can appear:

*	indicates that it is not remappable
&	indicates that only script-local mappings are remappable
@	indicates a buffer-local mapping

Everything from the first non-blank after {lhs} up to the end of the line (or '|') is considered to be part of {rhs}. This allows the {rhs} to end with a space.

Note: When using mappings for Visual mode, you can use the "'<" mark, which is the start of the last selected Visual area in the current buffer '< .

The :filter command can be used to select what mappings to list. The pattern is matched against the {lhs} and {rhs} in the raw form.

While mappings are being listed, it is not possible to add or clear mappings, e.g. from a timer callback. E1309

:map-verbose

When 'verbose' is non-zero, the detected and used 'keyprotocol' value will be displayed in the first line. Also a key map will also display where it was last defined. Example:

```
:verbose map <C-W>*
```

```
Kitty keyboard protocol: Cleared
n <C-W>*      * <C-W><C-S>*
      Last set from /home/abcd/.vimrc
```

See `:verbose-cmd` for more information.

1.5 MAPPING SPECIAL KEYS

`:map-special-keys`

There are three ways to map a special key:

1. The Vi-compatible method: Map the key code. Often this is a sequence that starts with `<Esc>`. To enter a mapping like this you type `:map` and then you have to type **CTRL-V** before hitting the function key. **Note** that when the key code for the key is in the termcap (the `t_` options), it will automatically be translated into the internal code and become the second way of mapping (unless the `'k'` flag is included in `'coptions'`).
2. The second method is to use the internal code for the function key. To enter such a mapping type **CTRL-K** and then hit the function key, or use the form `"#1"`, `"#2"`, .. `"#9"`, `"#0"`, `"<Up>"`, `"<S-Down>"`, `"<S-F7>"`, etc. (see table of keys `key-notation`, all keys from `<Up>` can be used). The first ten function keys can be defined in two ways: Just the number, like `"#2"`, and with `"<F>"`, like `"<F2>"`. Both stand for function key 2. `"#0"` refers to function key 10, defined with option `'t_f10'`, which may be function key zero on some keyboards. The `<>` form cannot be used when `'coptions'` includes the `'<'` flag.
3. Use the termcap entry, with the form `<t_xx>`, where `"xx"` is the name of the termcap entry. Any string entry can be used. For example:

```
:map <t_F3> G
```

Maps function key 13 to `"G"`. This does not work if `'coptions'` includes the `'<'` flag.

The advantage of the second and third method is that the mapping will work on different terminals without modification (the function key will be translated into the same internal code or the actual key code, no matter what terminal you are using. The termcap must be correct for this to work, and you must use the same mappings).

DETAIL: Vim first checks if a sequence from the keyboard is mapped. If it isn't the terminal key codes are tried (see `terminal-options`). If a terminal code is found it is replaced with the internal code. Then the check for a mapping is done again (so you can map an internal code to something else). What is written into the script file depends on what is recognized. If the terminal key code was recognized as a mapping the key code itself is written to the script file. If it was recognized as a terminal code the internal code is written to the script file.

1.6 SPECIAL CHARACTERS

`:map-special-chars`

`map_backslash` `map-backslash`

Note that only **CTRL-V** is mentioned here as a special character for mappings and abbreviations. When `'coptions'` does not contain `'B'`, a backslash can also be used like **CTRL-V**. The `<>` notation can be fully used then `<>`. But you cannot use `"<C-V>"` like **CTRL-V** to escape the special meaning of what follows.

To map a backslash, or use a backslash literally in the {rhs}, the special sequence "<Bslash>" can be used. This avoids the need to double backslashes when using nested mappings.

map_CTRL-C map-CTRL-C

Using **CTRL-C** in the {lhs} is possible, but it will only work when Vim is waiting for a key, not when Vim is busy with something. When Vim is busy **CTRL-C** interrupts/breaks the command.

When using the GUI version on MS-Windows **CTRL-C** can be mapped to allow a Copy command to the clipboard. Use **CTRL-Break** to interrupt Vim.

map_space_in_lhs map-space_in_lhs

To include a space in {lhs} precede it with a **CTRL-V** (type two **CTRL-V**s for each space).

map_space_in_rhs map-space_in_rhs

If you want a {rhs} that starts with a space, use "<Space>". To be fully Vi compatible (but unreadable) don't use the <> notation, precede {rhs} with a single **CTRL-V** (you have to type **CTRL-V** two times).

map_empty_rhs map-empty-rhs

You can create an empty {rhs} by typing nothing after a single **CTRL-V** (you have to type **CTRL-V** two times). Unfortunately, you cannot do this in a vimrc file.

<Nop>

An easier way to get a mapping that doesn't produce anything, is to use "<Nop>" for the {rhs}. This only works when the <> notation is enabled.

For example, to make sure that function key 8 does nothing at all:

```
:map <F8> <Nop>
:map! <F8> <Nop>
```

map-multibyte

It is possible to map multibyte characters, but only the whole character. You cannot map the first byte only. This was done to prevent problems in this scenario:

```
:set encoding=latin1
:imap <M-C> foo
:set encoding=utf-8
```

The mapping for <M-C> is defined with the latin1 encoding, resulting in a 0xc3 byte. If you type the character á (0xe1 <M-a>) in UTF-8 encoding this is the two bytes 0xc3 0xa1. You don't want the 0xc3 byte to be mapped then or otherwise it would be impossible to type the á character.

<Leader> mapleader

To define a mapping which uses the "g:mapleader" variable, the special string "<Leader>" can be used. It is replaced with the string value of "g:mapleader". If "g:mapleader" is not set or empty, a backslash is used instead. Example:

```
map <Leader>A oanother line<Esc>
```

Works like:

```
map \A oanother line<Esc>
```

But after (legacy script):

```
let mapleader = ","
```

Or (Vim9 script):

```
g:mapleader = ","
```

It works like:

```
map ,A oanother line<Esc>
```

Note that the value of "g:mapleader" is used at the moment the mapping is defined. Changing "g:mapleader" after that has no effect for already defined mappings.

<LocalLeader> maplocalleader

<LocalLeader> is just like **<Leader>**, except that it uses "maplocalleader" instead of "mapleader". **<LocalLeader>** is to be used for mappings which are local to a buffer. Example:

```
:map <buffer> <LocalLeader>A oanother line<Esc>
```

In a global plugin **<Leader>** should be used and in a filetype plugin **<LocalLeader>**. "mapleader" and "maplocalleader" can be equal. Although, if you make them different, there is a smaller chance of mappings from global plugins to clash with mappings for filetype plugins. For example, you could keep "mapleader" at the default backslash, and set "maplocalleader" to an underscore.

map-<SID>

In a script the special key name "**<SID>**" can be used to define a mapping that's local to the script. See **<SID>** for details.

<Plug>

The special key name "**<Plug>**" can be used for an internal mapping, which is not to be matched with any key sequence. This is useful in plugins **using-<Plug>** .

<MouseMove>

The special key name "**<MouseMove>**" can be used to handle mouse movement. It needs to be enabled with '**mousemoveevent**'. Currently only works in the GUI. The **getmousepos()** function can be used to obtain the mouse position.

<Char> **<Char->**

To map a character by its decimal, octal or hexadecimal number the **<Char>** construct can be used:

```
<Char-123>    character 123
<Char-033>    character 27
<Char-0x7f>   character 127
<S-Char-114> character 114 ('r') shifted ('R')
```

This is useful to specify a (multibyte) character in a '**keymap**' file. Upper and lowercase differences are ignored.

map-comments

It is not possible to put a comment after these commands, because the '"' character is considered to be part of the **{lhs}** or **{rhs}**. However, one can use '|', since this starts a new, empty command with a comment.

map_bar **map-bar**

Since the '|' character is used to separate a map command from the next command, you will have to do something special to include a '|' in **{rhs}**. There are three methods:

```
use          works when          example
```

```

<Bar>      '<' is not in 'coptions'      :map _l :!ls <Bar> more^M
\|         'b' is not in 'coptions'     :map _l :!ls \| more^M
^V|        always, in Vim and Vi        :map _l :!ls ^V| more^M

```

(here ^V stands for **CTRL-V**; to get one **CTRL-V** you have to type it twice; you cannot use the <> notation "<C-V>" here).

All three work when you use the default setting for 'coptions'.

When 'b' is present in 'coptions', "\|" will be recognized as a mapping ending in a '\' and then another command. This is Vi compatible, but illogical when compared to other commands.

map_return map-return

When you have a mapping that contains an Ex command, you need to put a line terminator after it to have it executed. The use of <CR> is recommended for this (see <>). Example:

```
:map _ls :!ls -l %:S<CR>:echo "the end"<CR>
```

To avoid mapping of the characters you type in insert or Command-line mode, type a **CTRL-V** first. The mapping in Insert mode is disabled if the 'paste' option is on.

map-error

Note that when an error is encountered (that causes an error message or might cause a beep) the rest of the mapping is not executed. This is Vi-compatible.

Note that the second character (argument) of the commands @zZtTfF[]rm``v and **CTRL-X** is not mapped. This was done to be able to use all the named registers and marks, even when the command with the same name has been mapped.

1.7 WHAT KEYS TO MAP

map-which-keys

If you are going to map something, you will need to choose which key(s) to use for the {lhs}. You will have to avoid keys that are used for Vim commands, otherwise you would not be able to use those commands anymore. Here are a few suggestions:

- Function keys <F2>, <F3>, etc.. Also the shifted function keys <S-F1>, <S-F2>, etc. **Note** that <F1> is already used for the help command.
- Any key with the Alt or Meta key pressed. Depending on your keyboard accented characters may be used as well. [:map-alt-keys](#)
- Use the '_' or ',' character and then any other character. The "_" and "," commands do exist in Vim (see _ and ,), but you probably never use them.
- Use a key that is a synonym for another command. For example: **CTRL-P** and **CTRL-N**. Use an extra character to allow more mappings.
- The key defined by <Leader> and one or more other keys. This is especially useful in scripts. [mapleader](#)

See the file "index" for keys that are not used and thus can be mapped without losing any builtin function. You can also use ":help {key}^D" to find out if a key is used for some command. ({key} is the specific key you want to find out about, ^D is **CTRL-D**).

1.8 EXAMPLES

map-examples

A few examples (given as you type them, for "<CR>" you type four characters; the '<' flag must not be present in '**coptions**' for this to work).

```
:map <F3> o#include
:map <M-g> /foo<CR>cwbar<Esc>
:map _x d/END/e<CR>
:map! qq quadrillion questions
```

Multiplying a count

When you type a count before triggering a mapping, it's like the count was typed before the {lhs}. For example, with this mapping:

```
:map <F4> 3w
```

Typing 2<F4> will result in "23w". Thus not moving 2 * 3 words but 23 words. If you want to multiply counts use the expression register:

```
:map <F4> @='3w'<CR>
```

The part between quotes is the expression being executed. @=

1.9 USING MAPPINGS

map-typing

Vim will compare what you type with the start of a mapped sequence. If there is an incomplete match, it will get more characters until there either is a complete match or until there is no match at all. Example: If you map! "qq", the first 'q' will not appear on the screen until you type another character. This is because Vim cannot know if the next character will be a 'q' or not. If the '**timeout**' option is on (which is the default) Vim will only wait for one second (or as long as specified with the '**timeoutlen**' option). After that it assumes that the 'q' is to be interpreted as such. If you type slowly, or your system is slow, reset the '**timeout**' option. Then you might want to set the '**ttimeout**' option.

map-precedence

Buffer-local mappings (defined using `:map-<buffer>`) take precedence over global mappings. When a buffer-local mapping is the same as a global mapping, Vim will use the buffer-local mapping. In addition, Vim will use a complete mapping immediately if it was defined with `<nowait>`, even if a longer mapping has the same prefix. For example, given the following two mappings:

```
:map <buffer> <nowait> \a :echo "Local \a"<CR>
:map \abc :echo "Global \abc"<CR>
```

When typing \a the buffer-local mapping will be used immediately. Vim will not wait for more characters to see if the user might be typing \abc.

map-keys-fails

There are situations where key codes might not be recognized:

- Vim can only read part of the key code. Mostly this is only the first character. This happens on some Unix versions in an xterm.
- The key code is after character(s) that are mapped. E.g., "<F1><F1>" or "g<F1>".

The result is that the key code is not recognized in this situation, and the mapping fails. There are two actions needed to avoid this problem:

- Remove the 'K' flag from **'cptions'**. This will make Vim wait for the rest of the characters of the function key.
- When using <F1> to <F4> the actual key code generated may correspond to <xF1> to <xF4>. There are mappings from <xF1> to <F1>, <xF2> to <F2>, etc., but these are not recognized after another half a mapping. Make sure the key codes for <F1> to <F4> are correct:

```
:set <F1>=<type CTRL-V><type F1>
```

Type the <F1> as four characters. The part after the "=" must be done with the actual keys, not the literal text.

Another solution is to use the actual key code in the mapping for the second special key:

```
:map <F1><Esc>OP :echo "yes"<CR>
```

Don't type a real <Esc>, Vim will recognize the key code and replace it with <F1> anyway.

Another problem may be that when keeping ALT or Meta pressed the terminal prepends ESC instead of setting the 8th bit. See [:map-alt-keys](#) .

recursive_mapping

If you include the {lhs} in the {rhs} you have a recursive mapping. When {lhs} is typed, it will be replaced with {rhs}. When the {lhs} which is included in {rhs} is encountered it will be replaced with {rhs}, and so on. This makes it possible to repeat a command an infinite number of times. The only problem is that the only way to stop this is by causing an error. The macros to solve a maze uses this, look there for an example. There is one exception: If the {rhs} starts with {lhs}, the first character is not mapped again (this is Vi compatible).

For example:

```
:map ab abcd
```

will execute the "a" command and insert "bcd" in the text. The "ab" in the {rhs} will not be mapped again.

If you want to exchange the meaning of two keys you should use the :noremap command. For example:

```
:noremap k j
```

```
:noremap j k
```

This will exchange the cursor up and down commands.

With the normal :map command, when the **'remap'** option is on, mapping takes place until the text is found not to be a part of a {lhs}. For example, if you use:

```
:map x y
```

```
:map y x
```

Vim will replace x with y, and then y with x, etc. When this has happened **'maxmapdepth'** times (default 1000), Vim will give the error message "recursive mapping".

:map-undo

If you include an undo command inside a mapped sequence, this will bring the text back in the state before executing the macro. This is compatible with the original Vi, as long as there is only one undo command in the mapped

sequence (having two undo commands in a mapped sequence did not make sense in the original Vi, you would get back the text before the first undo).

1.10 MAPPING ALT-KEYS

[:map-alt-keys](#)

For a readable mapping command the `<A-k>` form can be used. **Note** that `<A-k>` and `<A-K>` are different, the latter will use an upper case letter. Actually, `<A-K>` and `<A-S-K>` are the same. Instead of "A" you can use "M". If you have an actual Meta modifier key, please see [:map-meta-keys](#) .

In the GUI Vim handles the Alt key itself, thus mapping keys with ALT should always work. But in a terminal Vim gets a sequence of bytes and has to figure out whether ALT was pressed or not.

If the terminal supports the `modifyOtherKeys` mode and it has been enabled, then Vim can recognize more key combinations, see [modifyOtherKeys](#) below. The Kitty keyboard protocol works in a similar way, see [kitty-keyboard-protocol](#) .

By default Vim assumes that pressing the ALT key sets the 8th bit of a typed character. Most decent terminals can work that way, such as xterm, aterm and rxvt. If your `<A-k>` mappings don't work it might be that the terminal is prefixing the character with an ESC character. But you can just as well type ESC before a character, thus Vim doesn't know what happened (except for checking the delay between characters, which is not reliable).

As of this writing, some mainstream terminals like `gnome-terminal` and `konsole` use the ESC prefix. There doesn't appear a way to have them use the 8th bit instead. Xterm should work well by default. Aterm and rxvt should work well when started with the `--meta8` argument. You can also tweak resources like `metaSendsEscape`, `eightBitInput` and `eightBitOutput`.

On the Linux console, this behavior can be toggled with the `setmetamode` command. Bear in mind that not using an ESC prefix could get you in trouble with other programs. You should make sure that bash has the `convert-meta` option set to "on" in order for your Meta keybindings to still work on it (it's the default readline behavior, unless changed by specific system configuration). For that, you can add the line:

```
set convert-meta on
```

to your `~/.inputrc` file. If you're creating the file, you might want to use:

```
$include /etc/inputrc
```

as the first line, if that file exists on your system, to keep global options. This may cause a problem for entering special characters, such as the umlaut. Then you should use **CTRL-V** before that character.

Bear in mind that `convert-meta` has been reported to have troubles when used in UTF-8 locales. On terminals like xterm, the `metaSendsEscape` resource can be toggled on the fly through the "Main Options" menu, by pressing `Ctrl-LeftClick` on the terminal; that's a good last resource in case you want to send ESC when

using other applications but not when inside Vim.

1.11 MAPPING META-KEYS

`:map-meta-keys`

Mapping keys with the Meta modifier works very similar to using the Alt key. What key on your keyboard produces the Meta modifier depends on your keyboard and configuration.

Note that mapping `<M-a>` actually is for using the Alt key. That can be confusing! It cannot be changed, it would not be backwards compatible.

For the Meta modifier the "T" character is used. For example, to map Meta-b in Insert mode:

```
:imap <T-b> terrible
```

1.12 MAPPING SUPER-KEYS or COMMAND-KEYS

`:map-super-keys` `:map-cmd-key`

The Super modifier is available in GUI mode (when `gui_running` is 1) for GVim on Linux and MacVim on Mac OS. If you're on a Mac, this represents the Command key, on Linux with the GTK GUI it represents the Super key. The character "D" is used for the Super / Command modifier.

For example, to map Command-b in Insert mode:

```
:imap <D-b> baritone
```

1.13 MAPPING IN modifyOtherKeys mode

`modifyOtherKeys`

Xterm and a few other terminals can be put in a mode where keys with modifiers are sent with a special escape code. Vim recognizes these codes and can then make a difference between **CTRL-H** and Backspace, even when Backspace sends the character 8. And many more special keys, such as Tab and **CTRL-I**, which cannot be mapped separately otherwise.

For xterm `modifyOtherKeys` is enabled in the builtin termcap entry. If this is not used you can enable `modifyOtherKeys` with these lines in your vimrc:

```
let &t_TI = "\<Esc>[>4;2m"
let &t_TE = "\<Esc>[>4;m"
```

This sets `modifyOtherKeys` to level 2. **Note** that `modifyOtherKeys` level 1 does not work. Some terminals do not support level 2 and then send key codes that Vim will not be able to correctly recognize.

In case the `modifyOtherKeys` mode causes problems you can disable it:

```
let &t_TI = ""
let &t_TE = ""
```

It does not take effect immediately. To have this work without restarting Vim execute a shell command, e.g.: ``!ls`` Or put the lines in your `vimrc`.

When `modifyOtherKeys` is enabled you can map `<C->` and `<C-S->`:

```
imap <C-> [[[
imap <C-> {{{
```

Without `modifyOtherKeys` `<C->` and `<C->` are indistinguishable from Esc.

Note that `<C->` is used and not `<C-S->` or `<C-S->`. This works on most

keyboards. Similarly, `<C-}>` is used instead of `<C-S-]>` or `<C-S-}>` and `<C-|>` instead of `<C-S-|>` or `<C-S-|>`. Note that `|` has a special meaning in a mapping, see `map-bar`.

WARNING: if you map `<C-[>` you may very well break any key codes that start with Esc. Make sure it comes AFTER other mappings.

Starting with xterm version 377 Vim can detect the `modifyOtherKeys` state by requesting it. For this the `'t_RK'` termcap entry is used. When the response is found then Vim will know whether `modifyOtherKeys` level 2 is enabled, and handle mappings accordingly.

Before version 377 Vim automatically detects if the `modifyOtherKeys` mode was enabled when it spots an escape sequence that must have been created by it. To see if Vim detected such an escape sequence use ``:verbose map``, the first line will then show "Seen `modifyOtherKeys`: true" (possibly translated).

This automatic detection depends on receiving an escape code starting with `"<1b>[27;"`. This is the normal way xterm sends these key codes. However, if the `formatOtherKeys` resource is set another form is used that is not recognized, therefore you must not set `formatOtherKeys`.

A known side effect is that in Insert mode the raw escape sequence is inserted after the `CTRL-V` key. This can be used to check whether `modifyOtherKeys` is enabled: In Insert mode type `CTRL-SHIFT-V CTRL-V`, if you get one byte then `modifyOtherKeys` is off, if you get `<1b>[27;5;118~` then it is on.

Note that xterm up to version 376 has a bug that makes Shift-Esc send a regular Esc code, the Shift modifier is dropped.

When the `'esckeys'` option is off, then `modifyOtherKeys` will be disabled in Insert mode to avoid every key with a modifier causing Insert mode to end.

1.14 MAPPING WITH KITTY KEYBOARD PROTOCOL `kitty-keyboard-protocol`

If the value of `'term'` contains "kitty" then Vim will send out an escape sequence to enable the Kitty keyboard protocol. This can be changed with the `'keyprotocol'` option.

Like `modifyOtherKeys`, this will make it possible to distinguish between more keys with modifiers. Also, this protocol sends an escape sequence for the Esc key, so that Vim does not need to use a timeout to know whether receiving an Esc character means the Esc key was pressed or it's the start of an escape sequence.

Vim automatically detects if the Kitty keyboard protocol was enabled when it spots the response to the status request (this should be part of the `t_TI` termcap entry). To see if Vim detected such an escape sequence use:

```
:verbose map
```

The first line will then show "Kitty keyboard protocol: `{value}`" (possibly translated). The meaning of `{value}`:

Unknown	no status received yet
Off	protocol is not used

On	protocol is used
Disabled	protocol was used but expected to have been disabled by ' t_TE '
Cleared	protocol expected to have been disabled by ' t_TE ', previous state is unknown

1.15 MAPPING AN OPERATOR

:map-operator

An operator is used before a `{motion}` command. To define your own operator you must create a mapping that first sets the '**operatorfunc**' option and then invoke the `g@` operator. After the user types the `{motion}` command the specified function will be called.

`g@{motion}` Call the function set by the '**operatorfunc**' option. The '[' mark is positioned at the start of the text moved over by `{motion}`, the ']' mark on the last character of the text. The function is called with one String argument:

```

"line"      {motion} was linewise
"char"      {motion} was characterwise
"block"     {motion} was blockwise-visual

```

The type can be forced, see **forced-motion**.
{not available when compiled without the **+eval** feature}

Here is an example that counts the number of spaces with `<F4>`:

```

nnoremap <expr> <F4> CountSpaces()
xnoremap <expr> <F4> CountSpaces()
" doubling <F4> works on a line
nnoremap <expr> <F4><F4> CountSpaces() .. '_'

function CountSpaces(context = {}, type = '') abort
  if a:type == ''
    let context = #{
      \ dot_command: v:false,
      \ extend_block: '',
      \ virtualedit: [&l:virtualedit, &g:virtualedit],
      \ }
    let &operatorfunc = function('CountSpaces', [context])
    set virtualedit=block
    return 'g@'
  endif

let save = #{
  \ clipboard: &clipboard,
  \ selection: &selection,
  \ virtualedit: [&l:virtualedit, &g:virtualedit],
  \ register: getreginfo(''),
  \ visual_marks: [getpos("<"), getpos(">")],
  \ }

```

```

try
  set clipboard= selection=inclusive virtualedit=
  let commands = #{
    \ line: "'[V']",
    \ char: "`[v`]",
    \ block: "`[\<C-V>`]",
    \ }[a:type]
  let [_, _, col, off] = getpos("")
  if off != 0
    let vcol = getline("")->strpart(0, col + off)->strdisplaywidth()
    if vcol >= [line("", '$')->virtcol() - 1
      let a:context.extend_block = '$'
    else
      let a:context.extend_block = vcol .. '|'
    endif
  endif
  if a:context.extend_block != ''
    let commands .= 'o0' .. a:context.extend_block
  endif
  let commands .= 'y'
  execute 'silent noautocmd keepjumps normal! ' .. commands
  echomsg getreg('')->count(' ')
finally
  call setreg('', save.register)
  call setpos('<', save.visual_marks[0])
  call setpos('>', save.visual_marks[1])
  let &clipboard = save.clipboard
  let &selection = save.selection
  let [&l:virtualedit, &g:virtualedit] = get(a:context.dot_command ? save : a:con
  let a:context.dot_command = v:true
endtry
endfunction

```

An `<expr>` mapping is used to be able to fetch any prefixed count and register. This also avoids using a command line, which would trigger `CmdlineEnter` and `CmdlineLeave` autocommands.

Note that the `'selection'` option is temporarily set to "inclusive" to be able to yank exactly the right text by using Visual mode from the '[' to the ']' mark.

Also **note** that the `'clipboard'` option is temporarily emptied to avoid clobbering the ``"*`` or ``"+`` registers, if its value contains the item ``unnamed`` or ``unnamedplus``.

The ``mode()`` function will return the state as it will be after applying the operator.

Here is an example for using a lambda function to create a normal-mode operator to add quotes around text in the current line:

```

nnoremap <F4> <Cmd>let &opfunc='{t ->
                \ getline(".")
                \ ->split("\zs")

```

```

\ ->insert("\'", col("'"'])")
\ ->insert("\'", col("'"[") - 1)
\ ->join("")
\ ->setline(".")}'<CR>g@

```

2. Abbreviations

abbreviations Abbreviations

Abbreviations are used in Insert mode, Replace mode and Command-line mode. If you enter a word that is an abbreviation, it is replaced with the word it stands for. This can be used to save typing for often used long words. And you can use it to automatically correct obvious spelling errors.

Examples:

```

:iab ms Microsoft
:iab tihs this

```

There are three types of abbreviations:

full-id The "full-id" type consists entirely of keyword characters (letters and characters from '**iskeyword**' option). This is the most common abbreviation.

Examples: "foo", "g3", "-1"

end-id The "end-id" type ends in a keyword character, but all the other characters are not keyword characters.

Examples: "#i", "..f", "\$/7"

non-id The "non-id" type ends in a non-keyword character, the other characters may be of any type, excluding space and tab. {this type is not supported by Vi}

Examples: "def#", "4/7\$"

Examples of strings that cannot be abbreviations: "a.b", "#def", "a b", "_\$r"

An abbreviation is only recognized when you type a non-keyword character. This can also be the <Esc> that ends Insert mode or the <CR> that ends a command. The non-keyword character which ends the abbreviation is inserted after the expanded abbreviation. An exception to this is the character <C-]>, which is used to expand an abbreviation without inserting any extra characters.

Example:

```

:ab hh            hello
"hh<Space>" is expanded to "hello<Space>"
"hh<C-]>" is expanded to "hello"

```

The characters before the cursor must match the abbreviation. Each type has an additional rule:

full-id In front of the match is a non-keyword character, or this is where

the line or insertion starts. Exception: When the abbreviation is only one character, it is not recognized if there is a non-keyword character in front of it, other than a space or a tab. However, for the command line "'<,'>" (or any other marks) is ignored, as if the command line starts after it.

end-id In front of the match is a keyword character, or a space or a tab, or this is where the line or insertion starts.

non-id In front of the match is a space, tab or the start of the line or the insertion.

Examples: ({CURSOR} is where you type a non-keyword character)

```
:ab foo four old otters
    "foo{CURSOR}" is expanded to "four old otters"
    "foobar{CURSOR}" is not expanded
    "barfoo{CURSOR}" is not expanded

:ab #i #include
    "#i{CURSOR}" is expanded to "#include"
    ">#i{CURSOR}" is not expanded

:ab ;; <endofline>
    "test;;" is not expanded
    "test ;;" is expanded to "test <endofline>"
```

To avoid the abbreviation in Insert mode: Type **CTRL-V** before the character that would trigger the abbreviation. E.g. **CTRL-V** <Space>. Or type part of the abbreviation, exit insert mode with <Esc>, re-enter insert mode with "a" and type the rest.

To avoid the abbreviation in Command-line mode: Type **CTRL-V** twice somewhere in the abbreviation to avoid it to be replaced. A **CTRL-V** in front of a normal character is mostly ignored otherwise.

It is possible to move the cursor after an abbreviation:

```
:iab if if (<Left>
```

This does not work if '**cpoptions**' includes the '<' flag. <>

You can even do more complicated things. For example, to consume the space typed after an abbreviation:

```
func Eatchar(pat)
    let c = nr2char(getchar(0))
    return (c =~ a:pat) ? ' ' : c
endfunc
iabbr <silent> if if (<Left><C-R>=Eatchar('\s')<CR>
```

There are no default abbreviations.

Abbreviations are never recursive. You can use ":ab f f-o-o" without any problem. But abbreviations can be mapped. {some versions of Vi support recursive abbreviations, for no apparent reason}

Abbreviations are disabled if the '**paste**' option is on.

`:abbreviate-local` `:abbreviate-<buffer>`

Just like mappings, abbreviations can be local to a buffer. This is mostly used in a `filetype-plugin` file. Example for a C plugin file:

```
:abb <buffer> FF for (i = 0; i < ; ++i)
```

`:ab` `:abbreviate`

`:ab[breviate]` list all abbreviations. The character in the first column indicates the mode where the abbreviation is used: 'i' for insert mode, 'c' for Command-line mode, '!' for both. These are the same as for mappings, see `map-listing`.

`:abbreviate-verbose`

When **'verbose'** is non-zero, listing an abbreviation will also display where it was last defined. Example:

```
:verbose abbreviate
! teh the
Last set from /home/abcd/vim/abbr.vim
```

See `:verbose-cmd` for more information.

`:ab[breviate] {lhs}` list the abbreviations that start with `{lhs}`. You may need to insert a **CTRL-V** (type it twice) to avoid that a typed `{lhs}` is expanded, since command-line abbreviations apply here.

`:ab[breviate] [<expr>] [<buffer>] {lhs} {rhs}` add abbreviation for `{lhs}` to `{rhs}`. If `{lhs}` already existed it is replaced with the new `{rhs}`. `{rhs}` may contain spaces. See `:map-<expr>` for the optional `<expr>` argument. See `:map-<buffer>` for the optional `<buffer>` argument.

`:una` `:unabbreviate`

`:una[bbreviate] [<buffer>] {lhs}` Remove abbreviation for `{lhs}` from the list. If none is found, remove abbreviations in which `{lhs}` matches with the `{rhs}`. This is done so that you can even remove abbreviations after expansion. To avoid expansion insert a **CTRL-V** (type it twice).

`:norea` `:noreabbrev`

`:norea[bbrev] [<expr>] [<buffer>] [lhs] [rhs]` Same as `":ab"`, but no remapping for this `{rhs}`.

`:ca` `:cab` `:cabbrev`

`:ca[bbrev] [<expr>] [<buffer>] [lhs] [rhs]` Same as `":ab"`, but for Command-line mode only.

`:cuna` `:cunabbrev`

`:cuna[bbrev] [<buffer>] {lhs}` Same as `":una"`, but for Command-line mode only.

Stored as: `esc` `^V^["`

After parsing, the abbreviation's short form ("`esc`") and long form (the two characters "`^V^["`) are stored in the abbreviation table. If you give the `:ab` command with no arguments, this is how the abbreviation will be displayed.

Later, when the abbreviation is expanded because the user typed in the word "`esc`", the long form is subjected to the same type of `^V` interpretation as keyboard input. So the `^V` protects the `^["` character from being interpreted as the "exit Insert mode" character. Instead, the `^["` is inserted into the text.

Expands to: `^["`

[example given by Steve Kirkendall]

3. Local mappings and functions

`script-local`

When using several Vim script files, there is the danger that mappings and functions used in one script use the same name as in other scripts. To avoid this, they can be made local to the script.

`<SID>` `<SNR>` `E81`

The string "`<SID>`" can be used in a mapping or menu. This requires that the '`<`' flag is not present in '`cpoptions`'. This is useful if you have a `script-local` function that you want to call from a mapping in the same script.

When executing the map command, Vim will replace "`<SID>`" with the special key code `<SNR>`, followed by a number that's unique for the script, and an underscore. Example:

```
:map <SID>Add
```

would define a mapping "`<SNR>23_Add`".

When defining a function in a script, "`s:`" can be prepended to the name to make it local to the script (in `Vim9` script functions without a prefix are local to the script). But when a mapping is executed from outside of the script, it doesn't know in which script the function was defined. To avoid this problem, use "`<SID>`" instead of "`s:`". The same translation is done as for mappings. This makes it possible to define a call to the function in a mapping.

When a local function is executed, it runs in the context of the script it was defined in. This means that new functions and mappings it defines can also use "`s:`" or "`<SID>`" and it will use the same unique number as when the function itself was defined. Also, the "`s:var`" local script variables can be used.

When executing an autocommand or a user command, it will run in the context of the script it was defined in. This makes it possible that the command calls a local function or uses a local mapping.

In case the value is used in a context where `<SID>` cannot be correctly

expanded, use the `expand()` function:

```
let &includexpr = expand('<SID>') .. 'My_includeexpr()'
```

Otherwise, using "`<SID>`" outside of a script context is an error.

If you need to get the script number to use in a complicated script, you can use this function:

```
func s:ScriptNumber()  
    return matchstr(expand('<SID>'), '<SNR>\zs\d\+\ze_')  
endfunc
```

The "`<SNR>`" will be shown when listing functions and mappings. This is useful to find out what they are defined to.

The `:scriptnames` command can be used to see which scripts have been sourced and what their `<SNR>` number is.

This is all {not available when compiled without the `|+eval|` feature}.

4. User-defined commands

user-commands

It is possible to define your own Ex commands. A user-defined command can act just like a built-in command (it can have a range or arguments, arguments can be completed as filenames or buffer names, etc), except that when the command is executed, it is transformed into a normal Ex command and then executed.

For starters: See section [40.2](#) in the user manual.

E183 E841 user-cmd-ambiguous

All user defined commands must start with an uppercase letter, to avoid confusion with builtin commands. Exceptions are these builtin commands:

```
:Next  
:X
```

They cannot be used for a user defined command. `:Print` is also an existing command, but it is deprecated and can be overruled.

The other characters of the user command can be uppercase letters, lowercase letters or digits. When using digits, [note](#) that other commands that take a numeric argument may become ambiguous. For example, the command `:Cc2` could be the user command `:Cc2` without an argument, or the command `:Cc` with argument `"2"`. It is advised to put a space between the command name and the argument to avoid these problems.

When using a user-defined command, the command can be abbreviated. However, if an abbreviation is not unique, an error will be issued. Furthermore, a built-in command will always take precedence.

Example:

```
:command Rename ...  
:command Renumber ...  
:Rena " Means "Rename"  
:Renu " Means "Renumber"  
:Ren " Error - ambiguous
```



```

:command Paste ...
:P                               " The built-in :Print

```

It is recommended that full names for user-defined commands are used in scripts.

```

:com[mand]                                :com  :command
List all user-defined commands. When listing commands,
the characters in the first columns are:
!   Command has the -bang attribute
"   Command has the -register attribute
|   Command has the -bar attribute
b   Command is local to current buffer
(see below for details on attributes)
The list can be filtered on command name with
:filter , e.g., to list all commands with "Pyth" in
the name:
    filter Pyth command

```

```

:com[mand] {cmd}                          List the user-defined commands that start with {cmd}

```

When **'verbose'** is non-zero, listing a command will also display where it was last defined and any completion argument. Example:

```

:verbose command TOhtml
Name      Args Range Complete Definition
TOhtml    0    %           :call Convert2HTML(<line1>, <line2>)
Last set from /usr/share/vim/vim-7.0/plugin/tohtml.vim

```

See `:verbose-cmd` for more information.

```

:com[mand][!] [{attr}...] {cmd} {repl}
Define a user command. The name of the command is
{cmd} and its replacement text is {repl}. The
command's attributes (see below) are {attr}. If the
command already exists, an error is reported, unless a
! is specified, in which case the command is
redefined. There is one exception: When sourcing a
script again, a command that was previously defined in
that script will be silently replaced.

```

```

:delc[ommand] {cmd}                       :delc  :delcommand  E184
Delete the user-defined command {cmd}.
This is not allowed while listing commands, e.g. from
a timer. E1311

```

```

:delc[ommand] -buffer {cmd}               E1237
Delete the user-defined command {cmd} that was defined
for the current buffer.

```

```

:comc[lear]                                :comc  :comclear

```

Delete all user-defined commands.

Command attributes

[command-attributes](#)

User-defined commands are treated by Vim just like any other Ex commands. They can have arguments, or have a range specified. Arguments are subject to completion as filenames, buffers, etc. Exactly how this works depends upon the command's attributes, which are specified when the command is defined.

When defining a user command in a script, it will be able to call functions local to the script and use mappings local to the script. When the user invokes the user command, it will run in the context of the script it was defined in. This matters if `<SID>` is used in a command.

There are a number of attributes, split into four categories: argument handling, completion behavior, range handling, and special cases. The attributes are described below, by category.

Argument handling

[E175](#) [E176](#) [:command-nargs](#)

By default, a user defined command will take no arguments (and an error is reported if any are supplied). However, it is possible to specify that the command can take arguments, using the `-nargs` attribute. Valid cases are:

<code>-nargs=0</code>	No arguments are allowed (the default)
<code>-nargs=1</code>	Exactly one argument is required, it includes spaces
<code>-nargs=*</code>	Any number of arguments are allowed (0, 1, or many), separated by white space
<code>-nargs=?</code>	0 or 1 arguments are allowed
<code>-nargs=+</code>	Arguments must be supplied, but any number are allowed

Arguments are considered to be separated by (unescaped) spaces or tabs in this context, except when there is one argument, then the white space is part of the argument.

Note that arguments are used as text, not as expressions. Specifically, `"s:var"` will use the script-local variable in the script where the command was defined, not where it is invoked! Example:

```
script1.vim:
  :let s:error = "None"
  :command -nargs=1 Error echoerr <args>
script2.vim:
  :source script1.vim
  :let s:error = "Wrong!"
  :Error s:error
```

Executing `script2.vim` will result in "None" being echoed. Not what you intended! Calling a function may be an alternative.

Completion behavior

[:command-completion](#) [E179](#) [E180](#) [E181](#)
[:command-complete](#)

By default, the arguments of user defined commands do not undergo completion. However, by specifying one or the other of the following attributes, argument completion can be enabled:

-complete=arglist	file names in argument list
-complete=augroup	autocmd groups
-complete=buffer	buffer names
-complete=behave	:behave suboptions
-complete=color	color schemes
-complete=command	Ex command (and arguments)
-complete=compiler	compilers
-complete=cscope	:cscope suboptions
-complete=dir	directory names
-complete=environment	environment variable names
-complete=event	autocommand events
-complete=expression	Vim expression
-complete=file	file and directory names
-complete=file_in_path	file and directory names in 'path'
-complete=filetype	filetype names 'filetype'
-complete=function	function name
-complete=help	help subjects
-complete=highlight	highlight groups
-complete=history	:history suboptions
-complete=keymap	keyboard mappings
-complete=locale	locale names (as output of locale -a)
-complete=mapclear	buffer argument
-complete=mapping	mapping name
-complete=menu	menus
-complete=messages	:messages suboptions
-complete=option	options
-complete=packadd	optional package pack-add names
-complete=shellcmd	Shell command
-complete=sign	:sign suboptions
-complete=syntax	syntax file names 'syntax'
-complete=syntime	:syntime suboptions
-complete=tag	tags
-complete=tag_listfiles	tags, file names are shown when CTRL-D is hit
-complete=user	user names
-complete=var	user variables
-complete=custom,{func}	custom completion, defined via {func}
-complete=customlist,{func}	custom completion, defined via {func}

If you specify completion while there is nothing to complete (-nargs=0, the default) then you get error **E1208** .

Note: That some completion methods might expand environment variables.

Custom completion

:command-completion-custom

:command-completion-customlist **E467** **E468**

It is possible to define customized completion schemes via the "custom,{func}" or the "customlist,{func}" completion argument. The {func} part should be a function with the following signature:

```
:function {func}(ArgLead, CmdLine, CursorPos)
```

The function need not use all these arguments. The function should provide the completion candidates as the return value.

For the "custom" argument, the function should return the completion candidates one per line in a newline separated string.

E1303

For the "customlist" argument, the function should return the completion candidates as a Vim List. Non-string items in the list are ignored.

The function arguments are:

ArgLead	the leading portion of the argument currently being completed on
CmdLine	the entire command line
CursorPos	the cursor position in it (byte index)

The function may use these for determining context. For the "custom" argument, it is not necessary to filter candidates against the (implicit pattern in) ArgLead. Vim will filter the candidates with its regexp engine after function return, and this is probably more efficient in most cases. If **'wildoptions'** contains "fuzzy", then the candidates will be filtered using **fuzzy-matching**. For the "customlist" argument, Vim will not filter the returned completion candidates and the user supplied function should filter the candidates.

The following example lists user names to a Finger command

```
:com -complete=custom,ListUsers -nargs=1 Finger !finger <args>
:fun ListUsers(A,L,P)
:   return system("cut -d: -f1 /etc/passwd")
:endfun
```

The following example completes filenames from the directories specified in the **'path'** option:

```
:com -nargs=1 -bang -complete=customlist,EditFileComplete
\ EditFile edit<bang> <args>
:fun EditFileComplete(A,L,P)
:   return split(globpath(&path, a:A), "\n")
:endfun
```

This example does not work for file names with spaces!

Range handling

E177 E178 :command-range :command-count

By default, user-defined commands do not accept a line number range. However, it is possible to specify that the command does take a range (the **-range** attribute), or that it takes an arbitrary count value, either in the line number position (**-range=N**, like the **:split** command) or as a "count" argument (**-count=N**, like the **:Next** command). The count will then be available in the argument with **<count>**.

Possible attributes are:

-range	Range allowed, default is current line
---------------	--

-range=% Range allowed, default is whole file (1,\$)
 -range=N A count (default N) which is specified in the line number position (like `:split`); allows for zero line number.
 -count=N A count (default N) which is specified either in the line number position, or as an initial argument (like `:Next`).
 -count Acts like `-count=0`

Note that `-range=N` and `-count=N` are mutually exclusive - only one should be specified.

:command-addr

It is possible that the special characters in the range like `.`, `$` or `%` which by default correspond to the current line, last line and the whole buffer, relate to arguments, (loaded) buffers, windows or tab pages.

Possible values are (second column is the short name used in listing):

-addr=lines		Range of lines (this is the default for <code>-range</code>)
-addr=arguments	arg	Range for arguments
-addr=buffers	buf	Range for buffers (also not loaded buffers)
-addr=loaded_buffers	load	Range for loaded buffers
-addr=windows	win	Range for windows
-addr=tabs	tab	Range for tab pages
-addr=quickfix	qf	Range for quickfix entries
-addr=other	?	Other kind of range; can use <code>."</code> , <code>"\$</code> and <code>"%</code> as with "lines" (this is the default for <code>-count</code>)

Special cases

[:command-bang](#) [:command-bar](#)
[:command-register](#) [:command-buffer](#)
[:command-keepsript](#)

There are some special cases as well:

-bang The command can take a `!` modifier (like `:q` or `:w`)
 -bar The command can be followed by a `"|"` and another command. A `"|"` inside the command argument is not allowed then. Also checks for a `"` to start a comment.
 -register The first argument to the command can be an optional register name (like `:del`, `:put`, `:yank`).
 -buffer The command will only be available in the current buffer.
 -keepsript Do not use the location of where the user command was defined for verbose messages, use the location of where the user command was invoked.

In the cases of the `-count` and `-register` attributes, if the optional argument is supplied, it is removed from the argument list and is available to the replacement text separately.

Note that these arguments can be abbreviated, but that is a deprecated feature. Use the full name for new scripts.

Replacement text

`:command-repl`

The `{repl}` argument is normally one long string, possibly with "|" separated commands. A special case is when the argument is "{", then the following lines, up to a line starting with "}" are used and Vim9 syntax applies. Example:

```
:command MyCommand {
    echo 'hello'
    g:calledMyCommand = true
}
```

E1231

There must be white space before the "{". No nesting is supported, inline functions cannot be used. Commands where a "|" may appear in the argument, such as commands with an expression argument, cannot be followed by a "|" and another command.

If the command is defined in Vim9 script (a script that starts with `:vim9script` and in a :def` function) then {repl} will be executed as in Vim9 script. Thus this depends on where the command is defined, not where it is used.`

The replacement text `{repl}` for a user defined command is scanned for special escape sequences, using `<...>` notation. Escape sequences are replaced with values from the entered command line, and all other text is copied unchanged. The resulting string is executed as an Ex command. To avoid the replacement use `<lt>` in place of the initial `<`. Thus to include "`<bang>`" literally use "`<lt>bang>`".

The valid escape sequences are

- `<line1>` The starting line of the command range.
 - `<line2>` The final line of the command range.
 - `<range>` The number of items in the command range: 0, 1 or 2
 - `<count>` Any count supplied (as described for the `'-range'` and `'-count'` attributes).
 - `<bang>` (See the `'-bang'` attribute) Expands to a ! if the command was executed with a ! modifier, otherwise expands to nothing.
 - `<mods>` The command modifiers, if specified. Otherwise, expands to nothing. Supported modifiers are `:aboveleft`, `:belowright`, `:botright`, `:browse`, `:confirm`, `:hide`, `:horizontal`, `:keepalt`, `:keepjumps`, `:keepmarks`, `:keeppatterns`, `:leftabove`, `:lockmarks`, `:noautocmd`, `:noswapfile`, `:rightbelow`, `:sandbox`, `:silent`, `:tab`, `:topleft`, `:unsilent`, `:verbose`, and `:vertical`.
Note that `:filter` is not supported.
- Examples:

```
command! -nargs=+ -complete=file MyEdit
    \ for f in expand(<q-args>, 0, 1) |
```

```

\ exe '<mods> split ' .. f |
\ endfor

```

```

function! SpecialEdit(files, mods)
  for f in expand(a:files, 0, 1)
    exe a:mods .. ' split ' .. f
  endfor
endfunction
command! -nargs=+ -complete=file Sedit
  \ call SpecialEdit(<q-args>, <q-mods>)

```

- <reg>** (See the '-register' attribute) The optional register, if specified. Otherwise, expands to nothing. **<register>** is a synonym for this.
- <args>** The command arguments, exactly as supplied (but as noted above, any count or register can consume some of the arguments, which are then not part of **<args>**).
- <lt>** A single '<' (Less-Than) character. This is needed if you want to get a literal copy of one of these escape sequences into the expansion - for example, to get **<bang>**, use **<lt>bang<**.

<q-args>

If the first two characters of an escape sequence are "q-" (for example, **<q-args>**) then the value is quoted in such a way as to make it a valid value for use in an expression. This uses the argument as one single value. When there is no argument **<q-args>** is an empty string. See the [q-args-example](#) below.

<f-args>

To allow commands to pass their arguments on to a user-defined function, there is a special form **<f-args>** ("function args"). This splits the command arguments at spaces and tabs, quotes each argument individually, and the **<f-args>** sequence is replaced by the comma-separated list of quoted arguments. See the [Mycmd](#) example below. If no arguments are given **<f-args>** is removed.

To embed whitespace into an argument of **<f-args>**, prepend a backslash. **<f-args>** replaces every pair of backslashes (\\) with one backslash. A backslash followed by a character other than white space or a backslash remains unmodified. Also see [f-args-example](#) below. Overview:

command	<f-args>
XX ab	'ab'
XX a\b	'a\b'
XX a\ b	'a b'
XX a\ b	'a ', 'b'
XX a\\b	'a\b'
XX a\\ b	'a\ ', 'b'
XX a\\\b	'a\\b'
XX a\\\ b	'a\ b'
XX a\\\\b	'a\\b'
XX a\\\\ b	'a\\ ', 'b'
XX	[nothing]

Note that if the "no arguments" situation is to be handled, you have to make sure that the function can be called without arguments. For a compiled function you might want to use variable arguments, see [vim9-variable-arguments](#) .

Examples for user commands:

```
" Delete everything after here to the end
:com Ddel +,$d

" Rename the current buffer
:com -nargs=1 -bang -complete=file Ren f <args>|w<bang>

" Replace a range with the contents of a file
" (Enter this all as one line)
:com -range -nargs=1 -complete=file
    Replace <line1>-pu_|<line1>,<line2>d|r <args>|<line1>d

" Count the number of lines in the range
:com! -range -nargs=0 Lines echo <line2> - <line1> + 1 "lines"
```

f-args-example

Call a user function (example of <f-args>)

```
:com -nargs=* Mycmd call Myfunc(<f-args>)
```

When executed as:

```
:Mycmd arg1 arg2
```

This will invoke:

```
:call Myfunc("arg1","arg2")
```

q-args-example

A more substantial example:

```
:function Allargs(command)
:   let i = 0
:   while i < argc()
:     if filereadable(argv(i))
:       execute "e " .. argv(i)
:       execute a:command
:     endif
:     let i = i + 1
:   endwhile
:endifunction
:command -nargs+= -complete=command Allargs call Allargs(<q-args>)
```

The command Allargs takes any Vim command(s) as argument and executes it on all files in the argument list. Usage example (note use of the "e" flag to ignore errors and the "update" command to write modified buffers):

```
:Allargs %s/foo/bar/ge|update
```

This will invoke:

```
:call Allargs("%s/foo/bar/ge|update")
```


Tags and special searches

tags-and-searches

See section 29.1 of the user manual for an introduction.

- 1. Jump to a tag [tag-commands](#)
- 2. Tag stack [tag-stack](#)
- 3. Tag match list [tag-matchlist](#)
- 4. Tags details [tag-details](#)
- 5. Tags file format [tags-file-format](#)
- 6. Include file searches [include-search](#)
- 7. Using 'tagfunc' [tag-function](#)



1. Jump to a tag

[tag-commands](#)

[tag](#) [tags](#)

A tag is an identifier that appears in a "tags" file. It is a sort of label that can be jumped to. For example: In C programs each function name can be used as a tag. The "tags" file has to be generated by a program like ctags, before the tag commands can be used.

With the ":tag" command the cursor will be positioned on the tag. With the **CTRL-]** command, the keyword on which the cursor is standing is used as the tag. If the cursor is not on a keyword, the first keyword to the right of the cursor is used.

The ":tag" command works very well for C programs. If you see a call to a function and wonder what that function does, position the cursor inside of the function name and hit **CTRL-]**. This will bring you to the function definition. An easy way back is with the **CTRL-T** command. Also read about the tag stack below.

[:ta](#) [:tag](#) [E426](#) [E429](#)

`:[count]ta[g][!] {name}`

Jump to the definition of {name}, using the information in the tags file(s). Put {name} in the tag stack. See [tag-!](#) for [!]. {name} can be a regexp pattern, see [tag-regexp](#). When there are several matching tags for {name}, jump to the [count] one. When [count] is omitted the first one is jumped to. See [tag-matchlist](#) for jumping to other matching tags.

`g<LeftMouse>`
`<C-LeftMouse>`
CTRL-]

[g<LeftMouse>](#)
[<C-LeftMouse>](#) [CTRL-\]](#)

Jump to the definition of the keyword under the cursor. Same as ":tag {name}", where {name} is the keyword under or after cursor.

When there are several matching tags for {name}, jump to the [count] one. When no [count] is given the first one is jumped to. See [tag-matchlist](#) for jumping to other matching tags.

{Visual}CTRL-] v_CTRL-]
Same as ":tag {name}", where {name} is the text that is highlighted.

CTRL-] telnet-CTRL-]
is the default telnet escape key. When you type **CTRL-]** to jump to a tag, you will get the telnet prompt instead. Most versions of telnet allow changing or disabling the default escape key. See the telnet man page. You can 'telnet -E {Hostname}' to disable the escape character, or 'telnet -e {EscapeCharacter} {Hostname}' to specify another escape character. If possible, try to use "ssh" instead of "telnet" to avoid this problem.

tag-priority
When there are multiple matches for a tag, this priority is used:

1. "FSC" A full matching static tag for the current file.
2. "F C" A full matching global tag for the current file.
3. "F " A full matching global tag for another file.
4. "FS " A full matching static tag for another file.
5. " SC" An ignore-case matching static tag for the current file.
6. " C" An ignore-case matching global tag for the current file.
7. " " An ignore-case matching global tag for another file.
8. " S " An ignore-case matching static tag for another file.

Note that when the current file changes, the priority list is mostly not changed, to avoid confusion when using ":tnext". It is changed when using ":tag {name}".

The ignore-case matches are not found for a ":tag" command when:

- 'tagcase' is "followic" and the 'ignorecase' option is off
- 'tagcase' is "followscs" and the 'ignorecase' option is off and the 'smartcase' option is off or the pattern contains an upper case character.
- 'tagcase' is "match"
- 'tagcase' is "smart" and the pattern contains an upper case character.

The ignore-case matches are found when:

- a pattern is used (starting with a "/"
- for ":tselect"
- when 'tagcase' is "followic" and 'ignorecase' is on
- when 'tagcase' is "followscs" and 'ignorecase' is on or the 'smartcase' option is on and the pattern does not contain an upper case character
- when 'tagcase' is "ignore"
- when 'tagcase' is "smart" and the pattern does not contain an upper case character

Note that using ignore-case tag searching disables binary searching in the tags file, which causes a slowdown. This can be avoided by fold-case sorting the tag file. See the 'tagbsearch' option for an explanation.

=====

2. Tag stack

tag-stack tagstack E425

On the tag stack is remembered which tags you jumped to, and from where. Tags are only pushed onto the stack when the '**tagstack**' option is set.

<code>g<RightMouse></code>	<code>g<RightMouse></code>
<code><C-RightMouse></code>	<code><C-RightMouse></code> <code>CTRL-T</code>
CTRL-T	Jump to <code>[count]</code> older entry in the tag stack (default 1).
<code>:[count]po[p][!]</code>	<code>:po</code> <code>:pop</code> <code>E555</code> <code>E556</code> Jump to <code>[count]</code> older entry in tag stack (default 1). See <code>tag-!</code> for <code>[!]</code> .
<code>:[count]ta[g][!]</code>	Jump to <code>[count]</code> newer entry in tag stack (default 1). See <code>tag-!</code> for <code>[!]</code> .
<code>:tags</code>	<code>:tags</code> Show the contents of the tag stack. The active entry is marked with a '>'.

The output of `":tags"` looks like this:

```
# T0 tag      FROM line  in file/text
1  1 main          1  harddisk2:text/vim/test
> 2  2 FuncA       58  i = FuncA(10);
3  1 FuncC       357  harddisk2:text/vim/src/amiga.c
```

This list shows the tags that you jumped to and the cursor position before that jump. The older tags are at the top, the newer at the bottom.

The '>' points to the active entry. This is the tag that will be used by the next `":tag"` command. The **CTRL-T** and `":pop"` command will use the position above the active entry.

Below the "T0" is the number of the current match in the match list. [Note](#) that this doesn't change when using `":pop"` or `":tag"`.

The line number and file name are remembered to be able to get back to where you were before the tag command. The line number will be correct, also when deleting/inserting lines, unless this was done by another program (e.g. another instance of Vim).

For the current file, the "file/text" column shows the text at the position. An indent is removed and a long line is truncated to fit in the window.

You can jump to previously used tags with several commands. Some examples:

<code>":pop"</code> or CTRL-T	to position before previous tag
<code>{count}CTRL-T</code>	to position before <code>{count}</code> older tag
<code>":tag"</code>	to newer tag
<code>":0tag"</code>	to last used tag

The most obvious way to use this is while browsing through the call graph of

a program. Consider the following call graph:

```
main ---> FuncA ---> FuncC
      ---> FuncB
```

(Explanation: main calls FuncA and FuncB; FuncA calls FuncC). You can get from main to FuncA by using **CTRL-]** on the call to FuncA. Then you can **CTRL-]** to get to FuncC. If you now want to go back to main you can use **CTRL-T** twice. Then you can **CTRL-]** to FuncB.

If you issue a `":ta {name}"` or **CTRL-]** command, this tag is inserted at the current position in the stack. If the stack was full (it can hold up to 20 entries), the oldest entry is deleted and the older entries shift one position up (their index number is decremented by one). If the last used entry was not at the bottom, the entries below the last used one are deleted. This means that an old branch in the call graph is lost. After the commands explained above the tag stack will look like this:

```
# TO tag      FROM line  in file/text
1  1 main          1  harddisk2:text/vim/test
2  1 FuncB         59  harddisk2:text/vim/src/main.c
```

The `gettagstack()` function returns the tag stack of a specified window. The `settagstack()` function modifies the tag stack of a window.

tagstack-examples

Write to the tag stack just like `':tag'` but with a user-defined `jumper#jump_to_tag` function:

```
" Store where we're jumping from before we jump.
let tag = expand('<cword>')
let pos = [bufnr()] + getcurpos()[1:]
let item = {'bufnr': pos[0], 'from': pos, 'tagname': tag}
if jumper#jump_to_tag(tag)
    " Jump was successful, write previous location to tag stack.
    let winid = win_getid()
    let stack = gettagstack(winid)
    let stack['items'] = [item]
    call settagstack(winid, stack, 't')
endif
```

Set current index of the tag stack to 4:

```
call settagstack(1005, {'curidx' : 4})
```

Push a new item onto the tag stack:

```
let pos = [bufnr('myfile.txt'), 10, 1, 0]
let newtag = [{'tagname' : 'mytag', 'from' : pos}]
call settagstack(2, {'items' : newtag}, 'a')
```

E73

When you try to use the tag stack while it doesn't contain anything you will get an error message.

=====
3. Tag match list

tag-matchlist E427 E428

When there are several matching tags, these commands can be used to jump between them. **Note** that these commands don't change the tag stack, they keep the same entry.

`:ts[elect][!] [name]` `:ts` `:tselect`
List the tags that match `[name]`, using the information in the tags file(s).
When `[name]` is not given, the last tag name from the tag stack is used.
See `tag-!` for `[!]`.
With a `'>'` in the first column is indicated which is the current position in the list (if there is one).
`[name]` can be a regexp pattern, see `tag-regexp`.
See `tag-priority` for the priorities used in the listing.
Example output:

```
# pri kind tag file
1 F f mch_delay os_amiga.c
   mch_delay(msec, ignoreinput)
> 2 F f mch_delay os_msdos.c
   mch_delay(msec, ignoreinput)
3 F f mch_delay os_unix.c
   mch_delay(msec, ignoreinput)
Type number and <Enter> (empty cancels):
```

See `tag-priority` for the "pri" column. **Note** that this depends on the current file, thus using `":tselect xxx"` can produce different results.
The "kind" column gives the kind of tag, if this was included in the tags file.
The "info" column shows information that could be found in the tags file. It depends on the program that produced the tags file.
When the list is long, you may get the `more-prompt`.
If you already see the tag you want to use, you can type 'q' and enter the number.

`:sts[elect][!] [name]` `:sts` `:stselect`
Does `":tselect[!] [name]"` and splits the window for the selected tag.

`g]` `g]`
Like **CTRL-]**, but use `":tselect"` instead of `":tag"`.

`{Visual}g]` `v_g]`
Same as `"g]"`, but use the highlighted text as the identifier.

`:tj[ump][!] [name]` `:tj` `:tjump`
Like `":tselect"`, but jump to the tag directly when there is only one match.

<code>:stj[ump][!] [name]</code>	Does <code>":tjump[!] [name]"</code> and splits the window for the selected tag.	<code>:stj</code> <code>:stjump</code>
<code>g CTRL-]</code>	Like <code>CTRL-]</code> , but use <code>":tjump"</code> instead of <code>":tag"</code> .	<code>g_CTRL-]</code>
<code>{Visual}g CTRL-]</code>	Same as <code>"g CTRL-]"</code> , but use the highlighted text as the identifier.	<code>v_g_CTRL-]</code>
<code>:[count]tn[ext][!]</code>	Jump to <code>[count]</code> next matching tag (default 1). See <code>tag-!</code> for <code>[!]</code> .	<code>:tn</code> <code>:tnext</code>
<code>:[count]tp[revious][!]</code>	Jump to <code>[count]</code> previous matching tag (default 1). See <code>tag-!</code> for <code>[!]</code> .	<code>:tp</code> <code>:tprevious</code>
<code>:[count]tN[ext][!]</code>	Same as <code>":tprevious"</code> .	<code>:tN</code> <code>:tNext</code>
<code>:[count]tr[ewind][!]</code>	Jump to first matching tag. If <code>[count]</code> is given, jump to <code>[count]</code> th matching tag. See <code>tag-!</code> for <code>[!]</code> .	<code>:tr</code> <code>:trewind</code>
<code>:[count]tf[irst][!]</code>	Same as <code>":trewind"</code> .	<code>:tf</code> <code>:tfirst</code>
<code>:tl[ast][!]</code>	Jump to last matching tag. See <code>tag-!</code> for <code>[!]</code> .	<code>:tl</code> <code>:tlast</code>
<code>:lt[ag][!] [name]</code>	Jump to tag <code>[name]</code> and add the matching tags to a new location list for the current window. <code>[name]</code> can be a regexp pattern, see <code>tag-regexp</code> . When <code>[name]</code> is not given, the last tag name from the tag stack is used. The search pattern to locate the tag line is prefixed with <code>"\V"</code> to escape all the special characters (very nomagic). The location list showing the matching tags is independent of the tag stack. See <code>tag-!</code> for <code>[!]</code> .	<code>:lt</code> <code>:ltag</code>

When there is no other message, Vim shows which matching tag has been jumped to, and the number of matching tags:

`tag 1 of 3 or more`

The " or more" is used to indicate that Vim didn't try all the tags files yet. When using `":tnext"` a few times, or with `":tlast"`, more matches may be found.

When you didn't see this message because of some other message, or you just want to know where you are, this command will show it again (and jump to the same tag as last time):

`:0tn`

tag-skip-file

When a matching tag is found for which the file doesn't exist, this match is skipped and the next matching tag is used. Vim reports this, to notify you of missing files. When the end of the list of matches has been reached, an error message is given.

tag-preview

The tag match list can also be used in the preview window. The commands are the same as above, with a "p" prepended.

{not available when compiled without the |+quickfix| feature}

:pts :ptselect

`:pts[elect][!] [name]` Does `:tselect[!] [name]` and shows the new tag in a "Preview" window. See `:ptag` for more info.

:ptj :ptjump

`:ptj[ump][!] [name]` Does `:tjump[!] [name]` and shows the new tag in a "Preview" window. See `:ptag` for more info.

:ptn :ptnext

`:[count]ptn[ext][!]` `":tnext"` in the preview window. See `:ptag`.

:ptp :ptprevious

`:[count]ptp[revious][!]` `":tprevious"` in the preview window. See `:ptag`.

:ptN :ptNext

`:[count]ptN[ext][!]` Same as `":ptprevious"`.

:ptr :ptrewind

`:[count]ptr[ewind][!]` `":trewind"` in the preview window. See `:ptag`.

:ptf :ptfirst

`:[count]ptf[irst][!]` Same as `":ptrewind"`.

:ptl :ptlast

`:ptl[ast][!]` `":tlast"` in the preview window. See `:ptag`.

4. Tags details

tag-details

static-tag

A static tag is a tag that is defined for a specific file. In a C program this could be a static function.

In Vi jumping to a tag sets the current search pattern. This means that the "n" command after jumping to a tag does not search for the same pattern that it did before jumping to the tag. Vim does not do this as we consider it to be a bug. If you really want the old Vi behavior, set the 't' flag in **'cptions'**.

tag-binary-search

Vim uses binary searching in the tags file to find the desired tag quickly (when enabled at compile time `+tag_binary`). But this only works if the

tags file was sorted on ASCII byte value. Therefore, if no match was found, another try is done with a linear search. If you only want the linear search, reset the **'tagbsearch'** option. Or better: Sort the tags file!

Note that the binary searching is disabled when not looking for a tag with a specific name. This happens when ignoring case and when a regular expression is used that doesn't start with a fixed string. Tag searching can be a lot slower then. The former can be avoided by case-fold sorting the tags file. See **'tagbsearch'** for details.

tag-regex

The `":tag"` and `":tselect"` commands accept a regular expression argument. See **pattern** for the special characters that can be used.

When the argument starts with `'/'`, it is used as a pattern. If the argument does not start with `'/'`, it is taken literally, as a full tag name.

Examples:

```
:tag main
    jumps to the tag "main" that has the highest priority.
:tag /^get
    jumps to the tag that starts with "get" and has the highest priority.
:tag /norm
    lists all the tags that contain "norm", including "id_norm".
```

When the argument both exists literally, and match when used as a regexp, a literal match has a higher priority. For example, `":tag /open"` matches `"open"` before `"open_file"` and `"file_open"`.

When using a pattern case is ignored. If you want to match case use `"\C"` in the pattern.

tag-!

If the tag is in the current file this will always work. Otherwise the performed actions depend on whether the current file was changed, whether a `!` is added to the command and on the **'autowrite'** option:

tag in	file			autowrite	
current	changed	!	wifixbuf	option	action
yes	x	x	no	x	goto tag
no	no	x	no	x	read other file, goto tag
no	yes	yes	no	x	abandon current file, read other file, goto tag
no	yes	no	no	on	write current file, read other file, goto tag
no	yes	no	no	off	fail
yes	x	yes	x	x	goto tag
no	no	no	yes	x	fail
no	yes	no	yes	x	fail
no	yes	no	yes	on	fail
no	yes	no	yes	off	fail

- If the tag is in the current file, the command will always work.
- If the tag is in another file and the current file was not changed, the other file will be made the current file and read into the buffer.
- If the tag is in another file, the current file was changed and a `!` is

- added to the command, the changes to the current file are lost, the other file will be made the current file and read into the buffer.
- If the tag is in another file, the current file was changed and the **'autowrite'** option is on, the current file will be written, the other file will be made the current file and read into the buffer.
 - If the tag is in another file, the current file was changed and the **'autowrite'** option is off, the command will fail. If you want to save the changes, use the **":w"** command and then use **":tag"** without an argument. This works because the tag is put on the stack anyway. If you want to lose the changes you can use the **":tag!"** command.
 - If the tag is in another file and the window includes **'winfixbuf'**, the command will fail. If the tag is in the same file then it may succeed.

tag-security

Note that Vim forbids some commands, for security reasons. This works like using the **'secure'** option for `exrc/vimrc` files in the current directory. See [trojan-horse](#) and [sandbox](#) .

When the `{tagaddress}` changes a buffer, you will get a warning message:

```
"WARNING: tag command changed a buffer!!!"
```

In a future version changing the buffer will be impossible. All this for security reasons: Somebody might hide a nasty command in the tags file, which would otherwise go unnoticed. Example:

```
:$d|/tag-function-name/
```

In Vi the **":tag"** command sets the last search pattern when the tag is searched for. In Vim this is not done, the previous search pattern is still remembered, unless the **'t'** flag is present in **'coptions'**.

emacs-tags emacs_tags E430

Emacs style tag files are only supported if Vim was compiled with the **+emacs_tags** feature enabled. Sorry, there is no explanation about Emacs tag files here, it is only supported for backwards compatibility :-).

Lines in Emacs tags files can be very long. Vim only deals with lines of up to about 510 bytes. To see whether lines are ignored set **'verbose'** to 5 or higher. Non-Emacs tags file lines can be any length.

tags-option

The **'tags'** option is a list of file names. Each of these files is searched for the tag. This can be used to use a different tags file than the default file "tags". It can also be used to access a common tags file.

The next file in the list is not used when:

- A matching static tag for the current buffer has been found.
- A matching global tag has been found.

This also depends on whether case is ignored. Case is ignored when:

- **'tagcase'** is "followic" and **'ignorecase'** is set
- **'tagcase'** is "ignore"
- **'tagcase'** is "smart" and the pattern only contains lower case characters.
- **'tagcase'** is "followscs" and **'smartcase'** is set and the pattern only contains lower case characters.

If case is not ignored, and the tags file only has a match without matching case, the next tags file is searched for a match with matching case. If no

tag with matching case is found, the first match without matching case is used. If case is ignored, and a matching global tag with or without matching case is found, this one is used, no further tags files are searched.

When a tag file name starts with "./", the '.' is replaced with the path of the current file. This makes it possible to use a tags file in the directory where the current file is (no matter what the current directory is). The idea of using "." is that you can define which tag file is searched first: In the current directory ("tags,./tags") or in the directory of the current file ("./tags,tags").

For example:

```
:set tags=./tags,tags,/home/user/commontags
```

In this example the tag will first be searched for in the file "tags" in the directory where the current file is. Next the "tags" file in the current directory. If it is not found there, then the file "/home/user/commontags" will be searched for the tag.

This can be switched off by including the 'd' flag in '**cptions**', to make it Vi compatible. "./tags" will then be the tags file in the current directory, instead of the tags file in the directory where the current file is.

Instead of the comma a space may be used. Then a backslash is required for the space to be included in the string option:

```
:set tags=tags\ /home/user/commontags
```

To include a space in a file name use three backslashes. To include a comma in a file name use two backslashes. For example, use:

```
:set tags=tag\\\ file,/home/user/common\\,tags
```

for the files "tag file" and "/home/user/common,tags". The '**tags**' option will have the value "tag\ file,/home/user/common,tags".

If the '**tagrelative**' option is on (which is the default) and using a tag file in another directory, file names in that tag file are relative to the directory where the tag file is.

5. Tags file format tags-file-format E431

ctags jtags

A tags file can be created with an external command, for example "ctags". It will contain a tag for each function. Some versions of "ctags" will also make a tag for each "#defined" macro, typedefs, enums, etc.

Some programs that generate tags files:

ctags As found on most Unix systems. Only supports C. Only does the basic work.

universal ctags A maintained version of ctags based on exuberant ctags. See <https://ctags.io>.

Exuberant_ctags

exuberant ctags This is a very good one. It works for C, C++, Java,

	Fortran, Eiffel and others. It can generate tags for many items. See http://ctags.sourceforge.net . No new version since 2009.
etags	Connected to Emacs. Supports many languages.
JTags	For Java, in Java. It can be found at http://www.fleiner.com/jtags/ .
ptags.py	For Python, in Python. Found in your Python source directory at Tools/scripts/ptags.py.
ptags	For Perl, in Perl. It can be found at http://www.eleves.ens.fr:8080/home/nthiery/Tags/ .
gnatxref	For Ada. See http://www.gnuada.org/ . gnatxref is part of the gnat package.

The lines in the tags file must have one of these two formats:

1. {tagname} {TAB} {tagfile} {TAB} {tagaddress}
2. {tagname} {TAB} {tagfile} {TAB} {tagaddress} {term} {field} ..

Previously an old format was supported, see [tag-old-static](#) .

The first format is a normal tag, which is completely compatible with Vi. It is the only format produced by traditional ctags implementations. This is often used for functions that are global, also referenced in other files.

The lines in the tags file can end in <NL> or <CR><NL>. On the Macintosh <CR> also works. The <CR> and <NL> characters can never appear inside a line.

The second format is new. It includes additional information in optional fields at the end of each line. It is backwards compatible with Vi. It is only supported by new versions of ctags (such as Universal ctags or Exuberant ctags).

{tagname}	The identifier. Normally the name of a function, but it can be any identifier. It cannot contain a <Tab>.
{TAB}	One <Tab> character. Note: previous versions allowed any white space here. This has been abandoned to allow spaces in {tagfile}.
{tagfile}	The file that contains the definition of {tagname}. It can have an absolute or relative path. It may contain environment variables and wildcards (although the use of wildcards is doubtful). It cannot contain a <Tab>.
{tagaddress}	The Ex command that positions the cursor on the tag. It can be any Ex command, although restrictions apply (see tag-security). Posix only allows line numbers and search commands, which are mostly used.
{term}	;" The two characters semicolon and double quote. This is interpreted by Vi as the start of a comment, which makes the following be ignored. This is for backwards compatibility with Vi, it ignores the following fields. Example: <pre>APP file /^static int APP;\$/" v</pre> When {tagaddress} is not a line number or search pattern, then {term} must be ;" . Here the bar ends the command (excluding the bar) and ;" is used to have Vi ignore the rest of the

line. Example:
APP file.c call cursor(3, 4)|;" v

{field} .. A list of optional fields. Each field has the form:

```
<Tab>{fieldname}:{value}
```

The {fieldname} identifies the field, and can only contain alphabetical characters [a-zA-Z].

The {value} is any string, but cannot contain a <Tab>.

These characters are special:

"\t" stands for a <Tab>

"\r" stands for a <CR>

"\n" stands for a <NL>

"\" stands for a single \" character

There is one field that doesn't have a ':'. This is the kind of the tag. It is handled like it was preceded with "kind:". See the documentation of ctags for the kinds it produces.

The only other field currently recognized by Vim is "file:" (with an empty value). It is used for a static tag.

The first lines in the tags file can contain lines that start with

```
!_TAG_
```

These are sorted to the first lines, only rare tags that start with "!" can sort to before them. Vim recognizes two items. The first one is the line that indicates if the file was sorted. When this line is found, Vim uses binary searching for the tags file:

```
!_TAG_FILE_SORTED<Tab>1<Tab>{anything}
```

A tag file may be case-fold sorted to avoid a linear search when case is ignored. (Case is ignored when 'ignorecase' is set and 'tagcase' is "followic", or when 'tagcase' is "ignore".) See 'tagbsearch' for details. The value '2' should be used then:

```
!_TAG_FILE_SORTED<Tab>2<Tab>{anything}
```

The other tag that Vim recognizes is the encoding of the tags file:

```
!_TAG_FILE_ENCODING<Tab>utf-8<Tab>{anything}
```

Here "utf-8" is the encoding used for the tags. Vim will then convert the tag being searched for from 'encoding' to the encoding of the tags file. And when listing tags the reverse happens. When the conversion fails the unconverted tag is used.

tag-search

The command can be any Ex command, but often it is a search command.

Examples:

```
tag1 file1 /^main(argc, argv)/  
tag2 file2 108
```

The command is always executed with 'magic' not set. The only special characters in a search pattern are "^" (begin-of-line) and "\$" (<EOL>). See [pattern](#) . Note that you must put a backslash before each backslash in

the search text. This is for backwards compatibility with Vi.

E434 E435

If the command is a normal search command (it starts and ends with "/" or "?"), some special handling is done:

- Searching starts on line 1 of the file.

The direction of the search is forward for "/", backward for "?".

Note that **'wrapscan'** does not matter, the whole file is always searched.

- If the search fails, another try is done ignoring case. If that fails too, a search is done for:

```
"^tagname[ \t]*("
```

(the tag with '^' prepended and "[\t]*(" appended). When using function names, this will find the function name when it is in column 0. This will help when the arguments to the function have changed since the tags file was made. If this search also fails another search is done with:

```
"^[#a-zA-Z_].*\<tagname[ \t]*("
```

This means: A line starting with '#' or an identifier and containing the tag followed by white space and a '(' . This will find macro names and function names with a type prepended.

tag-old-static

Until March 2019 (patch 8.1.1092) an outdated format was supported:

```
{tagfile}:{tagname} {TAB} {tagfile} {TAB} {tagaddress}
```

This format is for a static tag only. It is obsolete now, replaced by the second format. It is only supported by Elvis 1.x, older Vim versions and a few versions of ctags. A static tag is often used for functions that are local, only referenced in the file {tagfile}. **Note** that for the static tag, the two occurrences of {tagfile} must be exactly the same. Also see [tags-option](#) below, for how static tags are used.

The support was removed, since when you can update to the new [Vim version you](#) should also be able to update ctags to one that supports the second format.

6. Include file searches

[include-search](#) [definition-search](#)
E387 E388 E389

These commands look for a string in the current file and in all encountered included files (recursively). This can be used to find the definition of a variable, function or macro. If you only want to search in the current buffer, use the commands listed at [pattern-searches](#) .

These commands are not available when the [+find_in_path](#) feature was disabled at compile time.

When a line is encountered that includes another file, that file is searched before continuing in the current buffer. Files included by included files are also searched. When an include file could not be found it is silently ignored. Use the [:checkpath](#) command to discover which files could not be found, possibly your **'path'** option is not set up correctly. **Note:** the included file is searched, not a buffer that may be editing that file. Only for the current file the lines in the buffer are used.

The string can be any keyword or a defined macro. For the keyword any match will be found. For defined macros only lines that match with the **'define'** option will be found. The default is `^\#\s*define`, which is for C programs. For other languages you probably want to change this. See **'define'** for an example for C++. The string cannot contain an end-of-line, only matches within a line are found.

When a match is found for a defined macro, the displaying of lines continues with the next line when a line ends in a backslash.

The commands that start with `"["` start searching from the start of the current file. The commands that start with `"]"` start at the current cursor position.

The **'include'** option is used to define a line that includes another file. The default is `^\#\s*include`, which is for C programs. **Note:** Vim does not recognize C syntax, if the **'include'** option matches a line inside `"#ifdef/#endif"` or inside a comment, it is searched anyway. The **'isfname'** option is used to recognize the file name that comes after the matched pattern.

The **'path'** option is used to find the directory for the include files that do not have an absolute path.

The **'comments'** option is used for the commands that display a single line or jump to a line. It defines patterns that may start a comment. Those lines are ignored for the search, unless `[!]` is used. One exception: When the line matches the pattern `^\# *define` it is not considered to be a comment.

If you want to list matches, and then select one to jump to, you could use a mapping to do that for you. Here is an example:

```
:map <F4> [I:let nr = input("Which one: ")<Bar>exe "normal " .. nr .. "[\t"<CR>
```

```
[i                                     [i
    Display the first line that contains the keyword
    under the cursor. The search starts at the beginning
    of the file. Lines that look like a comment are
    ignored (see 'comments' option). If a count is given,
    the count'th matching line is displayed, and comment
    lines are not ignored.
```

```
]i                                     ]i
    like "[i", but start at the current cursor position.
```

```
:[range]is[earch][!] [count] [/]pattern[/]          :is  :isearch
    Like "[i" and "]i", but search in [range] lines
    (default: whole file).
    See :search-args for [/] and [!].
```

```
[I                                     [I
    Display all lines that contain the keyword under the
    cursor. Filenames and line numbers are displayed
```

for the found lines. The search starts at the beginning of the file.

`]I` like "[I", but start at the current cursor position.

`:[range]il[ist][!] [/]pattern[/]`
Like "[I" and "]I", but search in [range] lines (default: whole file).
See [:search-args](#) for [/] and [!].

`[CTRL-I` Jump to the first line that contains the keyword under the cursor. The search starts at the beginning of the file. Lines that look like a comment are ignored (see '[comments](#)' option). If a count is given, the count'th matching line is jumped to, and comment lines are not ignored.

`] CTRL-I` like "[CTRL-I", but start at the current cursor position.

`:[range]ij[ump][!] [count] [/]pattern[/]`
Like "[CTRL-I" and "] CTRL-I", but search in [range] lines (default: whole file).
See [:search-args](#) for [/] and [!].

`CTRL-W CTRL-I`
`CTRL-W i` Open a new window, with the cursor on the first line that contains the keyword under the cursor. The search starts at the beginning of the file. Lines that look like a comment line are ignored (see '[comments](#)' option). If a count is given, the count'th matching line is jumped to, and comment lines are not ignored.

`:[range]isp[lit][!] [count] [/]pattern[/]`
Like "CTRL-W i" and "CTRL-W i", but search in [range] lines (default: whole file).
See [:search-args](#) for [/] and [!].

`[d` Display the first macro definition that contains the macro under the cursor. The search starts from the beginning of the file. If a count is given, the count'th matching line is displayed.

`]d` like "[d", but start at the current cursor position.

:ds :dsearch

:**[range]ds[earch][!]** **[count]** **[/]string[/]**
 Like "**d**" and "**d**", but search in **[range]** lines (default: whole file).
 See **:search-args** for **[/]** and **[!]**.

[D

Display all macro definitions that contain the macro under the cursor. Filenames and line numbers are displayed for the found lines. The search starts from the beginning of the file.

]D

like "[D", but start at the current cursor position.

:dli :dlist

:**[range]dli[st][!]** **[/]string[/]**
 Like **`[D`** and **`]D`**, but search in **[range]** lines (default: whole file).
 See **:search-args** for **[/]** and **[!]**.
Note that **`:dl`** works like **`:delete`** with the "l" flag, not **`:dlist`**.

[_CTRL-D

[**CTRL-D** Jump to the first macro definition that contains the keyword under the cursor. The search starts from the beginning of the file. If a count is given, the count'th matching line is jumped to.

]_CTRL-D

] **CTRL-D** like "[**CTRL-D**", but start at the current cursor position.

:dj :djump

:**[range]dj[ump][!]** **[count]** **[/]string[/]**
 Like "**[CTRL-D**" and "**] CTRL-D**", but search in **[range]** lines (default: whole file).
 See **:search-args** for **[/]** and **[!]**.

CTRL-W CTRL-D **CTRL-W_CTRL-D** **CTRL-W_d**
CTRL-W d Open a new window, with the cursor on the first macro definition line that contains the keyword under the cursor. The search starts from the beginning of the file. If a count is given, the count'th matching line is jumped to.

:dsp :dsplit

:**[range]dsp[lit][!]** **[count]** **[/]string[/]**
 Like "**CTRL-W d**", but search in **[range]** lines (default: whole file).
 See **:search-args** for **[/]** and **[!]**.

:che :chec :check :checkpath

:che[ckpath] List all the included files that could not be found.

`:che[ckpath]!` List all the included files.

`:search-args`

Common arguments for the commands above:

[!] When included, find matches in lines that are recognized as comments. When excluded, a match is ignored when the line is recognized as a comment (according to `'comments'`), or the match is in a C comment (after `"//"` or inside `/* */`). Note that a match may be missed if a line is recognized as a comment, but the comment ends halfway the line. And if the line is a comment, but it is not recognized (according to `'comments'`) a match may be found in it anyway. Example:

```
/* comment
  foobar */
```

A match for "foobar" is found, because this line is not recognized as a comment (even though syntax highlighting does recognize it).

Note: Since a macro definition mostly doesn't look like a comment, the [!] makes no difference for `":dlist"`, `":dsearch"` and `":djump"`.

[/] A pattern can be surrounded by `'/'`. Without `'/'` only whole words are matched, using the pattern `"\<pattern\>"`. Only after the second `'/'` a next command can be appended with `'|'`. Example:

```
:isearch /string/ | echo "the last one"
```

For a `":djump"`, `":dsplit"`, `":dlist"` and `":dsearch"` command the pattern is used as a literal string, not as a search pattern.

7. Using `'tagfunc'`

`tag-function`

It is possible to provide Vim with a function which will generate a list of tags used for commands like `:tag`, `:tselect` and Normal mode tag commands like `CTRL-]`.

The function used for generating the taglist is specified by setting the `'tagfunc'` option. The function will be called with three arguments:

pattern	The tag identifier or pattern used during the tag search.
flags	String containing flags to control the function behavior.
info	Dict containing the following entries:
	buf_ffname Full filename which can be used for priority.
	user_data Custom data String, if stored in the tag stack previously by tagfunc.

Note that in a legacy function `"a:"` needs to be prepended to the argument name when using it.

Currently up to three flags may be passed to the tag function:

'c' The function was invoked by a normal command being processed (mnemonic: the tag function may use the context around the cursor to perform a better job of generating the tag list.)

'i' In Insert mode, the user was completing a tag (with `i_CTRL-X_CTRL-]` or `'completeopt'` contains ``t``).

'r' The first argument to tagfunc should be interpreted as a pattern (see `tag-regex`), such as when using:

```
:tag /pat
```

It is also given when completing in insert mode.

If this flag is not present, the argument is usually taken literally as the full tag name.

Note that when **'tagfunc'** is set, the priority of the tags described in **tag-priority** does not apply. Instead, the priority is exactly as the ordering of the elements in the list returned by the function.

E987

The function should return a List of Dict entries. Each Dict must at least include the following entries and each value must be a string:

name	Name of the tag.
filename	Name of the file where the tag is defined. It is either relative to the current directory or a full path.
cmd	Ex command used to locate the tag in the file. This can be either an Ex search pattern or a line number.

Note that the format is similar to that of **taglist()**, which makes it possible to use its output to generate the result.

The following fields are optional:

kind	Type of the tag.
user_data	String of custom data stored in the tag stack which can be used to disambiguate tags between operations.

If the function returns **v:null** instead of a List, a standard tag lookup will be performed instead.

It is not allowed to change the tagstack from inside **'tagfunc'**. E986

It is not allowed to close a window or change window from inside **'tagfunc'**.

E1299

The following is a hypothetical example of a function used for **'tagfunc'**. It uses the output of **taglist()** to generate the result: a list of tags in the inverse order of file names.

```
function TagFunc(pattern, flags, info)
  function CompareFileNames(item1, item2)
    let f1 = a:item1['filename']
    let f2 = a:item2['filename']
    return f1 >=# f2 ?
      \ -1 : f1 <=# f2 ? 1 : 0
  endfunction

  let result = taglist(a:pattern)
  call sort(result, "CompareFileNames")

  return result
endfunc
set tagfunc=TagFunc
```

Editing with multiple windows and buffers. [windows](#) [buffers](#)

The commands which have been added to use multiple windows and buffers are explained here. Additionally, there are explanations for commands that work differently when used in combination with more than one window.

The basics are explained in chapter 7 and 8 of the user manual [usr_07.txt](#) [usr_08.txt](#) .

- | | |
|---|------------------------------------|
| 1. Introduction | windows-intro |
| 2. Starting Vim | windows-starting |
| 3. Opening and closing a window | opening-window |
| 4. Moving cursor to other windows | window-move-cursor |
| 5. Moving windows around | window-moving |
| 6. Window resizing | window-resize |
| 7. Argument and buffer list commands | buffer-list |
| 8. Do a command in all buffers or windows | list-repeat |
| 9. Tag or file name under the cursor | window-tag |
| 10. The preview window | preview-window |
| 11. Using hidden buffers | buffer-hidden |
| 12. Special kinds of buffers | special-buffers |

{not able to use multiple windows when the [+windows](#) feature was disabled at compile time}

-
1. Introduction [windows-intro](#) [window](#)

Summary:

- A buffer is the in-memory text of a file.
- A window is a viewport on a buffer.
- A tab page is a collection of windows.

A window is a viewport onto a buffer. You can use multiple windows on one buffer, or several windows on different buffers.

A buffer is a file loaded into memory for editing. The original file remains unchanged until you write the buffer to the file.

A buffer can be in one of three states:

[active-buffer](#)
active: The buffer is displayed in a window. If there is a file for this buffer, it has been read into the buffer. The buffer may have been modified since then and thus be different from the file.

[hidden-buffer](#)
hidden: The buffer is not displayed. If there is a file for this buffer, it has been read into the buffer. Otherwise it's the same as an active

buffer, you just can't see it.

inactive-buffer

inactive: The buffer is not displayed and does not contain anything. Options for the buffer are remembered if the file was once loaded. It can contain marks from the `viminfo` file. But the buffer doesn't contain text.

In a table:

state	displayed in window	loaded	":buffers" shows
active	yes	yes	'a'
hidden	no	yes	'h'
inactive	no	no	' '

Note: All **CTRL-W** commands can also be executed with `:wincmd`, for those places where a Normal mode command can't be used or is inconvenient.

The main Vim window can hold several split windows. There are also tab pages `tab-page`, each of which can hold multiple windows.

window-ID `winid` `windowid`

Each window has a unique identifier called the window ID. This identifier will not change within a Vim session. The `win_getid()` and `win_id2tabwin()` functions can be used to convert between the window/tab number and the identifier. There is also the window number, which may change whenever windows are opened or closed, see `winnr()`.

The window number is only valid in one specific tab. The window ID is valid across tabs. For most functions that take a window ID or a window number, the window number only applies to the current tab, while the window ID can refer to a window in any tab.

Each buffer has a unique number and the number will not change within a Vim session. The `bufnr()` and `bufname()` functions can be used to convert between a buffer name and the buffer number.

2. Starting Vim

windows-starting

By default, Vim starts with one window, just like Vi.

The `"-o"` and `"-O"` arguments to Vim can be used to open a window for each file in the argument list. The `"-o"` argument will split the windows horizontally; the `"-O"` argument will split the windows vertically. If both `"-o"` and `"-O"` are given, the last one encountered will be used to determine the split orientation. For example, this will open three windows, split horizontally:

```
vim -o file1 file2 file3
```

`"-oN"`, where N is a decimal number, opens N windows split horizontally. If there are more file names than windows, only N windows are opened and some files do not get a window. If there are more windows than file names, the last few windows will be editing empty buffers. Similarly, `"-ON"` opens N windows split vertically, with the same restrictions.

If there are many file names, the windows will become very small. You might

want to set the **'winheight'** and/or **'winwidth'** options to create a workable situation.

Buf/Win Enter/Leave **autocommand** s are not executed when opening the new windows and reading the files, that's only done when they are really entered.

status-line

A status line will be used to separate windows. The **'laststatus'** option tells when the last window also has a status line:

```
'laststatus' = 0      never a status line
'laststatus' = 1      status line if there is more than one window
'laststatus' = 2      always a status line
```

You can change the contents of the status line with the **'statusline'** option. This option can be local to the window, so that you can have a different status line in each window.

Normally, inversion is used to display the status line. This can be changed with the 's' character in the **'highlight'** option. For example, "sb" sets it to bold characters. If no highlighting is used for the status line ("sn"), the '^' character is used for the current window, and '=' for other windows. If the mouse is supported and enabled with the **'mouse'** option, a status line can be dragged to resize windows.

Note: If you expect your status line to be in reverse video and it isn't, check if the **'highlight'** option contains "si". In version 3.0, this meant to invert the status line. Now it should be "sr", reverse the status line, as "si" now stands for italic! If italic is not available on your terminal, the status line is inverted anyway; you will only see this problem on terminals that have termcap codes for italics.

filler-lines

The lines after the last buffer line in a window are called filler lines. By default, these lines start with a tilde (~) character. The **'eob'** item in the **'fillchars'** option can be used to change this character. By default, these characters are highlighted as NonText (**hl-NonText**). The EndOfBuffer highlight group (**hl-EndOfBuffer**) can be used to change the highlighting of the filler characters.

3. Opening and closing a window

opening-window

CTRL-W s
CTRL-W S
CTRL-W CTRL-S

CTRL-W_s
CTRL-W_S
CTRL-W_CTRL-S
:sp :split

:[N]sp[lit] [++opt] [+cmd] [file]

Split current window in two. The result is two viewports on the same file.

Make the new window N high (default is to use half the height of the current window). Reduces the current window height to create room (and others, if the **'equalalways'** option is set, **'eadirection'** isn't "hor", and one of them is higher than the current or the new window).

If `[file]` is given it will be edited in the new window. If it is not loaded in any buffer, it will be read. Else the new window will use the already loaded buffer.

Note: `CTRL-S` does not work on all terminals and might block further input, use `CTRL-Q` to get going again. Also see `++opt` and `+cmd` .

E242 E1159

Be careful when splitting a window in an autocommand, it may mess up the window layout if this happens while making other window layout changes.

`CTRL-W CTRL-V`

`CTRL-W v`

`:[N]vs[plit] [++opt] [+cmd] [file]`

`CTRL-W_CTRL-V`

`CTRL-W_v`

`:vs :vsplit`

Like `:split` , but split vertically. The windows will be spread out horizontally if

1. a width was not specified,
2. `'equalalways'` is set,
3. `'eadirection'` isn't "ver", and
4. one of the other windows is wider than the current or new window.

If N was given make the new window N columns wide, if possible.

Note: In other places `CTRL-Q` does the same as `CTRL-V`, but here it doesn't!

`CTRL-W n`

`CTRL-W CTRL-N`

`:[N]new [++opt] [+cmd]`

`CTRL-W_n`

`CTRL-W_CTRL-N`

`:new`

Create a new window and start editing an empty file in it. Make new window N high (default is to use half the existing height). Reduces the current window height to create room (and others, if the `'equalalways'` option is set and `'eadirection'` isn't "hor").

Also see `++opt` and `+cmd` .

If `'fileformats'` is not empty, the first format given will be used for the new buffer. If `'fileformats'` is empty, the `'fileformat'` of the current buffer is used. This can be overridden with the `++opt` argument.

Autocommands are executed in this order:

1. WinLeave for the current window
2. WinEnter for the new window
3. BufLeave for the current buffer
4. BufEnter for the new buffer

This behaves like a `":split"` first, and then an `":enew"` command.

`:[N]new [++opt] [+cmd] {file}`

`:[N]sp[lit] [++opt] [+cmd] {file}`

`:split_f`

Create a new window and start editing file `{file}` in it. This behaves almost like a `":split"` first, and then an `":edit"` command, but the alternate file name in the original window is

set to {file}.

If [+cmd] is given, execute the command when the file has been loaded +cmd .

Also see ++opt .

Make new window N high (default is to use half the existing height). Reduces the current window height to create room (and others, if the 'equalalways' option is set).

: [N]vne[w] [++opt] [+cmd] [file] :vne :vnew
Like :new , but split vertically. If 'equalalways' is set and 'eadirection' isn't "ver" the windows will be spread out horizontally, unless a width was specified.

: [N]sv[iew] [++opt] [+cmd] [file] :sv :sview splitview
Same as ":split", but set 'readonly' option for this buffer.

: [N]sf[ind] [++opt] [+cmd] {file} :sf :sfi :sfind splitfind
Same as ":split", but search for {file} in 'path' like in :find . Doesn't split if {file} is not found.

CTRL-W CTRL-^ CTRL-W_CTRL-^ CTRL-W_^
CTRL-W ^ Split the current window in two and edit the alternate file. When a count N is given, split the current window and edit buffer N. Similar to ":sp #" and ":sp #N", but it allows the other buffer to be unnamed. This command matches the behavior of CTRL-^ , except that it splits a window first.

CTRL-W : CTRL-W_
Does the same as typing : - enter a command line. Useful in a terminal window, where all Vim commands must be preceded with CTRL-W or 'termwinkey'.

Note that the 'splitbelow' and 'splitright' options influence where a new window will appear.

E36

Creating a window will fail if there is not enough room. Every window needs at least one screen line and column, sometimes more. Options 'winminheight' and 'winminwidth' are relevant.

:vert[ical] {cmd} :vert :vertical
Execute {cmd}. If it contains a command that splits a window, it will be split vertically. For `vertical wincmd =` windows will be equalized only vertically. Doesn't work for :execute and :normal .

:hor[izontal] {cmd} :hor :horizontal
Execute {cmd}. Currently only makes a difference for `horizontal wincmd =`, which will equalize windows only horizontally.

:lefta[bove] {cmd} :lefta :leftabove
:abo[veleft] {cmd} :abo :aboveleft

Execute {cmd}. If it contains a command that splits a window, it will be opened left (vertical split) or above (horizontal split) the current window. Overrides **'splitbelow'** and **'splitright'**.

Doesn't work for `:execute` and `:normal` .

`:rightb[elow] {cmd}` `:rightb` `:rightbelow`
`:bel[owright] {cmd}` `:bel` `:belowright`

Execute {cmd}. If it contains a command that splits a window, it will be opened right (vertical split) or below (horizontal split) the current window. Overrides **'splitbelow'** and **'splitright'**.

Doesn't work for `:execute` and `:normal` .

`:topleft` E442

`:to[pleft] {cmd}`

Execute {cmd}. If it contains a command that splits a window, it will appear at the top and occupy the full width of the Vim window. When the split is vertical the window appears at the far left and occupies the full height of the Vim window.

Doesn't work for `:execute` and `:normal` .

`:bo` `:botright`

`:bo[tright] {cmd}`

Execute {cmd}. If it contains a command that splits a window, it will appear at the bottom and occupy the full width of the Vim window. When the split is vertical the window appears at the far right and occupies the full height of the Vim window.

Doesn't work for `:execute` and `:normal` .

These command modifiers can be combined to make a vertically split window occupy the full height. Example:

`:vertical topleft split tags`

Opens a vertically split, full-height window on the "tags" file at the far left of the Vim window.

Closing a window

`:q[uit]`
`:{count}q[uit]` `:count_quit`
CTRL-W q `CTRL-W_q`
CTRL-W CTRL-Q `CTRL-W_CTRL-Q`

Without {count}: Quit the current window. If {count} is given quit the {count} window.

`edit-window`

When quitting the last edit window (not counting help or preview windows), exit Vim.

When **'hidden'** is set, and there is only one window for the current buffer, it becomes hidden. When **'hidden'** is not set, and there is only one window for the current buffer, and the buffer was changed, the command fails.

(Note: **CTRL-Q** does not work on all terminals).

If `[count]` is greater than the last window number the last window will be closed:

```
:1quit " quit the first window
:$quit " quit the last window
:9quit " quit the last window
      " if there are fewer than 9 windows opened
:-quit " quit the previous window
:+quit " quit the next window
:+2quit " quit the second next window
```

When closing a help window, and this is not the only window, Vim will try to restore the previous window layout, see `:helpclose` .

```
:q[uit]!
:{count}q[uit]!
```

Without `{count}`: Quit the current window. If `{count}` is given quit the `{count}` window.

If this was the last window for a buffer, any changes to that buffer are lost. When quitting the last window (not counting help windows), exit Vim. The contents of the buffer are lost, even when `'hidden'` is set.

```
:clo[se][!]
:{count}clo[se][!]
```

CTRL-W c

CTRL-W_c `:clo` `:close`

Without `{count}`: Close the current window. If `{count}` is given close the `{count}` window.

When the `'hidden'` option is set, or when the buffer was changed and the `[!]` is used, the buffer becomes hidden (unless there is another window editing it).

When there is only one `edit-window` in the current tab page and there is another tab page, this closes the current tab page. `tab-page` .

This command fails when:

E444

- There is only one window on the screen.
- When `'hidden'` is not set, `[!]` is not used, the buffer has changes, and there is no other window on this buffer.

Changes to the buffer are not written and won't get lost, so this is a "safe" command.

CTRL-W CTRL-C

CTRL-W_CTRL-C

You might have expected that **CTRL-W CTRL-C** closes the current window, but that does not work, because the **CTRL-C** cancels the command.

`:hide`

:hid[e]
:{count}hid[e]

Without {count}: Quit the current window, unless it is the last window on the screen.
If {count} is given quit the {count} window.

The buffer becomes hidden (unless there is another window editing it or 'bufhidden' is "unload", "delete" or "wipe").
If the window is the last one in the current tab page the tab page is closed. [tab-page](#)

The value of 'hidden' is irrelevant for this command. Changes to the buffer are not written and won't get lost, so this is a "safe" command.

:hid[e] {cmd} Execute {cmd} with 'hidden' is set. The previous value of 'hidden' is restored after {cmd} has been executed.

Example:

`:hide edit Makefile`

This will edit "Makefile", and hide the current buffer if it has any changes.

:on[ly][!]
:{count}on[ly][!]

CTRL-W o
CTRL-W CTRL-O

CTRL-W_o E445
CTRL-W_CTRL-O :on :only

Make the current window the only one on the screen. All other windows are closed. For {count} see the `:quit` command above [:count_quit](#) .

When the 'hidden' option is set, all buffers in closed windows become hidden.

When 'hidden' is not set, and the 'autowrite' option is set, modified buffers are written. Otherwise, windows that have buffers that are modified are not removed, unless the [!] is given, then they become hidden. But modified buffers are never abandoned, so changes cannot get lost.

=====

4. Moving cursor to other windows

[window-move-cursor](#)

CTRL-W <Down>
CTRL-W CTRL-J
CTRL-W j

CTRL-W_<Down>
CTRL-W_CTRL-J CTRL-W_j

Move cursor to Nth window below current one. Uses the cursor position to select between alternatives.

CTRL-W <Up>
CTRL-W CTRL-K
CTRL-W k

CTRL-W_<Up>
CTRL-W_CTRL-K CTRL-W_k

Move cursor to Nth window above current one. Uses the cursor position to select between alternatives.

CTRL-W <Left>
CTRL-W CTRL-H

CTRL-W_<Left>
CTRL-W_CTRL-H

CTRL-W <BS> CTRL-W_<BS> CTRL-W_h
CTRL-W h Move cursor to Nth window left of current one. Uses the cursor position to select between alternatives.

CTRL-W <Right> CTRL-W_<Right>
CTRL-W CTRL-L CTRL-W_CTRL-L CTRL-W_l
CTRL-W l Move cursor to Nth window right of current one. Uses the cursor position to select between alternatives.

CTRL-W w CTRL-W_w CTRL-W_CTRL-W
CTRL-W CTRL-W Without count: move cursor to window below/right of the current one. If there is no window below or right, go to top-left window.
With count: go to Nth window (windows are numbered from top-left to bottom-right). To obtain the window number see `bufwinnr()` and `winnr()`. When N is larger than the number of windows go to the last window.

CTRL-W_W
CTRL-W W Without count: move cursor to window above/left of current one. If there is no window above or left, go to bottom-right window. With count: go to Nth window, like with **CTRL-W w**.

CTRL-W t CTRL-W_t CTRL-W_CTRL-T
CTRL-W CTRL-T Move cursor to top-left window.

CTRL-W b CTRL-W_b CTRL-W_CTRL-B
CTRL-W CTRL-B Move cursor to bottom-right window.

CTRL-W p CTRL-W_p CTRL-W_CTRL-P
CTRL-W CTRL-P Go to previous (last accessed) window.

CTRL-W_P E441
CTRL-W P Go to preview window. When there is no preview window this is an error.
{not available when compiled without the |+quickfix| feature}

If Visual mode is active and the new window is not for the same buffer, the Visual mode is ended. If the window is on the same buffer, the cursor position is set to keep the same Visual area selected.

:winc :wincmd

These commands can also be executed with " :wincmd":

```
:[count]winc[md] {arg}
```

Like executing **CTRL-W [count] {arg}**. Example:
:wincmd j
Moves to the window below the current one.
This command is useful when a Normal mode cannot be used (for the `CursorHold` autocommand event). Or when a Normal mode command is inconvenient.
The count can also be a window number. Example:
:exe nr .. "wincmd w"
This goes to window "nr".

5. Moving windows around

window-moving

CTRL-W r CTRL-W_r CTRL-W_CTRL-R E443
CTRL-W CTRL-R Rotate windows downwards/rightwards. The first window becomes the second one, the second one becomes the third one, etc. The last window becomes the first window. The cursor remains in the same window.
This only works within the row or column of windows that the current window is in.

CTRL-W_R
CTRL-W R Rotate windows upwards/leftwards. The second window becomes the first one, the third one becomes the second one, etc. The first window becomes the last window. The cursor remains in the same window.
This only works within the row or column of windows that the current window is in.

CTRL-W x CTRL-W_x CTRL-W_CTRL-X
CTRL-W CTRL-X Without count: Exchange current window with next one. If there is no next window, exchange with previous window.
With count: Exchange current window with Nth window (first window is 1). The cursor is put in the other window.
When vertical and horizontal window splits are mixed, the exchange is only done in the row or column of windows that the current window is in.

The following commands can be used to change the window layout. For example, when there are two vertically split windows, **CTRL-W K** will change that in horizontally split windows. **CTRL-W H** does it the other way around.

CTRL-W_K
CTRL-W K Move the current window to be at the very top, using the full width of the screen. This works like `:topleft split`, except it is applied to the current window and no new window is created.

CTRL-W_J
CTRL-W J Move the current window to be at the very bottom, using the full width of the screen. This works like `:botright split`, except it is applied to the current window and no new window is created.

CTRL-W_H
CTRL-W H Move the current window to be at the far left, using the full height of the screen. This works like `:vert toleft split`, except it is applied to the current window and no new window is created.

CTRL-W_L
CTRL-W L Move the current window to be at the far right, using the full height of the screen. This works like `:vert botright split`,

except it is applied to the current window and no new window is created.

CTRL-W_T

CTRL-W T Move the current window to a new tab page. This fails if there is only one window in the current tab page. This works like `:tab split`, except the previous window is closed. When a count is specified the new tab page will be opened before the tab page with this index. Otherwise it comes after the current tab page.

6. Window resizing

window-resize

CTRL-W_ =

CTRL-W = Make all windows (almost) equally high and wide, but use `'winheight'` and `'winwidth'` for the current window. Windows with `'winfixheight'` set keep their height and windows with `'winfixwidth'` set keep their width. To equalize only vertically (make window equally high) use ``vertical wincmd =``. To equalize only horizontally (make window equally wide) use ``horizontal wincmd =``.

:res[ize] -N **:res** **:resize** **CTRL-W_-**
CTRL-W - Decrease current window height by N (default 1).
If used after `:vertical` : decrease width by N.

:res[ize] +N **CTRL-W_+**
CTRL-W + Increase current window height by N (default 1).
If used after `:vertical` : increase width by N.

:res[ize] [N]
CTRL-W CTRL-_ **CTRL-W_CTRL-_** **CTRL-W__**
CTRL-W _ Set current window height to N (default: highest possible).

:{winnr}res[ize] [+ -]N
Like `:resize`` above, but apply the size to window `{winnr}` instead of the current window.

z{nr}<CR> Set current window height to `{nr}`.

CTRL-W_<

CTRL-W < Decrease current window width by N (default 1).

CTRL-W_>

CTRL-W > Increase current window width by N (default 1).

:vert[ical] res[ize] [N] **:vertical-resize** **CTRL-W_bar**
CTRL-W | Set current window width to N (default: widest possible).

You can also resize a window by dragging a status line up or down with the mouse. Or by dragging a vertical separator line left or right. This only

works if the version of Vim that is being used supports the mouse and the **'mouse'** option has been set to enable it.

The option **'winheight'** ('wh') is used to set the minimal window height of the current window. This option is used each time another window becomes the current window. If the option is '0', it is disabled. Set **'winheight'** to a very large value, e.g., '9999', to make the current window always fill all available space. Set it to a reasonable value, e.g., '10', to make editing in the current window comfortable.

The equivalent **'winwidth'** ('wiw') option is used to set the minimal width of the current window.

When the option **'equalalways'** ('ea') is set, all the windows are automatically made the same size after splitting or closing a window. If you don't set this option, splitting a window will reduce the size of the current window and leave the other windows the same. When closing a window, the extra lines are given to the window above it.

The **'eadirection'** option limits the direction in which the **'equalalways'** option is applied. The default "both" resizes in both directions. When the value is "ver" only the heights of windows are equalized. Use this when you have manually resized a vertically split window and want to keep this width. Likewise, "hor" causes only the widths of windows to be equalized.

The option **'cmdheight'** ('ch') is used to set the height of the command-line. If you are annoyed by the **hit-enter** prompt for long messages, set this option to 2 or 3.

If there is only one window, resizing that window will also change the command line height. If there are several windows, resizing the current window will also change the height of the window below it (and sometimes the window above it).

The minimal height and width of a window is set with **'winminheight'** and **'winminwidth'**. These are hard values, a window will never become smaller.

WinScrolled and WinResized autocommands

win-scrolled-resized

If you want to get notified of changes in window sizes, the **WinResized** autocommand event can be used.

If you want to get notified of text in windows scrolling vertically or horizontally, the **WinScrolled** autocommand event can be used. This will also trigger in window size changes.

Exception: the events will not be triggered when the text scrolls for **'incsearch'**.

WinResized-event

The **WinResized** event is triggered after updating the display, several windows may have changed size then. A list of the IDs of windows that changed since last time is provided in the `v:event.windows` variable, for example:

```
[1003, 1006]
```

WinScrolled-event

The **WinScrolled** event is triggered after **WinResized** , and also if a window

was scrolled. That can be vertically (the text at the top of the window changed) or horizontally (when `'wrap'` is off or when the first displayed part of the first line changes). Note that `WinScrolled` will trigger many more times than `WinResized`, it may slow down editing a bit.

The information provided by `WinScrolled` is a dictionary for each window that has changes, using the window ID as the key, and a total count of the changes with the key "all". Example value for `v:event` (Vim9 syntax):

```
{
  all: {width: 0, height: 2, leftcol: 0, skipcol: 0, topline: 1, topfill: 0},
  1003: {width: 0, height: -1, leftcol: 0, skipcol: 0, topline: 0, topfill: 0},
  1006: {width: 0, height: 1, leftcol: 0, skipcol: 0, topline: 1, topfill: 0},
}
```

Note that the "all" entry has the absolute values of the individual windows accumulated.

If you need more information about what changed, or you want to "debounce" the events (not handle every event to avoid doing too much work), you may want to use the `'winlayout()'` and `'getwininfo()'` functions.

`WinScrolled` and `WinResized` do not trigger when the first autocommand is added, only after the first scroll or resize. They may trigger when switching to another tab page.

The commands executed are expected to not cause window size or scroll changes. If this happens anyway, the event will trigger again very soon. In other words: Just before triggering the event, the current sizes and scroll positions are stored and used to decide whether there was a change.

E1312

It is not allowed to change the window layout here (split, close or move windows).

=====

7. Argument and buffer list commands

buffer-list

args list	buffer list	meaning
1. <code>:[N]argument [N]</code>	11. <code>:[N]buffer [N]</code>	to arg/buf N
2. <code>:[N]next [file ..]</code>	12. <code>:[N]bnext [N]</code>	to Nth next arg/buf
3. <code>:[N]Next [N]</code>	13. <code>:[N]bNext [N]</code>	to Nth previous arg/buf
4. <code>:[N]previous [N]</code>	14. <code>:[N]bprevious [N]</code>	to Nth previous arg/buf
5. <code>:rewind / :first</code>	15. <code>:brewind / :bfirst</code>	to first arg/buf
6. <code>:last</code>	16. <code>:blast</code>	to last arg/buf
7. <code>:all</code>	17. <code>:ball</code>	edit all args/buffers
	18. <code>:unhide</code>	edit all loaded buffers
	19. <code>:[N]bmod [N]</code>	to Nth modified buf

split & args list	split & buffer list	meaning
21. <code>:[N]sargument [N]</code>	31. <code>:[N]sbuffer [N]</code>	split + to arg/buf N
22. <code>:[N]snext [file ..]</code>	32. <code>:[N]sbnext [N]</code>	split + to Nth next arg/buf
23. <code>:[N]sNext [N]</code>	33. <code>:[N]sbNext [N]</code>	split + to Nth previous arg/buf
24. <code>:[N]sprevious [N]</code>	34. <code>:[N]sbprevious [N]</code>	split + to Nth previous arg/buf
25. <code>:srewind / :sfirst</code>	35. <code>:sbrewind / :sbfirst</code>	split + to first arg/buf
26. <code>:slast</code>	36. <code>:sblast</code>	split + to last arg/buf

27. :sall	37. :sball	edit all args/buffers
	38. :sunhide	edit all loaded buffers
	39. :[N]sbmod [N]	split + to Nth modified buf
40. :args	list of arguments	
41. :buffers	list of buffers	

The meaning of [N] depends on the command:

[N] is the number of buffers to go forward/backward on 2/12/22/32,
3/13/23/33, and 4/14/24/34

[N] is an argument number, defaulting to current argument, for 1 and 21

[N] is a buffer number, defaulting to current buffer, for 11 and 31

[N] is a count for 19 and 39

Note: ":next" is an exception, because it must accept a list of file names for compatibility with Vi.

The argument list and multiple windows

The current position in the argument list can be different for each window. Remember that when doing ":e file", the position in the argument list stays the same, but you are not editing the file at that position. To indicate this, the file message (and the title, if you have one) shows "(file (N) of M)", where "(N)" is the current position in the file list, and "M" the number of files in the file list.

All the entries in the argument list are added to the buffer list. Thus, you can also get to them with the buffer list commands, like ":bnext".

```
:[N]al[l][!] [N]           :al   :all  :sal  :sall
:[N]sal[l][!] [N]
```

Rearrange the screen to open one window for each argument. All other windows are closed. When a count is given, this is the maximum number of windows to open.

With the **:tab** modifier open a tab page for each argument. When there are more arguments than '**tabpagemax**' further ones become split windows in the last tab page.

When the '**hidden**' option is set, all buffers in closed windows become hidden.

When '**hidden**' is not set, and the '**autowrite**' option is set, modified buffers are written. Otherwise, windows that have buffers that are modified are not removed, unless the [!] is given, then they become hidden. But modified buffers are never abandoned, so changes cannot get lost.

[N] is the maximum number of windows to open. '**winheight**' also limits the number of windows opened ('winwidth' if **:vertical** was prepended).

Buf/Win Enter/Leave autocommands are not executed for the new windows here, that's only done when they are really entered.

If autocommands change the window layout while this command is busy an error will be given. **E249**

`:[N]sa[rgument][!] [++opt] [+cmd] [N]` :sa :sargument
Short for `":split | argument [N]"`: split window and go to Nth argument. But when there is no such argument, the window is not split. Also see `++opt` and `+cmd` .

`:[N]sn[ext][!] [++opt] [+cmd] [file ..]` :sn :snext
Short for `":split | [N]next"`: split window and go to Nth next argument. But when there is no next file, the window is not split. Also see `++opt` and `+cmd` .

`:[N]spr[evious][!] [++opt] [+cmd] [N]` :spr :sprevious
`:[N]sN[ext][!] [++opt] [+cmd] [N]` :sN :sNext
Short for `":split | [N]Next"`: split window and go to Nth previous argument. But when there is no previous file, the window is not split. Also see `++opt` and `+cmd` .

:sre :srewind
`:sre[wind][!] [++opt] [+cmd]`
Short for `":split | rewind"`: split window and go to first argument. But when there is no argument list, the window is not split. Also see `++opt` and `+cmd` .

:sfir :sfirst
`:sfir[st] [++opt] [+cmd]`
Same as `":srewind"`.

:sla :slast
`:sla[st][!] [++opt] [+cmd]`
Short for `":split | last"`: split window and go to last argument. But when there is no argument list, the window is not split. Also see `++opt` and `+cmd` .

:dr :drop
`:dr[op] [++opt] [+cmd] {file} ..`
Edit the first `{file}` in a window.
- If the file is already open in a window change to that window.
- If the file is not open in a window edit the file in the current window. If the current buffer can't be `abandoned`, the window is split first.
- Windows that are not in the argument list or are not full width will be closed if possible.
The `argument-list` is set, like with the `:next` command. The purpose of this command is that it can be used from a program that wants Vim to edit another file, e.g., a debugger. When using the `:tab` modifier each argument is opened in a tab page. The last window is used if it's empty.
Also see `++opt` and `+cmd` .

=====
8. Do a command in all buffers or windows list-repeat

:windo
`:[range]windo {cmd}` Execute `{cmd}` in each window or if `[range]` is given

only in windows for which the window number lies in the [range]. It works like doing this:

```
CTRL-W t
:{cmd}
CTRL-W w
:{cmd}
etc.
```

This only operates in the current tab page.

When an error is detected on one window, further windows will not be visited.

The last window (or where an error occurred) becomes the current window.

{cmd} can contain '|' to concatenate several commands.

{cmd} must not open or close windows or reorder them.

Also see :tabdo , :argdo , :bufdo , :cdo , :ldo , :cfdo and :lfdo

:bufdo
:[range]bufdo[!] {cmd} Execute {cmd} in each buffer in the buffer list or if [range] is given only for buffers for which their buffer number is in the [range]. It works like doing this:

```
:bfirst
:{cmd}
:bnext
:{cmd}
etc.
```

When the current file can't be abandoned and the [!] is not present, the command fails.

When an error is detected on one buffer, further buffers will not be visited.

Unlisted buffers are skipped.

The last buffer (or where an error occurred) becomes the current buffer.

{cmd} can contain '|' to concatenate several commands.

{cmd} must not delete buffers or add buffers to the buffer list.

Note: While this command is executing, the Syntax autocommand event is disabled by adding it to 'eventignore'. This considerably speeds up editing each buffer.

Also see :tabdo , :argdo , :windo , :cdo , :ldo , :cfdo and :lfdo

Examples:

```
:windo set nolist foldcolumn=0 | normal! zn
```

This resets the 'list' option and disables folding in all windows.

```
:bufdo set fileencoding= | update
```

This resets the **'fileencoding'** in each buffer and writes it if this changed the buffer. The result is that all buffers will use the **'encoding'** encoding (if conversion succeeds).

-
9. Tag or file name under the cursor window-tag
- :sta** **:stag**
- :sta**[g][!] [tagname] Does **":tag**[!] [tagname]" and splits the window for the found tag. See also **:tag** .
- CTRL-W]** **CTRL-W_]** **CTRL-W_CTRL-]**
CTRL-W CTRL-] Split current window in two. Use identifier under cursor as a tag and jump to it in the new upper window.
In Visual mode uses the Visually selected text as a tag.
Make new window N high.
- CTRL-W g]** **CTRL-W_g]** Split current window in two. Use identifier under cursor as a tag and perform **":tselect**" on it in the new upper window.
In Visual mode uses the Visually selected text as a tag.
Make new window N high.
- CTRL-W g CTRL-]** **CTRL-W_g_CTRL-]** Split current window in two. Use identifier under cursor as a tag and perform **":tjump**" on it in the new upper window.
In Visual mode uses the Visually selected text as a tag.
Make new window N high.
- CTRL-W f** **CTRL-W_f** **CTRL-W_CTRL-F**
CTRL-W CTRL-F Split current window in two. Edit file name under cursor. Like **":split gf"**, but window isn't split if the file does not exist.
Uses the **'path'** variable as a list of directory names where to look for the file. Also the path for current file is used to search for the file name.
If the name is a hypertext link that looks like **"type://machine/path"**, only **"/path"** is used.
If a count is given, the count'th matching file is edited.
- CTRL-W F** **CTRL-W_F** Split current window in two. Edit file name under cursor and jump to the line number following the file name. See **gF** for details on how the line number is obtained.
- CTRL-W gf** **CTRL-W_gf** Open a new tab page and edit the file name under the cursor. Like **"tab split"** and **"gf"**, but the new tab page isn't created if the file does not exist.
- CTRL-W gF** **CTRL-W_gF** Open a new tab page and edit the file name under the cursor and jump to the line number following the file name. Like

"tab split" and "gF", but the new tab page isn't created if the file does not exist.

CTRL-W gt CTRL-W_gt
Go to next tab page, same as `gt`.

CTRL-W gT CTRL-W_gT
Go to previous tab page, same as `gT`.

Also see [CTRL-W_CTRL-I](#) : open window for an included file that includes the keyword under the cursor.

10. The preview window preview-window

The preview window is a special window to show (preview) another file. It is normally a small window used to show an include file or definition of a function.

{not available when compiled without the |+quickfix| feature}

There can be only one preview window (per tab page). It is created with one of the commands below. The '**previewheight**' option can be set to specify the height of the preview window when it's opened. The '**previewwindow**' option is set in the preview window to be able to recognize it. The '**winfixheight**' option is set to have it keep the same height when opening/closing other windows.

preview-popup

Alternatively, a popup window can be used by setting the '**previewpopup**' option. When set, it overrules the '**previewwindow**' and '**previewheight**' settings. The option is a comma-separated list of values:

height	maximum height of the popup
width	maximum width of the popup
highlight	highlight group of the popup (default is Pmenu)

Example:
`:set previewpopup=height:10,width:60`

A few peculiarities:

- If the file is in a buffer already, it will be re-used. This will allow for editing the file while it's visible in the popup window.
- No ATTENTION dialog will be used, since you can't edit the file in the popup window. However, if you later open the same buffer in a normal window, you may not notice it's edited elsewhere. And when then using ":edit" to trigger the ATTENTION and responding "A" for Abort, the preview window will become empty.

:pta :ptag

`:pta[g][!] [tagname]`
Does `":tag[!] [tagname]"` and shows the found tag in a "Preview" window without changing the current buffer or cursor position. If a "Preview" window already exists, it is re-used (like a help window is). If a new one is opened, '**previewheight**' is used for the height of the window. See also `:tag` .
See below for an example. [CursorHold-example](#)

Small difference from `:tag` : When `[tagname]` is equal to the already displayed tag, the position in the matching tag list is not reset. This makes the `CursorHold` example work after a `:ptnext` .

CTRL-W z CTRL-W_z
CTRL-W CTRL-Z CTRL-W_CTRL-Z :pc :pclose
`:pc[lose][!]` Close any "Preview" window currently open. When the `'hidden'` option is set, or when the buffer was changed and the `[!]` is used, the buffer becomes hidden (unless there is another window editing it). The command fails if any "Preview" buffer cannot be closed. See also `:close` .

`:pclose` :pp :ppop
`:[count]pp[op][!]` Does `:[count]pop[!]` in the preview window. See `:pop` and `:ptag` .

CTRL-W } CTRL-W_}
Use identifier under cursor as a tag and perform a `:ptag` on it. Make the new Preview window (if required) N high. If N is not given, `'previewheight'` is used.

CTRL-W g } CTRL-W_g}
Use identifier under cursor as a tag and perform a `:ptjump` on it. Make the new Preview window (if required) N high. If N is not given, `'previewheight'` is used.

`:ped[it][!] [+opt] [+cmd] {file}` :ped :pedit
Edit `{file}` in the preview window. The preview window is opened like with `:ptag` . The current window and cursor position isn't changed. Useful example:
`:pedit +/putc /usr/include/stdio.h`

`:[range]ps[earch][!] [count] [/]pattern[/]` :ps :pssearch
Works like `:ijump` but shows the found match in the preview window. The preview window is opened like with `:ptag` . The current window and cursor position isn't changed. Useful example:
`:pssearch popen`
Like with the `:ptag` command, you can use this to automatically show information about the word under the cursor. This is less clever than using `:ptag` , but you don't need a tags file and it will also find matches in system include files. Example:
`:au! CursorHold *. [ch] ++nested exe "silent! pssearch " .. expand("<word>")`
Warning: This can be slow.

Example CursorHold-example

`:au! CursorHold *. [ch] ++nested exe "silent! ptag " .. expand("<word>")`

This will cause a ":ptag" to be executed for the keyword under the cursor, when the cursor hasn't moved for the time set with 'updatetime'. The "nested" makes other autocommands be executed, so that syntax highlighting works in the preview window. The "silent!" avoids an error message when the tag could not be found. Also see [CursorHold](#) . To disable this again:

```
:au! CursorHold
```

A nice addition is to highlight the found tag, avoid the ":ptag" when there is no word under the cursor, and a few other things:

```
:au! CursorHold *. [ch] ++nested call PreviewWord()
:func PreviewWord()
:  if &previewwindow          " don't do this in the preview window
:    return
:  endif
:  let w = expand("<word>")      " get the word under cursor
:  if w =~ '\a'                " if the word contains a letter
:
:    " Delete any existing highlight before showing another tag
:    silent! wincmd P          " jump to preview window
:    if &previewwindow        " if we really get there...
:      match none             " delete existing highlight
:      wincmd p               " back to old window
:    endif
:
:    " Try displaying a matching tag for the word under the cursor
:    try
:      exe "ptag " .. w
:    catch
:      return
:    endtry
:
:    silent! wincmd P          " jump to preview window
:    if &previewwindow        " if we really get there...
:      if has("folding")
:        silent! .foldopen    " don't want a closed fold
:      endif
:      call search("$", "b")    " to end of previous line
:      let w = substitute(w, '\\', '\\\\', "")
:      call search('\<V' .. w .. '\>') " position cursor on match
:      " Add a match highlight to the word at this position
:      hi previewWord term=bold ctermbg=green guibg=green
:      exe 'match previewWord "\%" .. line(".") .. 'l\%' .. col(".") .. 'c\k*'
:      wincmd p               " back to old window
:    endif
:  endif
:endifun
```

11. Using hidden buffers

buffer-hidden

A hidden buffer is not displayed in a window, but is still loaded into memory. This makes it possible to jump from file to file, without the need to read or

write the file every time you get another buffer in a window.

`:buffer-!`

If the option **'hidden'** ('hid') is set, abandoned buffers are kept for all commands that start editing another file: `":edit"`, `":next"`, `":tag"`, etc. The commands that move through the buffer list sometimes make the current buffer hidden although the **'hidden'** option is not set. This happens when a buffer is modified, but is forced (with `"!'`) to be removed from a window, and **'autowrite'** is off or the buffer can't be written.

You can make a hidden buffer not hidden by starting to edit it with any command, or by deleting it with the `":bdelete"` command.

The **'hidden'** is global, it is used for all buffers. The **'bufhidden'** option can be used to make an exception for a specific buffer. It can take these values:

<code><empty></code>	Use the value of 'hidden' .
<code>hide</code>	Hide this buffer, also when 'hidden' is not set.
<code>unload</code>	Don't hide but unload this buffer, also when 'hidden' is set.
<code>delete</code>	Delete the buffer.

`hidden-quit`

When you try to quit Vim while there is a hidden, modified buffer, you will get an error message and Vim will make that buffer the current buffer. You can then decide to write this buffer (`":wq"`) or quit without writing (`":q!"`). Be careful: there may be more hidden, modified buffers!

A buffer can also be unlisted. This means it exists, but it is not in the list of buffers. `unlisted-buffer`

```
:files[!] [flags]           :files
:buffers[!] [flags]       :buffers  :ls
:ls[!] [flags]
```

Show all buffers. Example:

```
1 #h  "/test/text"         line 1
2u    "asdf"               line 0
3 %a + "version.c"        line 1
```

When the `[!]` is included the list will show unlisted buffers (the term "unlisted" is a bit confusing then...).

Each buffer has a unique number. That number will not change, thus you can always go to a specific buffer with `":buffer N"` or `"N CTRL-^"`, where N is the buffer number.

For the file name these special values are used:

<code>[Prompt]</code>	<code>prompt-buffer</code>
<code>[Popup]</code>	buffer of a <code>popup-window</code>
<code>[Scratch]</code>	'buftype' is "nofile"
<code>[No Name]</code>	no file name specified

For a `terminal-window` buffer the status is used.

Indicators (chars in the same column are mutually exclusive):

u an unlisted buffer (only displayed when [!] is used)
unlisted-buffer
% the buffer in the current window
the alternate buffer for ":e #" and CTRL-^
a an active buffer: it is loaded and visible
h a hidden buffer: It is loaded, but currently not
displayed in a window hidden-buffer
- a buffer with 'modifiable' off
= a readonly buffer
R a terminal buffer with a running job
F a terminal buffer with a finished job
? a terminal buffer without a job: `:terminal NONE`
+ a modified buffer
x a buffer with read errors

[flags] can be a combination of the following characters, which restrict the buffers to be listed:

+ modified buffers
- buffers with 'modifiable' off
= readonly buffers
a active buffers
u unlisted buffers (overrides the "!")
h hidden buffers
x buffers with a read error
% current buffer
alternate buffer
R terminal buffers with a running job
F terminal buffers with a finished job
? terminal buffers without a job: `:terminal NONE`
t show time last used and sort buffers

Combining flags means they are "and"ed together, e.g.:

h+ hidden buffers which are modified
a+ active buffers which are modified

When using :filter the pattern is matched against the displayed buffer name, e.g.:

```
filter /\.vim/ ls
```

```
:bad :badd
```

```
:bad[d] [+lnum] {fname}
```

Add file name {fname} to the buffer list, without loading it, if it wasn't listed yet. If the buffer was previously deleted, not wiped, it will be made listed again. If "lnum" is specified, the cursor will be positioned at that line when the buffer is first entered. Note that other commands after the + will be ignored.

```
:balt
```

```
:balt [+lnum] {fname}
```

Like `:badd` and also set the alternate file for the current window to {fname}.


```

:[N]bd[elete][!]                               :bd   :bdel   :bdelete  E516
:bd[elete][!] [N]
    Unload buffer [N] (default: current buffer) and delete it from
    the buffer list. If the buffer was changed, this fails,
    unless when [!] is specified, in which case changes are lost.
    The file remains unaffected. Any windows for this buffer are
    closed. If buffer [N] is the current buffer, another buffer
    will be displayed instead. This is the most recent entry in
    the jump list that points into a loaded buffer.
    Actually, the buffer isn't completely deleted, it is removed
    from the buffer list unlisted-buffer and option values,
    variables and mappings/abbreviations for the buffer are
    cleared. Examples:
        :.,$-bdelete    " delete buffers from the current one to
                        " last but one
        :%bdelete       " delete all buffers

:bdelete[!] {bufname}                           E93   E94
    Like ":bdelete[!] [N]", but buffer given by name, see
    {bufname} .

:bdelete[!] N1 N2 ...
    Do ":bdelete[!]" for buffer N1, N2, etc. The arguments can be
    buffer numbers or buffer names (but not buffer names that are
    a number). Insert a backslash before a space in a buffer
    name.

:N,Mbdelete[!] Do ":bdelete[!]" for all buffers in the range N to M
    inclusive .

:[N]bw[ipeout][!]                               :bw   :bwipe   :bwipeout E517
:bw[ipeout][!] {bufname}
:N,Mbw[ipeout][!]
:bw[ipeout][!] N1 N2 ...
    Like :bdelete , but really delete the buffer. Everything
    related to the buffer is lost. All marks in this buffer
    become invalid, option settings are lost, etc. Don't use this
    unless you know what you are doing. Examples:
        :.,$bwipeout    " wipe out all buffers after the current
                        " one
        :%bwipeout      " wipe out all buffers

:[N]bun[load][!]                               :bun   :bunload  E515
:bun[load][!] [N]
    Unload buffer [N] (default: current buffer). The memory
    allocated for this buffer will be freed. The buffer remains
    in the buffer list.
    If the buffer was changed, this fails, unless when [!] is
    specified, in which case the changes are lost.
    Any windows for this buffer are closed. If buffer [N] is the
    current buffer, another buffer will be displayed instead.
    This is the most recent entry in the jump list that points

```

into a loaded buffer.

`:bunload[!] {bufname}`

Like `":bunload[!] [N]"`, but buffer given by name.
Also see `{bufname}` .

`:N,Mbunload[!]` Do `":bunload[!]"` for all buffers in the range N to M
`inclusive` .

`:bunload[!] N1 N2 ...`

Do `":bunload[!]"` for buffer N1, N2, etc. The arguments can be
buffer numbers or buffer names (but not buffer names that are
a number). Insert a backslash before a space in a buffer
name.

`:[N]b[uffer][!] [+cmd] [N] :b :bu :buf :buffer E86`

Edit buffer [N] from the buffer list. If [N] is not given,
the current buffer remains being edited. See `:buffer-!` for
[!]. This will also edit a buffer that is not in the buffer
list, without setting the `'buflisted'` flag.
The notation with single quotes does not work here,
``:buf 12'345'`` uses 12'345 as a buffer name.
Also see `+cmd` .

`:[N]b[uffer][!] [+cmd] {bufname} {bufname}`

Edit buffer for `{bufname}` from the buffer list. A partial
name also works, so long as it is unique in the list of
buffers.
Note that a buffer whose name is a number cannot be referenced
by that name; use the buffer number instead.
Insert a backslash before a space in a buffer name.
See `:buffer-!` for [!].
This will also edit a buffer that is not in the buffer list,
without setting the `'buflisted'` flag.
Also see `+cmd` .

`:[N]sb[uffer] [+cmd] [N] :sb :sbuffer`

Split window and edit buffer [N] from the buffer list. If [N]
is not given, the current buffer is edited. Respects the
"useopen" setting of `'switchbuf'` when splitting. This will
also edit a buffer that is not in the buffer list, without
setting the `'buflisted'` flag.
Also see `+cmd` .

`:[N]sb[uffer] [+cmd] {bufname}`

Split window and edit buffer for `{bufname}` from the buffer
list. This will also edit a buffer that is not in the buffer
list, without setting the `'buflisted'` flag.

Note: If what you want to do is split the buffer, make a copy
under another name, you can do it this way:

`:w foobar | sp #`

Also see `+cmd` .

`:[N]bn[ext][!] [+cmd] [N] :bn :bnext E87`

Go to [N]th next buffer in buffer list. [N] defaults to one. Wraps around the end of the buffer list.

See `:buffer-!` for [!].

Also see `+cmd` .

If you are in a help buffer, this takes you to the next help buffer (if there is one). Similarly, if you are in a normal (non-help) buffer, this takes you to the next normal buffer. This is so that if you have invoked help, it doesn't get in the way when you're browsing code/text buffers. The next three commands also work like this.

`:sbn` `:sbnext`

`:[N]sbn[ext] [+cmd] [N]`

Split window and go to [N]th next buffer in buffer list.

Wraps around the end of the buffer list. Uses `'switchbuf'`

Also see `+cmd` .

`:[N]bN[ext][!] [+cmd] [N]`

`:bN` `:bNext` `:bp` `:bprevious` E88

`:[N]bp[revious][!] [+cmd] [N]`

Go to [N]th previous buffer in buffer list. [N] defaults to one. Wraps around the start of the buffer list.

See `:buffer-!` for [!] and `'switchbuf'`.

Also see `+cmd` .

`:[N]sbN[ext] [+cmd] [N]`

`:sbN` `:sbNext` `:sbp` `:sbprevious`

`:[N]sbp[revious] [+cmd] [N]`

Split window and go to [N]th previous buffer in buffer list.

Wraps around the start of the buffer list.

Uses `'switchbuf'`.

Also see `+cmd` .

`:br[ewind][!] [+cmd]`

`:br` `:bre` `:brewind`

Go to first buffer in buffer list. If the buffer list is empty, go to the first unlisted buffer.

See `:buffer-!` for [!].

`:bf[irst] [+cmd]`

`:bf` `:bfirst`

Same as `:brewind` .

Also see `+cmd` .

`:sbr[ewind] [+cmd]`

`:sbr` `:sbrewind`

Split window and go to first buffer in buffer list. If the buffer list is empty, go to the first unlisted buffer.

Respects the `'switchbuf'` option.

Also see `+cmd` .

`:sbf[irst] [+cmd]`

`:sbf` `:sbfirst`

Same as `":sbrewind"`.

`:bl[ast][!] [+cmd]`

`:bl` `:blast`

Go to last buffer in buffer list. If the buffer list is empty, go to the last unlisted buffer.

See `:buffer-!` for [!].

`:sbl[ast] [+cmd]` `:sbl` `:sblast`
 Split window and go to last buffer in buffer list. If the buffer list is empty, go to the last unlisted buffer. Respects `'switchbuf'` option.

`:[N]bm[odified][!] [+cmd] [N]` `:bm` `:bmodified` `E84`
 Go to [N]th next modified buffer. **Note:** this command also finds unlisted buffers. If there is no modified buffer the command fails.

`:[N]sbm[odified] [+cmd] [N]` `:sbm` `:sbmodified`
 Split window and go to [N]th next modified buffer. Respects `'switchbuf'` option. **Note:** this command also finds buffers not in the buffer list.

`:[N]unh[ide] [N]` `:unh` `:unhide` `:sun` `:sunhide`
`:[N]sun[hide] [N]`
 Rearrange the screen to open one window for each loaded buffer in the buffer list. When a count is given, this is the maximum number of windows to open.

`:[N]ba[ll] [N]` `:ba` `:ball` `:sba` `:sball`
`:[N]sba[ll] [N]` Rearrange the screen to open one window for each buffer in the buffer list. When a count is given, this is the maximum number of windows to open. `'winheight'` also limits the number of windows opened (`'winwidth'` if `:vertical` was prepended). Buf/Win Enter/Leave autocommands are not executed for the new windows here, that's only done when they are really entered. When the `:tab` modifier is used new windows are opened in a new tab, up to `'tabpagemax'`.

Note: All the commands above that start editing another buffer, keep the `'readonly'` flag as it was. This differs from the `":edit"` command, which sets the `'readonly'` flag each time the file is read.

12. Special kinds of buffers special-buffers

Instead of containing the text of a file, buffers can also be used for other purposes. A few options can be set to change the behavior of a buffer:

<code>'bufhidden'</code>	what happens when the buffer is no longer displayed in a window.
<code>'buftype'</code>	what kind of a buffer this is
<code>'swapfile'</code>	whether the buffer will have a swap file
<code>'buflisted'</code>	buffer shows up in the buffer list

A few useful kinds of a buffer:

`quickfix` Used to contain the error list or the location list. See `:cwindow` and `:lwindow`. This command sets the `'buftype'` option to `"quickfix"`. You are not supposed to change this! `'swapfile'` is off.

`help` Contains a help file. Will only be created with the `:help`

command. The flag that indicates a help buffer is internal and can't be changed. The **'buflisted'** option will be reset for a help buffer.

terminal A terminal window buffer, see [terminal](#) . The contents cannot be read or changed until the job ends.

directory Displays directory contents. Can be used by a file explorer plugin. The buffer is created with these settings:

```
:setlocal buftype=nowrite  
:setlocal bufhidden=delete  
:setlocal noswapfile
```

The buffer name is the name of the directory and is adjusted when using the `:cd` command.

scratch Contains text that can be discarded at any time. It is kept when closing the window, it must be deleted explicitly.

Settings:

```
:setlocal buftype=nofile  
:setlocal bufhidden=hide  
:setlocal noswapfile
```

The buffer name can be used to identify the buffer, if you give it a meaningful name.

unlisted The buffer is not in the buffer list. It is not used for normal editing, but to show a help file, remember a file name or marks. The `:bdelete` command will also set this option, thus it doesn't completely delete the buffer. Settings:

```
:setlocal nobuflisted
```


Editing with windows in multiple tab pages. tab-page tabpage

The commands which have been added to use multiple tab pages are explained here. Additionally, there are explanations for commands that work differently when used in combination with more than one tab page.

- 1. Introduction tab-page-intro
- 2. Commands tab-page-commands
- 3. Other items tab-page-other
- 4. Setting **'tabline'** setting-tabline
- 5. Setting **'guitablabel'** setting-guitablabel

{not able to use multiple tab pages when the `+windows` feature was disabled at compile time}

1. Introduction tab-page-intro

A tab page holds one or more windows. You can easily switch between tab pages, so that you have several collections of windows to work on different things.

Usually you will see a list of labels at the top of the Vim window, one for each tab page. With the mouse you can click on the label to jump to that tab page. There are other ways to move between tab pages, see below.

Most commands work only in the current tab page. That includes the `CTRL-W` commands, `:windo`, `:all` and `:ball` (when not using the `:tab` modifier). The commands that are aware of other tab pages than the current one are mentioned below.

Tabs are also a nice way to edit a buffer temporarily without changing the current window layout. Open a new tab page, do whatever you want to do and close the tab page.

2. Commands tab-page-commands

OPENING A NEW TAB PAGE:

When starting Vim `"vim -p filename ..."` opens each file argument in a separate tab page (up to **'tabpagemax'**). See `-p`

A double click with the mouse in the non-GUI tab pages line opens a new, empty tab page. It is placed left of the position of the click. The first click may select another tab page first, causing an extra screen update.

This also works in a few GUI versions, esp. Win32 and Motif. But only when

clicking right of the labels.

In the GUI tab pages line you can use the right mouse button to open menu.
`tabline-menu` .

For the related autocommands see `tabnew-autocmd` .

```
:[count]tabe[dit]                :tabe   :tabedit   :tabnew
:[count]tabnew
```

Open a new tab page with an empty window, after the current tab page. If [count] is given the new tab page appears after the tab page [count] otherwise the new tab page will appear after the current one.

```
    :tabnew      " opens tabpage after the current one
    :.tabnew     " as above
    :+tabnew     " opens tabpage after the next tab page
                  " note: it is one further than :tabnew
    :-tabnew     " opens tabpage before the current one
    :0tabnew     " opens tabpage before the first one
    :$tabnew     " opens tabpage after the last one
```

```
:[count]tabe[dit] [++opt] [+cmd] {file}
:[count]tabnew [++opt] [+cmd] {file}
    Open a new tab page and edit {file}, like with :edit .
    For [count] see :tabnew above.
```

```
:[count]tabf[ind] [++opt] [+cmd] {file}                :tabf   :tabfind
    Open a new tab page and edit {file} in 'path', like with
    :find . For [count] see :tabnew above.
```

```
:[count]tab {cmd}                :tab
    Execute {cmd} and when it opens a new window open a new tab
    page instead. Doesn't work for :diffsplit , :diffpatch ,
    :execute and :normal .
    If [count] is given the new tab page appears after the tab
    page [count] otherwise the new tab page will appear after the
    current one.
```

Examples:

```
    :tab split      " opens current buffer in new tab page
    :tab help gt    " opens tab page with help for "gt"
    :.tab help gt   " as above
    :+tab help      " opens tab page with help after the next
                  " tab page
    :-tab help     " opens tab page with help before the
                  " current one
    :0tab help     " opens tab page with help before the
                  " first one
    :$tab help     " opens tab page with help after the last
                  " one
```

CTRL-W gf Open a new tab page and edit the file name under the cursor.
See `CTRL-W_gf` .

CTRL-W gF Open a new tab page and edit the file name under the cursor

and jump to the line number following the file name.
See `CTRL-W_gF` .

CLOSING A TAB PAGE:

Closing the last window of a tab page closes the tab page too, unless there is only one tab page.

Using the mouse: If the tab page line is displayed you can click in the "X" at the top right to close the current tab page. A custom `'tabline'` may show something else.

```
                                :tabc    :tabclose
:tabc[lose][!]  Close current tab page.
                This command fails when:
                - There is only one tab page on the screen.           E784
                - When 'hidden' is not set, [!] is not used, a buffer has
                  changes, and there is no other window on this buffer.
                Changes to the buffer are not written and won't get lost, so
                this is a "safe" command.
                :tabclose      " close the current tab page

:{count}tabc[lose][!]
:tabc[lose][!] {count}
                Close tab page {count}. Fails in the same way as `:tabclose`
                above.
                :-tabclose    " close the previous tab page
                :+tabclose    " close the next tab page
                :1tabclose    " close the first tab page
                :$tabclose    " close the last tab page
                :tabclose -2  " close the 2nd previous tab page
                :tabclose +   " close the next tab page
                :tabclose 3   " close the third tab page
                :tabclose $   " close the last tab page
                :tabclose #   " close the last accessed tab page
```

When a tab is closed the next tab page will become the current one.

```
                                :tabo    :tabonly
:tabo[nly][!]  Close all other tab pages.
                When the 'hidden' option is set, all buffers in closed windows
                become hidden.
                When 'hidden' is not set, and the 'autowrite' option is set,
                modified buffers are written. Otherwise, windows that have
                buffers that are modified are not removed, unless the [!] is
                given, then they become hidden. But modified buffers are
                never abandoned, so changes cannot get lost.
                :tabonly      " close all tab pages except the current
                " one

:{count}tabo[nly][!]
:tabo[nly][!] {count}
                Close all tab pages except {count} one.
                :.tabonly    " as above
```

```

:-tabonly      " close all tab pages except the previous
               " one
:+tabonly      " close all tab pages except the next one
:1tabonly      " close all tab pages except the first one
:$tabonly      " close all tab pages except the last one
:tabonly -     " close all tab pages except the previous
               " one
:tabonly +2    " close all tab pages except the two next
               " one
:tabonly 1     " close all tab pages except the first one
:tabonly $     " close all tab pages except the last one
:tabonly #     " close all tab pages except the last
               " accessed one

```

SWITCHING TO ANOTHER TAB PAGE:

Using the mouse: If the tab page line is displayed you can click in a tab page label to switch to that tab page. Click where there is no label to go to the next tab page. `'tabline'`

```

:tabn[ext]          :tabn      :tabnext      gt
<C-PageDown>      CTRL-<PageDown>  <C-PageDown>
gt                 i_CTRL-<PageDown>  i_<C-PageDown>
                  Go to the next tab page.  Wraps around from the last to the
                  first one.

```

```

:{count}tabn[ext]
:tabn[ext] {count}
                  Go to tab page {count}.  The first tab page has number one.
:-tabnext        " go to the previous tab page
:+tabnext        " go to the next tab page
:+2tabnext       " go to the two next tab page
:1tabnext        " go to the first tab page
:$tabnext        " go to the last tab page
:tabnext $       " as above
:tabnext #       " go to the last accessed tab page
:tabnext -       " go to the previous tab page
:tabnext -1      " as above
:tabnext +       " go to the next tab page
:tabnext +1      " as above

```

```

{count}<C-PageDown>
{count}gt        Go to tab page {count}.  The first tab page has number one.

```

```

:tabp[revious]    :tabp      :tabprevious  gT  :tabN
:tabN[ext]        :tabNext    CTRL-<PageUp>
<C-PageUp>      <C-PageUp>  i_CTRL-<PageUp>  i_<C-PageUp>
gT               Go to the previous tab page.  Wraps around from the first one
                  to the last one.

```

```

:tabp[revious] {count}
:tabN[ext] {count}

```

`{count}<C-PageUp>`
`{count}gT` Go `{count}` tab pages back. Wraps around from the first one to the last one. Note that the use of `{count}` is different from `:tabnext`, where it is used as the tab page number.

`:tabr[ewind]` `:tabfir` `:tabfirst` `:tabr` `:tabrewind`
`:tabfir[st]` Go to the first tab page.

`:tabl[ast]` `:tabl` `:tablast`
Go to the last tab page.

`g<Tab>` `g<Tab>` `CTRL-W_g<Tab>` `<C-Tab>`
Go to the last accessed tab page.

Other commands:

`:tabs` `:tabs`
List the tab pages and the windows they contain.
Shows a ">" for the current window.
Shows a "+" for modified buffers.
For example:

```
Tab page 1
+ tabpage.txt
  ex_docmd.c
Tab page 2
> main.c
```

REORDERING TAB PAGES:

`:tabm[ove] [N]` `:tabm` `:tabmove`
`:[N]tabm[ove]`

Move the current tab page to after tab page N. Use zero to make the current tab page the first one. N is counted before the move, thus if the second tab is the current one, ``:tabmove 1`` and ``:tabmove 2`` have no effect. Without N the tab page is made the last one.

```
:.tabmove " do nothing
:-tabmove " move the tab page to the left
:+tabmove " move the tab page to the right
:@tabmove " move the tab page to the beginning of the tab
" list
:tabmove 0 " as above
:tabmove " move the tab page to the last
:$tabmove " as above
:tabmove $ " as above
:tabmove # " move the tab page after the last accessed
" tab page
```

`:tabm[ove] +[N]`
`:tabm[ove] -[N]`

Move the current tab page N places to the right (with +) or to the left (with -).

```
:tabmove - " move the tab page to the left
:tabmove -1 " as above
```

```
:tabmove + " move the tab page to the right
:tabmove +1 " as above
```

Note that although it is possible to move a tab behind the N-th one by using :Ntabmove. And move it by N places by using :+Ntabmove. For clarification what +N means in this context see [\[range\]](#) .

LOOPING OVER TAB PAGES:

```
:[range]tabd[o] {cmd} :tabd :tabdo
Execute {cmd} in each tab page or if [range] is given only in
tab pages which tab page number is in the [range]. It works
like doing this:
    :tabfirst
    :{cmd}
    :tabnext
    :{cmd}
    etc.
```

This only operates in the current window of each tab page. When an error is detected on one tab page, further tab pages will not be visited.

The last tab page (or where an error occurred) becomes the current tab page.

{cmd} can contain '|' to concatenate several commands.

{cmd} must not open or close tab pages or reorder them.

Also see [:windo](#) , [:argdo](#) , [:bufdo](#) , [:cdo](#) , [:ldo](#) , [:cfdo](#) and [:lfdo](#)

3. Other items

[tab-page-other](#)

[tabline-menu](#)

The GUI tab pages line has a popup menu. It is accessed with a right click. The entries are:

Close	Close the tab page under the mouse pointer. The current one if there is no label under the mouse pointer.
New Tab	Open a tab page, editing an empty buffer. It appears to the left of the mouse pointer.
Open Tab...	Like "New Tab" and additionally use a file selector to select a file to edit.

Diff mode works per tab page. You can see the diffs between several files within one tab page. Other tab pages can show differences between other files.

Variables local to a tab page start with "t:". [tabpage-variable](#)

Currently there is only one option local to a tab page: **'cmdheight'**.

[tabnew-autocmd](#)

The TabLeave and TabEnter autocommand events can be used to do something when switching from one tab page to another. The exact order depends on what you are doing. When creating a new tab page this works as if you create a new window on the same buffer and then edit another buffer. Thus ":tabnew" triggers:

WinLeave	leave current window
TabLeave	leave current tab page
WinEnter	enter window in new tab page
TabEnter	enter new tab page
BufLeave	leave current buffer
BufEnter	enter new empty buffer

When switching to another tab page the order is:

```
BufLeave
WinLeave
TabLeave
TabEnter
WinEnter
BufEnter
```

4. Setting 'tabline'

setting-tabline

The 'tabline' option specifies what the line with tab pages labels looks like. It is only used when there is no GUI tab line.

You can use the 'showtabline' option to specify when you want the line with tab page labels to appear: never, when there is more than one tab page or always.

The highlighting of the tab pages line is set with the groups TabLine TabLineSel and TabLineFill. hl-TabLine hl-TabLineSel hl-TabLineFill

A "+" will be shown for a tab page that has a modified window. The number of windows in a tabpage is also shown. Thus "3+" means three windows and one of them has a modified buffer.

The 'tabline' option allows you to define your preferred way to tab pages labels. This isn't easy, thus an example will be given here.

For basics see the 'statusline' option. The same items can be used in the 'tabline' option. Additionally, the `tabpagebuflist()`, `tabpagenr()` and `tabpagewinnr()` functions are useful.

Since the number of tab labels will vary, you need to use an expression for the whole option. Something like:

```
:set tabline=%!MyTabLine()
```

Then define the MyTabLine() function to list all the tab pages labels. A convenient method is to split it in two parts: First go over all the tab pages and define labels for them. Then get the label for each tab page.

```
function MyTabLine()
  let s = ''
```

```

for i in range(tabpagenr('$'))
  " select the highlighting
  if i + 1 == tabpagenr()
    let s .= '%#TabLineSel#'
  else
    let s .= '%#TabLine#'
  endif

  " set the tab page number (for mouse clicks)
  let s .= '%' .. (i + 1) .. 'T'

  " the label is made by MyTabLabel()
  let s .= ' %{MyTabLabel(' .. (i + 1) .. ')} '
endfor

" after the last tab fill with TabLineFill and reset tab page nr
let s .= '%#TabLineFill#%T'

" right-align the label to close the current tab page
if tabpagenr('$') > 1
  let s .= '%=%#TabLine#%999Xclose'
endif

return s
endfunction

```

Now the `MyTabLabel()` function is called for each tab page to get its label.

```

function MyTabLabel(n)
  let buflist = tabpagebuflist(a:n)
  let winnr = tabpagewinnr(a:n)
  return bufname(buflist[winnr - 1])
endfunction

```

This is just a simplistic example that results in a tab pages line that resembles the default, but without adding a + for a modified buffer or truncating the names. You will want to reduce the width of labels in a clever way when there is not enough room. Check the **'columns'** option for the space available.

5. Setting **'guitablabel'**

[setting-guitablabel](#)

When the GUI tab pages line is displayed, **'guitablabel'** can be used to specify the label to display for each tab page. Unlike **'tabline'**, which specifies the whole tab pages line at once, **'guitablabel'** is used for each label separately.

'guitabtooltip' is very similar and is used for the tooltip of the same label. This only appears when the mouse pointer hovers over the label, thus it usually is longer. Only supported on some systems though.

See the **'statusline'** option for the format of the value.

The "%N" item can be used for the current tab page number. The `v:lnum` variable is also set to this number when the option is evaluated. The items that use a file name refer to the current window of the tab page.

Note that syntax highlighting is not used for the option. The %T and %X items are also ignored.

A simple example that puts the tab page number and the buffer name in the label:

```
:set guitablabel=%N\ %f
```

An example that resembles the default **'guitablabel'**: Show the number of windows in the tab page and a '+' if there is a modified buffer:

```
function GuiTabLabel()
  let label = ''
  let bufnrlist = tabpagebuflist(v:lnum)

  " Add '+' if one of the buffers in the tab page is modified
  for bufnr in bufnrlist
    if getbufvar(bufnr, "&modified")
      let label = '+'
      break
    endif
  endfor

  " Append the number of windows in the tab page if more than one
  let wincount = tabpagewinnr(v:lnum, '$')
  if wincount > 1
    let label .= wincount
  endif
  if label != ''
    let label .= ' '
  endif

  " Append the buffer name
  return label .. bufname(bufnrlist[tabpagewinnr(v:lnum) - 1])
endfunction

set guitablabel=%{GuiTabLabel()}
```

Note that the function must be defined before setting the option, otherwise you get an error message for the function not being known.

If you want to fall back to the default label, return an empty string.

If you want to show something specific for a tab page, you might want to use a tab page local variable. `t:var`

Spell checking

spell

- | | |
|------------------------------|-------------------|
| 1. Quick start | spell-quickstart |
| 2. Remarks on spell checking | spell-remarks |
| 3. Generating a spell file | spell-mkspell |
| 4. Spell file format | spell-file-format |

{not available when the |+syntax| feature has been disabled at compile time}

Note: There also is a vimspell plugin. If you have it you can do ":help vimspell" to find about it. But you will probably want to get rid of the plugin and use the '**spell**' option instead, it works better.

-
- | | | |
|----------------|------------------|------|
| 1. Quick start | spell-quickstart | E756 |
|----------------|------------------|------|

This command switches on spell checking:

```
:setlocal spell spelllang=en_us
```

This switches on the '**spell**' option and specifies to check for US English.

The words that are not recognized are highlighted with one of these:

SpellBad	word not recognized	hl-SpellBad
SpellCap	word not capitalised	hl-SpellCap
SpellRare	rare word	hl-SpellRare
SpellLocal	wrong spelling for selected region	hl-SpellLocal

Vim only checks words for spelling, there is no grammar check.

If the '**mousemodel**' option is set to "popup" and the cursor is on a badly spelled word or it is "popup_setpos" and the mouse pointer is on a badly spelled word, then the popup menu will contain a submenu to replace the bad word. **Note:** this slows down the appearance of the popup menu. **Note** for GTK: don't release the right mouse button until the menu appears, otherwise it won't work.

To search for the next misspelled word:

]	Move to next misspelled word after the cursor. A count before the command can be used to repeat. ' wrapsan ' applies.
---	--

[Like "]" but search backwards, find the misspelled word before the cursor. Doesn't recognize words split over two lines, thus may stop at words that are
---	--

not highlighted as bad. Does not stop at word with missing capital at the start of a line.

]`S` Like "`]`s" but only stop at bad words, not at rare words or words for another region.

[`S` Like "`]`S" but search backwards.

To add words to your own word list:

`zg` Add word under the cursor as a good word to the first name in '`spellfile`'. A count may precede the command to indicate the entry in '`spellfile`' to be used. A count of two uses the second entry.

In Visual mode the selected characters are added as a word (including white space!).

When the cursor is on text that is marked as badly spelled then the marked text is used.

Otherwise the word under the cursor, separated by non-word characters, is used.

If the word is explicitly marked as bad word in another spell file the result is unpredictable.

`zG` Like "`zg`" but add the word to the internal word list `internal-wordlist`.

`zw` Like "`zg`" but mark the word as a wrong (bad) word. If the word already appears in '`spellfile`' it is turned into a comment line. See `spellfile-cleanup` for getting rid of those.

`zW` Like "`zw`" but add the word to the internal word list `internal-wordlist`.

`zug` Undo `zw` and `zg`, remove the word from the entry in '`spellfile`'. Count used as with `zg`.

`zuW` Undo `zW` and `zG`, remove the word from the internal word list. Count used as with `zg`.

`:[count]spe[llgood] {word}` `:spe` `:spellgood` E1280
Add `{word}` as a good word to '`spellfile`', like with

`zg` . Without count the first name is used, with a count of two the second entry, etc.

`:spe[llgood]! {word}` Add `{word}` as a good word to the internal word list, like with `zG` .

`:spellw` `:spellwrong`
`:[count]spellw[rong] {word}`
Add `{word}` as a wrong (bad) word to `'spellfile'`, as with `zw` . Without count the first name is used, with a count of two the second entry, etc.

`:spellw[rong]! {word}` Add `{word}` as a wrong (bad) word to the internal word list, like with `zW` .

`:spellra` `:spellrare`
`:[count]spellra[re] {word}`
Add `{word}` as a rare word to `'spellfile'`, similar to `zw` . Without count the first name is used, with a count of two the second entry, etc.

There are no normal mode commands to mark words as rare as this is a fairly uncommon command and all intuitive commands for this are already taken. If you want you can add mappings with e.g.:

```
nnoremap z? :exe ':spellrare ' .. expand('<cWORD>')<CR>
nnoremap z/ :exe ':spellrare! ' .. expand('<cWORD>')<CR>
:spellundo , zuw , or zuW can be used to undo this.
```

`:spellra[re]! {word}` Add `{word}` as a rare word to the internal word list, similar to `zW` .

`:spellu` `:spellundo`
`:[count]spellu[ndo] {word}`
Like `zuw` . `[count]` used as with `:spellgood` .

`:spellu[ndo]! {word}` Like `zuW` . `[count]` used as with `:spellgood` .

After adding a word to `'spellfile'` with the above commands its associated `".spl"` file will automatically be updated and reloaded. If you change `'spellfile'` manually you need to use the `:mkspell` command. This sequence of commands mostly works well:

```
:edit <file in 'spellfile'>
(make changes to the spell file)
:mkspell! %
```

More details about the `'spellfile'` format below [spell-wordlist-format](#) .

internal-wordlist

The internal word list is used for all buffers where `'spell'` is set. It is not stored, it is lost when you exit Vim. It is also cleared when `'encoding'` is set.

Finding suggestions for bad words:

`z=` For the word under/after the cursor suggest correctly spelled words. This also works to find alternatives for a word that is not highlighted as a bad word, e.g., when the word after it is bad. In Visual mode the highlighted text is taken as the word to be replaced. The results are sorted on similarity to the word being replaced. This may take a long time. Hit **CTRL-C** when you get bored.

If the command is used without a count the alternatives are listed and you can enter the number of your choice or press `<Enter>` if you don't want to replace. You can also use the mouse to click on your choice (only works if the mouse can be used in Normal mode and when there are no line wraps). Click on the first line (the header) to cancel.

The suggestions listed normally replace a highlighted bad word. Sometimes they include other text, in that case the replaced text is also listed after a "<".

If a count is used that suggestion is used, without prompting. For example, "1z=" always takes the first suggestion.

If **'verbose'** is non-zero a score will be displayed with the suggestions to indicate the likeliness to the badly spelled word (the higher the score the more different).

When a word was replaced the redo command "." will repeat the word replacement. This works like "ciw", the good word and `<Esc>`. This does NOT work for Thai and other languages without spaces between words.

`:spellr [epall]` `:spellr` `:spellrepall` E752 E753 Repeat the replacement done by `z=` for all matches with the replaced word in the current window.

In Insert mode, when the cursor is after a badly spelled word, you can use **CTRL-X s** to find suggestions. This works like Insert mode completion. Use **CTRL-N** to use the next suggestion, **CTRL-P** to go back. `i_CTRL-X_s`

The **'spellsuggest'** option influences how the list of suggestions is generated and sorted. See **'spellsuggest'** .

The **'spellcapcheck'** option is used to check the first word of a sentence starts with a capital. This doesn't work for the first word in the file. When there is a line break right after a sentence the highlighting of the next line may be postponed. Use **CTRL-L** when needed. Also see `set-spc-auto` for how it can be set automatically when **'spelllang'** is set.

The **'spelloptions'** option has a few more flags that influence the way spell checking works. For example, "camel" splits CamelCased words so that each part of the word is spell-checked separately.

Vim counts the number of times a good word is encountered. This is used to sort the suggestions: words that have been seen before get a small bonus, words that have been seen often get a bigger bonus. The COMMON item in the affix file can be used to define common words, so that this mechanism also works in a new or short file `spell-COMMON` .

2. Remarks on spell checking

`spell-remarks`

PERFORMANCE

Vim does on-the-fly spell checking. To make this work fast the word list is loaded in memory. Thus this uses a lot of memory (1 Mbyte or more). There might also be a noticeable delay when the word list is loaded, which happens when **'spell'** is set and when **'spelllang'** is set while **'spell'** was already set. To minimize the delay each word list is only loaded once, it is not deleted when **'spelllang'** is made empty or **'spell'** is reset. When **'encoding'** is set all the word lists are reloaded, thus you may notice a delay then too.

REGIONS

A word may be spelled differently in various regions. For example, English comes in (at least) these variants:

en	all regions
en_au	Australia
en_ca	Canada
en_gb	Great Britain
en_nz	New Zealand
en_us	USA

Words that are not used in one region but are used in another region are highlighted with SpellLocal `hl-SpellLocal` .

Always use lowercase letters for the language and region names.

When adding a word with `zg` or another command it's always added for all regions. You can change that by manually editing the **'spellfile'**. See `spell-wordlist-format` . Note that the regions as specified in the files in **'spellfile'** are only used when all entries in **'spelllang'** specify the same region (not counting files specified by their .spl name).

`spell-german`

Specific exception: For German these special regions are used:

de	all German words accepted
de_de	old and new spelling
de_19	old spelling
de_20	new spelling

de_at	Austria
de_ch	Switzerland

spell-russian

Specific exception: For Russian these special regions are used:

ru	all Russian words accepted
ru_ru	"IE" letter spelling
ru_yo	"YO" letter spelling

spell-yiddish

Yiddish requires using "utf-8" encoding, because of the special characters used. If you are using latin1 Vim will use transliterated (romanized) Yiddish instead. If you want to use transliterated Yiddish with utf-8 use "yi-tr".

In a table:

'encoding'	'spelllang'	
utf-8	yi	Yiddish
latin1	yi	transliterated Yiddish
utf-8	yi-tr	transliterated Yiddish

spell-cjk

Chinese, Japanese and other East Asian characters are normally marked as errors, because spell checking of these characters is not supported. If **'spelllang'** includes "cjk", these characters are not marked as errors. This is useful when editing text with spell checking while some Asian words are present.

SPELL FILES

spell-load

Vim searches for spell files in the "spell" subdirectory of the directories in **'runtimepath'**. The name is: LL.EEE.spl, where:

LL	the language name
EEE	the value of 'encoding'

The value for "LL" comes from **'spelllang'**, but excludes the region name.

Examples:

'spelllang'	LL
en_us	en
en-rare	en-rare
medical_ca	medical

Only the first file is loaded, the one that is first in **'runtimepath'**. If this succeeds then additionally files with the name LL.EEE.add.spl are loaded. All the ones that are found are used.

If no spell file is found the [SpellFileMissing](#) autocommand event is triggered. This may trigger the [spellfile.vim](#) plugin to offer you downloading the spell file.

Additionally, the files related to the names in **'spellfile'** are loaded. These are the files that [zg](#) and [zw](#) add good and wrong words to.

Exceptions:

- Vim uses "latin1" when **'encoding'** is "iso-8859-15". The euro sign doesn't

matter for spelling.

- When no spell file for **'encoding'** is found "ascii" is tried. This only works for languages where nearly all words are ASCII, such as English. It helps when **'encoding'** is not "latin1", such as iso-8859-2, and English text is being edited. For the ".add" files the same name as the found main spell file is used.

For example, with these values:

```
'runtimepath' is "~/vim,/usr/share/vim82,~/vim/after"  
'encoding'     is "iso-8859-2"  
'spelllang'   is "pl"
```

Vim will look for:

1. ~/.vim/spell/pl.iso-8859-2.spl
2. /usr/share/vim82/spell/pl.iso-8859-2.spl
3. ~/.vim/spell/pl.iso-8859-2.add.spl
4. /usr/share/vim82/spell/pl.iso-8859-2.add.spl
5. ~/.vim/after/spell/pl.iso-8859-2.add.spl

This assumes 1. is not found and 2. is found.

If **'encoding'** is "latin1" Vim will look for:

1. ~/.vim/spell/pl.latin1.spl
2. /usr/share/vim82/spell/pl.latin1.spl
3. ~/.vim/after/spell/pl.latin1.spl
4. ~/.vim/spell/pl.ascii.spl
5. /usr/share/vim82/spell/pl.ascii.spl
6. ~/.vim/after/spell/pl.ascii.spl

This assumes none of them are found (Polish doesn't make sense when leaving out the non-ASCII characters).

Spelling for EBCDIC is currently not supported.

A spell file might not be available in the current **'encoding'**. See [spell-mkspell](#) about how to create a spell file. Converting a spell file with "iconv" will NOT work!

Note: on VMS ".{enc}.spl" is changed to "_{enc}.spl" to avoid trouble with filenames.

[spell-sug-file](#) E781

If there is a file with exactly the same name as the ".spl" file but ending in ".sug", that file will be used for giving better suggestions. It isn't loaded before suggestions are made to reduce memory use.

[E758](#) [E759](#) [E778](#) [E779](#) [E780](#) [E782](#)

When loading a spell file Vim checks that it is properly formatted. If you get an error the file may be truncated, modified or intended for another Vim version.

SPELLFILE CLEANUP

[spellfile-cleanup](#)

The `zw` command turns existing entries in '`spellfile`' into comment lines. This avoids having to write a new file every time, but results in the file only getting longer, never shorter. To clean up the comment lines in all ".add" spell files do this:

```
:runtime spell/cleanadd.vim
```

This deletes all comment lines, except the ones that start with "##". Use "##" lines to add comments that you want to keep.

You can invoke this script as often as you like. A variable is provided to skip updating files that have been changed recently. Set it to the number of seconds that has passed since a file was changed before it will be cleaned. For example, to clean only files that were not changed in the last hour:

```
let g:spell_clean_limit = 60 * 60
```

The default is one second.

WORDS

Vim uses a fixed method to recognize a word. This is independent of '`iskeyword`', so that it also works in help files and for languages that include characters like '-' in '`iskeyword`'. The word characters do depend on '`encoding`'.

The table with word characters is stored in the main .spl file. Therefore it matters what the current locale is when generating it! A .add.spl file does not contain a word table though.

For a word that starts with a digit the digit is ignored, unless the word as a whole is recognized. Thus if "3D" is a word and "D" is not then "3D" is recognized as a word, but if "3D" is not a word then only the "D" is marked as bad. Hex numbers in the form 0x12ab and 0X12AB are recognized.

WORD COMBINATIONS

It is possible to spell-check words that include a space. This is used to recognize words that are invalid when used by themselves, e.g. for "et al.". It can also be used to recognize "the the" and highlight it.

The number of spaces is irrelevant. In most cases a line break may also appear. However, this makes it difficult to find out where to start checking for spelling mistakes. When you make a change to one line and only that line is redrawn Vim won't look in the previous line, thus when "et" is at the end of the previous line "al." will be flagged as an error. And when you type "the<CR>the" the highlighting doesn't appear until the first line is redrawn. Use `CTRL-L` to redraw right away. "[s" will also stop at a word combination with a line break.

When encountering a line break Vim skips characters such as '*', '>' and "'", so that comments in C, shell and Vim code can be spell checked.

SYNTAX HIGHLIGHTING

[spell-syntax](#)

Files that use syntax highlighting can specify where spell checking should be done:

1. everywhere default
2. in specific items use "contains=@Spell"
3. everywhere but specific items use "contains=@NoSpell"

For the second method adding the @NoSpell cluster will disable spell checking again. This can be used, for example, to add @Spell to the comments of a program, and add @NoSpell for items that shouldn't be checked. Also see `:syn-spell` for text that is not in a syntax item.

VIM SCRIPTS

If you want to write a Vim script that does something with spelling, you may find these functions useful:

<code>spellbadword()</code>	find badly spelled word at the cursor
<code>spellsuggest()</code>	get list of spelling suggestions
<code>soundfold()</code>	get the sound-a-like version of a word

SETTING '`spellcapcheck`' AUTOMATICALLY

`set-spc-auto`

After the '`spelllang`' option has been set successfully, Vim will source the files "spell/LANG.vim" in '`runtimepath`'. "LANG" is the value of '`spelllang`' up to the first comma, dot or underscore. This can be used to set options specifically for the language, especially '`spellcapcheck`'.

The distribution includes a few of these files. Use this command to see what they do:

```
:next $VIMRUNTIME/spell/*.vim
```

Note that the default scripts don't set '`spellcapcheck`' if it was changed from the default value. This assumes the user prefers another value then.

DOUBLE SCORING

`spell-double-scoring`

The '`spellsuggest`' option can be used to select "double" scoring. This mechanism is based on the principle that there are two kinds of spelling mistakes:

1. You know how to spell the word, but mistype something. This results in a small editing distance (character swapped/omitted/inserted) and possibly a word that sounds completely different.
2. You don't know how to spell the word and type something that sounds right. The edit distance can be big but the word is similar after sound-folding.

Since scores for these two mistakes will be very different we use a list for each and mix them.

The sound-folding is slow and people that know the language won't make the second kind of mistakes. Therefore **'spellsuggest'** can be set to select the preferred method for scoring the suggestions.

3. Generating a spell file

spell-mkspell

Vim uses a binary file format for spelling. This greatly speeds up loading the word list and keeps it small.

.aff .dic Myspell

You can create a Vim spell file from the **.aff** and **.dic** files that Myspell uses. Myspell is used by OpenOffice.org and Mozilla. The OpenOffice **.oxt** files are zip files which contain the **.aff** and **.dic** files. You should be able to find them here:

<http://extensions.services.openoffice.org/dictionary>

The older, OpenOffice 2 files may be used if this doesn't work:

<http://wiki.services.openoffice.org/wiki/Dictionaryes>

You can also use a plain word list. The results are the same, the choice depends on what word lists you can find.

If you install Aap (from www.a-a-p.org) you can use the recipes in the `runtime/spell/??/` directories. Aap will take care of downloading the files, apply patches needed for Vim and build the **.spl** file.

Make sure your current locale is set properly, otherwise Vim doesn't know what characters are upper/lower case letters. If the locale isn't available (e.g., when using an MS-Windows codepage on Unix) add tables to the **.aff** file

spell-affix-chars. If the **.aff** file doesn't define a table then the word table of the currently active spelling is used. If spelling is not active then Vim will try to guess.

:mksp :mkspell

:mksp[ell][!] [-ascii] {outname} {iname} ...

Generate a Vim spell file from word lists. Example:

:mkspell /tmp/nl nl_NL.words

E751

When **{outname}** ends in **".spl"** it is used as the output file name. Otherwise it should be a language name, such as **"en"**, without the region name. The file written will be **"{outname}.{encoding}.spl"**, where **{encoding}** is the value of the **'encoding'** option.

When the output file already exists **[!]** must be used to overwrite it.

When the **[-ascii]** argument is present, words with non-ascii characters are skipped. The resulting file ends in **"ascii.spl"**.

The input can be the Myspell format files **{iname}.aff** and **{iname}.dic**. If **{iname}.aff** does not exist then **{iname}** is used as the file name of a plain word list.

Multiple `{inname}` arguments can be given to combine regions into one Vim spell file. Example:
`:mkspell ~/.vim/spell/en /tmp/en_US /tmp/en_CA /tmp/en_AU`
This combines the English word lists for US, CA and AU into one `en.spl` file.

Up to eight regions can be combined. `E754` `E755`
The REP and SAL items of the first `.aff` file where they appear are used. `spell-REP` `spell-SAL`
`E845`

This command uses a lot of memory, required to find the optimal word tree (Polish, Italian and Hungarian require several hundred Mbyte). The final result will be much smaller, because compression is used. To avoid running out of memory compression will be done now and then. This can be tuned with the `'mkspellmem'` option.

After the spell file was written and it was being used in a buffer it will be reloaded automatically.

`:mksp[ell] [-ascii] {name}.{enc}.add`
Like `":mkspell"` above, using `{name}.{enc}.add` as the input file and producing an output file in the same directory that has `".spl"` appended.

`:mksp[ell] [-ascii] {name}`
Like `":mkspell"` above, using `{name}` as the input file and producing an output file in the same directory that has `".{enc}.spl"` appended.

Vim will report the number of duplicate words. This might be a mistake in the list of words. But sometimes it is used to have different prefixes and suffixes for the same basic word to avoid them combining (e.g. Czech uses this). If you want Vim to report all duplicate words set the `'verbose'` option.

Since you might want to change a Myspell word list for use with Vim the following procedure is recommended:

1. Obtain the `xx_YY.aff` and `xx_YY.dic` files from Myspell.
2. Make a copy of these files to `xx_YY.orig.aff` and `xx_YY.orig.dic`.
3. Change the `xx_YY.aff` and `xx_YY.dic` files to remove bad words, add missing words, define word characters with FOL/LOW/UPP, etc. The distributed `"*.diff"` files can be used.
4. Start Vim with the right locale and use `:mkspell` to generate the Vim spell file.
5. Try out the spell file with `":set spell spelllang=xx"` if you wrote it in a spell directory in `'runtimepath'`, or `":set spelllang=xx.enc.spl"` if you wrote it somewhere else.

When the Myspell files are updated you can merge the differences:

1. Obtain the new Myspell files as `xx_YY.new.aff` and `xx_UU.new.dic`.
2. Use `Vimdiff` to see what changed:

`vimdiff xx_YY.orig.dic xx_YY.new.dic`

3. Take over the changes you like in `xx_YY.dic`.
You may also need to change `xx_YY.aff`.
4. Rename `xx_YY.new.dic` to `xx_YY.orig.dic` and `xx_YY.new.aff` to `xx_YY.orig.aff`.

SPELL FILE VERSIONS

E770 E771 E772

Spell checking is a relatively new feature in Vim, thus it's possible that the `.spl` file format will be changed to support more languages. Vim will check the validity of the spell file and report anything wrong.

E771: Old spell file, needs to be updated

This spell file is older than your Vim. You need to update the `.spl` file.

E772: Spell file is for newer version of Vim

This means the spell file was made for a later version of Vim. You need to update Vim.

E770: Unsupported section in spell file

This means the spell file was made for a later version of Vim and contains a section that is required for the spell file to work. In this case it's probably a good idea to upgrade your Vim.

SPELL FILE DUMP

If for some reason you want to check what words are supported by the currently used spelling files, use this command:

	<code>:spelldump</code> <code>:spelld</code>
<code>:spelld[ump]</code>	Open a new window and fill it with all currently valid words. Compound words are not included. Note: For some languages the result may be enormous, causing Vim to run out of memory.
<code>:spelld[ump]!</code>	Like <code>:spelldump</code> and include the word count. This is the number of times the word was found while updating the screen. Words that are in <code>COMMON</code> items get a starting count of 10.

The format of the word list is used `spell-wordlist-format`. You should be able to read it with `:mkspell` to generate one `.spl` file that includes all the words.

When all entries to `'spelllang'` use the same regions or no regions at all then the region information is included in the dumped words. Otherwise only words for the current region are included and no `"/regions"` line is generated.

Comment lines with the name of the `.spl` file are used as a header above the words that were generated from that `.spl` file.

SPELL FILE MISSING

`spell-SpellFileMissing` `spellfile.vim`

If the spell file for the language you are using is not available, you will get an error message. But if the "spellfile.vim" plugin is active it will offer you to download the spell file. Just follow the instructions, it will ask you where to write the file (there must be a writable directory in **'runtimepath'** for this).

The plugin has a default place where to look for spell files, on the Vim ftp server. The protocol used is SSL (https://) for security. If you want to use another location or another protocol, set the `g:spellfile_URL` variable to the directory that holds the spell files. You can use `http://` or `ftp://`, but you are taking a security risk then. The `netrw` plugin is used for getting the file, look there for the specific syntax of the URL. Example:

```
let g:spellfile_URL = 'https://ftp.nluug.nl/vim/runtime/spell'
```

You may need to escape special characters.

The plugin will only ask about downloading a language once. If you want to try again anyway restart Vim, or set `g:spellfile_URL` to another value (e.g., prepend a space).

To avoid using the "spellfile.vim" plugin do this in your vimrc file:

```
let loaded_spellfile_plugin = 1
```

Instead of using the plugin you can define a `SpellFileMissing` autocommand to handle the missing file yourself. You can use it like this:

```
:au SpellFileMissing * call Download_spell_file(expand('<amatch>'))
```

Thus the `<amatch>` item contains the name of the language. Another important value is **'encoding'**, since every encoding has its own spell file. With two exceptions:

- For ISO-8859-15 (latin9) the name "latin1" is used (the encodings only differ in characters not used in dictionary words).
- The name "ascii" may also be used for some languages where the words use only ASCII letters for most of the words.

The default "spellfile.vim" plugin uses this autocommand, if you define your autocommand afterwards you may want to use `:au! SpellFileMissing` to overrule it. If you define your autocommand before the plugin is loaded it will notice this and not do anything.

E797

Note that the `SpellFileMissing` autocommand must not change or destroy the buffer the user was editing.

=====

4. Spell file format

spell-file-format

This is the format of the files that are used by the person who creates and maintains a word list.

Note that we avoid the word "dictionary" here. That is because the goal of spell checking differs from writing a dictionary (as in the book). For spelling we need a list of words that are OK, thus should not be highlighted.

Person and company names will not appear in a dictionary, but do appear in a word list. And some old words are rarely used while they are common misspellings. These do appear in a dictionary but not in a word list.

There are two formats: A straight list of words and a list using affix compression. The files with affix compression are used by Myspell (Mozilla and OpenOffice.org). This requires two files, one with .aff and one with .dic extension.

FORMAT OF STRAIGHT WORD LIST

spell-wordlist-format

The words must appear one per line. That is all that is required.

Additionally the following items are recognized:

- Empty and blank lines are ignored.

`# comment`

- Lines starting with a # are ignored (comment lines).

`/encoding=utf-8`

- A line starting with `"/encoding="`, before any word, specifies the encoding of the file. After the second '=' comes an encoding name. This tells Vim to setup conversion from the specified encoding to **'encoding'**. Thus you can use one word list for several target encodings.

`/regions=usca`

- A line starting with `"/regions="` specifies the region names that are supported. Each region name must be two ASCII letters. The first one is region 1. Thus `"/regions=usca` has region 1 "us" and region 2 "ca". In an addition word list the region names should be equal to the main word list!

- Other lines starting with `'/'` are reserved for future use. The ones that are not recognized are ignored. You do get a warning message, so that you know something won't work.

- A `"/` may follow the word with the following items:

`=` Case must match exactly.

`?` Rare word.

`!` Bad (wrong) word.

`1 to 9` A region in which the word is valid. If no regions are specified the word is valid in all regions.

Example:

```
# This is an example word list      comment
/encoding=latin1                   encoding of the file
/regions=uscagb                    regions "us", "ca" and "gb"
example                             word for all regions
blah/12                             word for regions "us" and "ca"
vim/!                               bad word
Campbell/?3                         rare word in region 3 "gb"
```

's mornings/=

keep-case word

Note that when "/=" is used the same word with all upper-case letters is not accepted. This is different from a word with mixed case that is automatically marked as keep-case, those words may appear in all upper-case letters.

FORMAT WITH .AFF AND .DIC FILES

aff-dic-format

There are two files: the basic word list and an affix file. The affix file specifies settings for the language and can contain affixes. The affixes are used to modify the basic words to get the full word list. This significantly reduces the number of words, especially for a language like Polish. This is called affix compression.

The basic word list and the affix file are combined with the ":mkspell" command and results in a binary spell file. All the preprocessing has been done, thus this file loads fast. The binary spell file format is described in the source code (src/spell.c). But only developers need to know about it.

The preprocessing also allows us to take the Myspell language files and modify them before the Vim word list is made. The tools for this can be found in the "src/spell" directory.

The format for the affix and word list files is based on what Myspell uses (the spell checker of Mozilla and OpenOffice.org). A description can be found here:

<http://linguocomponent.openoffice.org/affix.readme>

Note that affixes are case sensitive, this isn't obvious from the description.

Vim supports quite a few extras. They are described below [spell-affix-vim](#). Attempts have been made to keep this compatible with other spell checkers, so that the same files can often be used. One other project that offers more than Myspell is Hunspell (<http://hunspell.sf.net>).

WORD LIST FORMAT

spell-dic-format

A short example, with line numbers:

```
1      1234
2      aan
3      Als
4      Etten-Leur
5      et al.
6      's-Gravenhage
7      's-Gravenhaags
8      # word that differs between regions
9      kado/1
10     cadeau/2
11     TCP,IP
12     /the S affix may add a 's'
13     bedel/S
```

The first line contains the number of words. Vim ignores it, but you do get an error message if it's not there. [E760](#)

What follows is one word per line. White space at the end of the line is ignored, all other white space matters. The encoding is specified in the affix file [spell-SET](#) .

Comment lines start with '#' or '/'. See the example lines 8 and 12. [Note](#) that putting a comment after a word is NOT allowed:

```
someword # comment that causes an error!
```

After the word there is an optional slash and flags. Most of these flags are letters that indicate the affixes that can be used with this word. These are specified with SFX and PFX lines in the .aff file, see [spell-SFX](#) and [spell-PFX](#) . Vim allows using other flag types with the FLAG item in the affix file [spell-FLAG](#) .

When the word only has lower-case letters it will also match with the word starting with an upper-case letter.

When the word includes an upper-case letter, this means the upper-case letter is required at this position. The same word with a lower-case letter at this position will not match. When some of the other letters are upper-case it will not match either.

The word with all upper-case characters will always be OK,

word list	matches	does not match
als	als Als ALS	ALs ALS aLs aLS
Als	Als ALS	als ALs ALs aLs aLS
ALS	ALS	als Als ALS ALs aLs aLS
ALS	ALS ALS	als Als ALS aLs aLS

The KEEPCASE affix ID can be used to specifically match a word with identical case only, see below [spell-KEEPCASE](#) .

[Note](#): in line 5 to 7 non-word characters are used. You can include any character in a word. When checking the text a word still only matches when it appears with a non-word character before and after it. For Myspell a word starting with a non-word character probably won't work.

In line 12 the word "TCP/IP" is defined. Since the slash has a special meaning the comma is used instead. This is defined with the SLASH item in the affix file, see [spell-SLASH](#) . [Note](#) that without this SLASH item the word will be "TCP,IP".

AFFIX FILE FORMAT

[spell-aff-format](#) [spell-affix-vim](#)

[spell-affix-comment](#)

Comment lines in the .aff file start with a '#':

```
# comment line
```


Items with a fixed number of arguments can be followed by a comment. But only if none of the arguments can contain white space. The comment must start with a "#" character. Example:

```
KEEPCASE = # fix case for words with this flag
```

ENCODING

spell-SET

The affix file can be in any encoding that is supported by "iconv". However, in some cases the current locale should also be set properly at the time `:mkspell` is invoked. Adding FOL/LOW/UPP lines removes this requirement `spell-FOL`.

The encoding should be specified before anything where the encoding matters. The encoding applies both to the affix file and the dictionary file. It is done with a SET line:

```
SET utf-8
```

The encoding can be different from the value of the `'encoding'` option at the time `:mkspell` is used. Vim will then convert everything to `'encoding'` and generate a spell file for `'encoding'`. If some of the used characters do not fit in `'encoding'` you will get an error message.

spell-affix-mbyte

When using a multibyte encoding it's possible to use more different affix flags. But Myspell doesn't support that, thus you may not want to use it anyway. For compatibility use an 8-bit encoding.

INFORMATION

These entries in the affix file can be used to add information to the spell file. There are no restrictions on the format, but they should be in the right encoding.

	spell-NAME	spell-VERSION	spell-HOME
	spell-AUTHOR	spell-EMAIL	spell-COPYRIGHT
NAME	Name of the language		
VERSION	1.0.1 with fixes		
HOME	http://www.myhome.eu		
AUTHOR	John Doe		
EMAIL	john AT Doe DOT net		
COPYRIGHT	LGPL		

These fields are put in the `.spl` file as-is. The `:spellinfo` command can be used to view the info.

```
:spelli[nfo]          :spellinfo :spelli  
Display the information for the spell file(s) used for  
the current buffer.
```

CHARACTER TABLES

[spell-affix-chars](#)

When using an 8-bit encoding the affix file should define what characters are word characters. This is because the system where ":mkspell" is used may not support a locale with this encoding and isalpha() won't work. For example when using "cp1250" on Unix.

[E761](#) [E762](#) [spell-FOL](#)
[spell-LOW](#) [spell-UPP](#)

Three lines in the affix file are needed. Simplistic example:

```
FOL  áëñ
LOW  áëñ
UPP  ÁËÑ
```

All three lines must have exactly the same number of characters.

The "FOL" line specifies the case-folded characters. These are used to compare words while ignoring case. For most encodings this is identical to the lower case line.

The "LOW" line specifies the characters in lower-case. Mostly it's equal to the "FOL" line.

The "UPP" line specifies the characters with upper-case. That is, a character is upper-case where it's different from the character at the same position in "FOL".

An exception is made for the German sharp s ß. The upper-case version is "SS". In the FOL/LOW/UPP lines it should be included, so that it's recognized as a word character, but use the ß character in all three.

ASCII characters should be omitted, Vim always handles these in the same way. When the encoding is UTF-8 no word characters need to be specified.

[E763](#)

Vim allows you to use spell checking for several languages in the same file. You can list them in the '[spelllang](#)' option. As a consequence all spell files for the same encoding must use the same word characters, otherwise they can't be combined without errors.

If you get an E763 warning that the word tables differ you need to update your ".spl" spell files. If you downloaded the files, get the latest version of all spell files you use. If you are only using one, e.g., German, then also download the recent English spell files. Otherwise generate the .spl file again with [:mkspell](#) . If you still get errors check the FOL, LOW and UPP lines in the used .aff files.

The XX.ascii.spl spell file generated with the "-ascii" argument will not contain the table with characters, so that it can be combine with spell files for any encoding. The .add.spl files also do not contain the table.

MID-WORD CHARACTERS

[spell-midword](#)

Some characters are only to be considered word characters if they are used in between two ordinary word characters. An example is the single quote: It is often used to put text in quotes, thus it can't be recognized as a word character, but when it appears in between word characters it must be part of the word. This is needed to detect a spelling error such as they'are. That should be they're, but since "they" and "are" are words themselves that would go unnoticed.

These characters are defined with MIDWORD in the .aff file. Example:

```
MIDWORD '-
```

FLAG TYPES

spell-FLAG

Flags are used to specify the affixes that can be used with a word and for other properties of the word. Normally single-character flags are used. This limits the number of possible flags, especially for 8-bit encodings. The FLAG item can be used if more affixes are to be used. Possible values:

FLAG long	use two-character flags
FLAG num	use numbers, from 1 up to 65000
FLAG caplong	use one-character flags without A-Z and two-character flags that start with A-Z

With "FLAG num" the numbers in a list of affixes need to be separated with a comma: "234,2143,1435". This method is inefficient, but useful if the file is generated with a program.

When using "caplong" the two-character flags all start with a capital: "Aa", "B1", "BB", etc. This is useful to use one-character flags for the most common items and two-character flags for uncommon items.

Note: When using utf-8 only characters up to 65000 may be used for flags.

Note: even when using "num" or "long" the number of flags available to compounding and prefixes is limited to about 250.

AFFIXES

spell-PFX spell-SFX

The usual PFX (prefix) and SFX (suffix) lines are supported (see the Myspell documentation or the Aspell manual:

<http://aspell.net/man-html/Affix-Compression.html>).

Summary:

```
SFX L Y 2
SFX L 0 re [^x]
SFX L 0 ro x
```

The first line is a header and has four fields:

```
SFX {flag} {combine} {count}
```

{flag} The name used for the suffix. Mostly it's a single letter,

but other characters can be used, see [spell-FLAG](#) .

`{combine}` Can be 'Y' or 'N'. When 'Y' then the word plus suffix can also have a prefix. When 'N' then a prefix is not allowed.

`{count}` The number of lines following. If this is wrong you will get an error message.

For PFX the fields are exactly the same.

The basic format for the following lines is:

```
SFX {flag} {strip} {add} {condition} {extra}
```

`{flag}` Must be the same as the `{flag}` used in the first line.

`{strip}` Characters removed from the basic word. There is no check if the characters are actually there, only the length is used (in bytes). This better match the `{condition}`, otherwise strange things may happen. If the `{strip}` length is equal to or longer than the basic word the suffix won't be used. When `{strip}` is 0 (zero) then nothing is stripped.

`{add}` Characters added to the basic word, after removing `{strip}`. Optionally there is a '/' followed by flags. The flags apply to the word plus affix. See [spell-affix-flags](#)

`{condition}` A simplistic pattern. Only when this matches with a basic word will the suffix be used for that word. This is normally for using one suffix letter with different `{add}` and `{strip}` fields for words with different endings.

When `{condition}` is a . (dot) there is no condition.

The pattern may contain:

- Literal characters.
- A set of characters in []. `[abc]` matches a, b and c. A dash is allowed for a range `[a-c]`, but this is Vim-specific.
- A set of characters that starts with a ^, meaning the complement of the specified characters. `[^abc]` matches any character but a, b and c.

`{extra}` Optional extra text:

```
# comment          Comment is ignored
-                  Hunspell uses this, ignored
```

For PFX the fields are the same, but the `{strip}`, `{add}` and `{condition}` apply to the start of the word.

Note: Myspell ignores any extra text after the relevant info. Vim requires this text to start with a "#" so that mistakes don't go unnoticed. Example:

```
SFX F 0 in    [^i]n    # Spion > Spionin
SFX F 0 nen  in    # Bauerin > Bauerinnen
```

However, to avoid lots of errors in affix files written for Myspell, you can

add the IGNOREEXTRA flag.

Apparently Myspell allows an affix name to appear more than once. Since this might also be a mistake, Vim checks for an extra "S". The affix files for Myspell that use this feature apparently have this flag. Example:

```
SFX a Y 1 S
SFX a 0 an .

SFX a Y 2 S
SFX a 0 en .
SFX a 0 on .
```

AFFIX FLAGS

spell-affix-flags

This is a feature that comes from Hunspell: The affix may specify flags. This works similar to flags specified on a basic word. The flags apply to the basic word plus the affix (but there are restrictions). Example:

```
SFX S Y 1
SFX S 0 s .

SFX A Y 1
SFX A 0 able/S .
```

When the dictionary file contains "drink/AS" then these words are possible:

```
drink
drinks          uses S suffix
drinkable       uses A suffix
drinkables      uses A suffix and then S suffix
```

Generally the flags of the suffix are added to the flags of the basic word, both are used for the word plus suffix. But the flags of the basic word are only used once for affixes, except that both one prefix and one suffix can be used when both support combining.

Specifically, the affix flags can be used for:

- Suffixes on suffixes, as in the example above. This works once, thus you can have two suffixes on a word (plus one prefix).
- Making the word with the affix rare, by using the `spell-RARE` flag.
- Exclude the word with the affix from compounding, by using the `spell-COMPOUNDFORBIDFLAG` flag.
- Allow the word with the affix to be part of a compound word on the side of the affix with the `spell-COMPOUNDPERMITFLAG` .
- Use the NEEDCOMPOUND flag: word plus affix can only be used as part of a compound word. `spell-NEEDCOMPOUND`
- Compound flags: word plus affix can be part of a compound word at the end, middle, start, etc. The flags are combined with the flags of the basic word. `spell-compound`
- NEEDAFFIX: another affix is needed to make a valid word.
- CIRCUMFIX, as explained just below.

IGNOREEXTRA

spell-IGNOREEXTRA

Normally Vim gives an error for an extra field that does not start with '#'. This avoids errors going unnoticed. However, some files created for Myspell or Hunspell may contain many entries with an extra field. Use the IGNOREEXTRA flag to avoid lots of errors.

CIRCUMFIX

spell-CIRCUMFIX

The CIRCUMFIX flag means a prefix and suffix must be added at the same time. If a prefix has the CIRCUMFIX flag then only suffixes with the CIRCUMFIX flag can be added, and the other way around. An alternative is to only specify the suffix, and give that suffix two flags: the required prefix and the NEEDAFFIX flag. [spell-NEEDAFFIX](#)

PFXPOSTPONE

spell-PFXPOSTPONE

When an affix file has very many prefixes that apply to many words it's not possible to build the whole word list in memory. This applies to Hebrew (a list with all words is over a Gbyte). In that case applying prefixes must be postponed. This makes spell checking slower. It is indicated by this keyword in the .aff file:

[PFXPOSTPONE](#)

Only prefixes without a chop string and without flags can be postponed. Prefixes with a chop string or with flags will still be included in the word list. An exception if the chop string is one character and equal to the last character of the added string, but in lower case. Thus when the chop string is used to allow the following word to start with an upper case letter.

WORDS WITH A SLASH

spell-SLASH

The slash is used in the .dic file to separate the basic word from the affix letters and other flags. Unfortunately, this means you cannot use a slash in a word. Thus "TCP/IP" is not a word but "TCP" with the flags "IP". To include a slash in the word put a backslash before it: "TCP\IP". In the rare case you want to use a backslash inside a word you need to use two backslashes. Any other use of the backslash is reserved for future expansion.

KEEP-CASE WORDS

spell-KEEPCASE

In the affix file a KEEPCASE line can be used to define the affix name used for keep-case words. Example:

[KEEPCASE =](#)

This flag is not supported by Myspell. It has the meaning that case matters. This can be used if the word does not have the first letter in upper case at

the start of a sentence. Example:

word list	matches	does not match
's morgens/=	's morgens	'S morgens 's Morgens 'S MORGENS
's Morgens	's Morgens 'S MORGENS	'S morgens 's morgens

The flag can also be used to avoid that the word matches when it is in all upper-case letters.

RARE WORDS

spell-RARE

In the affix file a RARE line can be used to define the affix name used for rare words. Example:

RARE ?

Rare words are highlighted differently from bad words. This is to be used for words that are correct for the language, but are hardly ever used and could be a typing mistake anyway. When the same word is found as good it won't be highlighted as rare.

This flag can also be used on an affix, so that a basic word is not rare but the basic word plus affix is rare [spell-affix-flags](#) . However, if the word also appears as a good word in another way (e.g., in another region) it won't be marked as rare.

BAD WORDS

spell-BAD

In the affix file a BAD line can be used to define the affix name used for bad words. Example:

BAD !

This can be used to exclude words that would otherwise be good. For example "the the" in the .dic file:

the the/!

Once a word has been marked as bad it won't be undone by encountering the same word as good.

The flag also applies to the word with affixes, thus this can be used to mark a whole bunch of related words as bad.

spell-FORBIDDENWORD

FORBIDDENWORD can be used just like BAD. For compatibility with Hunspell.

spell-NEEDAFFIX

The NEEDAFFIX flag is used to require that a word is used with an affix. The word itself is not a good word (unless there is an empty affix). Example:

NEEDAFFIX +

COMPOUND WORDS

spell-compound

A compound word is a longer word made by concatenating words that appear in the .dic file. To specify which words may be concatenated a character is used. This character is put in the list of affixes after the word. We will call this character a flag here. Obviously these flags must be different from any affix IDs used.

spell-COMPOUNDFLAG

The Myspell compatible method uses one flag, specified with COMPOUNDFLAG. All words with this flag combine in any order. This means there is no control over which word comes first. Example:

```
COMPOUNDFLAG c
```

spell-COMPOUNDRULE

A more advanced method to specify how compound words can be formed uses multiple items with multiple flags. This is not compatible with Myspell 3.0. Let's start with an example:

```
COMPOUNDRULE c+
COMPOUNDRULE se
```

The first line defines that words with the "c" flag can be concatenated in any order. The second line defines compound words that are made of one word with the "s" flag and one word with the "e" flag. With this dictionary:

```
bork/c
onion/s
soup/e
```

You can make these words:

```
bork
borkbork
borkborkbork
(etc.)
onion
soup
onionsoup
```

The COMPOUNDRULE item may appear multiple times. The argument is made out of one or more groups, where each group can be:

```
one flag                e.g., c
alternate flags inside [] e.g., [abc]
```

Optionally this may be followed by:

```
*   the group appears zero or more times, e.g., sm*e
+   the group appears one or more times, e.g., c+
?   the group appears zero times or once, e.g., x?
```

This is similar to the regexp pattern syntax (but not the same!). A few examples with the sequence of word flags they require:

```
COMPOUNDRULE x+      x xx xxx etc.
COMPOUNDRULE yz      yz
COMPOUNDRULE x+z     xz xxz xxxz etc.
COMPOUNDRULE yx+     yx yxx yxxx etc.
```



```

COMPOUNDRULE xy?z      xz xyz

COMPOUNDRULE [abc]z    az bz cz
COMPOUNDRULE [abc]+z  az aaz abaz bz baz bcbz cz caz cbaz etc.
COMPOUNDRULE a[xyz]+  ax axx axyz ay ayx ayzz az azy azxy etc.
COMPOUNDRULE sm*e     se sme smme smmme etc.
COMPOUNDRULE s[xyz]*e se sxe sxye sxyxe sye syze sze szye szyxe etc.

```

A specific example: Allow a compound to be made of two words and a dash:

```

In the .aff file:
    COMPOUNDRULE sde
    NEEDAFFIX x
    COMPOUNDWORDMAX 3
    COMPOUNDMIN 1
In the .dic file:
    start/s
    end/e
    -/xd

```

This allows for the word "start-end", but not "startend".

An additional implied rule is that, without further flags, a word with a prefix cannot be compounded after another word, and a word with a suffix cannot be compounded with a following word. Thus the affix cannot appear on the inside of a compound word. This can be changed with the `spell-COMPOUNDPERMITFLAG`.

`spell-NEEDCOMPOUND`

The NEEDCOMPOUND flag is used to require that a word is used as part of a compound word. The word itself is not a good word. Example:

```
NEEDCOMPOUND &
```

`spell-ONLYINCOMPOUND`

The ONLYINCOMPOUND does exactly the same as NEEDCOMPOUND. Supported for compatibility with Hunspell.

`spell-COMPOUNDMIN`

The minimal character length of a word used for compounding is specified with COMPOUNDMIN. Example:

```
COMPOUNDMIN 5
```

When omitted there is no minimal length. Obviously you could just leave out the compound flag from short words instead, this feature is present for compatibility with Myspell.

`spell-COMPOUNDWORDMAX`

The maximum number of words that can be concatenated into a compound word is specified with COMPOUNDWORDMAX. Example:

```
COMPOUNDWORDMAX 3
```

When omitted there is no maximum. It applies to all compound words.

To set a limit for words with specific flags make sure the items in

COMPOUNDRULE where they appear don't allow too many words.

spell-COMPOUNDSYLMAX

The maximum number of syllables that a compound word may contain is specified with COMPOUNDSYLMAX. Example:

```
COMPOUNDSYLMAX 6
```

This has no effect if there is no SYLLABLE item. Without COMPOUNDSYLMAX there is no limit on the number of syllables.

If both COMPOUNDWORDMAX and COMPOUNDSYLMAX are defined, a compound word is accepted if it fits one of the criteria, thus is either made from up to COMPOUNDWORDMAX words or contains up to COMPOUNDSYLMAX syllables.

spell-COMPOUNDFORBIDFLAG

The COMPOUNDFORBIDFLAG specifies a flag that can be used on an affix. It means that the word plus affix cannot be used in a compound word. Example:

affix file:

```
COMPOUNDFLAG c
COMPOUNDFORBIDFLAG x
SFX a Y 2
SFX a 0 s .
SFX a 0 ize/x .
```

dictionary:

```
word/c
util/ac
```

This allows for "wordutil" and "wordutils" but not "wordutilize".

Note: this doesn't work for postponed prefixes yet.

spell-COMPOUNDPERMITFLAG

The COMPOUNDPERMITFLAG specifies a flag that can be used on an affix. It means that the word plus affix can also be used in a compound word in a way where the affix ends up halfway the word. Without this flag that is not allowed.

Note: this doesn't work for postponed prefixes yet.

spell-COMPOUNDROOT

The COMPOUNDROOT flag is used for words in the dictionary that are already a compound. This means it counts for two words when checking the compounding rules. Can also be used for an affix to count the affix as a compounding word.

spell-CHECKCOMPOUNDPATTERN

CHECKCOMPOUNDPATTERN is used to define patterns that, when matching at the position where two words are compounded together forbids the compound.

For example:

```
CHECKCOMPOUNDPATTERN o e
```

This forbids compounding if the first word ends in "o" and the second word starts with "e".

The arguments must be plain text, no patterns are actually supported, despite the item name. Case is always ignored.

The Hunspell feature to use three arguments and flags is not supported.

spell-NOCOMPOUNDSUGS

This item indicates that using compounding to make suggestions is not a good idea. Use this when compounding is used with very short or one-character words. E.g. to make numbers out of digits. Without this flag creating suggestions would spend most time trying all kind of weird compound words.

NOCOMPOUNDSUGS

spell-SYLLABLE

The SYLLABLE item defines characters or character sequences that are used to count the number of syllables in a word. Example:

```
SYLLABLE aáééííóóóöüüüÿ/aa/au/ea/ee/ei/ie/oa/oe/oo/ou/uu/ui
```

Before the first slash is the set of characters that are counted for one syllable, also when repeated and mixed, until the next character that is not in this set. After the slash come sequences of characters that are counted for one syllable. These are preferred over using characters from the set. With the example "ideeen" has three syllables, counted by "i", "ee" and "e".

Only case-folded letters need to be included.

Another way to restrict compounding was mentioned above: Adding the `spell-COMPOUNDFORBIDFLAG` flag to an affix causes all words that are made with that affix to not be used for compounding.

UNLIMITED COMPOUNDING

spell-NOBREAK

For some languages, such as Thai, there is no space in between words. This looks like all words are compounded. To specify this use the NOBREAK item in the affix file, without arguments:

```
NOBREAK
```

Vim will try to figure out where one word ends and a next starts. When there are spelling mistakes this may not be quite right.

spell-COMMON

Common words can be specified with the COMMON item. This will give better suggestions when editing a short file. Example:

```
COMMON the of to and a in is it you that he she was for on are
```

The words must be separated by white space, up to 25 per line.

When multiple regions are specified in a ":mkspell" command the common words for all regions are combined and used for all regions.

spell-NOSPLITSUGS

This item indicates that splitting a word to make suggestions is not a good idea. Split-word suggestions will appear only when there are few similar words.

NOSPLITSUGS

spell-NOSUGGEST

The flag specified with NOSUGGEST can be used for words that will not be suggested. Can be used for obscene words.

```
NOSUGGEST %
```

REPLACEMENTS

spell-REP

In the affix file REP items can be used to define common mistakes. This is used to make spelling suggestions. The items define the "from" text and the "to" replacement. Example:

```
REP 4
REP f ph
REP ph f
REP k ch
REP ch k
```

The first line specifies the number of REP lines following. Vim ignores the number, but it must be there (for compatibility with Myspell).

Don't include simple one-character replacements or swaps. Vim will try these anyway. You can include whole words if you want to, but you might want to use the "file:" item in '**spellsuggest**' instead.

You can include a space by using an underscore:

```
REP the_the the
```

SIMILAR CHARACTERS

spell-MAP E783

In the affix file MAP items can be used to define letters that are very much alike. This is mostly used for a letter with different accents. This is used to prefer suggestions with these letters substituted. Example:

```
MAP 2
MAP eéëêè
MAP uüùúû
```

The first line specifies the number of MAP lines following. Vim ignores the number, but the line must be there.

Each letter must appear in only one of the MAP items. It's a bit more efficient if the first letter is ASCII or at least one without accents.

.SUG FILE

spell-NOSUGFILE

When soundfolding is specified in the affix file then ":mkspell" will normally

produce a .sug file next to the .spl file. This file is used to find suggestions by their sound-a-like form quickly. At the cost of a lot of memory (the amount depends on the number of words, `:mkspell` will display an estimate when it's done).

To avoid producing a .sug file use this item in the affix file:

```
NOSUGFILE
```

Users can simply omit the .sug file if they don't want to use it.

SOUND-A-LIKE

`spell-SAL`

In the affix file SAL items can be used to define the sounds-a-like mechanism to be used. The main items define the "from" text and the "to" replacement. Simplistic example:

```
SAL CIA X
SAL CH X
SAL C K
SAL K K
```

There are a few rules and this can become quite complicated. An explanation how it works can be found in the Aspell manual:

<http://aspell.net/man-html/Phonetic-Code.html>.

There are a few special items:

```
SAL followup true
SAL collapse_result true
SAL remove_accents true
```

"1" has the same meaning as "true". Any other value means "false".

SIMPLE SOUNDFOLDING

`spell-SOFOFROM`

`spell-SOFOTO`

The SAL mechanism is complex and slow. A simpler mechanism is mapping all characters to another character, mapping similar sounding characters to the same character. At the same time this does case folding. You can not have both SAL items and simple soundfolding.

There are two items required: one to specify the characters that are mapped and one that specifies the characters they are mapped to. They must have exactly the same number of characters. Example:

```
SOFOFROM abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
SOFOTO ebctefghejklneprkstevvkesebctefghejklneprkstevvkes
```

In the example all vowels are mapped to the same character 'e'. Another method would be to leave out all vowels. Some characters that sound nearly the same and are often mixed up, such as 'm' and 'n', are mapped to the same character. Don't do this too much, all words will start looking alike.

Characters that do not appear in SOFOFROM will be left out, except that all white space is replaced by one space. Sequences of the same character in SOFOFROM are replaced by one.

You can use the `soundfold()` function to try out the results. Or set the `'verbose'` option to see the score in the output of the `z=` command.

UNSUPPORTED ITEMS

`spell-affix-not-supported`

These items appear in the affix file of other spell checkers. In Vim they are ignored, not supported or defined in another way.

ACCENT	(Hunspell) Use MAP instead. <code>spell-MAP</code>	<code>spell-ACCENT</code>
BREAK	(Hunspell) Define break points. Unclear how it works exactly. Not supported.	<code>spell-BREAK</code>
CHECKCOMPOUNDCASE	(Hunspell) Disallow uppercase letters at compound word boundaries. Not supported.	<code>spell-CHECKCOMPOUNDCASE</code>
CHECKCOMPOUNDDUP	(Hunspell) Disallow using the same word twice in a compound. Not supported.	<code>spell-CHECKCOMPOUNDDUP</code>
CHECKCOMPOUNDREP	(Hunspell) Something about using REP items and compound words. Not supported.	<code>spell-CHECKCOMPOUNDREP</code>
CHECKCOMPOUNDTRIPLE	(Hunspell) Forbid three identical characters when compounding. Not supported.	<code>spell-CHECKCOMPOUNDTRIPLE</code>
CHECKSHARPS	(Hunspell) SS letter pair in uppercased (German) words may be upper case sharp s (ß). Not supported.	<code>spell-CHECKSHARPS</code>
COMPLEXPREFIXES	(Hunspell) Enables using two prefixes. Not supported.	<code>spell-COMPLEXPREFIXES</code>
COMPOUND	(Hunspell) This is one line with the count of COMPOUND items, followed by that many COMPOUND lines with a pattern. Remove the first line with the count and rename the other items to COMPOUNDRULE <code>spell-COMPOUNDRULE</code>	<code>spell-COMPOUND</code>
COMPOUNDFIRST	(Hunspell) Use COMPOUNDRULE instead. <code>spell-COMPOUNDRULE</code>	<code>spell-COMPOUNDFIRST</code>
COMPOUNDBEGIN	(Hunspell)	<code>spell-COMPOUNDBEGIN</code>

	Words signed with COMPOUNDBEGIN may be first elements in compound words. Use COMPOUNDRULE instead. spell-COMPOUNDRULE
COMPOUNDLAST	(Hunspell) spell-COMPOUNDLAST Words signed with COMPOUNDLAST may be last elements in compound words. Use COMPOUNDRULE instead. spell-COMPOUNDRULE
COMPOUNDEND	(Hunspell) spell-COMPOUNDEND Probably the same as COMPOUNDLAST
COMPOUNDMIDDLE	(Hunspell) spell-COMPOUNDMIDDLE Words signed with COMPOUNDMIDDLE may be middle elements in compound words. Use COMPOUNDRULE instead. spell-COMPOUNDRULE
COMPOUNDRULES	(Hunspell) spell-COMPOUNDRULES Number of COMPOUNDRULE lines following. Ignored, but the argument must be a number.
COMPOUNDSYLLABLE	(Hunspell) spell-COMPOUNDSYLLABLE Use SYLLABLE and COMPOUNDSYLMAX instead. spell-SYLLABLE spell-COMPOUNDSYLMAX
KEY	(Hunspell) spell-KEY Define characters that are close together on the keyboard. Used to give better suggestions. Not supported.
LANG	(Hunspell) spell-LANG This specifies language-specific behavior. This actually moves part of the language knowledge into the program, therefore Vim does not support it. Each language property must be specified separately.
LEMMA_PRESENT	(Hunspell) spell-LEMMA_PRESENT Only needed for morphological analysis.
MAXNGRAMSUGS	(Hunspell) spell-MAXNGRAMSUGS Set number of n-gram suggestions. Not supported.
PSEUDOROOT	(Hunspell) spell-PSEUDOROOT Use NEEDAFFIX instead. spell-NEEDAFFIX
SUGSWITHDOTS	(Hunspell) spell-SUGSWITHDOTS Adds dots to suggestions. Vim doesn't need this.
SYLLABLENUM	(Hunspell) spell-SYLLABLENUM Not supported.
TRY	(Myspell, Hunspell, others) spell-TRY Vim does not use the TRY item, it is ignored. For making suggestions the actual characters in the words are used, that is much more efficient.

WORDCHARS

(Hunspell)

spell-WORDCHARS

Used to recognize words. Vim doesn't need it, because there is no need to separate words before checking them (using a trie instead of a hashtable).

VIM REFERENCE MANUAL by Bram Moolenaar

diff vimdiff gvimdiff diff-mode

This file describes the `+diff` feature: Showing differences between two to eight versions of the same file.

The basics are explained in section [08.7](#) of the user manual.

- | | |
|-----------------------|----------------------------|
| 1. Starting diff mode | <code>start-vimdiff</code> |
| 2. Viewing diffs | <code>view-diffs</code> |
| 3. Jumping to diffs | <code>jumpto-diffs</code> |
| 4. Copying diffs | <code>copy-diffs</code> |
| 5. Diff options | <code>diff-options</code> |

=====

1. Starting diff mode`start-vimdiff`

The easiest way to start editing in diff mode is with the `"vimdiff"` command. This starts Vim as usual, and additionally sets up for viewing the differences between the arguments.

```
vimdiff file1 file2 [file3 [file4]]
```

This is equivalent to:

```
vim -d file1 file2 [file3 [file4]]
```

You may also use `"gvimdiff"` or `"vim -d -g"`. The GUI is started then. You may also use `"viewdiff"` or `"gviewdiff"`. Vim starts in readonly mode then. `"r"` may be prepended for restricted mode (see [-Z](#)).

The second and following arguments may also be a directory name. Vim will then append the file name of the first argument to the directory name to find the file.

By default an internal diff library will be used. When `'diffopt'` or `'diffexpr'` has been set an external `"diff"` command will be used. This only works when such a diff program is available.

Diffs are local to the current tab page [tab-page](#). You can't see diffs with a window in another tab page. This does make it possible to have several diffs at the same time, each in their own tab page.

What happens is that Vim opens a window for each of the files. This is like using the `-O` argument. This uses vertical splits. If you prefer horizontal splits add the `-o` argument:

```
vimdiff -o file1 file2 [file3 [file4]]
```

If you always prefer horizontal splits include `"horizontal"` in `'diffopt'`.

In each of the edited files these options are set:

```
'diff'           on
'scrollbind'     on
'cursorbind'     on
'scrollopt'      includes "hor"
'wrap'           off, or leave as-is if 'diffopt' includes "followwrap"
'foldmethod'     "diff"
'foldcolumn'     value from 'diffopt', default is 2
```

These options are set local to the window. When editing another file they are reset to the global value.

The options can still be overruled from a modeline when re-editing the file. However, **'foldmethod'** and **'wrap'** won't be set from a modeline when **'diff'** is set.

See `:diffoff` for an easy way to revert the options.

The differences shown are actually the differences in the buffer. Thus if you make changes after loading a file, these will be included in the displayed diffs. You might have to do `:diffupdate` now and then, not all changes are immediately taken into account, especially when using an external diff command.

In your `.vimrc` file you could do something special when Vim was started in diff mode. You could use a construct like this:

```
if &diff
  setup for diff mode
else
  setup for non-diff mode
endif
```

While already in Vim you can start diff mode in three ways.

```
:diffs[plit] {filename} E98
:diffs :diffsplit
Open a new window on the file {filename}. The options are set
as for "vimdiff" for the current and the newly opened window.
Also see 'diffexpr'.
```

```
:diff[his] :diff :diffthis
Make the current window part of the diff windows. This sets
the options like for "vimdiff".
```

```
:diffp[atch] {patchfile} E816 :diffp :diffpatch
Use the current buffer, patch it with the diff found in
{patchfile} and open a buffer on the result. The options are
set as for "vimdiff".
{patchfile} can be in any format that the "patch" program
understands or 'patchexpr' can handle.
Note that {patchfile} should only contain a diff for one file,
the current file. If {patchfile} contains diffs for other
files as well, the results are unpredictable. Vim changes
directory to /tmp to avoid files in the current directory
```

accidentally being patched. But it may still result in various ".rej" files to be created. And when absolute path names are present these files may get patched anyway. Using the "patch" command is not allowed in `restricted-mode` .

To make these commands use a vertical split, prepend `:vertical` . Examples:

```
:vert diffsplit main.c~
:vert diffpatch /tmp/diff
```

If you always prefer a vertical split include "vertical" in `'diffopt'` .

E96

There can be up to eight buffers with `'diff'` set.

Since the option values are remembered with the buffer, you can edit another file for a moment and come back to the same file and be in diff mode again.

```
:diff[ff]          Switch off diff mode for the current window. Resets related
                   options also when 'diff' was not set.
                   :diff    :diffoff

:diff[ff]!        Switch off diff mode for the current window and in all windows
                   in the current tab page where 'diff' is set. Resetting
                   related options only happens in a window that has 'diff' set,
                   if the current window does not have 'diff' set then no options
                   in it are changed.
                   Hidden buffers are also removed from the list of diff'ed
                   buffers.
```

The `:diffoff` command resets the relevant options to the values they had when using `:diffsplit`, `:diffpatch`, `:diffthis` or starting Vim in diff mode. When using `:diffoff` twice the last saved values are restored. Otherwise they are set to their default value:

```
'diff'           off
'scrollbind'     off
'cursorbind'     off
'scrollopt'      without "hor"
'wrap'           on, or leave as-is if 'diffopt' includes "followwrap"
'foldmethod'     "manual"
'foldcolumn'     0
```

`'foldenable'` will most-likely be reset to off. That is when `'foldmethod'` is restored to "manual". The folds themselves are not cleared but they should not show up, resetting `'foldenable'` is the best way to do that.

2. Viewing diffs view-diffs

The effect is that the diff windows show the same text, with the differences highlighted. When scrolling the text, the `'scrollbind'` option will make the text in other windows to be scrolled as well. With vertical splits the text should be aligned properly.

The alignment of text will go wrong when:

- **'wrap'** is on, some lines will be wrapped and occupy two or more screen lines
- folds are open in one window but not another
- **'scrollbind'** is off
- changes have been made to the text
- "filler" is not present in **'diffopt'**, deleted/inserted lines makes the alignment go wrong

All the buffers edited in a window where the **'diff'** option is set will join in the diff. This is also possible for hidden buffers. They must have been edited in a window first for this to be possible. To get rid of the hidden buffers use `:diffoff!`.

`:DiffOrig diff-original-file`

Since **'diff'** is a window-local option, it's possible to view the same buffer in diff mode in one window and "normal" in another window. It is also possible to view the changes you have made to a buffer since the file was loaded. Since Vim doesn't allow having two buffers for the same file, you need another buffer. This command is useful:

```
command DiffOrig vert new | set bt=nofile | r ++edit # | 0d_
\ | diffthis | wincmd p | diffthis
```

(this is in `defaults.vim`). Use `:DiffOrig` to see the differences between the current buffer and the file it was loaded from.

A buffer that is unloaded cannot be used for the diff. But it does work for hidden buffers. You can use `:hide` to close a window without unloading the buffer. If you don't want a buffer to remain used for the diff do `:set nodiff` before hiding it.

`:dif :diff :diffupdate`

`:dif[fupdate][!]` Update the diff highlighting and folds.

Vim attempts to keep the differences updated when you make changes to the text. This mostly takes care of inserted and deleted lines. Changes within a line and more complicated changes do not cause the differences to be updated. To force the differences to be updated use:

`:diffupdate`

If the `!` is included Vim will check if the file was changed externally and needs to be reloaded. It will prompt for each changed file, like `:checktime` was used.

Vim will show filler lines for lines that are missing in one window but are present in another. These lines were inserted in another file or deleted in this file. Removing "filler" from the **'diffopt'** option will make Vim not display these filler lines.

Folds are used to hide the text that wasn't changed. See `folding` for all the commands that can be used with folds.

The context of lines above a difference that are not included in the fold can be set with the **'diffopt'** option. For example, to set the context to three lines:

```
:set diffopt=filler,context:3
```

The diffs are highlighted with these groups:

hl-DiffAdd	DiffAdd	Added (inserted) lines. These lines exist in this buffer but not in another.
hl-DiffChange	DiffChange	Changed lines.
hl-DiffText	DiffText	Changed text inside a Changed line. Vim finds the first character that is different, and the last character that is different (searching from the end of the line). The text in between is highlighted. This means that parts in the middle that are still the same are highlighted anyway. The 'diffopt' flags "iwhite" and "icase" are used here.
hl-DiffDelete	DiffDelete	Deleted lines. Also called filler lines, because they don't really exist in this buffer.

3. Jumping to diffs jumpto-diffs

Two commands can be used to jump to diffs:

[c	Jump backwards to the previous start of a change. When a count is used, do it that many times.	[c
]c	Jump forwards to the next start of a change. When a count is used, do it that many times.]c

It is an error if there is no change for the cursor to move to.

4. Diff copying copy-diffs E99 E100 E101 E102 E103 merge

There are two commands to copy text from one buffer to another. The result is that the buffers will be equal within the specified range.

```
:[range]diffg[et] [bufspec] :diffg :diffget  
Modify the current buffer to undo difference with another buffer. If [bufspec] is given, that buffer is used. If [bufspec] refers to the current buffer then nothing happens. Otherwise this only works if there is one other buffer in diff mode.  
See below for [range].
```

```
:[range]diffpu[t] [bufspec] :diffpu :diffput E793
```

Modify another buffer to undo difference with the current buffer. Just like `:diffget` but the other buffer is modified instead of the current one.

When `[bufspec]` is omitted and there is more than one other buffer in diff mode where `'modifiable'` is set this fails. See below for `[range]`.

`[count]``do` Same as `:diffget` without range. The "o" stands for "obtain" ("dg" can't be used, it could be the start of "dgg!"). **Note:** this doesn't work in Visual mode. If you give a `[count]`, it is used as the `[bufspec]` argument for `:diffget`.

`[count]``dp` Same as `:diffput` without range. **Note:** this doesn't work in Visual mode. If you give a `[count]`, it is used as the `[bufspec]` argument for `:diffput`.

When no `[range]` is given, the diff at the cursor position or just above it is affected. When `[range]` is used, Vim tries to only put or get the specified lines. When there are deleted lines, this may not always be possible.

There can be deleted lines below the last line of the buffer. When the cursor is on the last line in the buffer and there is no diff above this line, the `:diffget` and `do` commands will obtain lines from the other buffer.

To be able to get those lines from another buffer in a `[range]` it's allowed to use the last line number plus one. This command gets all diffs from the other buffer:

```
:1,$+1diffget
```

Note that deleted lines are displayed, but not counted as text lines. You can't move the cursor into them. To fill the deleted lines with the lines from another buffer use `:diffget` on the line below them.

E787

When the buffer that is about to be modified is read-only and the autocommand that is triggered by `FileChangedRO` changes buffers the command will fail. The autocommand must not change buffers.

The `[bufspec]` argument above can be a buffer number, a pattern for a buffer name or a part of a buffer name. Examples:

```
:diffget          Use the other buffer which is in diff mode
:diffget 3       Use buffer 3
:diffget v2      Use the buffer which matches "v2" and is in
                    diff mode (e.g., "file.c.v2")
```

5. Diff options

diff-options

Also see `'diffopt'` and the "diff" item of `'fillchars'` .

`diff-slow` `diff_translations`

For very long lines, the diff syntax highlighting might be slow, especially since it tries to match all different kind of localisations. To disable localisations and speed up the syntax highlighting, set the global variable `g:diff_translations` to zero:

```
let g:diff_translations = 0
```

After setting this variable, reload the syntax script:

```
set syntax=diff
```

FINDING THE DIFFERENCES

`diff-diffexpr`

The `'diffexpr'` option can be set to use something else than the internal diff support or the standard "diff" program to compare two files and find the differences. [E959](#)

When `'diffexpr'` is empty, Vim uses this command to find the differences between file1 and file2:

```
diff file1 file2 > outfile
```

The ">" is replaced with the value of `'shellredir'`.

The output of "diff" must be a normal "ed" style diff or a unified diff. A context diff will NOT work. For a unified diff no context lines can be used. Using "diff -u" will NOT work, use "diff -U0".

This example explains the format that Vim expects for the "ed" style diff:

```
1a2
> bbb
4d4
< 111
7c7
< GGG
---
> ggg
```

The "1a2" item appends the line "bbb".

The "4d4" item deletes the line "111".

The "7c7" item replaces the line "GGG" with "ggg".

When `'diffexpr'` is not empty, Vim evaluates it to obtain a diff file in the format mentioned. These variables are set to the file names used:

<code>v:fname_in</code>	original file
<code>v:fname_new</code>	new version of the same file
<code>v:fname_out</code>	where to write the resulting diff file

Additionally, **'diffexpr'** should take care of "icase" and "iwhite" in the **'diffopt'** option. **'diffexpr'** cannot change the value of **'lines'** and **'columns'**.

The advantage of using a function call without arguments is that it is faster, see [expr-option-function](#) .

Example (this does almost the same as **'diffexpr'** being empty):

```
set diffexpr=MyDiff()
function MyDiff()
  let opt = ""
  if &diffopt =~ "icase"
    let opt = opt .. "-i "
  endif
  if &diffopt =~ "iwhite"
    let opt = opt .. "-b "
  endif
  silent execute "!diff -a --binary " .. opt .. v:fname_in .. " " .. v:fname_new
    \ " > " .. v:fname_out
  redraw!
endfunction
```

The "-a" argument is used to force comparing the files as text, comparing as binaries isn't useful. The "--binary" argument makes the files read in binary mode, so that a **CTRL-Z** doesn't end the text on DOS.

The ``redraw!`` command may not be needed, depending on whether executing a shell command shows something on the display or not.

If the **'diffexpr'** expression starts with s: or `<SID>` , then it is replaced with the script ID ([local-function](#)). Example:

```
set diffexpr=s:MyDiffExpr()
set diffexpr=<SID>SomeDiffExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

E810 **E97**

Vim will do a test if the diff output looks alright. If it doesn't, you will get an error message. Possible causes:

- The "diff" program cannot be executed.
- The "diff" program doesn't produce normal "ed" style diffs (see above).
- The **'shell'** and associated options are not set correctly. Try if filtering works with a command like `":!sort"`.
- You are using **'diffexpr'** and it doesn't work.

If it's not clear what the problem is set the **'verbose'** option to one or more to see more messages.

The self-installing Vim for MS-Windows includes a diff program. If you don't have it you might want to download a diff.exe. For example from <http://gnuwin32.sourceforge.net/packages/diffutils.htm>.

USING PATCHES

diff-patchexpr

The **'patchexpr'** option can be set to use something else than the standard "patch" program.

When **'patchexpr'** is empty, Vim will call the "patch" program like this:

```
patch -o outfile origfile < patchfile
```

This should work fine with most versions of the "patch" program. **Note** that a CR in the middle of a line may cause problems, it is seen as a line break.

If the default doesn't work for you, set the **'patchexpr'** to an expression that will have the same effect. These variables are set to the file names used:

```
v:fname_in      original file
v:fname_diff    patch file
v:fname_out     resulting patched file
```

The advantage of using a function call without arguments is that it is faster, see [expr-option-function](#) .

Example (this does the same as **'patchexpr'** being empty):

```
set patchexpr=MyPatch()
function MyPatch()
  :call system("patch -o " .. v:fname_out .. " " .. v:fname_in ..
  \ " < " .. v:fname_diff)
endfunction
```

Make sure that using the "patch" program doesn't have unwanted side effects. For example, watch out for additionally generated files, which should be deleted. It should just patch the file and nothing else.

Vim will change directory to "/tmp" or another temp directory before evaluating **'patchexpr'**. This hopefully avoids that files in the current directory are accidentally patched. Vim will also delete files starting with v:fname_in and ending in ".rej" and ".orig".

If the **'patchexpr'** expression starts with s: or **<SID>** , then it is replaced with the script ID ([local-function](#)). Example:

```
set patchexpr=s:MyPatchExpr()
set patchexpr=<SID>SomePatchExpr()
```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

DIFF FUNCTION EXAMPLES

diff-func-examples

Some examples for using the [diff\(\)](#) function to compute the diff indices between two Lists of strings are below.

```
" some lines are changed
:echo diff(['abc', 'def', 'ghi'], ['abx', 'rrr', 'xhi'], {'output': 'indices'})
[{'from_idx': 0, 'from_count': 3, 'to_idx': 0, 'to_count': 3}]
```

```

" few lines added at the beginning
:echo diff(['ghi'], ['abc', 'def', 'ghi'], {'output': 'indices'})
[{'from_idx': 0, 'from_count': 0, 'to_idx': 0, 'to_count': 2}]

" few lines removed from the beginning
:echo diff(['abc', 'def', 'ghi'], ['ghi'], {'output': 'indices'})
[{'from_idx': 0, 'from_count': 2, 'to_idx': 0, 'to_count': 0}]

" few lines added in the middle
:echo diff(['abc', 'jkl'], ['abc', 'def', 'ghi', 'jkl'], {'output': 'indices'})
[{'from_idx': 1, 'from_count': 0, 'to_idx': 1, 'to_count': 2}]

" few lines removed in the middle
:echo diff(['abc', 'def', 'ghi', 'jkl'], ['abc', 'jkl'], {'output': 'indices'})
[{'from_idx': 1, 'from_count': 2, 'to_idx': 1, 'to_count': 0}]

" few lines added at the end
:echo diff(['abc'], ['abc', 'def', 'ghi'], {'output': 'indices'})
[{'from_idx': 1, 'from_count': 0, 'to_idx': 1, 'to_count': 2}]

" few lines removed from the end
:echo diff(['abc', 'def', 'ghi'], ['abc'], {'output': 'indices'})
[{'from_idx': 1, 'from_count': 2, 'to_idx': 1, 'to_count': 0}]

" disjointed changes
:echo diff(['ab', 'def', 'ghi', 'jkl'], ['abc', 'def', 'ghi', 'jk'], {'output': 'indices'})
[{'from_idx': 0, 'from_count': 1, 'to_idx': 0, 'to_count': 1},
 {'from_idx': 3, 'from_count': 1, 'to_idx': 3, 'to_count': 1}]

" disjointed changes with context length 1
:echo diff(['ab', 'def', 'ghi', 'jkl'], ['abc', 'def', 'ghi', 'jk'], {'output': 'indices'})
[{'from_idx': 0, 'from_count': 4, 'to_idx': 0, 'to_count': 4}]

```

Automatic commands [autocommand](#) [autocommands](#)

For a basic explanation, see section [40.3](#) in the user manual.

- | | |
|------------------------------|----------------------------------|
| 1. Introduction | autocmd-intro |
| 2. Defining autocommands | autocmd-define |
| 3. Removing autocommands | autocmd-remove |
| 4. Listing autocommands | autocmd-list |
| 5. Events | autocmd-events |
| 6. Patterns | autocmd-patterns |
| 7. Buffer-local autocommands | autocmd-buflocal |
| 8. Groups | autocmd-groups |
| 9. Executing autocommands | autocmd-execute |
| 10. Using autocommands | autocmd-use |
| 11. Disabling autocommands | autocmd-disable |

1. Introduction

[autocmd-intro](#)

You can specify commands to be executed automatically when reading or writing a file, when entering or leaving a buffer or window, and when exiting Vim. For example, you can create an autocommand to set the **'cindent'** option for files matching *.c. You can also use autocommands to implement advanced features, such as editing compressed files (see [gzip-example](#)). The usual place to put autocommands is in your .vimrc or .exrc file.

[E203](#) [E204](#) [E143](#) [E855](#) [E937](#) [E952](#)

WARNING: Using autocommands is very powerful, and may lead to unexpected side effects. Be careful not to destroy your text.

- It's a good idea to do some testing on an expendable copy of a file first. For example: If you use autocommands to decompress a file when starting to edit it, make sure that the autocommands for compressing when writing work correctly.
- Be prepared for an error halfway through (e.g., disk full). Vim will mostly be able to undo the changes to the buffer, but you may have to clean up the changes to other files by hand (e.g., compress a file that has been decompressed).
- If the BufRead* events allow you to edit a compressed file, the FileRead* events should do the same (this makes recovery possible in some rare cases). It's a good idea to use the same autocommands for the File* and Buf* events when possible.

Recommended use:

- Always use a group, so that it's easy to delete the autocommand.
- Keep the command itself short, call a function to do more work.
- Make it so that the script it is defined in can be sourced several times without the autocommand being repeated.

Example in Vim9 script:

```
autocmd_add({replace: true,
             group:   'DemoGroup',
             event:   'BufEnter',
             pattern: '*.txt',
             cmd:     'call DemoBufEnter()'
             })
```

In legacy script:

```
call autocmd_add(#{replace: v:true,
                  \ group: 'DemoGroup',
                  \ event: 'BufEnter',
                  \ pattern: '*.txt',
                  \ cmd: 'call DemoBufEnter()'
                  \ })
```

2. Defining autocommands

[autocmd-define](#)

```
:au[tocmd] [group] {event} {aupat} [++once] [++nested] {cmd}
                                     :au      :autocmd
Add {cmd} to the list of commands that Vim will
execute automatically on {event} for a file matching
{aupat} autocmd-patterns .
Here {event} cannot be "*". E1155
Note: A quote character is seen as argument to the
:autocmd and won't start a comment.
Vim always adds the {cmd} after existing autocommands,
so that the autocommands execute in the order in which
they were given.
See autocmd-nested for [++nested]. "nested"
(without the ++) can also be used, for backwards
compatibility, but not in Vim9 script. E1078
                                     autocmd-once
If [++once] is supplied the command is executed once,
then removed ("one shot").
```

The special pattern `<buffer>` or `<buffer=N>` defines a buffer-local autocommand. See [autocmd-buflocal](#) .

If the `:autocmd` is in Vim9 script (a script that starts with `:vim9script` and in a `:def` function) then {cmd} will be executed as in Vim9 script. Thus this depends on where the autocmd is defined, not where it is triggered.

```
                                     :autocmd-block
{cmd} can be a block, like with :command, see :command-repl . Example:
  au BufReadPost *.xml {
    setlocal matchpairs+=<:>
    /<start
  }
```

The `autocmd_add()` function can be used to add a list of autocmds and autocmd groups from a Vim script. It is preferred if you have anything that would

require using `:execute` with `:autocmd`.

Note: The `:autocmd` command can only be followed by another command when the `'|'` appears where the pattern is expected. This works:

```
:augroup mine | au! BufRead | augroup END
```

But this sees "augroup" as part of the defined command:

```
:augroup mine | au! BufRead * | augroup END
:augroup mine | au BufRead * set tw=70 | augroup END
```

Instead you can put the group name into the command:

```
:au! mine BufRead *
:au mine BufRead * set tw=70
```

Or use `:execute`:

```
:augroup mine | exe "au! BufRead *" | augroup END
:augroup mine | exe "au BufRead * set tw=70" | augroup END
```

autocmd-expand

Note that special characters (e.g., "%", "<cword>") in the `:autocmd` arguments are not expanded when the autocommand is defined. These will be expanded when the Event is recognized, and the `{cmd}` is executed. The only exception is that `<sfile>` is expanded when the `autocmd` is defined. Example:

```
:au BufNewFile,BufRead *.html so <sfile>:h/html.vim
```

Here Vim expands `<sfile>` to the name of the file containing this line.

`:autocmd` adds to the list of autocommands regardless of whether they are already present. When your `.vimrc` file is sourced twice, the autocommands will appear twice. To avoid this, define your autocommands in a group, so that you can easily clear them:

```
augroup vimrc
  " Remove all vimrc autocommands
  autocmd!
  au BufNewFile,BufRead *.html so <sfile>:h/html.vim
augroup END
```

If you don't want to remove all autocommands, you can instead use a variable to ensure that Vim includes the autocommands only once:

```
:if !exists("autocommands_loaded")
:  let autocommands_loaded = 1
:  au ...
:endif
```

When the `[group]` argument is not given, Vim uses the current group (as defined with `:augroup`); otherwise, Vim uses the group defined with `[group]`. **Note** that `[group]` must have been defined before. You cannot define a new group with `:au group ...`; use `:augroup` for that.

While testing autocommands, you might find the **'verbose'** option to be useful:

```
:set verbose=9
```

This setting makes Vim echo the autocommands as it executes them.

When defining an autocommand in a script, it will be able to call functions

local to the script and use mappings local to the script. When the event is triggered and the command executed, it will run in the context of the script it was defined in. This matters if `<SID>` is used in a command.

When executing the commands, the message from one command overwrites a previous message. This is different from when executing the commands manually. Mostly the screen will not scroll up, thus there is no hit-enter prompt. When one command outputs two messages this can happen anyway.

3. Removing autocommands

`autocmd-remove`

In addition to the below described commands, the `autocmd_delete()` function can be used to remove a list of autocmds and autocmd groups from a Vim script.

```
:au[tocmd]! [group] {event} {aupat} [++once] [++nested] {cmd}
    Remove all autocommands associated with {event} and
    {aupat}, and add the command {cmd}.
    See autocmd-once for [++once].
    See autocmd-nested for [++nested].
```

```
:au[tocmd]! [group] {event} {aupat}
    Remove all autocommands associated with {event} and
    {aupat}.
```

```
:au[tocmd]! [group] * {aupat}
    Remove all autocommands associated with {aupat} for
    all events.
```

```
:au[tocmd]! [group] {event}
    Remove ALL autocommands for {event}.
    Warning: You should not do this without a group for
    BufRead and other common events, it can break
    plugins, syntax highlighting, etc.
```

```
:au[tocmd]! [group]
    Remove ALL autocommands.
    Note: a quote will be seen as argument to the :autocmd
    and won't start a comment.
    Warning: You should normally not do this without a
    group, it breaks plugins, syntax highlighting, etc.
```

When the `[group]` argument is not given, Vim uses the current group (as defined with `":augroup"`); otherwise, Vim uses the group defined with `[group]`.

4. Listing autocommands

`autocmd-list`

```
:au[tocmd] [group] {event} {aupat}
    Show the autocommands associated with {event} and
    {aupat}.
```

```
:au[tocmd] [group] * {aupat}
    Show the autocommands associated with {aupat} for all
    events.
```

```
:au[tocmd] [group] {event}
                Show all autocmds for {event}.
```

```
:au[tocmd] [group]      Show all autocmds.
```

If you provide the `[group]` argument, Vim lists only the autocmds for `[group]`; otherwise, Vim lists the autocmds for ALL groups. **Note** that this argument behavior differs from that for defining and removing autocmds.

In order to list buffer-local autocmds, use a pattern in the form `<buffer>` or `<buffer=N>`. See `autocmd-buflocal`.

The `autocmd_get()` function can be used from a Vim script to get a list of autocmds.

`:autocmd-verbose`

When **'verbose'** is non-zero, listing an autocmd will also display where it was last defined. Example:

```
:verbose autocmd BufEnter
FileExplorer BufEnter
*          call s:LocalBrowse(expand("<amatch>"))
          Last set from /usr/share/vim/vim-7.0/plugin/NetrwPlugin.vim
```

See `:verbose-cmd` for more information.

5. Events

`autocmd-events` `E215` `E216`

You can specify a comma-separated list of event names. No white space can be used in this list. The command applies to all the events in the list.

For READING FILES there are four kinds of events possible:

BufNewFile		starting to edit a non-existent file
BufReadPre	BufReadPost	starting to edit an existing file
FilterReadPre	FilterReadPost	read the temp file with filter output
FileReadPre	FileReadPost	any other file read

Vim uses only one of these four kinds when reading a file. The "Pre" and "Post" events are both triggered, before and after reading the file.

Note that the autocmds for the `*ReadPre` events and all the Filter events are not allowed to change the current buffer (you will get an error message if this happens). This is to prevent the file to be read into the wrong buffer.

Note that the **'modified'** flag is reset AFTER executing the `BufReadPost` and `BufNewFile` autocmds. But when the **'modified'** option was set by the autocmds, this doesn't happen.

You can use the **'eventignore'** option to ignore a number of events or all events.

`autocommand-events` `{event}`

Vim recognizes the following events. Vim ignores the case of event names (e.g., you can use "BURead" or "bufread" instead of "BufRead").

First an overview by function with a short explanation. Then the list alphabetically with full explanations [autocmd-events-abc](#) .

Name	triggered by
Reading	
BufNewFile	starting to edit a file that doesn't exist
BufReadPre	starting to edit a new buffer, before reading the file
BufRead	starting to edit a new buffer, after reading the file
BufReadPost	starting to edit a new buffer, after reading the file
BufReadCmd	before starting to edit a new buffer Cmd-event
FileReadPre	before reading a file with a ":read" command
FileReadPost	after reading a file with a ":read" command
FileReadCmd	before reading a file with a ":read" command Cmd-event
FilterReadPre	before reading a file from a filter command
FilterReadPost	after reading a file from a filter command
StdinReadPre	before reading from stdin into the buffer
StdinReadPost	After reading from the stdin into the buffer
Writing	
BufWrite	starting to write the whole buffer to a file
BufWritePre	starting to write the whole buffer to a file
BufWritePost	after writing the whole buffer to a file
BufWriteCmd	before writing the whole buffer to a file Cmd-event
FileWritePre	starting to write part of a buffer to a file
FileWritePost	after writing part of a buffer to a file
FileWriteCmd	before writing part of a buffer to a file Cmd-event
FileAppendPre	starting to append to a file
FileAppendPost	after appending to a file
FileAppendCmd	before appending to a file Cmd-event
FilterWritePre	starting to write a file for a filter command or diff
FilterWritePost	after writing a file for a filter command or diff
Buffers	
BufAdd	just after adding a buffer to the buffer list
BufCreate	just after adding a buffer to the buffer list
BufDelete	before deleting a buffer from the buffer list
BufWipeout	before completely deleting a buffer
BufFilePre	before changing the name of the current buffer
BufFilePost	after changing the name of the current buffer
BufEnter	after entering a buffer
BufLeave	before leaving to another buffer
BufWinEnter	after a buffer is displayed in a window
BufWinLeave	before a buffer is removed from a window

BufUnload	before unloading a buffer
BufHidden	just before a buffer becomes hidden
BufNew	just after creating a new buffer
SwapExists	detected an existing swap file
Options	
FileType	when the 'filetype' option has been set
Syntax	when the 'syntax' option has been set
EncodingChanged	after the 'encoding' option has been changed
TermChanged	after the value of 'term' has changed
OptionSet	after setting any option
Startup and exit	
VimEnter	after doing all the startup stuff
GUIEnter	after starting the GUI successfully
GUIFailed	after starting the GUI failed
TermResponse	after the terminal response to t_RV is received
TermResponseAll	after the terminal response to t_RV and others is received
QuitPre	when using <code>:quit</code> , before deciding whether to exit
ExitPre	when using a command that may make Vim exit
VimLeavePre	before exiting Vim, before writing the viminfo file
VimLeave	before exiting Vim, after writing the viminfo file
VimSuspend	when suspending Vim
VimResume	when Vim is resumed after being suspended
Terminal	
TerminalOpen	after a terminal buffer was created
TerminalWinOpen	after a terminal buffer was created in a new window
Various	
FileChangedShell	Vim notices that a file changed since editing started
FileChangedShellPost	After handling a file changed since editing started
FileChangedRO	before making the first change to a read-only file
DiffUpdated	after diffs have been updated
DirChangedPre	before the working directory will change
DirChanged	after the working directory has changed
ShellCmdPost	after executing a shell command
ShellFilterPost	after filtering with a shell command
CmdUndefined	a user command is used but it isn't defined
FuncUndefined	a user function is used but it isn't defined
SpellFileMissing	a spell file is used but it can't be found
SourcePre	before sourcing a Vim script
SourcePost	after sourcing a Vim script
SourceCmd	before sourcing a Vim script Cmd-event
VimResized	after the Vim window size changed
FocusGained	Vim got input focus
FocusLost	Vim lost input focus

CursorHold	the user doesn't press a key for a while
CursorHoldI	the user doesn't press a key for a while in Insert mode
CursorMoved	the cursor was moved in Normal mode
CursorMovedI	the cursor was moved in Insert mode
WinNewPre	before creating a new window
WinNew	after creating a new window
TabNew	after creating a new tab page
WinClosed	after closing a window
TabClosed	after closing a tab page
WinEnter	after entering another window
WinLeave	before leaving a window
TabEnter	after entering another tab page
TabLeave	before leaving a tab page
CmdwinEnter	after entering the command-line window
CmdwinLeave	before leaving the command-line window
CmdlineChanged	after a change was made to the command-line text
CmdlineEnter	after the cursor moves to the command line
CmdlineLeave	before the cursor leaves the command line
InsertEnter	starting Insert mode
InsertChange	when typing <Insert> while in Insert or Replace mode
InsertLeave	when leaving Insert mode
InsertLeavePre	just before leaving Insert mode
InsertCharPre	when a character was typed in Insert mode, before inserting it
ModeChanged	after changing the mode
TextChanged	after a change was made to the text in Normal mode
TextChangedI	after a change was made to the text in Insert mode when popup menu is not visible
TextChangedP	after a change was made to the text in Insert mode when popup menu visible
TextChangedT	after a change was made to the text in Terminal mode
TextYankPost	after text has been yanked or deleted
SafeState	nothing pending, going to wait for the user to type a character
SafeStateAgain	repeated SafeState
ColorSchemePre	before loading a color scheme
ColorScheme	after loading a color scheme
RemoteReply	a reply from a server Vim was received
QuickFixCmdPre	before a quickfix command is run
QuickFixCmdPost	after a quickfix command is run
SessionLoadPost	after loading a session file
MenuPopup	just before showing the popup menu
CompleteChanged	after Insert mode completion menu changed

CompleteDonePre	after Insert mode completion is done, before clearing info
CompleteDone	after Insert mode completion is done, after clearing info
User SigUSR1	to be used in combination with ":doautocmd" after the SIGUSR1 signal has been detected
WinScrolled	after scrolling or resizing a window

The alphabetical list of autocommand events: [autocmd-events-abc](#)

	BufCreate BufAdd
BufAdd or BufCreate	Just after creating a new buffer which is added to the buffer list, or adding a buffer to the buffer list. Also used just after a buffer in the buffer list has been renamed. Not triggered for the initial buffers created during startup. The BufCreate event is for historic reasons. NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being created "<afile>".
	BufDelete
BufDelete	Before deleting a buffer from the buffer list. The BufUnload may be called first (if the buffer was loaded). Also used just before a buffer in the buffer list is renamed. NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being deleted "<afile>" and "<abuf>". Don't change to another buffer, it will cause problems.
	BufEnter
BufEnter	After entering a buffer. Useful for setting options for a file type. Also executed when starting to edit a buffer, after the BufReadPost autocommands.
	BufFilePost
BufFilePost	After changing the name of the current buffer with the ":file" or ":saveas" command.
	BufFilePre
BufFilePre	Before changing the name of the current buffer with the ":file" or ":saveas" command.
	BufHidden
BufHidden	Just before a buffer becomes hidden. That is, when there are no longer windows that show the buffer, but the buffer is not unloaded or deleted. Not used for ":qa" or ":q" when exiting Vim. NOTE: When this autocommand is executed, the

	current buffer "%" may be different from the buffer being unloaded "<afile>".
	BufLeave
BufLeave	Before leaving to another buffer. Also when leaving or closing the current window and the new current window is not for the same buffer. Not used for ":qa" or ":q" when exiting Vim.
	BufNew
BufNew	Just after creating a new buffer. Also used just after a buffer has been renamed. When the buffer is added to the buffer list BufAdd will be triggered too. NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being created "<afile>".
	BufNewFile
BufNewFile	When starting to edit a file that doesn't exist. Can be used to read in a skeleton file.
	BufRead BufReadPost
BufRead or BufReadPost	When starting to edit a new buffer, after reading the file into the buffer, before executing the modelines. See BufWinEnter for when you need to do something after processing the modelines. Also triggered: - when writing an unnamed buffer in a way that the buffer gets a name - after successfully recovering a file - for the filetypedetect group when executing ":filetype detect" Not triggered: - for the `:read file` command - when the file doesn't exist
	BufReadCmd
BufReadCmd	Before starting to edit a new buffer. Should read the file into the buffer. Cmd-event
	BufReadPre E200 E201
BufReadPre	When starting to edit a new buffer, before reading the file into the buffer. Not used if the file doesn't exist.
	BufUnload
BufUnload	Before unloading a buffer. This is when the text in the buffer is going to be freed. This may be after a BufWritePost and before a BufDelete. Also used for all buffers that are loaded when Vim is going to exit. NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being unloaded "<afile>". Don't change to another buffer or window, it will cause problems! When exiting and v:dying is 2 or more this event is not triggered.

BufWinEnter	<p style="text-align: right;">BufWinEnter</p> <p>After a buffer is displayed in a window. This can be when the buffer is loaded (after processing the modelines) or when a hidden buffer is displayed in a window (and is no longer hidden).</p> <p>Does not happen for <code>:split</code> without arguments, since you keep editing the same buffer, or <code>:split</code> with a file that's already open in a window, because it re-uses an existing buffer. But it does happen for a <code>:split</code> with the name of the current buffer, since it reloads that buffer.</p> <p>Does not happen for a terminal window, because it starts in Terminal-Job mode and Normal mode commands won't work. Use <code>TerminalOpen</code> instead.</p>
BufWinLeave	<p style="text-align: right;">BufWinLeave</p> <p>Before a buffer is removed from a window. Not when it's still visible in another window. Also triggered when exiting. It's triggered before BufUnload or BufHidden.</p> <p>NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being unloaded "<afile>".</p> <p>When exiting and v:dying is 2 or more this event is not triggered.</p>
BufWipeout	<p style="text-align: right;">BufWipeout</p> <p>Before completely deleting a buffer. The BufUnload and BufDelete events may be called first (if the buffer was loaded and was in the buffer list). Also used just before a buffer is renamed (also when it's not in the buffer list).</p> <p>NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer being deleted "<afile>".</p> <p>Don't change to another buffer, it will cause problems.</p>
BufWrite or BufWritePre	<p style="text-align: right;">BufWrite BufWritePre</p> <p>Before writing the whole buffer to a file.</p>
BufWriteCmd	<p style="text-align: right;">BufWriteCmd</p> <p>Before writing the whole buffer to a file. Should do the writing of the file and reset 'modified' if successful, unless '+' is in 'cpo' and writing to another file <code>cpo++</code>. The buffer contents should not be changed. When the command resets 'modified' the undo information is adjusted to mark older undo states as 'modified', like <code>:write</code> does.</p> <p style="text-align: right;">Cmd-event</p>
BufWritePost	<p style="text-align: right;">BufWritePost</p> <p>After writing the whole buffer to a file (should undo the commands for BufWritePre).</p> <p style="text-align: right;">CmdUndefined</p>

CmdUndefined	<p>When a user command is used but it isn't defined. Useful for defining a command only when it's used. The pattern is matched against the command name. Both <code><amatch></code> and <code><afile></code> are set to the name of the command.</p> <p>NOTE: Autocompletion won't work until the command is defined. An alternative is to always define the user command and have it invoke an autoloading function. See <code>autoload</code>.</p>
CmdlineChanged	<p>After a change was made to the text in the command line. Be careful not to mess up the command line, it may cause Vim to lock up.</p> <p><code><afile></code> is set to a single character, indicating the type of command-line.</p> <p><code>cmdwin-char</code></p>
CmdlineEnter	<p>After moving the cursor to the command line, where the user can type a command or search string; including non-interactive use of ":" in a mapping, but not when using <code><Cmd></code>.</p> <p>The pattern is matched against the character representing the type of command-line.</p> <p><code>cmdwin-char</code></p> <p><code><afile></code> is set to a single character, indicating the type of command-line.</p>
CmdlineLeave	<p>Before leaving the command line; including non-interactive use of ":" in a mapping, but not when using <code><Cmd></code>.</p> <p>Also when abandoning the command line, after typing CTRL-C or <code><Esc></code>.</p> <p>When the commands result in an error the command line is still executed.</p> <p><code><afile></code> is set to a single character, indicating the type of command-line.</p> <p><code>cmdwin-char</code></p>
CmdwinEnter	<p>After entering the command-line window. Useful for setting options specifically for this special type of window.</p> <p><code><afile></code> is set to a single character, indicating the type of command-line.</p> <p><code>cmdwin-char</code></p>
CmdwinLeave	<p>Before leaving the command-line window. Useful to clean up any global setting done with <code>CmdwinEnter</code>.</p> <p><code><afile></code> is set to a single character, indicating the type of command-line.</p> <p><code>cmdwin-char</code></p>
ColorScheme	<p>After loading a color scheme. <code>:colorscheme</code></p> <p>Not triggered if the color scheme is not</p>

found.

The pattern is matched against the colorscheme name. `<afile>` can be used for the name of the actual file where this option was set, and `<amatch>` for the new colorscheme name.

ColorSchemePre	<p>ColorSchemePre</p> <p>Before loading a color scheme. <code>:colorscheme</code> Useful to setup removing things added by a color scheme, before another one is loaded.</p>														
CompleteChanged	<p>CompleteChanged</p> <p>After each time the Insert mode completion menu changed. Not fired on popup menu hide, use <code>CompleteDonePre</code> or <code>CompleteDone</code> for that. Never triggered recursively.</p> <p>Sets these <code>v:event</code> keys:</p> <table><tr><td><code>completed_item</code></td><td>See <code>complete-items</code> .</td></tr><tr><td><code>height</code></td><td>nr of items visible</td></tr><tr><td><code>width</code></td><td>screen cells</td></tr><tr><td><code>row</code></td><td>top screen row</td></tr><tr><td><code>col</code></td><td>leftmost screen column</td></tr><tr><td><code>size</code></td><td>total nr of items</td></tr><tr><td><code>scrollbar</code></td><td>TRUE if visible</td></tr></table> <p>It is not allowed to change the text <code>textlock</code> .</p> <p>The size and position of the popup are also available by calling <code>pum_getpos()</code> .</p>	<code>completed_item</code>	See <code>complete-items</code> .	<code>height</code>	nr of items visible	<code>width</code>	screen cells	<code>row</code>	top screen row	<code>col</code>	leftmost screen column	<code>size</code>	total nr of items	<code>scrollbar</code>	TRUE if visible
<code>completed_item</code>	See <code>complete-items</code> .														
<code>height</code>	nr of items visible														
<code>width</code>	screen cells														
<code>row</code>	top screen row														
<code>col</code>	leftmost screen column														
<code>size</code>	total nr of items														
<code>scrollbar</code>	TRUE if visible														
CompleteDonePre	<p>CompleteDonePre</p> <p>After Insert mode completion is done. Either when something was completed or abandoning completion. <code>ins-completion</code> <code>complete_info()</code> can be used, the info is cleared after triggering CompleteDonePre. The <code>v:completed_item</code> variable contains information about the completed item.</p>														
CompleteDone	<p>CompleteDone</p> <p>After Insert mode completion is done. Either when something was completed or abandoning completion. <code>ins-completion</code> <code>complete_info()</code> cannot be used, the info is cleared before triggering CompleteDone. Use CompleteDonePre if you need it. The <code>v:completed_item</code> variable contains information about the completed item.</p>														
CursorHold	<p>CursorHold</p> <p>When the user doesn't press a key for the time specified with <code>'updatetime'</code>. Not triggered until the user has pressed a key (i.e. doesn't</p>														

fire every **'updatetime'** ms if you leave Vim to make some coffee. :) See [CursorHold-example](#) for previewing tags.

This event is only triggered in Normal mode. It is not triggered when waiting for a command argument to be typed, or a movement after an operator.

While recording the CursorHold event is not triggered. `q`

<CursorHold>

Internally the autocmd is triggered by the **<CursorHold>** key. In an expression mapping `getchar()` may see this character.

Note: Interactive commands cannot be used for this event. There is no hit-enter prompt, the screen is updated directly (when needed).

Note: In the future there will probably be another option to set the time.

Hint: to force an update of the status lines use:

```
:let &ro = &ro
```

```
{only on Amiga, Unix, Win32 and all GUI versions}
```

CursorHoldI

CursorHoldI

Just like CursorHold, but in Insert mode. Not triggered when waiting for another key, e.g. after **CTRL-V**, and not when in **CTRL-X** mode `insert_expand`.

CursorMoved

CursorMoved

After the cursor was moved in Normal or Visual mode. Also when the text of the cursor line has been changed, e.g., with "x", "rx" or "p". Not always triggered when there is typeahead, while executing commands in a script file, when an operator is pending or when moving to another window while remaining at the same cursor position.

For an example see [match-parens](#).

Note: This can not be skipped with ``:noautocmd``.

Careful: This is triggered very often, don't do anything that the user does not expect or that is slow.

CursorMovedI

CursorMovedI

After the cursor was moved in Insert mode. Not triggered when the popup menu is visible. Otherwise the same as CursorMoved.

DiffUpdated

DiffUpdated

After diffs have been updated. Depending on what kind of diff is being used (internal or external) this can be triggered on every change or when doing `:diffupdate`.

DirChangedPre	<p style="text-align: right;">DirChangedPre</p> <p>The working directory is going to be changed, as with DirChanged . The pattern is like with DirChanged . The new directory can be found in v:event.directory.</p>
DirChanged	<p style="text-align: right;">DirChanged</p> <p>The working directory has changed in response to the :cd or :tcd or :lcd commands, or as a result of the 'autochdir' option. The pattern can be:</p> <pre> "window" to trigger on `:lcd` "tabpage" to trigger on `:tcd` "global" to trigger on `:cd` "auto" to trigger on 'autochdir'. "drop" to trigger on editing a file </pre> <p><afile> is set to the new directory name.</p>
EncodingChanged	<p style="text-align: right;">EncodingChanged</p> <p>Fires off after the 'encoding' option has been changed. Useful to set up fonts, for example.</p>
ExitPre	<p style="text-align: right;">ExitPre</p> <p>When using :quit, :wq in a way it makes Vim exit, or using :qall, just after QuitPre . Can be used to close any non-essential window. Exiting may still be cancelled if there is a modified buffer that isn't automatically saved, use VimLeavePre for really exiting.</p>
FileAppendCmd	<p style="text-align: right;">FileAppendCmd</p> <p>Before appending to a file. Should do the appending to the file. Use the '[' and ']' marks for the range of lines. Cmd-event</p>
FileAppendPost	<p style="text-align: right;">FileAppendPost</p> <p>After appending to a file.</p>
FileAppendPre	<p style="text-align: right;">FileAppendPre</p> <p>Before appending to a file. Use the '[' and ']' marks for the range of lines.</p>
FileChangedRO	<p style="text-align: right;">FileChangedRO</p> <p>Before making the first change to a read-only file. Can be used to check-out the file from a source control system. Not triggered when the change was caused by an autocommand. This event is triggered when making the first change in a buffer or the first change after 'readonly' was set, just before the change is applied to the text.</p> <p>WARNING: If the autocommand moves the cursor the effect of the change is undefined.</p> <p style="text-align: right;">E788</p> <p>It is not allowed to change to another buffer here. You can reload the buffer but not edit another one.</p> <p style="text-align: right;">E881</p> <p>If the number of lines changes saving for undo may fail and the change will be aborted.</p>

FileChangedShell	<p style="text-align: right;">FileChangedShell</p> <p>When Vim notices that the modification time of a file has changed since editing started. Also when the file attributes of the file change or when the size of the file changes.</p> <p style="text-align: right;">timestamp</p> <p>Mostly triggered after executing a shell command, but also with a :checktime command or when gvim regains input focus. This autocommand is triggered for each changed file. It is not used when 'autoread' is set and the buffer was not changed. If a FileChangedShell autocommand is present the warning message and prompt is not given. The v:fcs_reason variable is set to indicate what happened and v:fcs_choice can be used to tell Vim what to do next.</p> <p>NOTE: When this autocommand is executed, the current buffer "%" may be different from the buffer that was changed, which is in "<afile>".</p> <p>NOTE: The commands must not change the current buffer, jump to another buffer or delete a buffer. E246 E811</p> <p>NOTE: This event never nests, to avoid an endless loop. This means that while executing commands for the FileChangedShell event no other FileChangedShell event will be triggered.</p>
FileChangedShellPost	<p style="text-align: right;">FileChangedShellPost</p> <p>After handling a file that was changed outside of Vim. Can be used to update the statusline.</p>
FileEncoding	<p style="text-align: right;">FileEncoding</p> <p>Obsolete. It still works and is equivalent to EncodingChanged .</p>
FileReadCmd	<p style="text-align: right;">FileReadCmd</p> <p>Before reading a file with a ":read" command. Should do the reading of the file. Cmd-event</p>
FileReadPost	<p style="text-align: right;">FileReadPost</p> <p>After reading a file with a ":read" command. Note that Vim sets the '[' and ']' marks to the first and last line of the read. This can be used to operate on the lines just read.</p>
FileReadPre	<p style="text-align: right;">FileReadPre</p> <p>Before reading a file with a ":read" command.</p>
FileType	<p style="text-align: right;">FileType</p> <p>When the 'filetype' option has been set. The pattern is matched against the filetype. <afile> can be used for the name of the file where this option was set, and <amatch> for the new value of 'filetype'. Navigating to another window or buffer is not allowed. See filetypes .</p>
FileWriteCmd	<p style="text-align: right;">FileWriteCmd</p> <p>Before writing to a file, when not writing the</p>

whole buffer. Should do the writing to the file. Should not change the buffer. Use the '[' and ']' marks for the range of lines.

`Cmd-event`

`FileWritePost`

FileWritePost After writing to a file, when not writing the whole buffer.

`FileWritePre`

FileWritePre Before writing to a file, when not writing the whole buffer. Use the '[' and ']' marks for the range of lines.

`FilterReadPost`

FilterReadPost After reading a file from a filter command. Vim checks the pattern against the name of the current buffer as with FilterReadPre. Not triggered when '`shelltemp`' is off.

`FilterReadPre` E135

FilterReadPre Before reading a file from a filter command. Vim checks the pattern against the name of the current buffer, not the name of the temporary file that is the output of the filter command. Not triggered when '`shelltemp`' is off.

`FilterWritePost`

FilterWritePost After writing a file for a filter command or making a diff with an external diff (see `DiffUpdated` for internal diff). Vim checks the pattern against the name of the current buffer as with FilterWritePre. Not triggered when '`shelltemp`' is off.

`FilterWritePre`

FilterWritePre Before writing a file for a filter command or making a diff with an external diff. Vim checks the pattern against the name of the current buffer, not the name of the temporary file that is the output of the filter command. Not triggered when '`shelltemp`' is off.

`FocusGained`

FocusGained When Vim got input focus. Only for the GUI version and a few console versions where this can be detected.

`FocusLost`

FocusLost When Vim lost input focus. Only for the GUI version and a few console versions where this can be detected. May also happen when a dialog pops up.

`FuncUndefined`

FuncUndefined When a user function is used but it isn't defined. Useful for defining a function only when it's used. The pattern is matched against the function name. Both `<amatch>` and `<afile>` are set to the name of the function. Not triggered when compiling a `Vim9`

function.

NOTE: When writing Vim scripts a better alternative is to use an autoloading function. See [autoload-functions](#).

	GUIEnter
GUIEnter	After starting the GUI successfully, and after opening the window. It is triggered before VimEnter when using gvim. Can be used to position the window from a .gvimrc file: <pre>:autocmd GUIEnter * winpos 100 50</pre>
	GUIFailed
GUIFailed	After starting the GUI failed. Vim may continue to run in the terminal, if possible (only on Unix and alikes, when connecting the X server fails). You may want to quit Vim: <pre>:autocmd GUIFailed * qall</pre>
	InsertChange
InsertChange	When typing <Insert> while in Insert or Replace mode. The <code>v:insertmode</code> variable indicates the new mode. Be careful not to move the cursor or do anything else that the user does not expect.
	InsertCharPre
InsertCharPre	When a character is typed in Insert mode, before inserting the char. The <code>v:char</code> variable indicates the char typed and can be changed during the event to insert a different character. When <code>v:char</code> is set to more than one character this text is inserted literally. It is not allowed to change the text <code>textlock</code> . The event is not triggered when <code>'paste'</code> is set. {only with the +eval feature}
	InsertEnter
InsertEnter	Just before starting Insert mode. Also for Replace mode and Virtual Replace mode. The <code>v:insertmode</code> variable indicates the mode. Be careful not to do anything else that the user does not expect. The cursor is restored afterwards. If you do not want that set <code>v:char</code> to a non-empty string.
	InsertLeavePre
InsertLeavePre	Just before leaving Insert mode. Also when using CTRL-O <code>i_CTRL-O</code> . Be careful not to change mode or use <code>:normal</code> , it will likely cause trouble.
	InsertLeave
InsertLeave	Just after leaving Insert mode. Also when using CTRL-O <code>i_CTRL-O</code> . But not for <code>i_CTRL-C</code> .
	MenuPopup
MenuPopup	Just before showing the popup menu (under the right mouse button). Useful for adjusting the menu for what is under the cursor or mouse

pointer.

The pattern is matched against one or two characters representing the mode:

n	Normal
v	Visual
o	Operator-pending
i	Insert
c	Command line
tl	Terminal

ModeChanged

ModeChanged

After changing the mode. The pattern is matched against `'old_mode:new_mode'`, for example match against `*:c*` to simulate `CmdlineEnter`.

The following values of `v:event` are set:

old_mode	The mode before it changed.
new_mode	The new mode as also returned by <code>mode()</code> called with a non-zero argument.

When ModeChanged is triggered, `old_mode` will have the value of `new_mode` when the event was last triggered.

This will be triggered on every minor mode change.

Usage example to use relative line numbers when entering Visual mode:

```
:au ModeChanged [vV\x16]*:* let &l:rnu = mode() =~# '[vV\x16]'  
:au ModeChanged *:[vV\x16]* let &l:rnu = mode() =~# '[vV\x16]'  
:au WinEnter,WinLeave * let &l:rnu = mode() =~# '[vV\x16]'
```

OptionSet

OptionSet

After setting an option. The pattern is matched against the long option name.

`<amatch>` indicates what option has been set.

`v:option_type` indicates whether it's global or local scoped.

`v:option_command` indicates what type of set/let command was used (follow the tag to see the table).

`v:option_new` indicates the newly set value.

`v:option_oldlocal` has the old local value.

`v:option_oldglobal` has the old global value.

`v:option_old` indicates the old option value.

`v:option_oldlocal` is only set when `:set` or `:setlocal` or a `modeline` was used to set the option. Similarly `v:option_oldglobal` is only set when `:set` or `:setglobal` was used.

This does not set `<abuf>`, you could use `bufnr()`.

Note that when setting a `global-local` string option with `:set`, then `v:option_old` is the

old global value. However, for all other kinds of options (local string options, global-local number options, ...) it is the old local value.

OptionSet is not triggered on startup and for the **'key'** option for obvious reasons.

Usage example: Check for the existence of the directory in the **'backupdir'** and **'undodir'** options, create the directory if it doesn't exist yet.

Note: It's a bad idea to reset an option during this autocommand, this may break a plugin. You can always use ``:noa`` to prevent triggering this autocommand.

When using `:set` in the autocommand the event is not triggered again.

QuickFixCmdPre

QuickFixCmdPre

Before a quickfix command is run (`:make` , `:lmake` , `:grep` , `:lgrep` , `:grepadd` , `:lgrepadd` , `:vimgrep` , `:lvimgrep` , `:vimgrepadd` , `:lvimgrepadd` , `:cscope` , `:cfile` , `:cgetfile` , `:caddfile` , `:lfile` , `:lgetfile` , `:laddfile` , `:helpgrep` , `:lhelpgrep` , `:cexpr` , `:cgetexpr` , `:caddexpr` , `:cbuffer` , `:cgetbuffer` , `:caddbuffer`).

The pattern is matched against the command being run. When `:grep` is used but **'grepprg'** is set to "internal" it still matches "grep". This command cannot be used to set the **'makeprg'** and **'grepprg'** variables. If this command causes an error, the quickfix command is not executed.

QuickFixCmdPost

QuickFixCmdPost

Like QuickFixCmdPre, but after a quickfix command is run, before jumping to the first location. For `:cfile` and `:lfile` commands it is run after the error file is read and before moving to the first error. See [QuickFixCmdPost-example](#) .

QuitPre

QuitPre

When using ``:quit`` , ``:wq`` or ``:qall`` , before deciding whether it closes the current window or quits Vim. For ``:wq`` the buffer is written before QuitPre is triggered. Can be used to close any non-essential window if the current window is the last ordinary window. Also see [ExitPre](#) .

RemoteReply

RemoteReply

When a reply from a Vim that functions as

server was received `server2client()` . The pattern is matched against the `{serverid}`. `<amatch>` is equal to the `{serverid}` from which the reply was sent, and `<afile>` is the actual reply string.

Note that even if an autocommand is defined, the reply should be read with `remote_read()` to consume it.

SafeState

SafeState

When nothing is pending, going to wait for the user to type a character.

This will not be triggered when:

- an operator is pending
- a register was entered with "r
- halfway executing a command
- executing a mapping
- there is typeahead
- Insert mode completion is active
- Command line completion is active

You can use ``mode()`` to find out what state Vim is in. That may be:

- Visual mode
- Normal mode
- Insert mode
- Command-line mode

Depending on what you want to do, you may also check more with ``state()``, e.g. whether the screen was scrolled for messages.

SafeStateAgain

SafeStateAgain

Like SafeState but after processing any messages and invoking callbacks. This may be triggered often, don't do something that takes time.

SessionLoadPost

SessionLoadPost

After loading the session file created using the `:mksession` command.

ShellCmdPost

ShellCmdPost

After executing a shell command with `!:cmd` , `:shell` , `:make` and `:grep` . Can be used to check for any changed files.

ShellFilterPost

ShellFilterPost

After executing a shell command with `":{range}!cmd"` , `":w !cmd"` or `":r !cmd"`. Can be used to check for any changed files.

SourcePre

SourcePre

Before sourcing a Vim script. `:source` `<afile>` is the name of the file being sourced.

SourcePost

SourcePost

After sourcing a Vim script. `:source` `<afile>` is the name of the file being sourced. Not triggered when sourcing was interrupted. Also triggered after a SourceCmd autocommand was triggered.

SourceCmd	<p style="text-align: right;">SourceCmd</p> <p>When sourcing a Vim script. <code>:source</code> <code><afile></code> is the name of the file being sourced. The autocommand must source this file. Cmd-event</p>												
SpellFileMissing	<p style="text-align: right;">SpellFileMissing</p> <p>When trying to load a spell checking file and it can't be found. The pattern is matched against the language. <code><amatch></code> is the language, 'encoding' also matters. See spell-SpellFileMissing .</p>												
StdinReadPost	<p style="text-align: right;">StdinReadPost</p> <p>After reading from the stdin into the buffer, before executing the modelines. Only used when the "-" argument was used when Vim was started <code>--</code> .</p>												
StdinReadPre	<p style="text-align: right;">StdinReadPre</p> <p>Before reading from stdin into the buffer. Only used when the "-" argument was used when Vim was started <code>--</code> .</p>												
SwapExists	<p style="text-align: right;">SwapExists</p> <p>Detected an existing swap file when starting to edit a file. Only when it is possible to select a way to handle the situation, when Vim would ask the user what to do. The <code>v:swapname</code> variable holds the name of the swap file found, <code><afile></code> the file being edited. <code>v:swapcommand</code> may contain a command to be executed in the opened file. The commands should set the <code>v:swapchoice</code> variable to a string with one character to tell Vim what should be done next:</p> <table style="margin-left: 40px;"> <tr><td>'o'</td><td>open read-only</td></tr> <tr><td>'e'</td><td>edit the file anyway</td></tr> <tr><td>'r'</td><td>recover</td></tr> <tr><td>'d'</td><td>delete the swap file</td></tr> <tr><td>'q'</td><td>quit, don't edit the file</td></tr> <tr><td>'a'</td><td>abort, like hitting CTRL-C</td></tr> </table> <p>When set to an empty string the user will be asked, as if there was no SwapExists autocmd.</p> <p style="text-align: right;">E812</p> <p>It is not allowed to change to another buffer, change a buffer name or change directory here. {only available with the +eval feature}</p>	'o'	open read-only	'e'	edit the file anyway	'r'	recover	'd'	delete the swap file	'q'	quit, don't edit the file	'a'	abort, like hitting CTRL-C
'o'	open read-only												
'e'	edit the file anyway												
'r'	recover												
'd'	delete the swap file												
'q'	quit, don't edit the file												
'a'	abort, like hitting CTRL-C												
Syntax	<p style="text-align: right;">Syntax</p> <p>When the 'syntax' option has been set. The pattern is matched against the syntax name. <code><afile></code> can be used for the name of the file where this option was set, and <code><amatch></code> for the new value of 'syntax'. See <code>:syn-on</code> .</p>												
TabClosed	<p style="text-align: right;">TabClosed</p> <p>After closing a tab page.</p>												

TabEnter	<p style="text-align: right;">TabEnter</p> <p>Just after entering a tab page. <code>tab-page</code> After triggering the WinEnter and before triggering the BufEnter event.</p>
TabLeave	<p style="text-align: right;">TabLeave</p> <p>Just before leaving a tab page. <code>tab-page</code> A WinLeave event will have been triggered first.</p>
TabNew	<p style="text-align: right;">TabNew</p> <p>When a tab page was created. <code>tab-page</code> A WinEnter event will have been triggered first, TabEnter follows.</p>
TermChanged	<p style="text-align: right;">TermChanged</p> <p>After the value of <code>'term'</code> has changed. Useful for re-loading the syntax file to update the colors, fonts and other terminal-dependent settings. Executed for all loaded buffers.</p>
TerminalOpen	<p style="text-align: right;">TerminalOpen</p> <p>Just after a terminal buffer was created, with <code>':terminal'</code> or <code>term_start()</code>. This event is triggered even if the buffer is created without a window, with the <code>++hidden</code> option.</p>
TerminalWinOpen	<p style="text-align: right;">TerminalWinOpen</p> <p>Just after a terminal buffer was created, with <code>':terminal'</code> or <code>term_start()</code>. This event is triggered only if the buffer is created with a window. Can be used to set window local options for the terminal window.</p>
TermResponse	<p style="text-align: right;">TermResponse</p> <p>After the response to <code>t_RV</code> is received from the terminal. The value of <code>v:termresponse</code> can be used to do things depending on the terminal version. Note that this event may be triggered halfway executing another event, especially if file I/O, a shell command or anything else that takes time is involved.</p>
TermResponseAll	<p style="text-align: right;">TermResponseAll</p> <p>After the response to <code>t_RV</code>, <code>t_RC</code>, <code>t_RS</code>, <code>t_RB</code>, <code>t_RF</code>, or <code>t_u7</code> are received from the terminal. The value of <code>v:termresponse</code>, <code>v:termblinkresp</code>, <code>v:termstyleresp</code>, <code>v:termrbgresp</code>, <code>v:termrfgresp</code>, and <code>v:termu7resp</code>, correspondingly, can be used. <code><amatch></code> will be set to any of:</p> <pre> "version", "cursorblink", "cursorshape", "background", "foreground", "ambiguouswidth" </pre> <p>Note that this event may be triggered halfway executing another event, especially if file I/O, a shell command or anything else that takes time is involved.</p>

TextChanged	<p style="text-align: center;">TextChanged</p> <p>After a change was made to the text in the current buffer in Normal mode. That is after <code>b:changedtick</code> has changed (also when that happened before the TextChanged autocommand was defined).</p> <p>Not triggered when there is typeahead or when an operator is pending.</p> <p>Note: This can not be skipped with <code>`:noautocmd`</code>.</p> <p>Careful: This is triggered very often, don't do anything that the user does not expect or that is slow.</p>												
TextChangedI	<p style="text-align: center;">TextChangedI</p> <p>After a change was made to the text in the current buffer in Insert mode.</p> <p>Not triggered when the popup menu is visible. Otherwise the same as TextChanged.</p>												
TextChangedP	<p style="text-align: center;">TextChangedP</p> <p>After a change was made to the text in the current buffer in Insert mode, only when the popup menu is visible. Otherwise the same as TextChanged.</p>												
TextChangedT	<p style="text-align: center;">TextChangedT</p> <p>After a change was made to the text in the current buffer in Terminal mode.</p> <p>Otherwise the same as TextChanged.</p>												
TextYankPost	<p style="text-align: center;">TextYankPost</p> <p>After text has been yanked or deleted in the current buffer. The following values of <code>v:event</code> can be used to determine the operation that triggered this autocmd:</p> <table> <tr> <td style="padding-right: 10px;">inclusive</td> <td>TRUE if the motion is <code>inclusive</code> else the motion is <code>exclusive</code>.</td> </tr> <tr> <td>operator</td> <td>The operation performed.</td> </tr> <tr> <td>regcontents</td> <td>Text that was stored in the register, as a list of lines, like with: <code>getreg(r, 1, 1)</code></td> </tr> <tr> <td>regname</td> <td>Name of the register or empty string for the unnamed register, see <code>registers</code>.</td> </tr> <tr> <td>regtype</td> <td>Type of the register, see <code>getregtype()</code>.</td> </tr> <tr> <td>visual</td> <td>True if the operation is performed on a <code>Visual</code> area.</td> </tr> </table> <p>Not triggered when <code>quote_</code> is used nor when called recursively.</p> <p>It is not allowed to change the buffer text, see <code>textlock</code>. E1064 {only when compiled with the +eval feature}</p>	inclusive	TRUE if the motion is <code>inclusive</code> else the motion is <code>exclusive</code> .	operator	The operation performed.	regcontents	Text that was stored in the register, as a list of lines, like with: <code>getreg(r, 1, 1)</code>	regname	Name of the register or empty string for the unnamed register, see <code>registers</code> .	regtype	Type of the register, see <code>getregtype()</code> .	visual	True if the operation is performed on a <code>Visual</code> area.
inclusive	TRUE if the motion is <code>inclusive</code> else the motion is <code>exclusive</code> .												
operator	The operation performed.												
regcontents	Text that was stored in the register, as a list of lines, like with: <code>getreg(r, 1, 1)</code>												
regname	Name of the register or empty string for the unnamed register, see <code>registers</code> .												
regtype	Type of the register, see <code>getregtype()</code> .												
visual	True if the operation is performed on a <code>Visual</code> area.												

User

User	<p>Never executed automatically. To be used for autocommands that are only executed with <code>":doautocmd"</code>.</p> <p>Note that when <code>`:doautocmd User MyEvent`</code> is used while there are no matching autocommands, you will get an error. If you don't want that, either check whether an autocommand is defined using <code>`exists('#User#MyEvent')`</code> or define a dummy autocommand yourself.</p> <p>Example:</p> <pre> if exists('#User#MyEvent') doautocmd User MyEvent endif </pre>
SigUSR1	<p style="text-align: right; color: magenta;">SigUSR1</p> <p>After the SIGUSR1 signal has been detected. Could be used if other ways of notifying Vim are not feasible. E.g. to check for the result of a build that takes a long time, or when a motion sensor is triggered.</p> <p><code>{only on Unix}</code></p>
UserGettingBored	<p style="text-align: right; color: magenta;">UserGettingBored</p> <p>When the user presses the same key 42 times. Just kidding! :-)</p>
VimEnter	<p style="text-align: right; color: magenta;">VimEnter</p> <p>After doing all the startup stuff, including loading <code>.vimrc</code> files, executing the <code>"-c cmd"</code> arguments, creating all windows and loading the buffers in them.</p> <p>Just before this event is triggered the <code>v:vim_did_enter</code> variable is set, so that you can do:</p> <pre> if v:vim_did_enter call s:init() else au VimEnter * call s:init() endif </pre>
VimLeave	<p style="text-align: right; color: magenta;">VimLeave</p> <p>Before exiting Vim, just after writing the <code>.viminfo</code> file. Executed only once, like <code>VimLeavePre</code>.</p> <p>To detect an abnormal exit use <code>v:dying</code>.</p> <p>When <code>v:dying</code> is 2 or more this event is not triggered.</p> <p>To get the exit code use <code>v:exiting</code>.</p>
VimLeavePre	<p style="text-align: right; color: magenta;">VimLeavePre</p> <p>Before exiting Vim, just before writing the <code>.viminfo</code> file. This is executed only once, if there is a match with the name of what happens to be the current buffer when exiting. Mostly useful with a <code>"*"</code> pattern.</p> <pre> :autocmd VimLeavePre * call CleanupStuff() </pre> <p>To detect an abnormal exit use <code>v:dying</code>.</p>

	<p>When <code>v:dying</code> is 2 or more this event is not triggered.</p> <p>To get the exit code use <code>v:exiting</code>.</p>
VimResized	<p><code>VimResized</code></p> <p>After the Vim window was resized, thus <code>'lines'</code> and/or <code>'columns'</code> changed. Not when starting up though.</p>
VimResume	<p><code>VimResume</code></p> <p>When the Vim instance is resumed after being suspended and <code>VimSuspend</code> was triggered. Useful for triggering <code>:checktime</code> and ensure the buffers content did not change while Vim was suspended:</p> <pre>:autocmd VimResume * checktime</pre>
VimSuspend	<p><code>VimSuspend</code></p> <p>When the Vim instance is suspended. Only when CTRL-Z was typed inside Vim, or when the SIGTSTP signal was sent to Vim, but not for SIGSTOP.</p>
WinClosed	<p><code>WinClosed</code></p> <p>When closing a window, just before it is removed from the window layout. The pattern is matched against the <code>window-ID</code>. Both <code><amatch></code> and <code><afile></code> are set to the <code>window-ID</code>. Non-recursive (event cannot trigger itself).</p>
WinEnter	<p><code>WinEnter</code></p> <p>After entering another window. Not done for the first window, when Vim has just started. Useful for setting the window height. If the window is for another buffer, Vim executes the BufEnter autocommands after the WinEnter autocommands.</p> <p>Note: For split and tabpage commands the WinEnter event is triggered after the split or tab command but before the file is loaded.</p>
WinLeave	<p><code>WinLeave</code></p> <p>Before leaving a window. If the window to be entered next is for a different buffer, Vim executes the BufLeave autocommands before the WinLeave autocommands (but not for <code>:"new"</code>). Not used for <code>:"qa"</code> or <code>:"q"</code> when exiting Vim.</p>
WinNewPre	<p><code>WinNewPre</code></p> <p>Before creating a new window. Triggered before commands that modify window layout by creating a split or new tab page. Not done for the first window, when Vim has just started. It is not allowed to modify window layout while executing commands for the WinNewPre event.</p> <p>Most useful to store current window layout and compare it with the new layout after the Window has been created.</p>

WinNew	<p style="text-align: right;">WinNew</p> <p>When a new window was created. Not done for the first window, when Vim has just started. Before a WinEnter event.</p>
WinScrolled	<p style="text-align: right;">WinScrolled</p> <p>After any window in the current tab page scrolled the text (horizontally or vertically) or changed width or height. See win-scrolled-resized .</p> <p>The pattern is matched against the <code>window-ID</code> of the first window that scrolled or resized. Both <code><amatch></code> and <code><afile></code> are set to the <code>window-ID</code> .</p> <p><code>v:event</code> is set with information about size and scroll changes. WinScrolled-event</p> <p>Only starts triggering after startup finished and the first screen redraw was done. Does not trigger when defining the first WinScrolled or WinResized event, but may trigger when adding more.</p> <p>Non-recursive: the event will not trigger while executing commands for the WinScrolled event. However, if the command causes a window to scroll or change size, then another WinScrolled event will be triggered later.</p>
WinResized	<p style="text-align: right;">WinResized</p> <p>After a window in the current tab page changed width or height. See win-scrolled-resized .</p> <p><code>v:event</code> is set with information about size changes. WinResized-event</p> <p>Same behavior as WinScrolled for the pattern, triggering and recursiveness.</p>

=====

6. Patterns **autocmd-patterns {aupat}**

The `{aupat}` argument of ``:autocmd`` can be a comma-separated list. This works as if the command was given with each pattern separately. Thus this command:

```
:autocmd BufRead *.txt,*.info set et
```

Is equivalent to:

```
:autocmd BufRead *.txt set et
:autocmd BufRead *.info set et
```

The file pattern `{aupat}` is tested for a match against the file name in one of two ways:

1. When there is no '/' in the pattern, Vim checks for a match against only the tail part of the file name (without its leading directory path).
2. When there is a '/' in the pattern, Vim checks for a match against both the short file name (as you typed it) and the full file name (after expanding it to a full path and resolving symbolic links).

The special pattern `<buffer>` or `<buffer=N>` is used for buffer-local autocommands `autocmd-buflocal`. This pattern is not matched against the name of a buffer.

Examples:

```
:autocmd BufRead *.txt      set et
```

Set the **'et'** option for all text files.

```
:autocmd BufRead /vim/src/*.c  set cindent
```

Set the **'cindent'** option for C files in the `/vim/src` directory.

```
:autocmd BufRead /tmp/*.c      set ts=5
```

If you have a link from `"/tmp/test.c"` to `"/home/nobody/vim/src/test.c"`, and you start editing `"/tmp/test.c"`, this autocommand will match.

Note: To match part of a path, but not from the root directory, use a '*' as the first character. Example:

```
:autocmd BufRead */doc/*.txt   set tw=78
```

This autocommand will for example be executed for `"/tmp/doc/xx.txt"` and `"/usr/home/piet/doc/yy.txt"`. The number of directories does not matter here.

The file name that the pattern is matched against is after expanding wildcards. Thus if you issue this command:

```
:e $ROOTDIR/main.$EXT
```

The argument is first expanded to:

```
/usr/root/main.py
```

Before it's matched with the pattern of the autocommand. Careful with this when using events like `FileReadCmd`, the value of `<amatch>` may not be what you expect.

Environment variables can be used in a pattern:

```
:autocmd BufRead $VIMRUNTIME/doc/*.txt set expandtab
```

And `~` can be used for the home directory (if `$HOME` is defined):

```
:autocmd BufWritePost ~/.vimrc so ~/.vimrc
```

```
:autocmd BufRead ~archive/*      set readonly
```

The environment variable is expanded when the autocommand is defined, not when the autocommand is executed. This is different from the command!

file-pattern

The pattern is interpreted like mostly used in file names:

- * matches any sequence of characters; Unusual: includes path separators
- ? matches any single character
- \? matches a '?'

```

.      matches a '.'
~      matches a '~'
,      separates patterns
\,     matches a ','
{ }    like \( \) in a pattern
,      inside { }: like \| in a pattern
\}     literal }
\{     literal {
\\{n,m} like \{n,m} in a pattern
\      special meaning like in a pattern
[ch]   matches 'c' or 'h'
[^ch]  match any character but 'c' and 'h'

```

Note that for all systems the '/' character is used for path separator (even for MS-Windows). This was done because the backslash is difficult to use in a pattern and to make the autocommands portable across different systems.

It is possible to use `pattern` items, but they may not work as expected, because of the translation done for the above.

autocmd-changes

Matching with the pattern is done when an event is triggered. Changing the buffer name in one of the autocommands, or even deleting the buffer, does not change which autocommands will be executed. Example:

```

au BufEnter *.foo bdel
au BufEnter *.foo set modified

```

This will delete the current buffer and then set **'modified'** in what has become the current buffer instead. Vim doesn't take into account that "*.foo" doesn't match with that buffer name. It matches "*.foo" with the name of the buffer at the moment the event was triggered.

However, buffer-local autocommands will not be executed for a buffer that has been wiped out with `:bwipe`. After deleting the buffer with `:bdel` the buffer actually still exists (it becomes unlisted), thus the autocommands are still executed.

=====

```

7. Buffer-local autocommands      autocmd-buflocal  autocmd-buffer-local
                                <buffer=N>      <buffer=abuf>  E680

```

Buffer-local autocommands are attached to a specific buffer. They are useful if the buffer does not have a name and when the name does not match a specific pattern. But it also means they must be explicitly added to each buffer.

Instead of a pattern buffer-local autocommands use one of these forms:

```

<buffer>      current buffer
<buffer=99>   buffer number 99
<buffer=abuf> using <abuf> (only when executing autocommands)
               <abuf>

```

Examples:

```

:au CursorHold <buffer> echo 'hold'

```

```

:au CursorHold <buffer=33> echo 'hold'
:au BufNewFile * au CursorHold <buffer=abuf> echo 'hold'

```

All the commands for autocommands also work with buffer-local autocommands, simply use the special string instead of the pattern. Examples:

```

:au! * <buffer>           " remove buffer-local autocommands for
                          " current buffer
:au! * <buffer=33>       " remove buffer-local autocommands for
                          " buffer #33
:bufdo :au! CursorHold <buffer> " remove autocmd for given event for all
                          " buffers
:au * <buffer>           " list buffer-local autocommands for
                          " current buffer

```

Note that when an autocommand is defined for the current buffer, it is stored with the buffer number. Thus it uses the form "<buffer=12>", where 12 is the number of the current buffer. You will see this when listing autocommands, for example.

To test for presence of buffer-local autocommands use the `exists()` function as follows:

```

:if exists("#CursorHold#<buffer=12>") | ... | endif
:if exists("#CursorHold#<buffer>") | ... | endif " for current buffer

```

When a buffer is wiped out its buffer-local autocommands are also gone, of course. **Note** that when deleting a buffer, e.g., with `:"bdel"`, it is only unlisted, the autocommands are still present. In order to see the removal of buffer-local autocommands:

```

:set verbose=6

```

It is not possible to define buffer-local autocommands for a non-existent buffer.

8. Groups autocmd-groups

Autocommands can be put together in a group. This is useful for removing or executing a group of autocommands. For example, all the autocommands for syntax highlighting are put in the "highlight" group, to be able to execute `:"doautoall highlight BufRead"` when the GUI starts.

When no specific group is selected, Vim uses the default group. The default group does not have a name. You cannot execute the autocommands from the default group separately; you can execute them only by executing autocommands for all groups.

Normally, when executing autocommands automatically, Vim uses the autocommands for all groups. The group only matters when executing autocommands with `:"doautocmd"` or `:"doautoall"`, or when defining or deleting autocommands.

The group name can contain any characters except white space. The group name "end" is reserved (also in uppercase).

The group name is case sensitive. **Note** that this is different from the event

name!

```
                                :aug   :augroup
:aug[roup] {name}               Define the autocmd group name for the
                                following ":autocmd" commands. The name "end"
                                or "END" selects the default group.
                                To avoid confusion, the name should be
                                different from existing {event} names, as this
                                most likely will not do what you intended.
```

```
                                :augroup-delete  E367  W19  E936
:aug[roup]! {name}             Delete the autocmd group {name}. Don't use
                                this if there is still an autocommand using
                                this group! You will get a warning if doing
                                it anyway. When the group is the current
                                group you will get error E936.
```

To enter autocommands for a specific group, use this method:

1. Select the group with ":augroup {name}".
2. Delete any old autocommands with ":au!".
3. Define the autocommands.
4. Go back to the default group with "augroup END".

Example:

```
:augroup uncompress
:  au!
:  au BufEnter *.gz      %!gunzip
:augroup END
```

This prevents having the autocommands defined twice (e.g., after sourcing the .vimrc file again).

FileExplorer

There is one group that is recognized by Vim: FileExplorer. If this group exists Vim assumes that editing a directory is possible and will trigger a plugin that lists the files in that directory. This is used by the `netrw` plugin. This allows you to do:

```
  browse edit
```

9. Executing autocommands

autocmd-execute

Vim can also execute Autocommands non-automatically. This is useful if you have changed autocommands, or when Vim has executed the wrong autocommands (e.g., the file pattern match was wrong).

Note that the `'eventignore'` option applies here too. Events listed in this option will not cause any commands to be executed.

```
                                :do     :doau   :doaut  :doautocmd  E217
:do[autocmd] [<nomodeline>] [group] {event} [fname]
                                Apply the autocommands matching [fname] (default:
                                current file name) for {event} to the current buffer.
                                You can use this when the current file name does not
```

match the right pattern, after changing settings, or to execute autocommands for a certain event. It's possible to use this inside an autocommand too, so you can base the autocommands for one extension on another extension. Example:

```
:au BufEnter *.cpp so ~/.vimrc_cpp
:au BufEnter *.cpp doau BufEnter x.c
```

Be careful to avoid endless loops. See [autocmd-nested](#) .

When the `[group]` argument is not given, Vim executes the autocommands for all groups. When the `[group]` argument is included, Vim executes only the matching autocommands for that group. **Note:** if you use an undefined group name, Vim gives you an error message.

`<nomodeline>`

After applying the autocommands the modelines are processed, so that their settings overrule the settings from autocommands, like what happens when editing a file. This is skipped when the `<nomodeline>` argument is present. You probably want to use `<nomodeline>` for events that are not used when loading a buffer, such as `User` .

Processing modelines is also skipped when no matching autocommands were executed.

`:doautoa` `:doautoall`

`:doautoa[ll] [<nomodeline>] [group] {event} [fname]`

Like `":doautocmd"`, but apply the autocommands to each loaded buffer. The current buffer is done last.

Note that `[fname]` is used to select the autocommands, not the buffers to which they are applied. Example:

```
augroup mine
  autocmd!
  autocmd FileType * echo expand('<amatch>')
augroup END
doautoall mine FileType Loaded-Buffer
```

Sourcing this script, you'll see as many "Loaded-Buffer" echoed as there are loaded buffers.

Careful: Don't use this for autocommands that delete a buffer, change to another buffer or change the contents of a buffer; the result is unpredictable. This command is intended for autocommands that set options, change highlighting, and things like that.

10. Using autocommands

`autocmd-use`

For WRITING FILES there are four possible sets of events. Vim uses only one of these sets for a write command:

BufWriteCmd	BufWritePre	BufWritePost	writing the whole buffer
-------------	-------------	--------------	--------------------------

	FilterWritePre	FilterWritePost	writing to filter temp file
FileAppendCmd	FileAppendPre	FileAppendPost	appending to a file
FileWriteCmd	FileWritePre	FileWritePost	any other file write

When there is a matching "*Cmd" autocmd, it is assumed it will do the writing. No further writing is done and the other events are not triggered.

Cmd-event

Note that the *WritePost commands should undo any changes to the buffer that were caused by the *WritePre commands; otherwise, writing the file will have the side effect of changing the buffer.

Before executing the autocmds, the buffer from which the lines are to be written temporarily becomes the current buffer. Unless the autocmds change the current buffer or delete the previously current buffer, the previously current buffer is made the current buffer again.

The *WritePre and *AppendPre autocmds must not delete the buffer from which the lines are to be written.

The '[' and ']' marks have a special position:

- Before the *ReadPre event the '[' mark is set to the line just above where the new lines will be inserted.
- Before the *ReadPost event the '[' mark is set to the first line that was just read, the ']' mark to the last line.
- Before executing the *WriteCmd, *WritePre and *AppendPre autocmds the '[' mark is set to the first line that will be written, the ']' mark to the last line.

Careful: '[' and ']' change when using commands that change the buffer.

In commands which expect a file name, you can use "<file>" for the file name that is being read :<file> (you can also use "%" for the current file name). "<buf>" can be used for the buffer number of the currently effective buffer. This also works for buffers that don't have a name. But it doesn't work for files without a buffer (e.g., with ":r file").

gzip-example

Examples for reading and writing compressed files:

```

:augroup gzip
: autocmd!
: autocmd BufReadPre,FileReadPre *.gz set bin
: autocmd BufReadPost,FileReadPost *.gz '[,']!gunzip
: autocmd BufReadPost,FileReadPost *.gz set nobin
: autocmd BufReadPost,FileReadPost *.gz execute ":doautocmd BufReadPost " .. expand("%")
: autocmd BufWritePost,FileWritePost *.gz !mv <file> <file>:r
: autocmd BufWritePost,FileWritePost *.gz !gzip <file>:r

: autocmd FileAppendPre *.gz !gunzip <file>
: autocmd FileAppendPre *.gz !mv <file>:r <file>
: autocmd FileAppendPost *.gz !mv <file> <file>:r
: autocmd FileAppendPost *.gz !gzip <file>:r
:augroup END

```

The "gzip" group is used to be able to delete any existing autocmds with

":autocmd!", for when the file is sourced twice.

("<afile>:r" is the file name without the extension, see `:_%:`)

The commands executed for the BufNewFile, BufRead/BufReadPost, BufWritePost, FileAppendPost and VimLeave events do not set or reset the changed flag of the buffer. When you decompress the buffer with the BufReadPost autocommands, you can still exit with ":q". When you use ":undo" in BufWritePost to undo the changes made by BufWritePre commands, you can still do ":q" (this also makes "ZZ" work). If you do want the buffer to be marked as modified, set the **'modified'** option.

To execute Normal mode commands from an autocommand, use the ":normal" command. Use with care! If the Normal mode command is not finished, the user needs to type characters (e.g., after ":normal m" you need to type a mark name).

If you want the buffer to be unmodified after changing it, reset the **'modified'** option. This makes it possible to exit the buffer with ":q" instead of ":q!".

autocmd-nested E218

By default, autocommands do not nest. For example, if you use ":e" or ":w" in an autocommand, Vim does not execute the BufRead and BufWrite autocommands for those commands. If you do want this, use the "nested" flag for those commands in which you want nesting. For example:

```
:autocmd FileChangedShell *.c ++nested e!
```

The nesting is limited to 10 levels to get out of recursive loops.

It's possible to use the ":au" command in an autocommand. This can be a self-modifying command! This can be useful for an autocommand that should execute only once.

If you want to skip autocommands for one command, use the `:noautocmd` command modifier or the **'eventignore'** option.

Note: When reading a file (with ":read file" or with a filter command) and the last line in the file does not have an <EOL>, Vim remembers this. At the next write (with ":write file" or with a filter command), if the same line is written again as the last line in a file AND **'binary'** is set, Vim does not supply an <EOL>. This makes a filter command on the just read lines write the same file as was read, and makes a write command on just filtered lines write the same file as was read from the filter. For example, another way to write a compressed file:

```
:autocmd FileWritePre *.gz set bin|'[,']!gzip
:autocmd FileWritePost *.gz undo|set nobin
```

autocommand-pattern

You can specify multiple patterns, separated by commas. Here are some examples:

```
:autocmd BufRead * set tw=79 nocin ic infercase fo=2croq
:autocmd BufRead .letter set tw=72 fo=2tcrq
```

```

:autocmd BufEnter .letter      set dict=/usr/lib/dict/words
:autocmd BufLeave  .letter      set dict=
:autocmd BufRead,BufNewFile    *.c,*.h set tw=0 cin noic
:autocmd BufEnter *.c,*.h      abbr FOR for (i = 0; i < 3; ++i)<CR>{<CR>}<Esc>O
:autocmd BufLeave  *.c,*.h      unabbr FOR

```

For makefiles (makefile, Makefile, imakefile, makefile.unix, etc.):

```

:autocmd BufEnter ?akefile*    set include=^s\=include
:autocmd BufLeave  ?akefile*    set include&

```

To always start editing C files at the first function:

```

:autocmd BufRead *.c,*.h      1;/^{

```

Without the "1;" above, the search would start from wherever the file was entered, rather than from the start of the file.

skeleton template

To read a skeleton (template) file when opening a new file:

```

:autocmd BufNewFile *.c        0r ~/vim/skeleton.c
:autocmd BufNewFile *.h        0r ~/vim/skeleton.h
:autocmd BufNewFile *.java     0r ~/vim/skeleton.java

```

To insert the current date and time in a *.html file when writing it:

```

:autocmd BufWritePre,FileWritePre *.html  ks|call LastMod()|'s
:fun LastMod()
:  if line("$") > 20
:    let l = 20
:  else
:    let l = line("$")
:  endif
:  exe "1," .. l .. "g/Last modified: /s/Last modified: ./Last modified: " ..
:  \ strftime("%Y %b %d")
:endifun

```

You need to have a line "Last modified: <date time>" in the first 20 lines of the file for this to work. Vim replaces <date time> (and anything in the same line after it) with the current date and time. Explanation:

```

ks          mark current position with mark 's'
call LastMod() call the LastMod() function to do the work
's         return the cursor to the old position

```

The LastMod() function checks if the file is shorter than 20 lines, and then uses the ":g" command to find lines that contain "Last modified: ". For those lines the ":s" command is executed to replace the existing date with the current one. The ":execute" command is used to be able to use an expression for the ":g" and ":s" commands. The date is obtained with the strftime() function. You can change its argument to get another date string.

When entering :autocmd on the command-line, completion of events and command names may be done (with <Tab>, **CTRL-D**, etc.) where appropriate.

Vim executes all matching autocommands in the order that you specify them. It is recommended that your first autocommand be used for all files by using "*" as the file pattern. This means that you can define defaults you like here for any settings, and if there is another matching autocommand it will override these. But if there is no other matching autocommand, then at least your default settings are recovered (if entering this file from another for which autocommands did match). **Note** that "*" will also match files starting with ".", unlike Unix shells.

autocmd-searchpat

Autocommands do not change the current search patterns. Vim saves the current search patterns before executing autocommands then restores them after the autocommands finish. This means that autocommands do not affect the strings highlighted with the **'hlsearch'** option. Within autocommands, you can still use search patterns normally, e.g., with the "n" command. If you want an autocommand to set the search pattern, such that it is used after the autocommand finishes, use the ":let @/ =" command. The search-highlighting cannot be switched off with ":nohlsearch" in an autocommand. Use the 'h' flag in the **'viminfo'** option to disable search-highlighting when starting Vim.

Cmd-event

When using one of the "*Cmd" events, the matching autocommands are expected to do the file reading, writing or sourcing. This can be used when working with a special kind of file, for example on a remote system. **CAREFUL:** If you use these events in a wrong way, it may have the effect of making it impossible to read or write the matching files! Make sure you test your autocommands properly. Best is to use a pattern that will never match a normal file name, for example "ftp://*".

When defining a BufReadCmd it will be difficult for Vim to recover a crashed editing session. When recovering from the original file, Vim reads only those parts of a file that are not found in the swap file. Since that is not possible with a BufReadCmd, use the **:preserve** command to make sure the original file isn't needed for recovery. You might want to do this only when you expect the file to be modified.

For file read and write commands the **v:cmdarg** variable holds the "++enc=" and "++ff=" argument that are effective. These should be used for the command that reads/writes the file. The **v:cmdbang** variable is one when "!" was used, zero otherwise.

See the \$VIMRUNTIME/plugin/netrwPlugin.vim for examples.

=====

11. Disabling autocommands autocmd-disable

To disable autocommands for some time use the **'eventignore'** option. **Note** that this may cause unexpected behavior, make sure you restore **'eventignore'** afterwards, using a **:try** block with **:finally** .

:noautocmd **:noa**

To disable autocommands for just one command use the ":noautocmd" command modifier. This will set **'eventignore'** to "all" for the duration of the

following command. Example:

```
:noautocmd w fname.gz
```

This will write the file without triggering the autocommands defined by the gzip plugin.

Note that some autocommands are not triggered right away, but only later. This specifically applies to `CursorMoved` and `TextChanged` .

VIM REFERENCE MANUAL by Bram Moolenaar

Expression evaluation [expression](#) [expr](#) [E15](#) [eval](#)
[E1002](#)

Using expressions is introduced in chapter 41 of the user manual [usr_41.txt](#) .

Note: Expression evaluation can be disabled at compile time. If this has been done, the features in this document are not available. See [+eval](#) and [no-eval-feature](#) .

This file is mainly about the backwards compatible (legacy) Vim script. For specifics of Vim9 script, which can execute much faster, supports type checking and much more, see [vim9.txt](#) . Where the syntax or semantics differ a remark is given.

- 1. Variables [variables](#)
 - 1.1 Variable types
 - 1.2 Function references [Funcref](#)
 - 1.3 Lists [Lists](#)
 - 1.4 Dictionaries [Dictionaries](#)
 - 1.5 Blobs [Blobs](#)
 - 1.6 More about variables [more-variables](#)
- 2. Expression syntax [expression-syntax](#)
- 3. Internal variable [internal-variables](#)
- 4. Builtin Functions [functions](#)
- 5. Defining functions [user-functions](#)
- 6. Curly braces names [curly-braces-names](#)
- 7. Commands [expression-commands](#)
- 8. Exception handling [exception-handling](#)
- 9. Examples [eval-examples](#)
- 10. Vim script version [vimscript-version](#)
- 11. No +eval feature [no-eval-feature](#)
- 12. The sandbox [eval-sandbox](#)
- 13. Textlock [textlock](#)
- 14. Vim script library [vim-script-library](#)

Testing support is documented in [testing.txt](#) .
Profiling is documented at [profiling](#) .

=====

- 1. Variables [variables](#)
 - 1.1 Variable types
 - [E712](#) [E896](#) [E897](#) [E899](#) [E1098](#)
 - [E1107](#) [E1135](#) [E1138](#)

There are ten types of variables:

		Number Integer
Number	A 32 or 64 bit signed number. expr-number	
	The number of bits is available in v:numbersize .	

Examples: -123 0x10 0177 0o177 0b1011

Float	A floating point number. floating-point-format Float Examples: 123.456 1.15e-6 -1.1e3
String	A NUL terminated string of 8-bit unsigned characters (bytes). expr-string Examples: "ab\txx\ --" 'x-z''a,c'
List	An ordered sequence of items, see List for details. Example: [1, 2, ['a', 'b']]
Dictionary	An associative, unordered array: Each entry has a key and a value. Dictionary Examples: <pre>{'blue': "#0000ff", 'red': "#ff0000"} #{blue: "#0000ff", red: "#ff0000"}</pre>
Funcref	A reference to a function Funcref . Example: function("strlen") It can be bound to a dictionary and arguments, it then works like a Partial. Example: function("Callback", [arg], myDict)
Special	v:false , v:true , v:none and v:null . Special
Job	Used for a job, see job_start() . Job Jobs
Channel	Used for a channel, see ch_open() . Channel Channels
Blob	Binary Large Object. Stores any sequence of bytes. See Blob for details Example: 0zFF00ED015DAF 0z is an empty Blob.

The Number and String types are converted automatically, depending on how they are used.

Conversion from a Number to a String is by making the ASCII representation of the Number. Examples:

```
Number 123    -->   String "123"  
Number 0     -->   String "0"  
Number -1    -->   String "-1"
```

[octal](#)

Conversion from a String to a Number only happens in legacy Vim script, not in Vim9 script. It is done by converting the first digits to a number.

Hexadecimal "0xf9", Octal "017" or "0o17", and Binary "0b10" numbers are recognized

NOTE: when using [Vim9](#) script or [scriptversion-4](#) [octal](#) with a leading "0" is not recognized. The 0o notation requires patch 8.2.0886.

If the String doesn't start with digits, the result is zero.

Examples:

```
String "456"  -->   Number 456  
String "6bar" -->   Number 6  
String "foo"  -->   Number 0
```

```
String "0xf1" --> Number 241
String "0100" --> Number 64
String "0o100" --> Number 64
String "0b101" --> Number 5
String "-8" --> Number -8
String "+8" --> Number 0
```

To force conversion from String to Number, add zero to it:

```
:echo "0100" + 0
64
```

To avoid a leading zero to cause octal conversion, or for using a different base, use `str2nr()` .

TRUE FALSE Boolean

For boolean operators Numbers are used. Zero is FALSE, non-zero is TRUE. You can also use `v:false` and `v:true` , in Vim9 script `false` and `true` . When TRUE is returned from a function it is the Number one, FALSE is the number zero.

Note that in the command:

```
:if "foo"
:" NOT executed
```

"foo" is converted to 0, which means FALSE. If the string starts with a non-zero number it means TRUE:

```
:if "8foo"
:" executed
```

To test for a non-empty string, use `empty()`:

```
:if !empty("foo")
```

falsy truthy

An expression can be used as a condition, ignoring the type and only using whether the value is "sort of true" or "sort of false". Falsy is:

```
the number zero
empty string, blob, list or dictionary
```

Other values are truthy. Examples:

```
0      falsy
1      truthy
-1     truthy
0.0    falsy
0.1    truthy
''     falsy
'x'    truthy
[]     falsy
[0]    truthy
{}     falsy
#{x: 1} truthy
0z     falsy
0z00   truthy
```

non-zero-arg

Function arguments often behave slightly different from **TRUE** : If the argument is present and it evaluates to a non-zero Number, `v:true` or a non-empty String, then the value is considered to be TRUE.

Note that " " and "0" are also non-empty strings, thus considered to be TRUE. A List, Dictionary or Float is not a Number or String, thus evaluate to FALSE.

E611 E745 E728 E703 E729 E730 E731 E908 E910
E913 E974 E975 E976 E1319 E1320 E1321 E1322
E1323 E1324

List , Dictionary , Funcref , Job , Channel , Blob , Class and object types are not automatically converted.

E805 E806 E808
When mixing Number and Float the Number is converted to Float. Otherwise there is no automatic conversion of Float. You can use str2float() for String to Float, printf() for Float to String and float2nr() for Float to Number.

E362 E891 E892 E893 E894 E907 E911 E914
When expecting a Float a Number can also be used, but nothing else.

no-type-checking
You will not get an error if you try to change the type of a variable.

1.2 Function references

E695 E718 E1192
A Funcref variable is obtained with the function() function, the funcref() function, (in Vim9 script) the name of a function, or created with the lambda expression expr-lambda . It can be used in an expression in the place of a function name, before the parenthesis around the arguments, to invoke the function it refers to. Example in Vim9 script:

```
:var Fn = MyFunc  
:echo Fn()
```

Legacy script:

```
:let Fn = function("MyFunc")  
:echo Fn()
```

E704 E705 E707
A Funcref variable must start with a capital, "s:", "w:", "t:" or "b:". You can use "g:" but the following name must still start with a capital. You cannot have both a Funcref variable and a function with the same name.

A special case is defining a function and directly assigning its Funcref to a Dictionary entry. Example:

```
:function dict.init() dict  
:  let self.val = 0  
:endfunction
```

The key of the Dictionary can start with a lower case letter. The actual function name is not used here. Also see numbered-function .

A Funcref can also be used with the :call command:

```
:call Fn()  
:call dict.init()
```

The name of the referenced function can be obtained with string() .

```
:let func = string(Fn)
```

You can use `call()` to invoke a Funcref and use a list variable for the arguments:

```
:let r = call(Fn, mylist)
```

Partial

A Funcref optionally binds a Dictionary and/or arguments. This is also called a Partial. This is created by passing the Dictionary and/or arguments to `function()` or `funcref()`. When calling the function the Dictionary and/or arguments will be passed to the function. Example:

```
let Cb = function('Callback', ['foo'], myDict)
call Cb('bar')
```

This will invoke the function as if using:

```
call myDict.Callback('foo', 'bar')
```

This is very useful when passing a function around, e.g. in the arguments of `ch_open()` .

Note that binding a function to a Dictionary also happens when the function is a member of the Dictionary:

```
let myDict.myFunction = MyFunction
call myDict.myFunction()
```

Here `MyFunction()` will get `myDict` passed as "self". This happens when the "myFunction" member is accessed. When making assigning "myFunction" to `otherDict` and calling it, it will be bound to `otherDict`:

```
let otherDict.myFunction = myDict.myFunction
call otherDict.myFunction()
```

Now "self" will be "otherDict". But when the dictionary was bound explicitly this won't happen:

```
let myDict.myFunction = function(MyFunction, myDict)
let otherDict.myFunction = myDict.myFunction
call otherDict.myFunction()
```

Here "self" will be "myDict", because it was bound explicitly.

1.3 Lists

[list](#) [List](#) [Lists](#) [E686](#)

A List is an ordered sequence of items. An item can be of any type. Items can be accessed by their index number. Items can be added and removed at any position in the sequence.

List creation

[E696](#) [E697](#)

A List is created with a comma-separated list of items in square brackets.

Examples:

```
:let mylist = [1, two, 3, "four"]
:let emptylist = []
```

An item can be any expression. Using a List for an item creates a List of Lists:

```
:let nestlist = [[11, 12], [21, 22], [31, 32]]
```

An extra comma after the last item is ignored.

List index

[list-index](#) E684

An item in the List can be accessed by putting the index in square brackets after the List. Indexes are zero-based, thus the first item has index zero.

```
:let item = mylist[0]      " get the first item: 1
:let item = mylist[2]      " get the third item: 3
```

When the resulting item is a list this can be repeated:

```
:let item = nestlist[0][1] " get the first list, second item: 12
```

A negative index is counted from the end. Index -1 refers to the last item in the List, -2 to the last but one item, etc.

```
:let last = mylist[-1]     " get the last item: "four"
```

To avoid an error for an invalid index use the `get()` function. When an item is not available it returns zero or the default value you specify:

```
:echo get(mylist, idx)
:echo get(mylist, idx, "NONE")
```

List concatenation

[list-concatenation](#)

Two lists can be concatenated with the "+" operator:

```
:let longlist = mylist + [5, 6]
:let longlist = [5, 6] + mylist
```

To prepend or append an item, turn it into a list by putting [] around it.

A list can be concatenated with another one in-place using `:let+=` or `extend()` :

```
:let mylist += [7, 8]
:call extend(mylist, [7, 8])
```

See [list-modification](#) below for more about changing a list in-place.

Sublist

[sublist](#)

A part of the List can be obtained by specifying the first and last index, separated by a colon in square brackets:

```
:let shortlist = mylist[2:-1] " get List [3, "four"]
```

Omitting the first index is similar to zero. Omitting the last index is similar to -1.

```

:let endlis = mylist[2:]      " from item 2 to the end: [3, "four"]
:let shortlist = mylist[2:2]  " List with one item: [3]
:let otherlist = mylist[:]    " make a copy of the List

```

Notice that the last index is inclusive. If you prefer using an exclusive index use the `slice()` method.

If the first index is beyond the last item of the List or the second item is before the first item, the result is an empty list. There is no error message.

If the second index is equal to or greater than the length of the list the length minus one is used:

```

:let mylist = [0, 1, 2, 3]
:echo mylist[2:8]          " result: [2, 3]

```

NOTE: `mylist[s:e]` means using the variable "s:e" as index. Watch out for using a single letter variable before the ":". Insert a space when needed: `mylist[s : e]`.

List identity

list-identity

When variable "aa" is a list and you assign it to another variable "bb", both variables refer to the same list. Thus changing the list "aa" will also change "bb":

```

:let aa = [1, 2, 3]
:let bb = aa
:call add(aa, 4)
:echo bb
[1, 2, 3, 4]

```

Making a copy of a list is done with the `copy()` function. Using `[:]` also works, as explained above. This creates a shallow copy of the list: Changing a list item in the list will also change the item in the copied list:

```

:let aa = [[1, 'a'], 2, 3]
:let bb = copy(aa)
:call add(aa, 4)
:let aa[0][1] = 'aaa'
:echo aa
[[1, aaa], 2, 3, 4]
:echo bb
[[1, aaa], 2, 3]

```

To make a completely independent list use `deepcopy()`. This also makes a copy of the values in the list, recursively. Up to a hundred levels deep.

The operator "is" can be used to check if two variables refer to the same List. "isnot" does the opposite. In contrast "==" compares if two lists have the same value.

```

:let alist = [1, 2, 3]
:let blist = [1, 2, 3]
:echo alist is blist

```

0

```
:echo alist == blist
1
```

Note about comparing lists: Two lists are considered equal if they have the same length and all items compare equal, as with using "==". There is one exception: When comparing a number with a string they are considered different. There is no automatic type conversion, as with using "==" on variables. Example:

```
echo 4 == "4"
1
echo [4] == ["4"]
0
```

Thus comparing Lists is more strict than comparing numbers and strings. You can compare simple values this way too by putting them in a list:

```
:let a = 5
:let b = "5"
:echo a == b
1
:echo [a] == [b]
0
```

List unpack

To unpack the items in a list to individual variables, put the variables in square brackets, like list items:

```
:let [var1, var2] = mylist
```

When the number of variables does not match the number of items in the list this produces an error. To handle any extra items from the list append ";" and a variable name:

```
:let [var1, var2; rest] = mylist
```

This works like:

```
:let var1 = mylist[0]
:let var2 = mylist[1]
:let rest = mylist[2:]
```

Except that there is no error if there are only two items. "rest" will be an empty list then.

List modification

list-modification

To change a specific item of a list use `:let` this way:

```
:let list[4] = "four"
:let listlist[0][3] = item
```

To change part of a list you can specify the first and last item to be modified. The value must at least have the number of items in the range:

```
:let list[3:5] = [3, 4, 5]
```


To add items to a List in-place, you can use `:let+=` (`list-concatenation`):

```
:let listA = [1, 2]
:let listA += [3, 4]
```

When two variables refer to the same List, changing one List in-place will cause the referenced List to be changed in-place:

```
:let listA = [1, 2]
:let listB = listA
:let listB += [3, 4]
:echo listA
[1, 2, 3, 4]
```

Adding and removing items from a list is done with functions. Here are a few examples:

```
:call insert(list, 'a')           " prepend item 'a'
:call insert(list, 'a', 3)        " insert item 'a' before list[3]
:call add(list, "new")           " append String item
:call add(list, [1, 2])          " append a List as one new item
:call extend(list, [1, 2])       " extend the list with two more items
:let i = remove(list, 3)         " remove item 3
:unset list[3]                  " idem
:let l = remove(list, 3, -1)     " remove items 3 to last item
:unset list[3 : ]               " idem
:call filter(list, 'v:val !~ "x"') " remove items with an 'x'
```

Changing the order of items in a list:

```
:call sort(list)                 " sort a list alphabetically
:call reverse(list)             " reverse the order of items
:call uniq(sort(list))          " sort and remove duplicates
```

For loop

The `:for` loop executes commands for each item in a List, String or Blob. A variable is set to each item in sequence. Example with a List:

```
:for item in mylist
:  call Doit(item)
:endfor
```

This works like:

```
:let index = 0
:while index < len(mylist)
:  let item = mylist[index]
:  :call Doit(item)
:  let index = index + 1
:endwhile
```

If all you want to do is modify each item in the list then the `map()` function will be a simpler method than a for loop.

Just like the `:let` command, `:for` also accepts a list of variables. This requires the argument to be a List of Lists.

```
:for [lnum, col] in [[1, 3], [2, 8], [3, 0]]
:  call Doit(lnum, col)
```

```
:endfor
```

This works like a `:let` command is done for each list item. Again, the types must remain the same to avoid an error.

It is also possible to put remaining items in a List variable:

```
:for [i, j; rest] in listlist
:  call Doit(i, j)
:  if !empty(rest)
:    echo "remainder: " .. string(rest)
:  endif
:endif
:endfor
```

For a Blob one byte at a time is used.

For a String one character, including any composing characters, is used as a String. Example:

```
for c in text
  echo 'This character is ' .. c
endfor
```

List functions

E714

Functions that are useful with a List:

```
:let r = call(funcname, list) " call a function with an argument list
:if empty(list) " check if list is empty
:let l = len(list) " number of items in list
:let big = max(list) " maximum value in list
:let small = min(list) " minimum value in list
:let xs = count(list, 'x') " count nr of times 'x' appears in list
:let i = index(list, 'x') " index of first 'x' in list
:let lines = getline(1, 10) " get ten text lines from buffer
:call append('$', lines) " append text lines in buffer
:let list = split("a b c") " create list from items in a string
:let string = join(list, ', ') " create string from list items
:let s = string(list) " String representation of list
:call map(list, "'>> " .. v:val') " prepend ">> " to each item
```

Don't forget that a combination of features can make things simple. For example, to add up all the numbers in a list:

```
:exe 'let sum = ' .. join(nrlist, '+')
```

1.4 Dictionaries

dict Dict Dictionaries Dictionary

A Dictionary is an associative array: Each entry has a key and a value. The entry can be located with the key. The entries are stored without a specific ordering.

Dictionary creation

E720 E721 E722 E723

A Dictionary is created with a comma-separated list of entries in curly

braces. Each entry has a key and a value, separated by a colon. Each key can only appear once. Examples:

```
:let mydict = {1: 'one', 2: 'two', 3: 'three'}
:let emptydict = {}
```

E713 E716 E717

A key is always a String. You can use a Number, it will be converted to a String automatically. Thus the String '4' and the number 4 will find the same entry. **Note** that the String '04' and the Number 04 are different, since the Number will be converted to the String '4', leading zeros are dropped. The empty string can also be used as a key.

In **Vim9** script a literal key can be used if it consists only of alphanumeric characters, underscore and dash, see [vim9-literal-dict](#).

literal-Dict #{}

To avoid having to put quotes around every key the #{} form can be used in legacy script. This does require the key to consist only of ASCII letters, digits, '-' and '_'. Example:

```
:let mydict = #{zero: 0, one_key: 1, two-key: 2, 333: 3}
```

Note that 333 here is the string "333". Empty keys are not possible with #{}. In **Vim9** script the #{} form cannot be used because it can be confused with the start of a comment.

A value can be any expression. Using a Dictionary for a value creates a nested Dictionary:

```
:let nestdict = {1: {11: 'a', 12: 'b'}, 2: {21: 'c'}}
```

An extra comma after the last entry is ignored.

Accessing entries

The normal way to access an entry is by putting the key in square brackets:

```
:let val = mydict["one"]
:let mydict["four"] = 4
```

You can add new entries to an existing Dictionary this way, unlike Lists.

For keys that consist entirely of letters, digits and underscore the following form can be used [expr-entry](#) :

```
:let val = mydict.one
:let mydict.four = 4
```

Since an entry can be any type, also a List and a Dictionary, the indexing and key lookup can be repeated:

```
:echo dict.key[idx].key
```

Dictionary to List conversion

You may want to loop over the entries in a dictionary. For this you need to turn the Dictionary into a List and pass it to [:for](#).

Most often you want to loop over the keys, using the [keys\(\)](#) function:

```
:for key in keys(mydict)
```

```
: echo key .. ': ' .. mydict[key]
:endifor
```

The List of keys is unsorted. You may want to sort them first:

```
:for key in sort(keys(mydict))
```

To loop over the values use the `values()` function:

```
:for v in values(mydict)
: echo "value: " .. v
:endifor
```

If you want both the key and the value use the `items()` function. It returns a List in which each item is a List with two items, the key and the value:

```
:for [key, value] in items(mydict)
: echo key .. ': ' .. value
:endifor
```

Dictionary identity

dict-identity

Just like Lists you need to use `copy()` and `deepcopy()` to make a copy of a Dictionary. Otherwise, assignment results in referring to the same Dictionary:

```
:let onedict = {'a': 1, 'b': 2}
:let adict = onedict
:let adict['a'] = 11
:echo onedict['a']
11
```

Two Dictionaries compare equal if all the key-value pairs compare equal. For more info see `list-identity`.

Dictionary modification

dict-modification

To change an already existing entry of a Dictionary, or to add a new entry, use `:let` this way:

```
:let dict[4] = "four"
:let dict['one'] = item
```

Removing an entry from a Dictionary is done with `remove()` or `:unlet`.

Three ways to remove the entry with key "aaa" from dict:

```
:let i = remove(dict, 'aaa')
:unlet dict.aaa
:unlet dict['aaa']
```

Merging a Dictionary with another is done with `extend()`:

```
:call extend(adict, bdict)
```

This extends `adict` with all entries from `bdict`. Duplicate keys cause entries in `adict` to be overwritten. An optional third argument can change this.

Note that the order of entries in a Dictionary is irrelevant, thus don't expect `:echo adict` to show the items from `bdict` after the older entries in `adict`.

Weeding out entries from a Dictionary can be done with `filter()` :

```
:call filter(dict, 'v:val =~ "x"')
```

This removes all entries from "dict" with a value not matching 'x'.

This can also be used to remove all entries:

```
call filter(dict, 0)
```

In some situations it is not allowed to remove or add entries to a Dictionary. Especially when iterating over all the entries. You will get [E1313](#) or another error in that case.

Dictionary function

[Dictionary-function](#) [self](#) [E725](#) [E862](#)

When a function is defined with the "dict" attribute it can be used in a special way with a dictionary. Example:

```
:function Mylen() dict
:  return len(self.data)
:endifunction
:let mydict = {'data': [0, 1, 2, 3], 'len': function("Mylen")}
:echo mydict.len()
```

This is like a method in object oriented programming. The entry in the Dictionary is a [Funcref](#) . The local variable "self" refers to the dictionary the function was invoked from. When using [Vim9](#) script you can use classes and objects, see `:class``.

It is also possible to add a function without the "dict" attribute as a [Funcref](#) to a Dictionary, but the "self" variable is not available then.

[numbered-function](#) [anonymous-function](#)

To avoid the extra name for the function it can be defined and directly assigned to a Dictionary in this way:

```
:let mydict = {'data': [0, 1, 2, 3]}
:function mydict.len()
:  return len(self.data)
:endifunction
:echo mydict.len()
```

The function will then get a number and the value of `dict.len` is a [Funcref](#) that references this function. The function can only be used through a [Funcref](#) . It will automatically be deleted when there is no [Funcref](#) remaining that refers to it.

It is not necessary to use the "dict" attribute for a numbered function.

If you get an error for a numbered function, you can find out what it is with a trick. Assuming the function is 42, the command is:

```
:function g:42
```

Functions for Dictionaries

[E715](#)

Functions that can be used with a Dictionary:

```
:if has_key(dict, 'foo') " TRUE if dict has entry with key "foo"
```

```

:if empty(dict)           " TRUE if dict is empty
:let l = len(dict)       " number of items in dict
:let big = max(dict)     " maximum value in dict
:let small = min(dict)  " minimum value in dict
:let xs = count(dict, 'x') " count nr of times 'x' appears in dict
:let s = string(dict)    " String representation of dict
:call map(dict, '">> " .. v:val') " prepend ">> " to each item

```

1.5 Blobs

blob Blob Blobs E978

A Blob is a binary object. It can be used to read an image from a file and send it over a channel, for example.

A Blob mostly behaves like a `List` of numbers, where each number has the value of an 8-bit byte, from 0 to 255.

Blob creation

A Blob can be created with a `blob-literal` :

```
:let b = 0zFF00ED015DAF
```

Dots can be inserted between bytes (pair of hex characters) for readability, they don't change the value:

```
:let b = 0zFF00.ED01.5DAF
```

A blob can be read from a file with `readfile()` passing the `{type}` argument set to "B", for example:

```
:let b = readfile('image.png', 'B')
```

A blob can be read from a channel with the `ch_readblob()` function.

Blob index

blob-index E979

A byte in the Blob can be accessed by putting the index in square brackets after the Blob. Indexes are zero-based, thus the first byte has index zero.

```

:let myblob = 0z00112233
:let byte = myblob[0]           " get the first byte: 0x00
:let byte = myblob[2]         " get the third byte: 0x22

```

A negative index is counted from the end. Index -1 refers to the last byte in the Blob, -2 to the last but one byte, etc.

```
:let last = myblob[-1]        " get the last byte: 0x33
```

To avoid an error for an invalid index use the `get()` function. When an item is not available it returns -1 or the default value you specify:

```

:echo get(myblob, idx)
:echo get(myblob, idx, 999)

```

Blob iteration

The `:for` loop executes commands for each byte of a Blob. The loop variable is

set to each byte in the Blob. Example:

```
:for byte in 0z112233
:  call Doit(byte)
:endifor
```

This calls Doit() with 0x11, 0x22 and 0x33.

Blob concatenation

[blob-concatenation](#)

Two blobs can be concatenated with the "+" operator:

```
:let longblob = myblob + 0z4455
:let longblob = 0z4455 + myblob
```

A blob can be concatenated with another one in-place using `:let+=` :

```
:let myblob += 0z6677
```

See [blob-modification](#) below for more about changing a blob in-place.

Part of a blob

A part of the Blob can be obtained by specifying the first and last index, separated by a colon in square brackets:

```
:let myblob = 0z00112233
:let shortblob = myblob[1:2]    " get 0z1122
:let shortblob = myblob[2:-1]   " get 0z2233
```

Omitting the first index is similar to zero. Omitting the last index is similar to -1.

```
:let endblob = myblob[2:]       " from item 2 to the end: 0z2233
:let shortblob = myblob[2:2]    " Blob with one byte: 0z22
:let otherblob = myblob[:]      " make a copy of the Blob
```

If the first index is beyond the last byte of the Blob or the second index is before the first index, the result is an empty Blob. There is no error message.

If the second index is equal to or greater than the length of the list the length minus one is used:

```
:echo myblob[2:8]              " result: 0z2233
```

Blob modification

[blob-modification](#) [E1182](#) [E1184](#)

To change a specific byte of a blob use `:let` this way:

```
:let blob[4] = 0x44
```

When the index is just one beyond the end of the Blob, it is appended. Any higher index is an error.

To change a sequence of bytes the `[:]` notation can be used:

```
let blob[1:3] = 0z445566
```

The length of the replaced bytes must be exactly the same as the value provided. [E972](#)

To change part of a blob you can specify the first and last byte to be modified. The value must have the same number of bytes in the range:

```
:let blob[3:5] = 0z334455
```

To add items to a Blob in-place, you can use `:let+=` (`blob-concatenation`):

```
:let blobA = 0z1122  
:let blobA += 0z3344
```

When two variables refer to the same Blob, changing one Blob in-place will cause the referenced Blob to be changed in-place:

```
:let blobA = 0z1122  
:let blobB = blobA  
:let blobB += 0z3344  
:echo blobA  
0z11223344
```

You can also use the functions `add()` , `remove()` and `insert()` .

Blob identity

Blobs can be compared for equality:

```
if blob == 0z001122
```

And for equal identity:

```
if blob is otherblob
```

`blob-identity` E977

When variable "aa" is a Blob and you assign it to another variable "bb", both variables refer to the same Blob. Then the "is" operator returns true.

When making a copy using `[:]` or `copy()` the values are the same, but the identity is different:

```
:let blob = 0z112233  
:let blob2 = blob  
:echo blob == blob2  
1  
:echo blob is blob2  
1  
:let blob3 = blob[:]  
:echo blob == blob3  
1  
:echo blob is blob3  
0
```

Making a copy of a Blob is done with the `copy()` function. Using `[:]` also works, as explained above.

1.6 More about variables

`more-variables`

If you need to know the type of a variable or expression, use the `type()` function.

When the '!' flag is included in the '`viminfo`' option, global variables that

start with an uppercase letter, and don't contain a lowercase letter, are stored in the viminfo file [viminfo-file](#) .

When the '[sessionoptions](#)' option contains "global", global variables that start with an uppercase letter and contain at least one lowercase letter are stored in the session file [session-file](#) .

variable name	can be stored where
my_var_6	not
My_Var_6	session file
MY_VAR_6	viminfo file

In legacy script it is possible to form a variable name with curly braces, see [curly-braces-names](#) .

2. Expression syntax

[expression-syntax](#)
E1143

Expression syntax summary, from least to most significant:

expr1	<code>expr2</code> <code>expr2 ? expr1 : expr1</code>	if-then-else
expr2	<code>expr3</code> <code>expr3 expr3 ...</code>	logical OR
expr3	<code>expr4</code> <code>expr4 && expr4 ...</code>	logical AND
expr4	<code>expr5</code> <code>expr5 == expr5</code> <code>expr5 != expr5</code> <code>expr5 > expr5</code> <code>expr5 >= expr5</code> <code>expr5 < expr5</code> <code>expr5 <= expr5</code> <code>expr5 =~ expr5</code> <code>expr5 !~ expr5</code> <code>expr5 ==? expr5</code> <code>expr5 ==# expr5</code> etc.	equal not equal greater than greater than or equal smaller than smaller than or equal regex matches regex doesn't match equal, ignoring case equal, match case As above, append ? for ignoring case, # for matching case
	<code>expr5 is expr5</code> <code>expr5 isnot expr5</code>	same List , Dictionary or Blob instance different List , Dictionary or Blob instance
expr5	<code>expr6</code> <code>expr6 << expr6</code> <code>expr6 >> expr6</code>	bitwise left shift bitwise right shift
expr6	<code>expr7</code>	

	expr7 + expr7 ...	number addition, list or blob concatenation
	expr7 - expr7 ...	number subtraction
	expr7 . expr7 ...	string concatenation
	expr7 .. expr7 ...	string concatenation
expr7	expr8	
	expr8 * expr8 ...	number multiplication
	expr8 / expr8 ...	number division
	expr8 % expr8 ...	number modulo
expr8	expr9	
	<type>expr9	type check and conversion (Vim9 only)
expr9	expr10	
	! expr9	logical NOT
	- expr9	unary minus
	+ expr9	unary plus
expr10	expr11	
	expr10[expr1]	byte of a String or item of a List
	expr10[expr1 : expr1]	substring of a String or sublist of a List
	expr10.name	entry in a Dictionary
	expr10(expr1, ...)	function call with Funcref variable
	expr10->name(expr1, ...)	method call
expr11	number	number constant
	"string"	string constant, backslash is special
	'string'	string constant, ' is doubled
	[expr1, ...]	List
	{expr1: expr1, ...}	Dictionary
	#{key: expr1, ...}	legacy Dictionary
	&option	option value
	(expr1)	nested expression
	variable	internal variable
	va{ria}ble	internal variable with curly braces
	\$VAR	environment variable
	@r	contents of register 'r'
	function(expr1, ...)	function call
	func{ti}on(expr1, ...)	function call with curly braces
	{args -> expr1}	legacy lambda expression
	(args) => expr1	Vim9 lambda expression

"..." indicates that the operations in this level can be concatenated.

Example:

```
&nu || &list && &shell == "csh"
```

All expressions within one level are parsed from left to right.

Expression nesting is limited to 1000 levels deep (300 when build with MSVC) to avoid running out of stack and crashing. E1169

expr1 expr1 ternary falsy-operator ?? E109

The ternary operator: `expr2 ? expr1 : expr1`
The falsy operator: `expr2 ?? expr1`

Ternary operator

In legacy script the expression before the '?' is evaluated to a number. If it evaluates to `TRUE`, the result is the value of the expression between the '?' and ':', otherwise the result is the value of the expression after the ':'.
:'

In `Vim9` script the first expression must evaluate to a boolean, see `vim9-boolean`.

Example:

```
:echo lnum == 1 ? "top" : lnum
```

Since the first expression is an "expr2", it cannot contain another ?:. The other two expressions can, thus allow for recursive use of ?:.

Example:

```
:echo lnum == 1 ? "top" : lnum == 1000 ? "last" : lnum
```

To keep this readable, using `line-continuation` is suggested:

```
:echo lnum == 1  
:\      ? "top"  
:\      : lnum == 1000  
:\      ? "last"  
:\      : lnum
```

You should always put a space before the ':', otherwise it can be mistaken for use in a variable such as "a:1".

Falsy operator

This is also known as the "null coalescing operator", but that's too complicated, thus we just call it the falsy operator.

The expression before the '??' is evaluated. If it evaluates to `truthy`, this is used as the result. Otherwise the expression after the '??' is evaluated and used as the result. This is most useful to have a default value for an expression that may result in zero or empty:

```
echo theList ?? 'list is empty'  
echo GetName() ?? 'unknown'
```

These are similar, but not equal:

```
expr2 ?? expr1  
expr2 ? expr2 : expr1
```

In the second line "expr2" is evaluated twice. And in `Vim9` script the type of expr2 before "?" must be a boolean.

expr2 and expr3

expr2 expr3

```

expr3 || expr3 ..      logical OR      expr-barbar
expr4 && expr4 ..      logical AND      expr-&&

```

The "||" and "&&" operators take one argument on each side.

In legacy script the arguments are (converted to) Numbers.

In Vim9 script the values must be boolean, see vim9-boolean . Use "!!" to convert any type to a boolean.

The result is:

input		output	
n1	n2	n1 n2	n1 && n2
FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE

The operators can be concatenated, for example:

```
&nu || &list && &shell == "csh"
```

Note that "&&" takes precedence over "||", so this has the meaning of:

```
&nu || (&list && &shell == "csh")
```

Once the result is known, the expression "short-circuits", that is, further arguments are not evaluated. This is like what happens in C. For example:

```
let a = 1
echo a || b
```

This is valid even if there is no variable called "b" because "a" is TRUE , so the result must be TRUE . Similarly below:

```
echo exists("b") && b == "yes"
```

This is valid whether "b" has been defined or not. The second clause will only be evaluated if "b" has been defined.

```
expr4 expr4 E1153
-----
```

```
expr5 {cmp} expr5
```

Compare two expr5 expressions. In legacy script the result is a 0 if it evaluates to false, or 1 if it evaluates to true. In Vim9 script the result is true or false .

```

expr==      expr!=      expr->      expr->=
expr-<      expr-<=      expr==~      expr-!~
expr==#      expr-!=#      expr->#      expr->=#

```

	<code>expr-<#</code>	<code>expr-<=#</code>	<code>expr==~#</code>	<code>expr-!~#</code>
	<code>expr==#?</code>	<code>expr-!#?</code>	<code>expr->?</code>	<code>expr->=?</code>
	<code>expr-<?</code>	<code>expr-<=?</code>	<code>expr==~?</code>	<code>expr-!~?</code>
	<code>expr-is</code>	<code>expr-isnot</code>	<code>expr-is#</code>	<code>expr-isnot#</code>
	<code>expr-is?</code>	<code>expr-isnot?</code>	E1072	
	use <code>'ignorecase'</code>	match case	ignore case	
equal	<code>==</code>	<code>==#</code>	<code>==?</code>	
not equal	<code>!=</code>	<code>!=#</code>	<code>!=?</code>	
greater than	<code>></code>	<code>>#</code>	<code>>?</code>	
greater than or equal	<code>>=</code>	<code>>=#</code>	<code>>=?</code>	
smaller than	<code><</code>	<code><#</code>	<code><?</code>	
smaller than or equal	<code><=</code>	<code><=#</code>	<code><=?</code>	
regexp matches	<code>==~</code>	<code>==~#</code>	<code>==~?</code>	
regexp doesn't match	<code>!~</code>	<code>!~#</code>	<code>!~?</code>	
same instance	<code>is</code>	<code>is#</code>	<code>is?</code>	
different instance	<code>isnot</code>	<code>isnot#</code>	<code>isnot?</code>	

Examples:

`"abc" ==# "Abc"` evaluates to 0

`"abc" ==? "Abc"` evaluates to 1

`"abc" == "Abc"` evaluates to 1 if `'ignorecase'` is set, 0 otherwise

NOTE: In Vim9 script `'ignorecase'` is not used.

E691 E692

A `List` can only be compared with a `List` and only "equal", "not equal", "is" and "isnot" can be used. This compares the values of the list, recursively. Ignoring case means case is ignored when comparing item values.

E735 E736

A `Dictionary` can only be compared with a `Dictionary` and only "equal", "not equal", "is" and "isnot" can be used. This compares the key/values of the `Dictionary` recursively. Ignoring case means case is ignored when comparing item values.

E694

A `Funcref` can only be compared with a `Funcref` and only "equal", "not equal", "is" and "isnot" can be used. Case is never ignored. Whether arguments or a `Dictionary` are bound (with a partial) matters. The `Dictionaries` must also be equal (or the same, in case of "is") and the arguments must be equal (or the same).

To compare `Funcrefs` to see if they refer to the same function, ignoring bound `Dictionary` and arguments, use `get()` to get the function name:

```
if get(Part1, 'name') == get(Part2, 'name')
    " Part1 and Part2 refer to the same function
```

E1037

Using "is" or "isnot" with a `List`, `Dictionary` or `Blob` checks whether the expressions are referring to the same `List`, `Dictionary` or `Blob` instance. A copy of a `List` is different from the original `List`. When using "is" without a `List`, `Dictionary` or `Blob`, it is equivalent to using "equal", using "isnot" equivalent to using "not equal". Except that a different type means the values are different:

```
echo 4 == '4'
1
```

```

echo 4 is '4'
0
echo 0 is []
0

```

"is#" / "isnot#" and "is?" / "isnot?" can be used to match and ignore case. In Vim9 script this doesn't work, two strings are never identical.

In legacy script, when comparing a String with a Number, the String is converted to a Number, and the comparison is done on Numbers. This means that:

```

echo 0 == 'x'
1

```

because 'x' converted to a Number is zero. However:

```

echo [0] == ['x']
0

```

Inside a List or Dictionary this conversion is not used.

In Vim9 script the types must match.

When comparing two Strings, this is done with strcmp() or stricmp(). This results in the mathematical difference (comparing byte values), not necessarily the alphabetical difference in the local language.

When using the operators with a trailing '#', or the short version and 'ignorecase' is off, the comparing is done with strcmp(): case matters.

When using the operators with a trailing '?', or the short version and 'ignorecase' is set, the comparing is done with stricmp(): case is ignored.

'smartcase' is not used.

The "=~" and "!~" operators match the lefthand argument with the righthand argument, which is used as a pattern. See [pattern](#) for what a pattern is. This matching is always done like 'magic' was set and 'coptions' is empty, no matter what the actual value of 'magic' or 'coptions' is. This makes scripts portable. To avoid backslashes in the regexp pattern to be doubled, use a single-quote string, see [literal-string](#).

Since a string is considered to be a single line, a multi-line pattern (containing \n, backslash-n) will not match. However, a literal NL character can be matched like an ordinary character. Examples:

```

"foo\nbar" =~ "\n"      evaluates to 1
"foo\nbar" =~ "\\n"    evaluates to 0

```

expr5

expr5 bitwise-shift

```

expr6 << expr6  bitwise left shift
expr6 >> expr6  bitwise right shift

```

```

expr-<<
expr->>

```

E1282 E1283

The "<<" and ">>" operators can be used to perform bitwise left or right shift of the left operand by the number of bits specified by the right operand. The operands are used as positive numbers. When shifting right with ">>" the topmost bit (sometimes called the sign bit) is cleared. If the right operand (shift amount) is more than the maximum number of bits in a number

(`v:numbersize`) the result is zero.

<code>expr6</code> and <code>expr7</code>	<code>expr6</code>	<code>expr7</code>	E1036	E1051
<code>expr7 + expr7</code>	Number addition,	<code>List</code> or <code>Blob</code>	concatenation	<code>expr++</code>
<code>expr7 - expr7</code>	Number subtraction			<code>expr--</code>
<code>expr7 . expr7</code>	String concatenation			<code>expr-</code>
<code>expr7 .. expr7</code>	String concatenation			<code>expr-..</code>

For `Lists` only "+" is possible and then both `expr7` must be a list. The result is a new list with the two lists Concatenated.

For String concatenation `".."` is preferred, since `"."` is ambiguous, it is also used for `Dict` member access and floating point numbers.

In `Vim9` script and when `vimscript-version` is 2 or higher, using `"."` is not allowed.

In `Vim9` script the arguments of `".."` are converted to String for simple types: Number, Float, Special and Bool. For other types `string()` should be used.

<code>expr8 * expr8</code>	Number multiplication	<code>expr-star</code>
<code>expr8 / expr8</code>	Number division	<code>expr-/</code>
<code>expr8 % expr8</code>	Number modulo	<code>expr-%</code>

In legacy script, for all operators except `"."` and `".."`, Strings are converted to Numbers.

For bitwise operators see `and()` , `or()` and `xor()` .

Note the difference between "+" and `".."` in legacy script:

```
"123" + "456" = 579
"123" .. "456" = "123456"
```

Since `'..'` has the same precedence as `+` and `-`, you need to read:

```
1 .. 90 + 90.0
```

As:

```
(1 .. 90) + 90.0
```

That works in legacy script, since the String "190" is automatically converted to the Number 190, which can be added to the Float 90.0. However:

```
1 .. 90 * 90.0
```

Should be read as:

```
1 .. (90 * 90.0)
```

Since `'..'` has lower precedence than `*`. This does NOT work, since this attempts to concatenate a Float and a String.

When dividing a Number by zero the result depends on the value:

```
0 / 0 = -0x80000000 (like NaN for Float)
>0 / 0 = 0x7fffffff (like positive infinity)
<0 / 0 = -0x7fffffff (like negative infinity)
(before Vim 7.2 it was always 0x7fffffff)
```

In `Vim9` script dividing a number by zero is an error. `E1154`

When 64-bit Number support is enabled:

```
0 / 0 = -0x8000000000000000 (like NaN for Float)
>0 / 0 = 0x7fffffffffffffff (like positive infinity)
<0 / 0 = -0x7fffffffffffffff (like negative infinity)
```

When the righthand side of '%' is zero, the result is 0.

None of these work for [Funcrefs](#).

(".", ".." and "%") do not work for Float. [E804](#) [E1035](#)

```
expr8                                     expr8
-----
<type>expr9
```

This is only available in [Vim9](#) script, see [type-casting](#).

```
expr9                                     expr9
-----
! expr9          logical NOT          expr-!
- expr9          unary minus          expr-unary--
+ expr9          unary plus           expr-unary-+
```

For '!' [TRUE](#) becomes [FALSE](#), [FALSE](#) becomes [TRUE](#) (one).
For '-' the sign of the number is changed.
For '+' the number is unchanged. [Note](#): "++" has no effect.

In legacy script a String will be converted to a Number first. [Note](#) that if the string does not start with a digit you likely don't get what you expect.

In [Vim9](#) script an error is given when "-" or "+" is used and the type is not a number.

In [Vim9](#) script "!" can be used for any type and the result is always a boolean. Use "!!" to convert any type to a boolean, according to whether the value is [falsy](#).

These three can be repeated and mixed. Examples:

```
!-1          == 0
!!8          == 1
--9          == 9
```

```
expr10                                     expr10
-----
```

This expression is either [expr11](#) or a sequence of the alternatives below, in any order. E.g., these are all possible:

```
expr10[expr1].name
expr10.name[expr1]
expr10(expr1, ...)[expr1].name
expr10->(expr1, ...)[expr1]
```

Evaluation is always from left to right.

A negative number can be used to measure from the end of the string. -1 is the last character, -2 the last but one, etc.

If an index goes out of range for the string characters are omitted. If `expr1b` is smaller than `expr1a` the result is an empty string.

Examples:

```
:let c = name[-1:]           " last byte of a string
:let c = name[0:-1]          " the whole string
:let c = name[-2:-2]         " last but one byte of a string
:let s = line(".")[4:]        " from the fifth byte to the end
:let s = s[:-3]              " remove last two bytes
```

slice

If `expr10` is a **List** this results in a new **List** with the items indicated by the indexes `expr1a` and `expr1b`. This works like with a **String**, as explained just above. Also see **sublist** below. Examples:

```
:let l = mylist[:3]          " first four items
:let l = mylist[4:4]         " List with one item
:let l = mylist[:]           " shallow copy of a List
```

If `expr10` is a **Blob** this results in a new **Blob** with the bytes in the indexes `expr1a` and `expr1b`, inclusive. Examples:

```
:let b = 0zDEADBEEF
:let bs = b[1:2]             " 0zADBE
:let bs = b[:]               " copy of 0zDEADBEEF
```

Using `expr10[expr1]` or `expr10[expr1a : expr1b]` on a **Funcref** results in an error.

Watch out for confusion between a namespace and a variable followed by a colon for a **sublist**:

```
mylist[n:]      " uses variable n
mylist[s:]      " uses namespace s:, error!
```

`expr10.name` entry in a **Dictionary** **expr-entry**
E1203 E1229

If `expr10` is a **Dictionary** and it is followed by a dot, then the following name will be used as a key in the **Dictionary**. This is just like: `expr10[name]`.

The name must consist of alphanumeric characters, just like a variable name, but it may start with a number. Curly braces cannot be used.

There must not be white space before or after the dot.

Examples:

```
:let dict = {"one": 1, 2: "two"}
:echo dict.one      " shows "1"
:echo dict.2        " shows "two"
:echo dict .2       " error because of space before the dot
```

Note that the dot is also used for String concatenation. To avoid confusion always put spaces around the dot for String concatenation.

`expr10(expr1, ...)` **Funcref** function call **E1085**

When `expr10` is a **Funcref** type variable, invoke the function it refers to.

`expr10->name([args])` method call **method** **->**
`expr10->{lambda}([args])` **E260** **E276** **E1265**

For methods that are also available as global functions this is the same as:
`name(expr10 [, args])`

There can also be methods specifically for the type of "expr10".

This allows for chaining, passing the value that one method returns to the next method:

`mylist->filter(filterexpr)->map(mapexpr)->sort()->join()`

Example of using a lambda:

`GetPercentage()->{x -> x * 100}()->printf('%d%%')`

When using `->` the **expr9** operators will be applied first, thus:

`-1.234->string()`

Is equivalent to:

`(-1.234)->string()`

And NOT:

`-(1.234->string())`

What comes after "`->`" can be a name, a simple expression (not containing any parenthesis), or any expression in parentheses:

`base->name(args)`
`base->some.name(args)`
`base->alist[idx](args)`
`base->(getFuncRef())(args)`

Note that in the last call the base is passed to the function resulting from "`getFuncRef()`", inserted before "args". **E1275**

E274

`"->name("` must not contain white space. There can be white space before the `"->`" and after the "`(`", thus you can split the lines like this:

```
mylist
\ ->filter(filterexpr)
\ ->map(mapexpr)
\ ->sort()
\ ->join()
```

When using the lambda form there must be no white space between the `}` and the `(`.

expr11

number

number

number constant

expr-number

`0x` hex-number `0o` octal-number `binary-number`
Decimal, Hexadecimal (starting with `0x` or `0X`), Binary (starting with `0b` or `0B`) and Octal (starting with `0`, `0o` or `0O`).

Assuming 64 bit numbers are used (see `v:numbersize`) an unsigned number is truncated to `0xffffffffffffffff` or `9223372036854775807`. You can use `-1` to get `0xffffffffffffffff`.

floating-point-format

Floating point numbers can be written in two forms:

```
[+]{N}.{M}
[+]{N}.{M}[eE][+]{exp}
```

`{N}` and `{M}` are numbers. Both `{N}` and `{M}` must be present and can only contain digits, except that in Vim9 script in `{N}` single quotes between digits are ignored.

`[+]` means there is an optional plus or minus sign.

`{exp}` is the exponent, power of 10.

Only a decimal point is accepted, not a comma. No matter what the current locale is.

Examples:

```
123.456
+0.0001
55.0
-0.123
1.234e03
1.0E-6
-3.1416e+88
```

These are INVALID:

```
3.          empty {M}
1e40       missing .{M}
```

Rationale:

Before floating point was introduced, the text "123.456" was interpreted as the two numbers "123" and "456", both converted to a string and concatenated, resulting in the string "123456". Since this was considered pointless, and we could not find it intentionally being used in Vim scripts, this backwards incompatibility was accepted in favor of being able to use the normal notation for floating point numbers.

float-pi float-e

A few useful values to copy&paste:

```
:let pi = 3.14159265359
:let e = 2.71828182846
```

Or, if you don't want to write them in as floating-point literals, you can also use functions, like the following:

```
:let pi = acos(-1.0)
:let e = exp(1.0)
```

floating-point-precision

The precision and range of floating points numbers depends on what "double" means in the library Vim was compiled with. There is no way to change this at runtime.

The default for displaying a `Float` is to use 6 decimal places, like using `printf("%g", f)`. You can select something else when using the `printf()` function. Example:

```
:echo printf('%.15e', atan(1))
7.853981633974483e-01
```

<code>string</code>	<code>string</code>	<code>String</code>	<code>expr-string</code>	<code>E114</code>
<code>"string"</code>	string constant	<code>expr-quote</code>		

Note that double quotes are used.

A string constant accepts these special characters:

- `\...` three-digit octal number (e.g., `"\316"`)
- `\..` two-digit octal number (must be followed by non-digit)
- `\.` one-digit octal number (must be followed by non-digit)
- `\x..` byte specified with two hex numbers (e.g., `"\x1f"`)
- `\x.` byte specified with one hex number (must be followed by non-hex char)
- `\X..` same as `\x..`
- `\X.` same as `\x.`
- `\u....` character specified with up to 4 hex numbers, stored according to the current value of `'encoding'` (e.g., `"\u02a4"`)
- `\U....` same as `\u` but allows up to 8 hex numbers.
- `\b` backspace `<BS>`
- `\e` escape `<Esc>`
- `\f` formfeed `0x0C`
- `\n` newline `<NL>`
- `\r` return `<CR>`
- `\t` tab `<Tab>`
- `\\` backslash
- `\"` double quote
- `\<xxx>` Special key named "xxx". e.g. `"\<C-W>"` for **CTRL-W**. This is for use in mappings, the `0x80` byte is escaped. To use the double quote character it must be escaped: `"<M-\>"`. Don't use `<Char-xxxx>` to get a UTF-8 character, use `\uxxxx` as mentioned above.
- `\<*xxx>` Like `\<xxx>` but prepends a modifier instead of including it in the character. E.g. `"\<C-w>"` is one character `0x17` while `"\<*C-w>"` is four bytes: 3 for the CTRL modifier and then character "W".

Note that `"\xff"` is stored as the byte 255, which may be invalid in some encodings. Use `"\u00ff"` to store character 255 according to the current value of `'encoding'`.

Note that `"\000"` and `"\x00"` force the end of the string.

blob-literal

blob-literal E973

Hexadecimal starting with 0z or 0Z, with an arbitrary number of bytes. The sequence must be an even number of hex characters. Example:

```
:let b = 0zFF00ED015DAF
```

literal-string

literal-string E115

'string' string constant

expr-'

Note that single quotes are used.

This string is taken as it is. No backslashes are removed or have a special meaning. The only exception is that two quotes stand for one quote.

Single quoted strings are useful for patterns, so that backslashes do not need to be doubled. These two commands are equivalent:

```
if a =~ "\\s*"
if a =~ '\s'
```

interpolated-string

\$quote interpolated-string

\$"string" interpolated string constant expr-\$quote
\$'string' interpolated literal string constant expr-\$'

Interpolated strings are an extension of the `string` and `literal-string`, allowing the inclusion of Vim script expressions (see `expr1`). Any expression returning a value can be enclosed between curly braces. The value is converted to a string. All the text and results of the expressions are concatenated to make a new string.

E1278 E1279

To include an opening brace '{' or closing brace '}' in the string content double it. For double quoted strings using a backslash also works. A single closing brace '}' will result in an error.

Examples:

```
let your_name = input("What's your name? ")
What's your name? Peter

echo
echo $"Hello, {your_name}!"
Hello, Peter!

echo $"The square root of {{9}} is {sqrt(9)}"
The square root of {9} is 3.0
```

string-offset-encoding

A string consists of multiple characters. How the characters are stored depends on '**encoding**'. Most common is UTF-8, which uses one byte for ASCII characters, two bytes for other latin characters and more bytes for other

characters.

A string offset can count characters or bytes. Other programs may use UTF-16 encoding (16-bit words) and an offset of UTF-16 words. Some functions use byte offsets, usually for UTF-8 encoding. Other functions use character offsets, in which case the encoding doesn't matter.

The different offsets for the string "a☺" are below:

UTF-8 offsets:

[0]: 61, [1]: C2, [2]: A9, [3]: F0, [4]: 9F, [5]: 98, [6]: 8A

UTF-16 offsets:

[0]: 0061, [1]: 00A9, [2]: D83D, [3]: DE0A

UTF-32 (character) offsets:

[0]: 00000061, [1]: 000000A9, [2]: 0001F60A

You can use the "g8" and "ga" commands on a character to see the decimal/hex/octal values.

The functions `byteidx()`, `utf16idx()` and `charidx()` can be used to convert between these indices. The functions `strlen()`, `strutf16len()` and `strcharlen()` return the number of bytes, UTF-16 code units and characters in a string respectively.

option expr-option E112 E113

<code>&option</code>	option value, local value if possible
<code>&g:option</code>	global option value
<code>&l:option</code>	local option value

Examples:

```
echo "tabstop is " .. &tabstop
if &insertmode
```

Any option name can be used here. See [options](#). When using the local value and there is no buffer-local or window-local value, the global value is used anyway.

register expr-register @r

@r contents of register 'r'

The result is the contents of the named register, as a single string. Newlines are inserted where required. To get the contents of the unnamed register use "@" or "@@". See [registers](#) for an explanation of the available registers.

When using the '=' register you get the expression itself, not what it evaluates to. Use `eval()` to evaluate it.

nesting expr-nesting E110

(expr1) nested expression

environment variable expr-env

\$VAR environment variable

The String value of any environment variable. When it is not defined, the result is an empty string.

The functions ``getenv()`` and ``setenv()`` can also be used and work for environment variables with non-alphanumeric names. The function ``environ()`` can be used to get a Dict with all environment variables.

expr-env-expand

Note that there is a difference between using \$VAR directly and using `expand("$VAR")`. Using it directly will only expand environment variables that are known inside the current Vim session. Using `expand()` will first try using the environment variables known inside the current Vim session. If that fails, a shell will be used to expand the variable. This can be slow, but it does expand all variables that the shell knows about. Example:

```
:echo $shell
:echo expand("$shell")
```

The first one probably doesn't echo anything, the second echoes the \$shell variable (if your shell supports it).

internal variable expr-variable E1015 E1089

variable internal variable
See below [internal-variables](#) .

function call expr-function E116 E118 E119 E120

function(expr1, ...) function call
See below [functions](#) .

lambda expression expr-lambda lambda

{args -> expr1} legacy lambda expression E451
(args) => expr1 Vim9 lambda expression

A lambda expression creates a new unnamed function which returns the result of evaluating `expr1` . Lambda expressions differ from [user-functions](#) in the following ways:

1. The body of the lambda expression is an `expr1` and not a sequence of `Ex` commands.
2. The prefix "a:" should not be used for arguments. E.g.:

```
:let F = {arg1, arg2 -> arg1 - arg2}
```



```
:echo F(5, 2)
3
```

The arguments are optional. Example:

```
:let F = {-> 'error function'}
:echo F('ignored')
error function
```

The [Vim9](#) lambda does not only use a different syntax, it also adds type checking and can be split over multiple lines, see [vim9-lambda](#) .

closure

Lambda expressions can access outer scope variables and arguments. This is often called a closure. Example where "i" and "a:arg" are used in a lambda while they already exist in the function scope. They remain valid even after the function returns:

```
:function Foo(arg)
: let i = 3
: return {x -> x + i - a:arg}
:endfunction
:let Bar = Foo(4)
:echo Bar(6)
5
```

Note that the variables must exist in the outer scope before the lambda is defined for this to work. See also [:func-closure](#) .

Lambda and closure support can be checked with:

```
if has('lambda')
```

Examples for using a lambda expression with [sort\(\)](#) , [map\(\)](#) and [filter\(\)](#) :

```
:echo map([1, 2, 3], {idx, val -> val + 1})
[2, 3, 4]
:echo sort([3,7,2,1,4], {a, b -> a - b})
[1, 2, 3, 4, 7]
```

The lambda expression is also useful for Channel, Job and timer:

```
:let timer = timer_start(500,
\ {-> execute("echo 'Handler called'", "")},
\ {'repeat': 3})
```

```
Handler called
Handler called
Handler called
```

Note that it is possible to cause memory to be used and not freed if the closure is referenced by the context it depends on:

```
function Function()
let x = 0
let F = {-> x}
endfunction
```

The closure uses "x" from the function scope, and "F" in that same scope refers to the closure. This cycle results in the memory not being freed. Recommendation: don't do this.

Notice how `execute()` is used to execute an Ex command. That's ugly though. In Vim9 script you can use a command block, see [inline-function](#) .

Although you can use the loop variable of a ``for`` command, it must still exist when the closure is called, otherwise you get an error. [E1302](#)

Lambda expressions have internal names like `'<lambda>42'`. If you get an error for a lambda expression, you can find what it is with the following command:

```
:function <lambda>42
```

See also: [numbered-function](#)

3. Internal variable [internal-variables](#) [E461](#) [E1001](#)

An internal variable name can be made up of letters, digits and `'_'`. But it cannot start with a digit. In legacy script it is also possible to use curly braces, see [curly-braces-names](#) .

In legacy script an internal variable is created with the `":let"` command `:let` . An internal variable is explicitly destroyed with the `":unlet"` command `:unlet` .

Using a name that is not an internal variable or refers to a variable that has been destroyed results in an error.

In Vim9 script ``:let`` is not used and variables work differently, see [:var](#) .

[variable-scope](#)

There are several name spaces for variables. Which one is to be used is specified by what is prepended:

	(nothing)	In a function: local to the function; in a legacy script: global; in a Vim9 script: local to the script
buffer-variable	b:	Local to the current buffer.
window-variable	w:	Local to the current window.
tabpage-variable	t:	Local to the current tab page.
global-variable	g:	Global.
local-variable	l:	Local to a function (only in a legacy function)
script-variable	s:	Local to a <code>:source</code> 'ed Vim script.
function-argument	a:	Function argument (only in a legacy function).
vim-variable	v:	Global, predefined by Vim.

The scope name by itself can be used as a [Dictionary](#) . For example, to delete all script-local variables:

```
:for k in keys(s:)
:  unlet s:[k]
:endfor
```

Note: in Vim9 script variables can also be local to a block of commands, see [vim9-scopes](#) .

[buffer-variable](#) [b:var](#) [b:](#)

A variable name that is preceded with `"b:"` is local to the current buffer.

Thus you can have several `"b:foo"` variables, one for each buffer.

This kind of variable is deleted when the buffer is wiped out or deleted with

`:bdelete .`

One local buffer variable is predefined:

`b:changedtick` `b:changedtick` `changetick`
The total number of changes to the current buffer. It is incremented for each change. An undo command is also a change in this case. Resetting `'modified'` when writing the buffer is also counted.
This can be used to perform an action only when the buffer has changed. Example:

```
:if my_changedtick != b:changedtick
:   let my_changedtick = b:changedtick
:   call My_Update()
:endif
```

You cannot change or delete the `b:changedtick` variable.
If you need more information about the change see `listener_add()` .

`w:var` `w:`
A variable name that is preceded with "w:" is local to the current window. It is deleted when the window is closed.

`t:var` `t:`
A variable name that is preceded with "t:" is local to the current tab page, It is deleted when the tab page is closed. {not available when compiled without the `+windows` feature}

`g:var` `g:`
Inside functions and in `Vim9` script global variables are accessed with "g:". Omitting this will access a variable local to a function or script. "g:" can also be used in any other place if you like.

`l:var` `l:`
Inside functions local variables are accessed without prepending anything. But you can also prepend "l:" if you like. However, without prepending "l:" you may run into reserved variable names. For example "count". By itself it refers to "v:count". Using "l:count" you can have a local variable with the same name.

`s:var`
In a legacy Vim script variables starting with "s:" can be used. They cannot be accessed from outside of the scripts, thus are local to the script. In `Vim9` script the "s:" prefix can be omitted, variables are script-local by default.

They can be used in:

- commands executed while the script is sourced
- functions defined in the script
- autocommands defined in the script
- functions and autocommands defined in functions and autocommands which were defined in the script (recursively)
- user defined commands defined in the script

Thus not in:

- other scripts sourced from this one

- mappings
- menus
- etc.

Script variables can be used to avoid conflicts with global variable names. Take this example:

```
let s:counter = 0
function MyCounter()
  let s:counter = s:counter + 1
  echo s:counter
endfunction
command Tick call MyCounter()
```

You can now invoke "Tick" from any script, and the "s:counter" variable in that script will not be changed, only the "s:counter" in the script where "Tick" was defined is used.

Another example that does the same:

```
let s:counter = 0
command Tick let s:counter = s:counter + 1 | echo s:counter
```

When calling a function and invoking a user-defined command, the context for script variables is set to the script where the function or command was defined.

The script variables are also available when a function is defined inside a function that is defined in a script. Example:

```
let s:counter = 0
function StartCounting(incr)
  if a:incr
    function MyCounter()
      let s:counter = s:counter + 1
    endfunction
  else
    function MyCounter()
      let s:counter = s:counter - 1
    endfunction
  endif
endfunction
```

This defines the MyCounter() function either for counting up or counting down when calling StartCounting(). It doesn't matter from where StartCounting() is called, the s:counter variable will be accessible in MyCounter().

When the same script is sourced again it will use the same script variables. They will remain valid as long as Vim is running. This can be used to maintain a counter:

```
if !exists("s:counter")
  let s:counter = 1
  echo "script executed for the first time"
```

```

else
  let s:counter = s:counter + 1
  echo "script executed " .. s:counter .. " times now"
endif

```

Note that this means that filetype plugins don't get a different set of script variables for each buffer. Use local buffer variables instead `b:var`.

PREDEFINED VIM VARIABLES

vim-variable v:var v:
E963 E1063

Some variables can be set by the user, but the type cannot be changed.

v:argv	<code>v:argv</code> argv-variable The command line arguments Vim was invoked with. This is a list of strings. The first item is the Vim command. See <code>v:progbath</code> for the command with full path.
v:beval_col	<code>v:beval_col</code> beval_col-variable The number of the column, over which the mouse pointer is. This is the byte index in the <code>v:beval_lnum</code> line. Only valid while evaluating the <code>'balloonexpr'</code> option.
v:beval_bufnr	<code>v:beval_bufnr</code> beval_bufnr-variable The number of the buffer, over which the mouse pointer is. Only valid while evaluating the <code>'balloonexpr'</code> option.
v:beval_lnum	<code>v:beval_lnum</code> beval_lnum-variable The number of the line, over which the mouse pointer is. Only valid while evaluating the <code>'balloonexpr'</code> option.
v:beval_text	<code>v:beval_text</code> beval_text-variable The text under or after the mouse pointer. Usually a word as it is useful for debugging a C program. <code>'iskeyword'</code> applies, but a dot and <code>"->"</code> before the position is included. When on a <code>']'</code> the text before it is used, including the matching <code>'['</code> and word before it. When on a Visual area within one line the highlighted text is used. Also see <code><cexpr></code> . Only valid while evaluating the <code>'balloonexpr'</code> option.
v:beval_winnr	<code>v:beval_winnr</code> beval_winnr-variable The number of the window, over which the mouse pointer is. Only valid while evaluating the <code>'balloonexpr'</code> option. The first window has number zero (unlike most other places where a window gets a number).
v:beval_winid	<code>v:beval_winid</code> beval_winid-variable The <code>window-ID</code> of the window, over which the mouse pointer is. Otherwise like <code>v:beval_winnr</code> .
v:char	<code>v:char</code> char-variable Argument for evaluating <code>'formatexpr'</code> and used for the typed character when using <code><expr></code> in an abbreviation <code>:map-<expr></code> . It is also used by the <code>InsertCharPre</code> and <code>InsertEnter</code> events.

`v:charconvert_from` `charconvert_from-variable`

`v:charconvert_from` The name of the character encoding of a file to be converted. Only valid while evaluating the `'charconvert'` option.

`v:charconvert_to` `charconvert_to-variable`

`v:charconvert_to` The name of the character encoding of a file after conversion. Only valid while evaluating the `'charconvert'` option.

`v:cmdarg` `cmdarg-variable`

`v:cmdarg` This variable is used for two purposes:

1. The extra arguments given to a file read/write command. Currently these are `"++enc="` and `"++ff="`. This variable is set before an autocommand event for a file read/write command is triggered. There is a leading space to make it possible to append this variable directly after the read/write command. **Note:** The `"cmd"` argument isn't included here, because it will be executed anyway.
2. When printing a PostScript file with `":hardcopy"` this is the argument for the `":hardcopy"` command. This can be used in `'printexpr'`.

`v:cmdbang` `cmdbang-variable`

`v:cmdbang` Set like `v:cmdarg` for a file read/write command. When a `"!"` was used the value is 1, otherwise it is 0. **Note** that this can only be used in autocommands. For user commands `<bang>` can be used.

`v:collate` `collate-variable`

`v:collate` The current locale setting for collation order of the runtime environment. This allows Vim scripts to be aware of the current locale encoding. Technical: it's the value of `LC_COLLATE`. When not using a locale the value is `"C"`. This variable can not be set directly, use the `:language` command.
See `multi-lang` .

`v:colornames`

`v:colornames` A dictionary that maps color names to hex color strings. These color names can be used with the `highlight-guifg` , `highlight-guibg` , and `highlight-guisp` parameters.

The key values in the dictionary (the color names) should be lower cased, because Vim looks up a color by its lower case name.

Updating an entry in `v:colornames` has no immediate effect on the syntax highlighting. The highlight commands (probably in a colorscheme script) need to be re-evaluated in order to use the updated color values. For example:

```
:let v:colornames['fuchsia'] = '#cf3ab4'
:let v:colornames['mauve'] = '#915f6d'
```

```
:highlight Normal guifg=fuscia guibg=mauve
```

This cannot be used to override the `cterm-colors` but it can be used to override other colors. For example, the X11 colors defined in the ``colors/lists/default.vim`` (previously defined in `rgb.txt`). When defining new color names in a plugin, the recommended practice is to set a color entry only when it does not already exist. For example:

```
:call extend(v:colornames, {  
  \ 'fuchsia': '#cf3ab4',  
  \ 'mauve': '#915f6d',  
  \ }, 'keep')
```

Using `extend()` with the `'keep'` option updates each color only if it did not exist in `v:colornames`. Doing so allows the user to choose the precise color value for a common name by setting it in their `.vimrc`.

It is possible to remove entries from this dictionary but doing so is NOT recommended, because it is disruptive to other scripts. It is also unlikely to achieve the desired result because the `:colorscheme` and `:highlight` commands will both automatically load all ``colors/lists/default.vim`` color scripts.

`v:completed_item` `completed_item-variable`

`v:completed_item`

`Dictionary` containing the `complete-items` for the most recently completed word after `CompleteDone`. The `Dictionary` is empty if the completion failed.

Note: Plugins can modify the value to emulate the builtin `CompleteDone` event behavior.

`v:count` `count-variable`

`v:count`

The count given for the last Normal mode command. Can be used to get the count before a mapping. Read-only. Example:

```
:map _x :<C-U>echo "the count is " .. v:count<CR>
```

Note: The `<C-U>` is required to remove the line range that you get when typing `:'` after a count.

When there are two counts, as in `"3d2w"`, they are multiplied, just like what happens in the command, `"d6w"` for the example.

Also used for evaluating the `'formatexpr'` option.

`"count"` also works, for backwards compatibility, unless `scriptversion` is 3 or higher.

`v:count1` `count1-variable`

`v:count1`

Just like `"v:count"`, but defaults to one when no count is used.

`v:ctype` `ctype-variable`

`v:ctype`

The current locale setting for characters of the runtime environment. This allows Vim scripts to be aware of the current locale encoding. Technical: it's the value of

LC_CTYPE. When not using a locale the value is "C". This variable can not be set directly, use the `:language` command. See `multi-lang` .

- v:dying** v:dying dying-variable
Normally zero. When a deadly signal is caught it's set to one. When multiple signals are caught the number increases. Can be used in an autocommand to check if Vim didn't terminate normally. {only works on Unix}
Example:
`:au VimLeave * if v:dying | echo "\nAAAAaaaarrggghhh!!!\n" | endif`
Note: if another deadly signal is caught when v:dying is one, VimLeave autocommands will not be executed.
- v:exiting** v:exiting exiting-variable
Vim exit code. Normally zero, non-zero when something went wrong. The value is v:null before invoking the `VimLeavePre` and `VimLeave` autocmds. See `:q` , `:x` and `:cquit` .
Example:
`:au VimLeave * echo "Exit value is " .. v:exiting`
- v:echospace** v:echospace echospace-variable
Number of screen cells that can be used for an `:echo` message in the last screen line before causing the `hit-enter-prompt` . Depends on `'showcmd'` , `'ruler'` and `'columns'` . You need to check `'cmdheight'` for whether there are full-width lines available above the last line.
- v:errmsg** v:errmsg errmsg-variable
Last given error message. It's allowed to set this variable.
Example:
`:let v:errmsg = ""`
`:silent! next`
`:if v:errmsg != ""`
`: ... handle error`
"errmsg" also works, for backwards compatibility, unless `scriptversion` is 3 or higher.
- v:errors** v:errors errors-variable assert-return
Errors found by assert functions, such as `assert_true()` . This is a list of strings.
The assert functions append an item when an assert fails. The return value indicates this: a one is returned if an item was added to v:errors, otherwise zero is returned.
To remove old results make it empty:
`:let v:errors = []`
If v:errors is set to anything but a list it is made an empty list by the assert function.
- v:event** v:event event-variable
Dictionary containing information about the current `autocommand` . See the specific event for what it puts in this dictionary.

The dictionary is emptied when the `autocommand` finishes, please refer to `dict-identity` for how to get an independent copy of it. Use `deepcopy()` if you want to keep the information after the event triggers. Example:

```
au TextYankPost * let g:foo = deepcopy(v:event)
```

`v:exception` `v:exception` `exception-variable`
The value of the exception most recently caught and not finished. See also `v:throwpoint` and `throw-variables`. Example:

```
:try  
: throw "oops"  
:catch /.*/  
: echo "caught " .. v:exception  
:endtry
```

Output: "caught oops".

`v:false` `v:false` `false-variable`
A Number with value zero. Used to put "false" in JSON. See `json_encode()`.

When used as a string this evaluates to "v:false".

```
echo v:false
```

```
v:false
```

That is so that `eval()` can parse the string back to the same value. Read-only.

In `Vim9` script "false" can be used which has a boolean type.

`v:fcs_reason` `v:fcs_reason` `fcs_reason-variable`
The reason why the `FileChangedShell` event was triggered. Can be used in an autocommand to decide what to do and/or what to set `v:fcs_choice` to. Possible values:

deleted	file no longer exists
conflict	file contents, mode or timestamp was changed and buffer is modified
changed	file contents has changed
mode	mode of file changed
time	only file timestamp changed

`v:fcs_choice` `v:fcs_choice` `fcs_choice-variable`
What should happen after a `FileChangedShell` event was triggered. Can be used in an autocommand to tell Vim what to do with the affected buffer:

reload	Reload the buffer (does not work if the file was deleted).
edit	Reload the buffer and detect the values for options such as <code>'fileformat'</code> , <code>'fileencoding'</code> , <code>'binary'</code> (does not work if the file was deleted).
ask	Ask the user what to do, as if there was no autocommand. Except that when only the timestamp changed nothing will happen.
<empty>	Nothing, the autocommand should do

everything that needs to be done.
The default is empty. If another (invalid) value is used then
Vim behaves like it is empty, there is no warning message.

<code>v:fname</code>	When evaluating <code>'includeexpr'</code> : the file name that was detected. Empty otherwise.
<code>v:fname_in</code>	The name of the input file. Valid while evaluating: option used for <code>'charconvert'</code> file to be converted <code>'diffexpr'</code> original file <code>'patchexpr'</code> original file <code>'printexpr'</code> file to be printed And set to the swap file name for <code>SwapExists</code> .
<code>v:fname_out</code>	The name of the output file. Only valid while evaluating: option used for <code>'charconvert'</code> resulting converted file (*) <code>'diffexpr'</code> output of diff <code>'patchexpr'</code> resulting patched file (*) When doing conversion for a write command (e.g., <code>":w file"</code>) it will be equal to <code>v:fname_in</code> . When doing conversion for a read command (e.g., <code>":e file"</code>) it will be a temporary file and different from <code>v:fname_in</code> .
<code>v:fname_new</code>	The name of the new version of the file. Only valid while evaluating <code>'diffexpr'</code> .
<code>v:fname_diff</code>	The name of the diff (patch) file. Only valid while evaluating <code>'patchexpr'</code> .
<code>v:folddashes</code>	Used for <code>'foldtext'</code> : dashes representing foldlevel of a closed fold. Read-only in the <code>sandbox</code> . <code>fold-foldtext</code>
<code>v:foldlevel</code>	Used for <code>'foldtext'</code> : foldlevel of closed fold. Read-only in the <code>sandbox</code> . <code>fold-foldtext</code>
<code>v:foldend</code>	Used for <code>'foldtext'</code> : last line of closed fold. Read-only in the <code>sandbox</code> . <code>fold-foldtext</code>
<code>v:foldstart</code>	Used for <code>'foldtext'</code> : first line of closed fold. Read-only in the <code>sandbox</code> . <code>fold-foldtext</code>

<code>v:hsearch</code>	<p style="text-align: right;"><code>v:hsearch</code> <i>hsearch-variable</i></p> <p>Variable that indicates whether search highlighting is on. Setting it makes sense only if <code>'hsearch'</code> is enabled which requires <code>+extra_search</code>. Setting this variable to zero acts like the <code>:nohsearch</code> command, setting it to one acts like <code>let &hsearch = &hsearch</code>. Note that the value is restored when returning from a function. <code>function-search-undo</code>.</p>
<code>v:insertmode</code>	<p style="text-align: right;"><code>v:insertmode</code> <i>insertmode-variable</i></p> <p>Used for the <code>InsertEnter</code> and <code>InsertChange</code> autocommand events. Values:</p> <ul style="list-style-type: none"> <code>i</code> Insert mode <code>r</code> Replace mode <code>v</code> Virtual Replace mode
<code>v:key</code>	<p style="text-align: right;"><code>v:key</code> <i>key-variable</i></p> <p>Key of the current item of a <code>Dictionary</code>. Only valid while evaluating the expression used with <code>map()</code> and <code>filter()</code>. Read-only.</p>
<code>v:lang</code>	<p style="text-align: right;"><code>v:lang</code> <i>lang-variable</i></p> <p>The current locale setting for messages of the runtime environment. This allows Vim scripts to be aware of the current language. Technical: it's the value of <code>LC_MESSAGES</code>. The value is system dependent. This variable can not be set directly, use the <code>:language</code> command. It can be different from <code>v:ctype</code> when messages are desired in a different language than what is used for character encoding. See <code>multi-lang</code>.</p>
<code>v:lc_time</code>	<p style="text-align: right;"><code>v:lc_time</code> <i>lc_time-variable</i></p> <p>The current locale setting for time messages of the runtime environment. This allows Vim scripts to be aware of the current language. Technical: it's the value of <code>LC_TIME</code>. This variable can not be set directly, use the <code>:language</code> command. See <code>multi-lang</code>.</p>
<code>v:lnum</code>	<p style="text-align: right;"><code>v:lnum</code> <i>lnum-variable</i></p> <p>Line number for the <code>'foldexpr'</code> <code>fold-expr</code>, <code>'formatexpr'</code> and <code>'indentexpr'</code> expressions, tab page number for <code>'guitablabel'</code> and <code>'guitabtooltip'</code>. Only valid while one of these expressions is being evaluated. Read-only when in the <code>sandbox</code>.</p>
<code>v:maxcol</code>	<p style="text-align: right;"><code>v:maxcol</code> <i>maxcol-variable</i></p> <p>Maximum line length. Depending on where it is used it can be screen columns, characters or bytes. The value currently is 2147483647 on all systems.</p>
<code>v:mouse_win</code>	<p style="text-align: right;"><code>v:mouse_win</code> <i>mouse_win-variable</i></p> <p>Window number for a mouse click obtained with <code>getchar()</code>. First window has number 1, like with <code>winnr()</code>. The value is</p>

zero when there was no mouse button click.

`v:mouse_winid` `v:mouse_winid` `mouse_winid-variable`
Window ID for a mouse click obtained with `getchar()` .
The value is zero when there was no mouse button click.

`v:mouse_lnum` `v:mouse_lnum` `mouse_lnum-variable`
Line number for a mouse click obtained with `getchar()` .
This is the text line number, not the screen line number. The
value is zero when there was no mouse button click.

`v:mouse_col` `v:mouse_col` `mouse_col-variable`
Column number for a mouse click obtained with `getchar()` .
This is the screen column number, like with `virtcol()` . The
value is zero when there was no mouse button click.

`v:none` `v:none` `none-variable` `None`
An empty String. Used to put an empty item in JSON. See
`json_encode()` .
This can also be used as a function argument to use the
default value, see `none-function_argument` .
When used as a number this evaluates to zero.
When used as a string this evaluates to "v:none".
`echo v:none`
`v:none`
That is so that `eval()` can parse the string back to the same
value. Read-only.
Note that using ``== v:none`` and ``!= v:none`` will often give
an error. Instead, use ``is v:none`` and ``isnot v:none`` .

`v:null` `v:null` `null-variable`
An empty String. Used to put "null" in JSON. See
`json_encode()` .
When used as a number this evaluates to zero.
When used as a string this evaluates to "v:null".
`echo v:null`
`v:null`
That is so that `eval()` can parse the string back to the same
value. Read-only.
In `Vim9` script ``null`` can be used without "v:".
In some places ``v:null`` and ``null`` can be used for a List,
Dict, Job, etc. that is not set. That is slightly different
than an empty List, Dict, etc.

`v:numbermax` `v:numbermax` `numbermax-variable`
Maximum value of a number.

`v:numbermin` `v:numbermin` `numbermin-variable`
Minimum value of a number (negative).

`v:numbersize` `v:numbersize` `numbersize-variable`
Number of bits in a Number. This is normally 64, but on some
systems it may be 32.

`v:oldfiles` `v:oldfiles` `oldfiles-variable`
List of file names that is loaded from the `viminfo` file on startup. These are the files that Vim remembers marks for. The length of the List is limited by the `'` argument of the `'viminfo'` option (default is 100). When the `viminfo` file is not used the List is empty. Also see `:oldfiles` and `c_#<`. The List can be modified, but this has no effect on what is stored in the `viminfo` file later. If you use values other than String this will cause trouble.
{only when compiled with the |+viminfo| feature}

`v:option_new` `v:option_new`
New value of the option. Valid while executing an `OptionSet` autocommand.

`v:option_old` `v:option_old`
Old value of the option. Valid while executing an `OptionSet` autocommand. Depending on the command used for setting and the kind of option this is either the local old value or the global old value.

`v:option_oldlocal` `v:option_oldlocal`
Old local value of the option. Valid while executing an `OptionSet` autocommand.

`v:option_oldglobal` `v:option_oldglobal`
Old global value of the option. Valid while executing an `OptionSet` autocommand.

`v:option_type` `v:option_type`
Scope of the set command. Valid while executing an `OptionSet` autocommand. Can be either "global" or "local"

`v:option_command` `v:option_command`
Command used to set the option. Valid while executing an `OptionSet` autocommand.

<code>value</code>	<code>option was set via</code>
<code>"setlocal"</code>	<code>:setlocal</code> or <code>":let l:xxx"</code>
<code>"setglobal"</code>	<code>:setglobal</code> or <code>":let g:xxx"</code>
<code>"set"</code>	<code>:set</code> or <code>:let</code>
<code>"modeline"</code>	<code>modeline</code>

`v:operator` `v:operator` `operator-variable`
The last operator given in Normal mode. This is a single character except for commands starting with `<g>` or `<z>`, in which case it is two characters. Best used alongside `v:prevcnt` and `v:register`. Useful if you want to cancel Operator-pending mode and then use the operator, e.g.:
`:omap 0 <Esc>:call MyMotion(v:operator)<CR>`
The value remains set until another operator is entered, thus don't expect it to be empty.
`v:operator` is not set for `:delete`, `:yank` or other Ex commands.
Read-only.

`v:prevcnt` `prevcnt-variable`

`v:prevcount` The count given for the last but one Normal mode command. This is the `v:count` value of the previous command. Useful if you want to cancel Visual or Operator-pending mode and then use the count, e.g.:

```
:vmap % <Esc>:call MyFilter(v:prevcount)<CR>
```

Read-only.

`v:profiling` `v:profiling` `profiling-variable`
Normally zero. Set to one after using `":profile start"`. See `profiling`.

`v:progrname` `v:progrname` `progrname-variable`
Contains the name (with path removed) with which Vim was invoked. Allows you to do special initialisations for `view`, `evim` etc., or any other name you might symlink to Vim. Read-only.

`v:progrpath` `v:progrpath` `progrpath-variable`
Contains the command with which Vim was invoked, in a form that when passed to the shell will run the same Vim executable as the current one (if `$PATH` remains unchanged). Useful if you want to message a Vim server using a `--remote-expr`.

To get the full path use:

```
echo exepath(v:progrpath)
```

If the command has a relative path it will be expanded to the full path, so that it still works after ``:cd``. Thus starting `./vim` results in `/home/user/path/to/vim/src/vim`. On Linux and other systems it will always be the full path. On Mac it may just be `"vim"` and using `exepath()` as mentioned above should be used to get the full path. On MS-Windows the executable may be called `"vim.exe"`, but the `".exe"` is not added to `v:progrpath`. Read-only.

`v:python3_version` `v:python3_version` `python3-version-variable`
Version of Python 3 that Vim was built against. When Python is loaded dynamically (`python-dynamic`), this version should exactly match the Python library up to the minor version (e.g. 3.10.2 and 3.10.3 are compatible as the minor version is "10", whereas 3.9.4 and 3.10.3 are not compatible). When `python-stable-abi` is used, this will be the minimum Python version that you can use instead. (e.g. if `v:python3_version` indicates 3.9, you can use 3.9, 3.10, or anything above).

This number is encoded as a hex number following Python ABI versioning conventions. Do the following to have a human-readable full version in hex:

```
echo printf("%08X", v:python3_version)
```

You can obtain only the minor version by doing:

```
echo and(v:python3_version>>16,0xff)
```

Read-only.

<code>v:register</code>	<p style="text-align: right;"><code>v:register</code> <code>register-variable</code></p> <p>The name of the register in effect for the current normal mode command (regardless of whether that command actually used a register). Or for the currently executing normal mode mapping (use this in custom commands that take a register). If none is supplied it is the default register <code>''</code>, unless <code>'clipboard'</code> contains "unnamed" or "unnamedplus", then it is <code>'*' or '+'</code>. Also see <code>getreg()</code> and <code>setreg()</code></p>
<code>v:scrollstart</code>	<p style="text-align: right;"><code>v:scrollstart</code> <code>scrollstart-variable</code></p> <p>String describing the script or function that caused the screen to scroll up. It's only set when it is empty, thus the first reason is remembered. It is set to "Unknown" for a typed command. This can be used to find out why your script causes the hit-enter prompt.</p>
<code>v:servername</code>	<p style="text-align: right;"><code>v:servername</code> <code>servername-variable</code></p> <p>The resulting registered <code>client-server-name</code> if any. Read-only.</p>
<code>v:searchforward</code>	<p style="text-align: right;"><code>v:searchforward</code> <code>searchforward-variable</code></p> <p>Search direction: 1 after a forward search, 0 after a backward search. It is reset to forward when directly setting the last search pattern, see <code>quote/</code> . Note that the value is restored when returning from a function. <code>function-search-undo</code> . Read-write.</p>
<code>v:shell_error</code>	<p style="text-align: right;"><code>v:shell_error</code> <code>shell_error-variable</code></p> <p>Result of the last shell command. When non-zero, the last shell command had an error. When zero, there was no problem. This only works when the shell returns the error code to Vim. The value -1 is often used when the command could not be executed. Read-only. Example:</p> <pre> :!<code>mv</code> foo bar :<code>if</code> v:shell_error : <code>echo</code> 'could not rename "foo" to "bar"!' :<code>endif</code> </pre> <p>"shell_error" also works, for backwards compatibility, unless <code>scriptversion</code> is 3 or higher.</p>
<code>v:sizeofint</code>	<p style="text-align: right;"><code>v:sizeofint</code> <code>sizeofint-variable</code></p> <p>Number of bytes in an int. Depends on how Vim was compiled. This is only useful for deciding whether a test will give the expected result.</p>
<code>v:sizeoflong</code>	<p style="text-align: right;"><code>v:sizeoflong</code> <code>sizeoflong-variable</code></p> <p>Number of bytes in a long. Depends on how Vim was compiled. This is only useful for deciding whether a test will give the expected result.</p>

	<code>v:sizeofpointer</code>	<code>sizeofpointer-variable</code>	Number of bytes in a pointer. Depends on how Vim was compiled. This is only useful for deciding whether a test will give the expected result.
	<code>v:statusmsg</code>	<code>statusmsg-variable</code>	Last given status message. It's allowed to set this variable.
	<code>v:swapname</code>	<code>swapname-variable</code>	Only valid when executing <code>SwapExists</code> autocommands: Name of the swap file found. Read-only.
	<code>v:swapchoice</code>	<code>swapchoice-variable</code>	<p><code>SwapExists</code> autocommands can set this to the selected choice for handling an existing swap file:</p> <ul style="list-style-type: none"> 'o' Open read-only 'e' Edit anyway 'r' Recover 'd' Delete swapfile 'q' Quit 'a' Abort <p>The value should be a single-character string. An empty value results in the user being asked, as would happen when there is no <code>SwapExists</code> autocommand. The default is empty.</p>
	<code>v:swapcommand</code>	<code>swapcommand-variable</code>	Normal mode command to be executed after a file has been opened. Can be used for a <code>SwapExists</code> autocommand to have another Vim open the file and jump to the right place. For example, when jumping to a tag the value is <code>":tag tagname\r"</code> . For <code>":edit +cmd file"</code> the value is <code>":cmd\r"</code> .
	<code>v:t_bool</code>	<code>t_bool-variable</code>	Value of <code>Boolean</code> type. Read-only. See: <code>type()</code>
	<code>v:t_channel</code>	<code>t_channel-variable</code>	Value of <code>Channel</code> type. Read-only. See: <code>type()</code>
	<code>v:t_dict</code>	<code>t_dict-variable</code>	Value of <code>Dictionary</code> type. Read-only. See: <code>type()</code>
	<code>v:t_float</code>	<code>t_float-variable</code>	Value of <code>Float</code> type. Read-only. See: <code>type()</code>
	<code>v:t_func</code>	<code>t_func-variable</code>	Value of <code>Funcref</code> type. Read-only. See: <code>type()</code>
	<code>v:t_job</code>	<code>t_job-variable</code>	Value of <code>Job</code> type. Read-only. See: <code>type()</code>
	<code>v:t_list</code>	<code>t_list-variable</code>	Value of <code>List</code> type. Read-only. See: <code>type()</code>
	<code>v:t_none</code>	<code>t_none-variable</code>	Value of <code>None</code> type. Read-only. See: <code>type()</code>
	<code>v:t_number</code>	<code>t_number-variable</code>	Value of <code>Number</code> type. Read-only. See: <code>type()</code>
	<code>v:t_string</code>	<code>t_string-variable</code>	Value of <code>String</code> type. Read-only. See: <code>type()</code>
		<code>t_blob-variable</code>	

v:t_blob Value of `Blob` type. Read-only. See: `type()`
v:t_class t_class-variable

v:t_class Value of `class` type. Read-only. See: `type()`
v:t_object t_object-variable

v:t_object Value of `object` type. Read-only. See: `type()`
v:t_typealias t_typealias-variable

v:t_typealias Value of `typealias` type. Read-only. See: `type()`
v:termresponse termresponse-variable

v:termresponse The escape sequence returned by the terminal for the `t_RV` termcap entry. It is set when Vim receives an escape sequence that starts with ESC [or CSI, then '>' or '?' and ends in a 'c', with only digits and ';' in between. When this option is set, the TermResponse autocommand event is fired, so that you can react to the response from the terminal. The TermResponseAll event is also fired, with `<amatch>` set to "version". You can use `terminalprops()` to see what Vim figured out about the terminal. The response from a new xterm is: "`<Esc>[> Pp ; Pv ; Pc c`". Pp is the terminal type: 0 for vt100 and 1 for vt220. Pv is the patch level (since this was introduced in patch 95, it's always 95 or higher). Pc is always zero. If Pv is 141 or higher then Vim will try to request terminal codes. This only works with xterm `xterm-codes`.
{only when compiled with |+termresponse| feature}

v:termblinkresp

v:termblinkresp The escape sequence returned by the terminal for the `t_RC` termcap entry. This is used to find out whether the terminal cursor is blinking. This is used by `term_getcursor()`. When this option is set, the TermResponseAll autocommand event is fired, with `<amatch>` set to "cursorblink".

v:termstyleresp

v:termstyleresp The escape sequence returned by the terminal for the `t_RS` termcap entry. This is used to find out what the shape of the cursor is. This is used by `term_getcursor()`. When this option is set, the TermResponseAll autocommand event is fired, with `<amatch>` set to "cursorshape".

v:termrbgresp

v:termrbgresp The escape sequence returned by the terminal for the `t_RB` termcap entry. This is used to find out what the terminal background color is; see '`background`'. When this option is set, the TermResponseAll autocommand event is fired, with `<amatch>` set to "background".

v:termrfgresp

v:termrfgresp The escape sequence returned by the terminal for the `t_RF` termcap entry. This is used to find out what the terminal foreground color is. When this option is set, the TermResponseAll autocommand event is fired, with `<amatch>` set to "foreground".

`v:termu7resp` `v:termu7resp`
The escape sequence returned by the terminal for the `t_u7` termcap entry. This is used to find out what the terminal does with ambiguous width characters, see `'ambiwidth'`. When this option is set, the `TermResponseAll` autocommand event is fired, with `<amatch>` set to `"ambiguouswidth"`.

`v:testing` `v:testing` `testing-variable`
Must be set before using ``test_garbagecollect_now()``. Also, when set certain error messages won't be shown for 2 seconds. (e.g. `'dictionary'` option is empty)

`v:this_session` `v:this_session` `this_session-variable`
Full filename of the last loaded or saved session file. See `:mksession`. It is allowed to set this variable. When no session file has been saved, this variable is empty. `"this_session"` also works, for backwards compatibility, unless `scriptversion` is 3 or higher

`v:throwpoint` `v:throwpoint` `throwpoint-variable`
The point where the exception most recently caught and not finished was thrown. Not set when commands are typed. See also `v:exception` and `throw-variables`.
Example:

```
:try
: throw "oops"
:catch /.*/
: echo "Exception from" v:throwpoint
:endtry
```

Output: "Exception from test.vim, line 2"

`v:true` `v:true` `true-variable`
A Number with value one. Used to put `"true"` in JSON. See `json_encode()`.
When used as a string this evaluates to `"v:true"`.

```
echo v:true
v:true
```

That is so that `eval()` can parse the string back to the same value. Read-only.
In `Vim9` script `"true"` can be used which has a boolean type.

`v:val` `v:val` `val-variable`
Value of the current item of a `List` or `Dictionary`. Only valid while evaluating the expression used with `map()` and `filter()`. Read-only.

`v:version` `v:version` `version-variable`
Version number of Vim: Major version number times 100 plus minor version number. Version 5.0 is 500. Version 5.1 is 501. Read-only. `"version"` also works, for backwards compatibility, unless `scriptversion` is 3 or higher. Use `has()` to check if a certain patch was included, e.g.:

```
if has("patch-7.4.123")
```

Note that patch numbers are specific to the version, thus both version 5.0 and 5.1 may have a patch 123, but these are

completely different.

`v:versionlong` Like `v:version`, but also including the patchlevel in the last four digits. Version 8.1 with patch 123 has value 8010123. This can be used like this:

```
if v:versionlong >= 8010123
```

However, if there are gaps in the list of patches included this will not work well. This can happen if a recent patch was included into an older version, e.g. for a security fix. Use the `has()` function to make sure the patch is actually included.

`v:vim_did_enter` Zero until most of startup is done. It is set to one just before `VimEnter` autocommands are triggered.

`v:warningmsg` Last given warning message. It's allowed to set this variable.

`v:windowid` When any X11 based GUI is running or when running in a terminal and Vim connects to the X server (`-X`) this will be set to the window ID. When an MS-Windows GUI is running this will be set to the window handle. Otherwise the value is zero.
Note: for windows inside Vim use `winnr()` or `win_getid()` , see `window-ID` .

4. Builtin Functions

[functions](#)

See [function-list](#) for a list grouped by what the function is used for.

The alphabetic list of all builtin functions and details are in a separate help file: [builtin-functions](#) .

5. Defining functions

[user-functions](#)

New functions can be defined. These can be called just like builtin functions. The function takes arguments, executes a sequence of Ex commands and can return a value.

You can find most information about defining functions in [userfunc.txt](#) . For Vim9 functions, which execute much faster, support type checking and more, see [vim9.txt](#) .

6. Curly braces names

[curly-braces-names](#)

In most places where you can use a variable, you can use a "curly braces name" variable. This is a regular variable name with one or more expressions

wrapped in braces {} like this:
`my_{adjective}_variable`

This only works in legacy Vim script, not in Vim9 script.

When Vim encounters this, it evaluates the expression inside the braces, puts that in place of the expression, and re-interprets the whole as a variable name. So in the above example, if the variable "adjective" was set to "noisy", then the reference would be to "my_noisy_variable", whereas if "adjective" was set to "quiet", then it would be to "my_quiet_variable".

One application for this is to create a set of variables governed by an option value. For example, the statement

```
echo my_{&background}_message
```

would output the contents of "my_dark_message" or "my_light_message" depending on the current value of 'background'.

You can use multiple brace pairs:

```
echo my_{adverb}_{adjective}_message
```

..or even nest them:

```
echo my_{ad{end_of_word}}_message
```

where "end_of_word" is either "verb" or "jective".

However, the expression inside the braces must evaluate to a valid single variable name, e.g. this is invalid:

```
:let foo='a + b'  
:echo c{foo}d
```

.. since the result of expansion is "ca + bd", which is not a variable name.

curly-braces-function-names

You can call and define functions by an evaluated name in a similar way.

Example:

```
:let func_end='whizz'  
:call my_func_{func_end}(parameter)
```

This would call the function "my_func_whizz(parameter)".

This does NOT work:

```
:let i = 3  
:let @{i} = '' " error  
:echo @{i} " error
```

7. Commands

expression-commands

Note: in Vim9 script `:let` is not used. `:var` is used for variable declarations and assignments do not use a command. vim9-declaration

```
:let {var-name} = {expr1}
```

:let E18

Set internal variable {var-name} to the result of the expression {expr1}. The variable will get the type from the {expr}. If {var-name} didn't exist yet, it is created.

`:let {var-name}[{idx}] = {expr1}` E689 E1141
 Set a list item to the result of the expression `{expr1}`. `{var-name}` must refer to a list and `{idx}` must be a valid index in that list. For nested list the index can be repeated.
 This cannot be used to add an item to a `List`.
 This cannot be used to set a byte in a `String`. You can do that like this:
`:let var = var[0:2] .. 'X' .. var[4:]`
 When `{var-name}` is a `Blob` then `{idx}` can be the length of the blob, in which case one byte is appended.

`:let {var-name}[{idx1}:{idx2}] = {expr1}` E711 E719 E1165 E1166 E1183
E708 E709 E710
 Set a sequence of items in a `List` to the result of the expression `{expr1}`, which must be a list with the correct number of items.
`{idx1}` can be omitted, zero is used instead.
`{idx2}` can be omitted, meaning the end of the list.
 When the selected range of items is partly past the end of the list, items will be added.

`:let {var} += {expr1}` Like `":let {var} = {var} + {expr1}"`.
`:let {var} -= {expr1}` Like `":let {var} = {var} - {expr1}"`.
`:let {var} *= {expr1}` Like `":let {var} = {var} * {expr1}"`.
`:let {var} /= {expr1}` Like `":let {var} = {var} / {expr1}"`.
`:let {var} %= {expr1}` Like `":let {var} = {var} % {expr1}"`.
`:let {var} .= {expr1}` Like `":let {var} = {var} . {expr1}"`.
`:let {var} ..= {expr1}` Like `":let {var} = {var} .. {expr1}"`.
 These fail if `{var}` was not set yet and when the type of `{var}` and `{expr1}` don't fit the operator.
``+=`` modifies a `List` or a `Blob` in-place instead of creating a new one.
``.`` is not supported with Vim script version 2 and later, see [vimscrip-version](#).

`:let ${env-name} = {expr1}` :let-environment :let-\$
 Set environment variable `{env-name}` to the result of the expression `{expr1}`. The type is always `String`.

 On some systems making an environment variable empty causes it to be deleted. Many systems do not make a difference between an environment variable that is not set and an environment variable that is empty.

`:let ${env-name} .= {expr1}`
 Append `{expr1}` to the environment variable `{env-name}`. If the environment variable didn't exist yet this works like `"=`".

```

:let @{reg-name} = {expr1}           :let-register :let-@
Write the result of the expression {expr1} in register
{reg-name}. {reg-name} must be a single letter, and
must be the name of a writable register (see
registers ). "@" can be used for the unnamed
register, "@/" for the search pattern.
If the result of {expr1} ends in a <CR> or <NL>, the
register will be linewise, otherwise it will be set to
characterwise.
This can be used to clear the last search pattern:
:let @/ = ""
This is different from searching for an empty string,
that would match everywhere.

:let @{reg-name} .= {expr1}
Append {expr1} to register {reg-name}. If the
register was empty it's like setting it to {expr1}.

:let &{option-name} = {expr1}       :let-option :let-&
Set option {option-name} to the result of the
expression {expr1}. A String or Number value is
always converted to the type of the option.
For an option local to a window or buffer the effect
is just like using the :set command: both the local
value and the global value are changed.
Example:
:let &path = &path .. ',/usr/local/include'
This also works for terminal codes in the form t_xx.
But only for alphanumerical names. Example:
:let &t_k1 = "\<Esc>[234;"
When the code does not exist yet it will be created as
a terminal key code, there is no error.

:let &{option-name} .= {expr1}
For a string option: Append {expr1} to the value.
Does not insert a comma like :set+= .

:let &{option-name} += {expr1}
:let &{option-name} -= {expr1}
For a number or boolean option: Add or subtract
{expr1}.

:let &l:{option-name} = {expr1}
:let &l:{option-name} .= {expr1}
:let &l:{option-name} += {expr1}
:let &l:{option-name} -= {expr1}
Like above, but only set the local value of an option
(if there is one). Works like :setlocal .

:let &g:{option-name} = {expr1}
:let &g:{option-name} .= {expr1}
:let &g:{option-name} += {expr1}
:let &g:{option-name} -= {expr1}

```

Like above, but only set the global value of an option (if there is one). Works like `:setglobal`.

`:let [{name1}, {name2}, ...] = {expr1}` E1093
`:let-unpack` E687 E688
{expr1} must evaluate to a `List`. The first item in the list is assigned to {name1}, the second item to {name2}, etc.
The number of names must match the number of items in the `List`.
Each name can be one of the items of the `":let"` command as mentioned above.
Example:

```
:let [s, item] = GetItem(s)
```

Detail: {expr1} is evaluated first, then the assignments are done in sequence. This matters if {name2} depends on {name1}. Example:

```
:let x = [0, 1]
:let i = 0
:let [i, x[i]] = [1, 2]
:echo x
```

The result is [0, 2].

```
:let [{name1}, {name2}, ...] .= {expr1}
:let [{name1}, {name2}, ...] += {expr1}
:let [{name1}, {name2}, ...] -= {expr1}
```

Like above, but append/add/subtract the value for each `List` item.

```
:let [{name}, ..., ; {lastname}] = {expr1}
```

E452

Like `:let-unpack` above, but the `List` may have more items than there are names. A list of the remaining items is assigned to {lastname}. If there are no remaining items {lastname} is set to an empty list.

Example:

```
:let [a, b; rest] = ["aval", "bval", 3, 4]
```

```
:let [{name}, ..., ; {lastname}] .= {expr1}
:let [{name}, ..., ; {lastname}] += {expr1}
:let [{name}, ..., ; {lastname}] -= {expr1}
```

Like above, but append/add/subtract the value for each `List` item.

```
:let=<< :let-heredoc
E990 E991 E172 E221 E1145
```

```
:let {var-name} =<< [trim] [eval] {endmarker}
text...
text...
{endmarker}
```

Set internal variable {var-name} to a `List` containing the lines of text bounded by the string {endmarker}.

If "eval" is not specified, then each line of text is used as a `literal-string`, except that single quotes

does not need to be doubled.

If "eval" is specified, then any Vim expression in the form {expr} is evaluated and the result replaces the expression, like with [interpolated-string](#) .

Example where \$HOME is expanded:

```
let lines =<< trim eval END
  some text
  See the file {$HOME}/.vimrc
  more text
END
```

There can be multiple Vim expressions in a single line but an expression cannot span multiple lines. If any expression evaluation fails, then the assignment fails.

{endmarker} must not contain white space.

{endmarker} cannot start with a lower case character.

The last line should end only with the {endmarker} string without any other character. Watch out for white space after {endmarker}!

Without "trim" any white space characters in the lines of text are preserved. If "trim" is specified before {endmarker}, then indentation is stripped so you can do:

```
let text =<< trim END
  if ok
    echo 'done'
  endif
END
```

Results in: ["if ok", " echo 'done'", "endif"]

The marker must line up with "let" and the indentation of the first line is removed from all the text lines. Specifically: all the leading indentation exactly matching the leading indentation of the first non-empty text line is stripped from the input lines. All leading indentation exactly matching the leading indentation before `let` is stripped from the line containing {endmarker}. [Note](#) that the difference between space and tab matters here.

If {var-name} didn't exist yet, it is created.

Cannot be followed by another command, but can be followed by a comment.

To avoid line continuation to be applied, consider adding 'C' to [cptions](#):

```
set cpo+=C
let var =<< END
  \ leading backslash
END
set cpo-=C
```

Examples:

```
let var1 =<< END
```



```

Sample text 1
  Sample text 2
Sample text 3
END

let data =<< trim DATA
      1 2 3 4
      5 6 7 8

DATA

let code =<< trim eval CODE
  let v = {10 + 20}
  let h = "{$HOME}"
  let s = "{Str1()} abc {Str2()}"
  let n = {MyFunc(3, 4)}
CODE

```

`:let {var-name} ..` List the value of variable `{var-name}`. Multiple variable names may be given. Special names recognized here: E121

```

g: global variables
b: local buffer variables
w: local window variables
t: local tab page variables
s: script-local variables
l: local function variables
v: Vim variables.

```

This does not work in Vim9 script. [vim9-declaration](#) E738

`:let` List the values of all variables. The type of the variable is indicated before the value:

```

<nothing> String
#         Number
*         Funcref

```

This does not work in Vim9 script. [vim9-declaration](#)

`:unl[et][!] {name} ...` :unlet :unl E108 E795 E1081

Remove the internal variable `{name}`. Several variable names can be given, they are all removed. The name may also be a [List](#) or [Dictionary](#) item.

With `[!]` no error message is given for non-existing variables.

One or more items from a [List](#) can be removed:

```

:unlet list[3]    " remove fourth item
:unlet list[3:]  " remove fourth item to last

```

One item from a [Dictionary](#) can be removed at a time:

```

:unlet dict['two']
:unlet dict.two

```

This is especially useful to clean up used global variables and script-local variables (these are not deleted when the script ends). Function-local variables are automatically deleted when the function ends.

In Vim9 script variables declared in a function or script cannot be removed.

```
:unl[et] ${env-name} ...                               :unlet-environment  :unlet-$
Remove environment variable {env-name}.
Can mix {name} and ${env-name} in one :unlet command.
No error message is given for a non-existing
variable, also without !.
If the system does not support deleting an environment
variable, it is made empty.
```

```
                                     :cons  :const  E1018
:cons[t] {var-name} = {expr1}
:cons[t] [{name1}, {name2}, ...] = {expr1}
:cons[t] [{name}, ..., ; {lastname}] = {expr1}
:cons[t] {var-name} =<< [trim] {marker}
text...
text...
{marker}
```

Similar to `:let`, but additionally lock the variable after setting the value. This is the same as locking the variable with `:lockvar` just after `:let`, thus:

```
:const x = 1
```

is equivalent to:

```
:let x = 1
```

```
:lockvar! x
```

NOTE: in Vim9 script `:const` works differently, see vim9-const`

This is useful if you want to make sure the variable is not modified. If the value is a List or Dictionary literal then the items also cannot be changed:

```
const ll = [1, 2, 3]
```

```
let ll[1] = 5 " Error!
```

Nested references are not locked:

```
let lvar = ['a']
```

```
const lconst = [0, lvar]
```

```
let lconst[0] = 2 " Error!
```

```
let lconst[1][0] = 'b' " OK
```

E995

`:const` does not allow to for changing a variable:

```
:let x = 1
```

```
:const x = 2 " Error!
```

E996

Note that environment variables, option values and register values cannot be used here, since they cannot be locked.

```
:cons[t]
:cons[t] {var-name}
```

If no argument is given or only `{var-name}` is given, the behavior is the same as `:let`.

```
:lockv[ar][!] [depth] {name} ...                       :lockvar  :lockv
Lock the internal variable {name}. Locking means that
```

it can no longer be changed (until it is unlocked).
A locked variable can be deleted:

```
:lockvar v
:let v = 'asdf' " fails!
:unlock v      " works
```

E741 E940 E1118 E1119 E1120 E1121 E1122

If you try to change a locked variable you get an error message: "E741: Value is locked: {name}".
If you try to lock or unlock a built-in variable you get an error message: "E940: Cannot lock or unlock variable {name}".

[depth] is relevant when locking a List or Dictionary . It specifies how deep the locking goes:

- 0 Lock the variable {name} but not its value.
- 1 Lock the List or Dictionary itself, cannot add or remove items, but can still change their values.
- 2 Also lock the values, cannot change the items. If an item is a List or Dictionary , cannot add or remove items, but can still change the values.
- 3 Like 2 but for the List / Dictionary in the List / Dictionary , one level deeper.

The default [depth] is 2, thus when {name} is a List or Dictionary the values cannot be changed.

Example with [depth] 0:

```
let mylist = [1, 2, 3]
lockvar 0 mylist
let mylist[0] = 77 " OK
call add(mylist, 4) " OK
let mylist = [7, 8, 9] " Error!
```

E743

For unlimited depth use [!] and omit [depth].
However, there is a maximum depth of 100 to catch loops.

Note that when two variables refer to the same List and you lock one of them, the List will also be locked when used through the other variable.

Example:

```
:let l = [0, 1, 2, 3]
:let cl = l
:lockvar l
:let cl[1] = 99 " won't work!
```

You may want to make a copy of a list to avoid this.
See `deepcopy()` .

E1391 E1392

Locking and unlocking object and class variables is

currently NOT supported.

`:unlo[ckvar][!] [depth] {name} ...` `:unlockvar` `:unlo` E1246
Unlock the internal variable `{name}`. Does the opposite of `:lockvar`.

If `{name}` does not exist:

- In `Vim9` script an error is given.
- In legacy script this is silently ignored.

`:if {expr1}` `:if` `:end` `:endif` `:en` E171 E579 E580
`:en[dif]`
Execute the commands until the next matching `:else`` or `:endif`` if `{expr1}` evaluates to non-zero. Although the short forms work, it is recommended to always use `:endif`` to avoid confusion and to make auto-indenting work properly.

From `Vim version 4.5` until 5.0, every Ex command in between the `:if`` and `:endif`` is ignored. These two commands were just to allow for future expansions in a backward compatible way. Nesting was allowed. **Note** that any `:else`` or `:elseif`` was ignored, the `:else`` part was not executed either.

You can use this to remain compatible with older versions:

```
:if version >= 500
: version-5-specific-commands
:endif
```

The commands still need to be parsed to find the `:endif``. Sometimes an older Vim has a problem with a new command. For example, `:silent`` is recognized as a `:substitute`` command. In that case `:execute`` can avoid problems:

```
:if version >= 600
: execute "silent 1,$delete"
:endif
```

In `Vim9` script `:endif`` cannot be shortened, to improve script readability.

NOTE: The `:append`` and `:insert`` commands don't work properly in between `:if`` and `:endif``.

`:el[se]` `:else` `:el` E581 E583
Execute the commands until the next matching `:else`` or `:endif`` if they previously were not being executed.

In `Vim9` script `:else`` cannot be shortened, to improve script readability.

`:elsei[f] {expr1}` `:elseif` `:elsei` E582 E584
Short for `:else` :if``, with the addition that there is no extra `:endif``.

In Vim9 script `:elseif` cannot be shortened, to improve script readability.

```
:wh[ile] {expr1}           :while   :endwhile   :wh   :endw
                           E170   E585   E588   E733
:endw[hile]
```

Repeat the commands between `:while` and `:endwhile`, as long as `{expr1}` evaluates to non-zero. When an error is detected from a command inside the loop, execution continues after the `:endwhile`.
Example:

```
:let lnum = 1
:while lnum <= line("$")
  :call FixLine(lnum)
  :let lnum = lnum + 1
:endwhile
```

In Vim9 script `:while` and `:endwhile` cannot be shortened, to improve script readability.
NOTE: The `:append` and `:insert` commands don't work properly inside a `:while` and `:for` loop.

```
:for {var} in {object}           :for     E690   E732
                                 :endfo   :endfor
:endfo[r]
```

Repeat the commands between `:for` and `:endfor` for each item in `{object}`. `{object}` can be a `List`, a `Blob` or a `String`. E1177

Variable `{var}` is set to the value of each item. In Vim9 script the loop variable must not have been declared yet, unless when it is a global/window/tab/buffer variable.

When an error is detected for a command inside the loop, execution continues after the `:endfor`. Changing `{object}` inside the loop affects what items are used. Make a copy if this is unwanted:
`:for item in copy(mylist)`

When `{object}` is a `List` and not making a copy, in legacy script Vim stores a reference to the next item in the `List` before executing the commands with the current item. Thus the current item can be removed without effect. Removing any later item means it will not be found. Thus the following example works (an inefficient way to make a `List` empty):

```
for item in mylist
  call remove(mylist, 0)
endfor
```

Note that reordering the `List` (e.g., with `sort()` or `reverse()`) may have unexpected effects. In Vim9 script the index is used. If an item before the current one is deleted the next item will be skipped.

When `{object}` is a `Blob`, Vim always makes a copy to iterate over. Unlike with `List`, modifying the `Blob` does not affect the iteration.

When `{object}` is a `String` each item is a string with one character, plus any combining characters.

In `Vim9` script `:endfor` cannot be shortened, to improve script readability.

```
:for [{var1}, {var2}, ...] in {listlist}
:endfo[r]
```

E1140

Like `:for` above, but each item in `{listlist}` must be a list, of which each item is assigned to `{var1}`, `{var2}`, etc. Example:

```
:for [lnum, col] in [[1, 3], [2, 5], [3, 8]]
:echo getline(lnum)[col]
:endfor
```

```
:con[tinue]
```

`:continue` `:con` E586

When used inside a `:while` or `:for` loop, jumps back to the start of the loop.

If it is used after a `:try` inside the loop but before the matching `:finally` (if present), the commands following the `:finally` up to the matching `:endtry` are executed first. This process applies to all nested `:try`'s inside the loop. The outermost `:endtry` then jumps back to the start of the loop.

In `Vim9` script `:cont` is the shortest form, to improve script readability.

```
:brea[k]
```

`:break` `:brea` E587

When used inside a `:while` or `:for` loop, skips to the command after the matching `:endwhile` or `:endfor`.

If it is used after a `:try` inside the loop but before the matching `:finally` (if present), the commands following the `:finally` up to the matching `:endtry` are executed first. This process applies to all nested `:try`'s inside the loop. The outermost `:endtry` then jumps to the command after the loop.

In `Vim9` script `:break` cannot be shortened, to improve script readability.

```
:try
```

`:try` `:endt` `:endtry`
E600 E601 E602 E1032

```
:endt[ry]
```

Change the error handling for the commands between `:try` and `:endtry` including everything being executed across `:source` commands, function calls, or autocommand invocations.

When an error or interrupt is detected and there is a `:finally` command following, execution continues

after the `:finally``. Otherwise, or when the `:endtry`` is reached thereafter, the next (dynamically) surrounding `:try`` is checked for a corresponding `:finally`` etc. Then the script processing is terminated. Whether a function definition has an "abort" argument does not matter.

Example:

```
try | call Unknown() | finally | echomsg "cleanup" | endtry
echomsg "not reached"
```

Moreover, an error or interrupt (dynamically) inside `:try`` and `:endtry`` is converted to an exception. It can be caught as if it were thrown by a `:throw`` command (see `:catch``). In this case, the script processing is not terminated.

The value "Vim:Interrupt" is used for an interrupt exception. An error in a Vim command is converted to a value of the form "Vim({command}):{errmsg}", other errors are converted to a value of the form "Vim:{errmsg}". `{command}` is the full command name, and `{errmsg}` is the message that is displayed if the error exception is not caught, always beginning with the error number.

Examples:

```
try | sleep 100 | catch /^Vim:Interrupt$/ | endtry
try | edit | catch /^Vim(edit):E\d\+/ | echo "error" | endtry
```

In Vim9 script `:endtry`` cannot be shortened, to improve script readability.

```

:cat[ch] /{pattern}/
                :cat  :catch
                E603  E604  E605  E654  E1033
```

The following commands until the next `:catch``, `:finally``, or `:endtry`` that belongs to the same `:try`` as the `:catch`` are executed when an exception matching `{pattern}` is being thrown and has not yet been caught by a previous `:catch``. Otherwise, these commands are skipped.

When `{pattern}` is omitted all errors are caught.

Examples:

```

:catch /^Vim:Interrupt$/           " catch interrupts (CTRL-C)
:catch /^Vim%\%((\a+)\)\|=E/       " catch all Vim errors
:catch /^Vim%\%((\a+)\)\|=:/       " catch errors and interrupts
:catch /^Vim(write):/              " catch all errors in :write
:catch /^Vim%\%((\a+)\)\|=E123:/    " catch error E123
:catch /my-exception/              " catch user exception
:catch /.*/                         " catch everything
:catch                             " same as /.*/
```

Another character can be used instead of / around the `{pattern}`, so long as it does not have a special meaning (e.g., '|' or '"') and doesn't occur inside `{pattern}`. E1067

Information about the exception is available in `v:exception` . Also see `throw-variables` .
NOTE: It is not reliable to `":catch"` the TEXT of an error message because it may vary in different locales.
In `Vim9` script `":catch`` cannot be shortened, to improve script readability.

`:fina[lly]` `:fina` `:finally` E606 E607
The following commands until the matching `":endtry`` are executed whenever the part between the matching `":try`` and the `":finally`` is left: either by falling through to the `":finally`` or by a `":continue``, `":break``, `":finish``, or `":return``, or by an error or interrupt or exception (see `":throw``) .

In `Vim9` script `":finally`` cannot be shortened, to improve script readability and avoid confusion with `":final`` .

`:th[ow] {expr1}` `:th` `:throw` E608 E1129
The `{expr1}` is evaluated and thrown as an exception. If the `":throw"` is used after a `":try`` but before the first corresponding `":catch``, commands are skipped until the first `":catch`` matching `{expr1}` is reached. If there is no such `":catch`` or if the `":throw"` is used after a `":catch`` but before the `":finally``, the commands following the `":finally`` (if present) up to the matching `":endtry`` are executed. If the `":throw`` is after the `":finally``, commands up to the `":endtry`` are skipped. At the `":endtry"`, this process applies again for the next dynamically surrounding `":try`` (which may be found in a calling function or sourcing script), until a matching `":catch`` has been found. If the exception is not caught, the command processing is terminated.

Example:

```
:try | throw "oops" | catch /^oo/ | echo "caught" | endtry
```

Note that `"catch"` may need to be on a separate line for when an error causes the parsing to skip the whole line and not see the `"|"` that separates the commands.

In `Vim9` script `":throw`` cannot be shortened, to improve script readability.

`:ec[ho] {expr1} ..` `:ec` `:echo`
Echoes each `{expr1}`, with a space in between. The first `{expr1}` starts on a new line. Also see `:comment` .
Use `"\n"` to start a new line. Use `"\r"` to move the cursor to the first column.
Uses the highlighting set by the `":echohl`` command. Cannot be followed by a comment.
Example:


```
:echo "the value of 'shell' is" &shell
```

:echo-redraw

A later redraw may make the message disappear again. And since Vim mostly postpones redrawing until it's finished with a sequence of commands this happens quite often. To avoid that a command from before the `:echo` causes a redraw afterwards (redraws are often postponed until you type something), force a redraw with the `:redraw` command. Example:

```
:new | redraw | echo "there is a new window"
```

:echon

`:echon {expr1} ..` Echoes each `{expr1}`, without anything added. Also see `:comment`.

Uses the highlighting set by the `:echohl` command. Cannot be followed by a comment.

Example:

```
:echon "the value of 'shell' is " &shell
```

Note the difference between using `:echo`, which is a Vim command, and `:!echo`, which is an external shell command:

```
:!echo % --> filename
```

The arguments of `:!"` are expanded, see `:_%`.

```
:!echo %" --> filename or "filename"
```

Like the previous example. Whether you see the double quotes or not depends on your `'shell'`.

```
:echo % --> nothing
```

The `'%` is an illegal character in an expression.

```
:echo %" --> %
```

This just echoes the `'%` character.

```
:echo expand("%") --> filename
```

This calls the `expand()` function to expand the `'%`.

:echoh **:echohl**

`:echoh[l] {name}` Use the highlight group `{name}` for the following `:echo`, `:echon` and `:echomsg` commands. Also used for the `input()` prompt. Example:

```
:echohl WarningMsg | echo "Don't panic!" | echohl None
```

Don't forget to set the group back to `"None"`, otherwise all following echo's will be highlighted.

:echom **:echomsg**

`:echom[sg] {expr1} ..` Echo the expression(s) as a true message, saving the message in the `message-history`.

Spaces are placed between the arguments as with the `:echo` command. But unprintable characters are displayed, not interpreted.

The parsing works slightly different from `:echo`, more like `:execute`. All the expressions are first evaluated and concatenated before echoing anything.

If expressions does not evaluate to a Number or String, `string()` is used to turn it into a string. Uses the highlighting set by the `:echohl` command.

Example:

```
:echomsg "It's a Zizzer Zazzer Zuzz, as you can plainly see."  
See :echo-redraw to avoid the message disappearing  
when the screen is redrawn.
```

:echow :echowin :echowindow

:**[N]**echow[indow] {**expr1**} ..

Like :echomsg but when the messages popup window is available the message is displayed there. This means it will show for three seconds and avoid a

hit-enter prompt. If you want to hide it before that, press Esc in Normal mode (when it would otherwise beep). If it disappears too soon you can use :messages to see the text.

When [N] is given then the window will show up for this number of seconds. The last :echowindow with a count matters, it is used once only.

The message window is available when Vim was compiled with the +timer and the +popupwin features.

:echoe :echoerr

:echoe[rr] {**expr1**} ..

Echo the expression(s) as an error message, saving the message in the message-history . When used in a script or function the line number will be added. Spaces are placed between the arguments as with the :echomsg command. When used inside a try conditional, the message is raised as an error exception instead (see try-echoerr).

Example:

```
:echoerr "This script just failed!"
```

If you just want a highlighted message use :echohl .

And to get a beep:

```
:exe "normal \<Esc>"
```

:echoc :echoconsole

:echoc[onsole] {**expr1**} ..

Intended for testing: works like :echomsg but when running in the GUI and started from a terminal write the text to stdout.

:eval

:eval {**expr**}

Evaluate {**expr**} and discard the result. Example:

```
:eval Getlist()->Filter()->append('$')
```

The expression is supposed to have a side effect, since the resulting value is not used. In the example the append() call appends the List with text to the buffer. This is similar to :call but works with any expression.

In Vim9 script an expression without an effect will result in error E1207 . This should help noticing mistakes.

The command can be shortened to :ev or :eva , but these are hard to recognize and therefore not to be

used.

The command cannot be followed by "|" and another command, since "|" is seen as part of the expression.

`:exe[cute] {expr1} ..` `:exe` `:execute`
Executes the string that results from the evaluation of {expr1} as an Ex command. Multiple arguments are concatenated, with a space in between. To avoid the extra space use the ".." operator to concatenate strings into one argument. {expr1} is used as the processed command, command line editing keys are not recognized. Cannot be followed by a comment.

Examples:

```
:execute "buffer" nextbuf
:execute "normal" count .. "w"
```

":execute" can be used to append a command to commands that don't accept a '|'. Example:

```
:execute '!ls' | echo "theend"
```

":execute" is also a nice way to avoid having to type control characters in a Vim script for a ":normal" command:

```
:execute "normal ixxx\<Esc>"
```

This has an <Esc> character, see [expr-string](#).

Be careful to correctly escape special characters in file names. The `fnameescape()` function can be used for Vim commands, `shellescape()` for `:!` commands.

Examples:

```
:execute "e " .. fnameescape(filename)
:execute "!ls " .. shellescape(filename, 1)
```

Note: The executed string may be any command-line, but starting or ending "if", "while" and "for" does not always work, because when commands are skipped the ":execute" is not evaluated and Vim loses track of where blocks start and end. Also "break" and "continue" should not be inside ":execute".

This example does not work, because the ":execute" is not evaluated and Vim does not see the "while", and gives an error for finding an ":endwhile":

```
:if 0
: execute 'while i > 5'
: echo "test"
: endwhile
:endif
```

It is allowed to have a "while" or "if" command completely in the executed string:

```
:execute 'while i < 5 | echo i | let i = i + 1 | endwhile'
```

[:exe-comment](#)

"`:execute`", "`:echo`" and "`:echon`" cannot be followed by a comment directly, because they see the `'` as the start of a string. But, you can use `|` followed by a comment. Example:

```
:echo "foo" | "this is a comment"
```

8. Exception handling

[exception-handling](#)

The Vim script language comprises an exception handling feature. This section explains how it can be used in a Vim script.

Exceptions may be raised by Vim on an error or on interrupt, see [catch-errors](#) and [catch-interrupt](#). You can also explicitly throw an exception by using the `:throw` command, see [throw-catch](#).

TRY CONDITIONALS

[try-conditionals](#)

Exceptions can be caught or can cause cleanup code to be executed. You can use a try conditional to specify catch clauses (that catch exceptions) and/or a finally clause (to be executed for cleanup).

A try conditional begins with a `:try` command and ends at the matching `:endtry` command. In between, you can use a `:catch` command to start a catch clause, or a `:finally` command to start a finally clause. There may be none or multiple catch clauses, but there is at most one finally clause, which must not be followed by any catch clauses. The lines before the catch clauses and the finally clause is called a try block.

```
:try
: ...
: ...
: ...
:catch /{pattern}/
: ...
: ...
: ...
:catch /{pattern}/
: ...
: ...
: ...
:finally
: ...
: ...
: ...
:endtry
```

TRY BLOCK

CATCH CLAUSE

CATCH CLAUSE

FINALLY CLAUSE

The try conditional allows to watch code for exceptions and to take the appropriate actions. Exceptions from the try block may be caught. Exceptions from the try block and also the catch clauses may cause cleanup actions.

When no exception is thrown during execution of the try block, the control

is transferred to the finally clause, if present. After its execution, the script continues with the line following the `":endtry"`.

When an exception occurs during execution of the try block, the remaining lines in the try block are skipped. The exception is matched against the patterns specified as arguments to the `":catch"` commands. The catch clause after the first matching `":catch"` is taken, other catch clauses are not executed. The catch clause ends when the next `":catch"`, `":finally"`, or `":endtry"` command is reached - whatever is first. Then, the finally clause (if present) is executed. When the `":endtry"` is reached, the script execution continues in the following line as usual.

When an exception that does not match any of the patterns specified by the `":catch"` commands is thrown in the try block, the exception is not caught by that try conditional and none of the catch clauses is executed. Only the finally clause, if present, is taken. The exception pends during execution of the finally clause. It is resumed at the `":endtry"`, so that commands after the `":endtry"` are not executed and the exception might be caught elsewhere, see [try-nesting](#) .

When during execution of a catch clause another exception is thrown, the remaining lines in that catch clause are not executed. The new exception is not matched against the patterns in any of the `":catch"` commands of the same try conditional and none of its catch clauses is taken. If there is, however, a finally clause, it is executed, and the exception pends during its execution. The commands following the `":endtry"` are not executed. The new exception might, however, be caught elsewhere, see [try-nesting](#) .

When during execution of the finally clause (if present) an exception is thrown, the remaining lines in the finally clause are skipped. If the finally clause has been taken because of an exception from the try block or one of the catch clauses, the original (pending) exception is discarded. The commands following the `":endtry"` are not executed, and the exception from the finally clause is propagated and can be caught elsewhere, see [try-nesting](#) .

The finally clause is also executed, when a `":break"` or `":continue"` for a `":while"` loop enclosing the complete try conditional is executed from the try block or a catch clause. Or when a `":return"` or `":finish"` is executed from the try block or a catch clause of a try conditional in a function or sourced script, respectively. The `":break"`, `":continue"`, `":return"`, or `":finish"` pends during execution of the finally clause and is resumed when the `":endtry"` is reached. It is, however, discarded when an exception is thrown from the finally clause.

When a `":break"` or `":continue"` for a `":while"` loop enclosing the complete try conditional or when a `":return"` or `":finish"` is encountered in the finally clause, the rest of the finally clause is skipped, and the `":break"`, `":continue"`, `":return"` or `":finish"` is executed as usual. If the finally clause has been taken because of an exception or an earlier `":break"`, `":continue"`, `":return"`, or `":finish"` from the try block or a catch clause, this pending exception or command is discarded.

For examples see [throw-catch](#) and [try-finally](#) .

NESTING OF TRY CONDITIONALS

[try-nesting](#)

Try conditionals can be nested arbitrarily. That is, a complete try conditional can be put into the try block, a catch clause, or the finally

clause of another try conditional. If the inner try conditional does not catch an exception thrown in its try block or throws a new exception from one of its catch clauses or its finally clause, the outer try conditional is checked according to the rules above. If the inner try conditional is in the try block of the outer try conditional, its catch clauses are checked, but otherwise only the finally clause is executed. It does not matter for nesting, whether the inner try conditional is directly contained in the outer one, or whether the outer one sources a script or calls a function containing the inner try conditional.

When none of the active try conditionals catches an exception, just their finally clauses are executed. Thereafter, the script processing terminates. An error message is displayed in case of an uncaught exception explicitly thrown by a `:"throw"` command. For uncaught error and interrupt exceptions implicitly raised by Vim, the error message(s) or interrupt message are shown as usual.

For examples see [throw-catch](#) .

EXAMINING EXCEPTION HANDLING CODE

[except-examine](#)

Exception handling code can get tricky. If you are in doubt what happens, set `'verbose'` to 13 or use the `:"13verbose"` command modifier when sourcing your script file. Then you see when an exception is thrown, discarded, caught, or finished. When using a verbosity level of at least 14, things pending in a finally clause are also shown. This information is also given in debug mode (see [debug-scripts](#)).

THROWING AND CATCHING EXCEPTIONS

[throw-catch](#)

You can throw any number or string as an exception. Use the `:"throw` command and pass the value to be thrown as argument:

```
:"throw 4711
:"throw "string"
```

[throw-expression](#)

You can also specify an expression argument. The expression is then evaluated first, and the result is thrown:

```
:"throw 4705 + strlen("string")
:"throw strpart("strings", 0, 6)
```

An exception might be thrown during evaluation of the argument of the `:"throw"` command. Unless it is caught there, the expression evaluation is abandoned. The `:"throw"` command then does not throw a new exception.

Example:

```
:"function! Foo(arg)
: try
:   throw a:arg
: catch /foo/
: endtry
: return 1
:endfunction
```

```

:
:function! Bar()
:  echo "in Bar"
:  return 4710
:endfunction
:
:throw Foo("arrgh") + Bar()

```

This throws "arrgh", and "in Bar" is not displayed since Bar() is not executed.

```

:throw Foo("foo") + Bar()

```

however displays "in Bar" and throws 4711.

Any other command that takes an expression as argument might also be abandoned by an (uncaught) exception during the expression evaluation. The exception is then propagated to the caller of the command.

Example:

```

:if Foo("arrgh")
:  echo "then"
:else
:  echo "else"
:endif

```

Here neither of "then" or "else" is displayed.

Exceptions can be caught by a try conditional with one or more [:catch](#) commands, see [try-conditionals](#). The values to be caught by each ":catch" command can be specified as a pattern argument. The subsequent catch clause gets executed when a matching exception is caught.

Example:

```

:function! Foo(value)
:  try
:    throw a:value
:  catch /^\

```

The first call to Foo() displays "Number thrown", the second "String thrown". An exception is matched against the ":catch" commands in the order they are specified. Only the first match counts. So you should place the more specific ":catch" first. The following order does not make sense:

```

:  catch /.*/
:    echo "String thrown"
:  catch /^\

```

```
: echo "Number thrown"
```

The first ":catch" here matches always, so that the second catch clause is never taken.

throw-variables

If you catch an exception by a general pattern, you may access the exact value in the variable `v:exception` :

```
: catch /^d\+$/  
: echo "Number thrown. Value is" v:exception
```

You may also be interested where an exception was thrown. This is stored in `v:throwpoint` . Note that "v:exception" and "v:throwpoint" are valid for the exception most recently caught as long it is not finished.

Example:

```
:function! Caught()  
: if v:exception != ""  
:   echo 'Caught "' . v:exception .. "' in ' .. v:throwpoint  
: else  
:   echo 'Nothing caught'  
: endif  
:endfunction  
:  
:function! Foo()  
: try  
:   try  
:     throw 4711  
:   finally  
:     call Caught()  
:   endtry  
:   catch /.*/  
:     call Caught()  
:     throw "oops"  
:   endtry  
:   catch /.*/  
:     call Caught()  
:   finally  
:     call Caught()  
:   endtry  
:endfunction  
:  
:call Foo()
```

This displays

```
Nothing caught  
Caught "4711" in function Foo, line 4  
Caught "oops" in function Foo, line 10  
Nothing caught
```

A practical example: The following command ":LineNumber" displays the line

number in the script or function where it has been used:

```
:function! LineNumber()  
:   return substitute(v:throwpoint, '.*\D\(\d\+).*', '\1', "")  
:endfunction  
:command! LineNumber try | throw "" | catch | echo LineNumber() | endtry
```

try-nested

An exception that is not caught by a try conditional can be caught by a surrounding try conditional:

```
:try  
:  try  
:    throw "foo"  
:  catch /foobar/  
:    echo "foobar"  
:  finally  
:    echo "inner finally"  
:  endtry  
:catch /foo/  
:  echo "foo"  
:endtry
```

The inner try conditional does not catch the exception, just its finally clause is executed. The exception is then caught by the outer try conditional. The example displays "inner finally" and then "foo".

throw-from-catch

You can catch an exception and throw a new one to be caught elsewhere from the catch clause:

```
:function! Foo()  
:  throw "foo"  
:endfunction  
:  
:function! Bar()  
:  try  
:    call Foo()  
:  catch /foo/  
:    echo "Caught foo, throw bar"  
:    throw "bar"  
:  endtry  
:endfunction  
:  
:try  
:  call Bar()  
:catch /.*/  
:  echo "Caught" v:exception  
:endtry
```

This displays "Caught foo, throw bar" and then "Caught bar".

rethrow

There is no real rethrow in the Vim script language, but you may throw

"v:exception" instead:

```
:function! Bar()
:  try
:    call Foo()
:  catch /.*/
:    echo "Rethrow" v:exception
:    throw v:exception
:  endtry
:endifunction
```

try-echoerr

Note that this method cannot be used to "rethrow" Vim error or interrupt exceptions, because it is not possible to fake Vim internal exceptions. Trying so causes an error exception. You should throw your own exception denoting the situation. If you want to cause a Vim error exception containing the original error exception value, you can use the **:echoerr** command:

```
:try
:  try
:    asdf
:  catch /.*/
:    echoerr v:exception
:  endtry
:catch /.*/
:  echo v:exception
:endifunction
```

This code displays

```
Vim(echoerr):Vim:E492: Not an editor command:  asdf
```

CLEANUP CODE

try-finally

Scripts often change global settings and restore them at their end. If the user however interrupts the script by pressing **CTRL-C**, the settings remain in an inconsistent state. The same may happen to you in the development phase of a script when an error occurs or you explicitly throw an exception without catching it. You can solve these problems by using a try conditional with a finally clause for restoring the settings. Its execution is guaranteed on normal control flow, on error, on an explicit **":throw"**, and on interrupt. (Note that errors and interrupts from inside the try conditional are converted to exceptions. When not caught, they terminate the script after the finally clause has been executed.)

Example:

```
:try
:  let s:saved_ts = &ts
:  set ts=17
:
:  " Do the hard work here.
:
:finally
:  let &ts = s:saved_ts
```

```
: unlet s:saved_ts
: endtry
```

This method should be used locally whenever a function or part of a script changes global settings which need to be restored on failure or normal exit of that function or script part.

break-finally

Cleanup code works also when the try block or a catch clause is left by a `":continue"`, `":break"`, `":return"`, or `":finish"`.

Example:

```
:let first = 1
:while 1
:  try
:    if first
:      echo "first"
:      let first = 0
:      continue
:    else
:      throw "second"
:    endif
:  catch ././
:    echo v:exception
:    break
:  finally
:    echo "cleanup"
:  endtry
:  echo "still in while"
: endwhile
: echo "end"
```

This displays "first", "cleanup", "second", "cleanup", and "end".

```
:function! Foo()
:  try
:    return 4711
:  finally
:    echo "cleanup\n"
:  endtry
:  echo "Foo still active"
: endfunction
:
: echo Foo() "returned by Foo"
```

This displays "cleanup" and "4711 returned by Foo". You don't need to add an extra `":return"` in the finally clause. (Above all, this would override the return value.)

except-from-finally

Using either of `":continue"`, `":break"`, `":return"`, `":finish"`, or `":throw"` in a finally clause is possible, but not recommended since it abandons the cleanup actions for the try conditional. But, of course, interrupt and error exceptions might get raised from a finally clause.

Example where an error in the finally clause stops an interrupt from working correctly:

```
:try
:  try
:    echo "Press CTRL-C for interrupt"
:    while 1
:      endwhile
:    finally
:      unlet novar
:    endtry
:catch /novar/
:endtry
:echo "Script still running"
:sleep 1
```

If you need to put commands that could fail into a finally clause, you should think about catching or ignoring the errors in these commands, see [catch-errors](#) and [ignore-errors](#) .

CATCHING ERRORS

[catch-errors](#)

If you want to catch specific errors, you just have to put the code to be watched in a try block and add a catch clause for the error message. The presence of the try conditional causes all errors to be converted to an exception. No message is displayed and `v:errmsg` is not set then. To find the right pattern for the `:catch` command, you have to know how the format of the error exception is.

Error exceptions have the following format:

```
Vim({cmdname}):{errmsg}
```

or

```
Vim:{errmsg}
```

`{cmdname}` is the name of the command that failed; the second form is used when the command name is not known. `{errmsg}` is the error message usually produced when the error occurs outside try conditionals. It always begins with a capital "E", followed by a two or three-digit error number, a colon, and a space.

Examples:

The command

```
:unlet novar
```

normally produces the error message

```
E108: No such variable: "novar"
```

which is converted inside try conditionals to an exception

```
Vim(unlet):E108: No such variable: "novar"
```

The command

```
:dwim
```

normally produces the error message

```
E492: Not an editor command: dwim
```

which is converted inside try conditionals to an exception
`Vim:E492: Not an editor command: dwim`

You can catch all `":unlet"` errors by a
`:catch /^Vim(unlet):/`
or all errors for misspelled command names by a
`:catch /^Vim:E492:/`

Some error messages may be produced by different commands:
`:function nofunc`

and

```
:delfunction nofunc
```

both produce the error message

```
E128: Function name must start with a capital: nofunc
```

which is converted inside try conditionals to an exception

```
Vim(function):E128: Function name must start with a capital: nofunc
```

or

```
Vim(delfunction):E128: Function name must start with a capital: nofunc
```

respectively. You can catch the error by its number independently on the command that caused it if you use the following pattern:

```
:catch /^Vim(\a+):E128:/
```

Some commands like

```
:let x = novar
```

produce multiple error messages, here:

```
E121: Undefined variable: novar
```

```
E15: Invalid expression: novar
```

Only the first is used for the exception value, since it is the most specific one (see [except-several-errors](#)). So you can catch it by

```
:catch /^Vim(\a+):E121:/
```

You can catch all errors related to the name "nofunc" by

```
:catch /\<nofunc\>/
```

You can catch all Vim errors in the `":write"` and `":read"` commands by

```
:catch /^Vim(\(write\|read\)):E\d\+:/
```

You can catch all Vim errors by the pattern

```
:catch /^Vim(\(\a+\))\=:E\d\+:/
```

catch-text

NOTE: You should never catch the error message text itself:

```
:catch /No such variable/
```

only works in the English locale, but not when the user has selected a different language by the `:language` command. It is however helpful to cite the message text in a comment:

```
:catch /^Vim(\a+):E108:/ " No such variable
```

IGNORING ERRORS

ignore-errors

You can ignore errors in a specific Vim command by catching them locally:

```
:try
```

```
: write
:catch
:endtry
```

But you are strongly recommended NOT to use this simple form, since it could catch more than you want. With the ":write" command, some autocommands could be executed and cause errors not related to writing, for instance:

```
:au BufWritePre * unlet novar
```

There could even be such errors you are not responsible for as a script writer: a user of your script might have defined such autocommands. You would then hide the error from the user.

It is much better to use

```
:try
: write
:catch /^Vim(write):/
:endtry
```

which only catches real write errors. So catch only what you'd like to ignore intentionally.

For a single command that does not cause execution of autocommands, you could even suppress the conversion of errors to exceptions by the ":silent!" command:

```
:silent! nunmap k
```

This works also when a try conditional is active.

CATCHING INTERRUPTS

catch-interrupt

When there are active try conditionals, an interrupt (CTRL-C) is converted to the exception "Vim:Interrupt". You can catch it like every exception. The script is not terminated, then.

Example:

```
:function! TASK1()
: sleep 10
:endfunction

:function! TASK2()
: sleep 20
:endfunction

:while 1
: let command = input("Type a command: ")
: try
:   if command == ""
:     continue
:   elseif command == "END"
:     break
:   elseif command == "TASK1"
:     call TASK1()
```

```

: elseif command == "TASK2"
:   call TASK2()
: else
:   echo "\nIllegal command:" command
:   continue
: endif
: catch /^Vim:Interrupt$/
:   echo "\nCommand interrupted"
:   " Caught the interrupt. Continue with next prompt.
: endtry
: endwhile

```

You can interrupt a task here by pressing **CTRL-C**; the script then asks for a new command. If you press **CTRL-C** at the prompt, the script is terminated.

For testing what happens when **CTRL-C** would be pressed on a specific line in your script, use the debug mode and execute the `>quit` or `>interrupt` command on that line. See [debug-scripts](#) .

CATCHING ALL

`catch-all`

The commands

```

:catch /.*/
:catch //
:catch

```

catch everything, error exceptions, interrupt exceptions and exceptions explicitly thrown by the `:throw` command. This is useful at the top level of a script in order to catch unexpected things.

Example:

```

:try
:
: " do the hard work here
:
:catch /MyException/
:
: " handle known problem
:
:catch /^Vim:Interrupt$/
:   echo "Script interrupted"
:catch /.*/
:   echo "Internal error (" .. v:exception .. ")"
:   echo " - occurred at " .. v:throwpoint
: endtry
:" end of script

```

Note: Catching all might catch more things than you want. Thus, you are strongly encouraged to catch only for problems that you can really handle by specifying a pattern argument to the `:catch`.

Example: Catching all could make it nearly impossible to interrupt a script by pressing **CTRL-C**:

```

:while 1
:  try
:    sleep 1
:  catch
:  endtry
:endwhile

```

EXCEPTIONS AND AUTOCOMMANDS

except-autocmd

Exceptions may be used during execution of autocommands. Example:

```

:autocmd User x try
:autocmd User x  throw "Oops!"
:autocmd User x catch
:autocmd User x  echo v:exception
:autocmd User x endtry
:autocmd User x throw "Arrgh!"
:autocmd User x echo "Should not be displayed"
:
:try
:  doautocmd User x
:catch
:  echo v:exception
:endtry

```

This displays "Oops!" and "Arrgh!".

except-autocmd-Pre

For some commands, autocommands get executed before the main action of the command takes place. If an exception is thrown and not caught in the sequence of autocommands, the sequence and the command that caused its execution are abandoned and the exception is propagated to the caller of the command.

Example:

```

:autocmd BufWritePre * throw "FAIL"
:autocmd BufWritePre * echo "Should not be displayed"
:
:try
:  write
:catch
:  echo "Caught:" v:exception "from" v:throwpoint
:endtry

```

Here, the ":write" command does not write the file currently being edited (as you can see by checking '**modified**'), since the exception from the BufWritePre autocommand abandons the ":write". The exception is then caught and the script displays:

```
Caught: FAIL from BufWrite Auto commands for "*"

```

except-autocmd-Post

For some commands, autocommands get executed after the main action of the

command has taken place. If this main action fails and the command is inside an active try conditional, the autocommands are skipped and an error exception is thrown that can be caught by the caller of the command.

Example:

```
:autocmd BufWritePost * echo "File successfully written!"
:
:try
: write /i/m/p/o/s/s/i/b/l/e
:catch
: echo v:exception
:endtry
```

This just displays:

```
Vim(write):E212: Can't open file for writing (/i/m/p/o/s/s/i/b/l/e)
```

If you really need to execute the autocommands even when the main action fails, trigger the event from the catch clause.

Example:

```
:autocmd BufWritePre * set noreadonly
:autocmd BufWritePost * set readonly
:
:try
: write /i/m/p/o/s/s/i/b/l/e
:catch
: doautocmd BufWritePost /i/m/p/o/s/s/i/b/l/e
:endtry
```

You can also use ":silent!":

```
:let x = "ok"
:let v:errmsg = ""
:autocmd BufWritePost * if v:errmsg != ""
:autocmd BufWritePost * let x = "after fail"
:autocmd BufWritePost * endif
:try
: silent! write /i/m/p/o/s/s/i/b/l/e
:catch
:endtry
:echo x
```

This displays "after fail".

If the main action of the command does not fail, exceptions from the autocommands will be catchable by the caller of the command:

```
:autocmd BufWritePost * throw "-("
:autocmd BufWritePost * echo "Should not be displayed"
:
:try
: write
:catch
```

```

: echo v:exception
:endtry

```

except-autocmd-Cmd

For some commands, the normal action can be replaced by a sequence of autocommands. Exceptions from that sequence will be catchable by the caller of the command.

Example: For the ":write" command, the caller cannot know whether the file had actually been written when the exception occurred. You need to tell it in some way.

```

:if !exists("cnt")
: let cnt = 0
:
: autocmd BufWriteCmd * if &modified
: autocmd BufWriteCmd * let cnt = cnt + 1
: autocmd BufWriteCmd * if cnt % 3 == 2
: autocmd BufWriteCmd * throw "BufWriteCmdError"
: autocmd BufWriteCmd * endif
: autocmd BufWriteCmd * write | set nomodified
: autocmd BufWriteCmd * if cnt % 3 == 0
: autocmd BufWriteCmd * throw "BufWriteCmdError"
: autocmd BufWriteCmd * endif
: autocmd BufWriteCmd * echo "File successfully written!"
: autocmd BufWriteCmd * endif
:endif
:
:try
: write
:catch /^BufWriteCmdError$/
: if &modified
: echo "Error on writing (file contents not changed)"
: else
: echo "Error after writing"
: endif
:catch /^Vim(write):/
: echo "Error on writing"
:endtry

```

When this script is sourced several times after making changes, it displays first

```
File successfully written!
```

then

```
Error on writing (file contents not changed)
```

then

```
Error after writing
```

etc.

except-autocmd-ill

You cannot spread a try conditional over autocommands for different events. The following code is ill-formed:

```

:autocmd BufWritePre * try
:

```

```

:autocmd BufWritePost * catch
:autocmd BufWritePost *   echo v:exception
:autocmd BufWritePost * endtry
:
:write

```

EXCEPTION HIERARCHIES AND PARAMETERIZED EXCEPTIONS

except-hier-param

Some programming languages allow to use hierarchies of exception classes or to pass additional information with the object of an exception class. You can do similar things in Vim.

In order to throw an exception from a hierarchy, just throw the complete class name with the components separated by a colon, for instance throw the string "EXCEPT:MATHERR:OVERFLOW" for an overflow in a mathematical library.

When you want to pass additional information with your exception class, add it in parentheses, for instance throw the string "EXCEPT:IO:WRITEERR(myfile)" for an error when writing "myfile".

With the appropriate patterns in the ":catch" command, you can catch for base classes or derived classes of your hierarchy. Additional information in parentheses can be cut out from `v:exception` with the ":substitute" command.

Example:

```

:function! CheckRange(a, func)
:  if a:a < 0
:    throw "EXCEPT:MATHERR:RANGE(" .. a:func .. ")"
:  endif
:endfunction
:
:function! Add(a, b)
:  call CheckRange(a:a, "Add")
:  call CheckRange(a:b, "Add")
:  let c = a:a + a:b
:  if c < 0
:    throw "EXCEPT:MATHERR:OVERFLOW"
:  endif
:  return c
:endfunction
:
:function! Div(a, b)
:  call CheckRange(a:a, "Div")
:  call CheckRange(a:b, "Div")
:  if (a:b == 0)
:    throw "EXCEPT:MATHERR:ZERODIV"
:  endif
:  return a:a / a:b
:endfunction
:
:function! Write(file)
:  try
:    execute "write" fnameescape(a:file)
:  catch /^Vim(write):/
:    throw "EXCEPT:IO(" .. getcwd() .. ", " .. a:file .. "):WRITEERR"
:  endtry

```

```

:endifunction
:
:try
:
: " something with arithmetic and I/O
:
:catch /^EXCEPT:MATHERR:RANGE/
: let function = substitute(v:exception, '.*(\(\a\+\)).*', '\1', "")
: echo "Range error in" function
:
:catch /^EXCEPT:MATHERR/          " catches OVERFLOW and ZERODIV
: echo "Math error"
:
:catch /^EXCEPT:IO/
: let dir = substitute(v:exception, '.*(\(.+\),\s*.\+).*', '\1', "")
: let file = substitute(v:exception, '.*(\.+\,\s*\(.+\)).*', '\1', "")
: if file !~ '^/'
:   let file = dir .. "/" .. file
: endif
: echo 'I/O error for "' .. file .. "'"
:
:catch /^EXCEPT/
: echo "Unspecified error"
:
:endtry

```

The exceptions raised by Vim itself (on error or when pressing **CTRL-C**) use a flat hierarchy: they are all in the "Vim" class. You cannot throw yourself exceptions with the "Vim" prefix; they are reserved for Vim.

Vim error exceptions are parameterized with the name of the command that failed, if known. See [catch-errors](#) .

PECULIARITIES

except-compact

The exception handling concept requires that the command sequence causing the exception is aborted immediately and control is transferred to finally clauses and/or a catch clause.

In the Vim script language there are cases where scripts and functions continue after an error: in functions without the "abort" flag or in a command after ":",silent!", control flow goes to the following line, and outside functions, control flow goes to the line following the outermost ":",endwhile" or ":",endif". On the other hand, errors should be catchable as exceptions (thus, requiring the immediate abortion).

This problem has been solved by converting errors to exceptions and using immediate abortion (if not suppressed by ":",silent!") only when a try conditional is active. This is no restriction since an (error) exception can be caught only from an active try conditional. If you want an immediate termination without catching the error, just use a try conditional without catch clause. (You can cause cleanup code being executed before termination by specifying a finally clause.)

When no try conditional is active, the usual abortion and continuation behavior is used instead of immediate abortion. This ensures compatibility of scripts written for Vim 6.1 and earlier.

However, when sourcing an existing script that does not use exception handling commands (or when calling one of its functions) from inside an active try conditional of a new script, you might change the control flow of the existing script on error. You get the immediate abortion on error and can catch the error in the new script. If however the sourced script suppresses error messages by using the ":silent!" command (checking for errors by testing `v:errmsg` if appropriate), its execution path is not changed. The error is not converted to an exception. (See `:silent`.) So the only remaining cause where this happens is for scripts that don't care about errors and produce error messages. You probably won't want to use such code from your new scripts.

except-syntax-err

Syntax errors in the exception handling commands are never caught by any of the ":catch" commands of the try conditional they belong to. Its finally clauses, however, is executed.

Example:

```
:try
: try
:   throw 4711
: catch /\(/
:   echo "in catch with syntax error"
: catch
:   echo "inner catch-all"
: finally
:   echo "inner finally"
: endtry
:catch
: echo 'outer catch-all caught "' .. v:exception .. "'"
: finally
:   echo "outer finally"
: endtry
```

This displays:

```
inner finally
outer catch-all caught "Vim(catch):E54: Unmatched \("
outer finally
```

The original exception is discarded and an error exception is raised, instead.

except-single-line

The ":try", ":catch", ":finally", and ":endtry" commands can be put on a single line, but then syntax errors may make it difficult to recognize the "catch" line, thus you better avoid this.

Example:

```
:try | unlet! foo # | catch | endtry
```

raises an error exception for the trailing characters after the ":unlet!" argument, but does not see the ":catch" and ":endtry" commands, so that the error exception is discarded and the "E488: Trailing characters" message gets displayed.

except-several-errors

When several errors appear in a single command, the first error message is usually the most specific one and therefore converted to the error exception.

Example:

```
echo novar
```

causes

```
E121: Undefined variable: novar
```

```
E15: Invalid expression: novar
```

The value of the error exception inside try conditionals is:

```
Vim(echo):E121: Undefined variable: novar
```

except-syntax-error

But when a syntax error is detected after a normal error in the same command, the syntax error is used for the exception being thrown.

Example:

```
unlet novar #
```

causes

```
E108: No such variable: "novar"
```

```
E488: Trailing characters
```

The value of the error exception inside try conditionals is:

```
Vim(unlet):E488: Trailing characters
```

This is done because the syntax error might change the execution path in a way not intended by the user. Example:

```
try
  try | unlet novar # | catch | echo v:exception | endtry
catch /.*/
  echo "outer catch:" v:exception
endtry
```

This displays "outer catch: Vim(unlet):E488: Trailing characters", and then a "E600: Missing :endtry" error message is given, see [except-single-line](#).

9. Examples

eval-examples

Printing in Binary

```
:" The function Nr2Bin() returns the binary string representation of a number.
:func Nr2Bin(nr)
:  let n = a:nr
:  let r = ""
:  while n
:    let r = '01'[n % 2] .. r
:    let n = n / 2
:  endwhile
:  return r
:endfunc
```

```
:" The function String2Bin() converts each character in a string to a
:" binary string, separated with dashes.
:func String2Bin(str)
:  let out = ''
:  for ix in range(strlen(a:str))
:    let out = out .. '-' .. Nr2Bin(char2nr(a:str[ix]))
:  endfor
```

```
: return out[1:]
: endfunc
```

Example of its use:

```
: echo Nr2Bin(32)
result: "100000"
: echo String2Bin("32")
result: "110011-110010"
```

Sorting lines

This example sorts lines with a specific compare function.

```
: func SortBuffer()
:   let lines = getline(1, '$')
:   call sort(lines, function("Strcmp"))
:   call setline(1, lines)
: endfunction
```

As a one-liner:

```
: call setline(1, sort(getline(1, '$'), function("Strcmp")))
```

scanf() replacement

sscanf

There is no sscanf() function in Vim. If you need to extract parts from a line, you can use matchstr() and substitute() to do it. This example shows how to get the file name, line number and column number out of a line like "foobar.txt, 123, 45".

```
:" Set up the match bit
: let mx='\\(\\f+\\),\\s*\\(\\d+\\),\\s*\\(\\d+\\)'
:" get the part matching the whole expression
: let l = matchstr(line, mx)
:" get each item out of the match
: let file = substitute(l, mx, '\\1', '')
: let lnum = substitute(l, mx, '\\2', '')
: let col = substitute(l, mx, '\\3', '')
```

The input is in the variable "line", the results in the variables "file", "lnum" and "col". (idea from Michael Geddes)

getting the scriptnames in a Dictionary

scriptnames-dictionary

The `:scriptnames` command can be used to get a list of all script files that have been sourced. There is also the `getscriptinfo()` function, but the information returned is not exactly the same. In case you need to manipulate the list, this code can be used as a base:

```
# Create or update scripts dictionary, indexed by SNR, and return it.
def Scripts(scripts: dict<string> = {}): dict<string>
  for info in getscriptinfo()
    if scripts->has_key(info.sid)
```

```

        continue
    endif
    scripts[info.sid] = info.name
endfor
return scripts
enddef

```

10. Vim script versions vimscript-version vimscript-versions scriptversion

Over time many features have been added to Vim script. This includes Ex commands, functions, variable types, etc. Each individual feature can be checked with the `has()` and `exists()` functions.

Sometimes old syntax of functionality gets in the way of making Vim better. When support is taken away this will break older Vim scripts. To make this explicit the `:scriptversion` command can be used. When a Vim script is not compatible with older versions of Vim this will give an explicit error, instead of failing in mysterious ways.

When using a legacy function, defined with `:function``, in Vim9 script then `scriptversion 4` is used.

scriptversion-1

`:scriptversion 1`

This is the original Vim script, same as not using a `:scriptversion` command. Can be used to go back to old syntax for a range of lines. Test for support with:

```
has('vimscript-1')
```

scriptversion-2

`:scriptversion 2`

String concatenation with `."` is not supported, use `.."` instead. This avoids the ambiguity using `."` for Dict member access and floating point numbers. Now `".5"` means the number 0.5.

scriptversion-3

`:scriptversion 3`

All `vim-variable` s must be prefixed by `"v:"`. E.g. `"version"` doesn't work as `v:version` anymore, it can be used as a normal variable. Same for some obvious names as `"count"` and others.

Test for support with:

```
has('vimscript-3')
```

scriptversion-4

`:scriptversion 4`

Numbers with a leading zero are not recognized as octal. `"0o"` or `"0O"` is still recognized as octal. With the previous version you get:

```

echo 017 " displays 15 (octal)
echo 0o17 " displays 15 (octal)
echo 018 " displays 18 (decimal)

```

with script version 4:


```
echo 017 " displays 17 (decimal)
echo 0o17 " displays 15 (octal)
echo 018 " displays 18 (decimal)
```

Also, it is possible to use single quotes inside numbers to make them easier to read:

```
echo 1'000'000
```

The quotes must be surrounded by digits.

Test for support with:

```
has('vimscript-4')
```

11. No +eval feature

no-eval-feature

When the `+eval` feature was disabled at compile time, none of the expression evaluation commands are available. To prevent this from causing Vim scripts to generate all kinds of errors, the `:if` and `:endif` commands are still recognized, though the argument of the `:if` and everything between the `:if` and the matching `:endif` is ignored. Nesting of `:if` blocks is allowed, but only if the commands are at the start of the line. The `:else` command is not recognized.

Example of how to avoid executing commands when the `+eval` feature is missing:

```
:if 1
: echo "Expression evaluation is compiled in"
:else
: echo "You will _never_ see this message"
:endif
```

To execute a command only when the `+eval` feature is disabled can be done in two ways. The simplest is to exit the script (or Vim) prematurely:

```
if 1
  echo "commands executed with +eval"
  finish
endif
args " command executed without +eval
```

If you do not want to abort loading the script you can use a trick, as this example shows:

```
silent! while 0
  set history=111
silent! endwhile
```

When the `+eval` feature is available the command is skipped because of the `"while 0"`. Without the `+eval` feature the `"while 0"` is an error, which is silently ignored, and the command is executed.

12. The sandbox

eval-sandbox sandbox

The `'foldexpr'`, `'formatexpr'`, `'includeexpr'`, `'indentexpr'`, `'statusline'` and

'foldtext' options may be evaluated in a sandbox. This means that you are protected from these expressions having nasty side effects. This gives some safety for when these options are set from a modeline. It is also used when the command from a tags file is executed and for **CTRL-R** = in the command line. The sandbox is also used for the `:sandbox` command.

E48

These items are not allowed in the sandbox:

- changing the buffer text
- defining or changing mapping, autocommands, user commands
- setting certain options (see `option-summary`)
- setting certain v: variables (see `v:var`) E794
- executing a shell command
- reading or writing a file
- jumping to another buffer or editing a file
- executing Python, Perl, etc. commands

This is not guaranteed 100% secure, but it should block most attacks.

`:san[dbox] {cmd}` Execute {cmd} in the sandbox. Useful to evaluate an option that may have been set from a modeline, e.g. **'foldexpr'**.

`:san` `:sandbox`

`sandbox-option`

A few options contain an expression. When this expression is evaluated it may have to be done in the sandbox to avoid a security risk. But the sandbox is restrictive, thus this only happens when the option was set from an insecure location. Insecure in this context are:

- sourcing a `.vimrc` or `.exrc` in the current directory
- while executing in the sandbox
- value coming from a modeline
- executing a function that was defined in the sandbox

Note that when in the sandbox and saving an option value and restoring it, the option will still be marked as it was set in the sandbox.

13. Textlock

`textlock`

In a few situations it is not allowed to change the text in the buffer, jump to another window and some other things that might confuse or break what Vim is currently doing. This mostly applies to things that happen when Vim is actually doing something else. For example, evaluating the **'balloonexpr'** may happen any moment the mouse cursor is resting at some position.

This is not allowed when the textlock is active:

- changing the buffer text
- jumping to another buffer or window
- editing another file
- closing a window or quitting Vim
- etc.

14. Vim script library

`vim-script-library`

Vim comes bundled with a Vim script library, that can be used by runtime, script authors. Currently, it only includes very few functions, but it may grow over time.

The functions are available as `Vim9-script` as well as using legacy Vim script (to be used for non Vim 9.0 versions and Neovim).

The functions make use of the autoloader prefix `dist#vim` (for legacy Vim script and Neovim) and `dist#vim9` for Vim9 script.

The following functions are available:

```
dist#vim#IsSafeExecutable(filetype, executable)
dist#vim9#IsSafeExecutable(filetype:string, executable:string): bool
```

This function takes a filetype and an executable and checks whether it is safe to execute the given executable. For security reasons users may not want to have Vim execute random executables or may have forbidden to do so for specific filetypes by setting the `"<filetype>_exec"` variable (`plugin_exec`).

It returns `true` or `false` to indicate whether the plugin should run the given executable. It takes the following arguments:

argument	type
filetype	string
executable	string

Builtin functions

builtin-functions

Note: Expression evaluation can be disabled at compile time, the builtin functions are not available then. See +eval and no-eval-feature .

For functions grouped by what they are used for see function-list .

- 1. Overview [builtin-function-list](#)
- 2. Details [builtin-function-details](#)
- 3. Feature list [feature-list](#)
- 4. Matching a pattern in a String [string-match](#)



- 1. Overview [builtin-function-list](#)

Use CTRL-] on the function name to jump to the full explanation.

USAGE	RESULT	DESCRIPTION
abs({expr})	Float or Number	absolute value of {expr}
acos({expr})	Float	arc cosine of {expr}
add({object}, {item})	List/Blob	append {item} to {object}
and({expr}, {expr})	Number	bitwise AND
append({lnum}, {text})	Number	append {text} below line {lnum}
appendbufline({expr}, {lnum}, {text})	Number	append {text} below line {lnum} in buffer {expr}
argc([winid])	Number	number of files in the argument list
argidx()	Number	current index in the argument list
arglistid([winnr] [, tabnr])	Number	argument list id
argv({nr} [, winid])	String	{nr} entry of the argument list
argv([-1, winid])	List	the argument list
asin({expr})	Float	arc sine of {expr}
assert_beeps({cmd})	Number	assert {cmd} causes a beep
assert_equal({exp}, {act} [, msg])	Number	assert {exp} is equal to {act}
assert_equalfile({fname-one}, {fname-two} [, msg])	Number	assert file contents are equal
assert_exception({error} [, msg])	Number	assert {error} is in v:exception
assert_fails({cmd} [, error] [, msg] [, lnum] [, context])	Number	assert {cmd} fails
assert_false({actual} [, msg])	Number	assert {actual} is false
assert_inrange({lower}, {upper}, {actual} [, msg])	Number	assert {actual} is inside the range
assert_match({pat}, {text} [, msg])	Number	assert {pat} matches {text}

assert_nobeep({cmd})	Number	assert {cmd} does not cause a beep
assert_notequal({exp}, {act} [, {msg}])	Number	assert {exp} is not equal {act}
assert_notmatch({pat}, {text} [, {msg}])	Number	assert {pat} not matches {text}
assert_report({msg})	Number	report a test failure
assert_true({actual} [, {msg}])	Number	assert {actual} is true
atan({expr})	Float	arc tangent of {expr}
atan2({expr1}, {expr2})	Float	arc tangent of {expr1} / {expr2}
autocmd_add({acmds})	Bool	add a list of autocmds and groups
autocmd_delete({acmds})	Bool	delete a list of autocmds and groups
autocmd_get([opts])	List	return a list of autocmds
balloon_gettext()	String	current text in the balloon
balloon_show({expr})	none	show {expr} inside the balloon
balloon_split({msg})	List	split {msg} as used for a balloon
blob2list({blob})	List	convert {blob} into a list of numbers
browse({save}, {title}, {initdir}, {default})	String	put up a file requester
browsedir({title}, {initdir})	String	put up a directory requester
bufadd({name})	Number	add a buffer to the buffer list
bufexists({buf})	Number	TRUE if buffer {buf} exists
buflisted({buf})	Number	TRUE if buffer {buf} is listed
bufload({buf})	Number	load buffer {buf} if not loaded yet
bufloaded({buf})	Number	TRUE if buffer {buf} is loaded
bufname([{buf}])	String	Name of the buffer {buf}
bufnr([{buf} [, {create}])	Number	Number of the buffer {buf}
bufwinid({buf})	Number	window ID of buffer {buf}
bufwinnr({buf})	Number	window number of buffer {buf}
byte2line({byte})	Number	line number at byte count {byte}
byteidx({expr}, {nr} [, {utf16}])	Number	byte index of {nr}'th char in {expr}
byteidxcomp({expr}, {nr} [, {utf16}])	Number	byte index of {nr}'th char in {expr}
call({func}, {arglist} [, {dict}])	any	call {func} with arguments {arglist}
ceil({expr})	Float	round {expr} up
ch_canread({handle})	Number	check if there is something to read
ch_close({handle})	none	close {handle}
ch_close_in({handle})	none	close in part of {handle}
ch_evalexpr({handle}, {expr} [, {options}])	any	evaluate {expr} on JSON {handle}
ch_evalraw({handle}, {string} [, {options}])	any	evaluate {string} on raw {handle}
ch_getbufnr({handle}, {what})	Number	get buffer number for {handle}/{what}
ch_getjob({channel})	Job	get the Job of {channel}
ch_info({handle})	String	info about channel {handle}
ch_log({msg} [, {handle}])	none	write {msg} in the channel log file
ch_logfile({fname} [, {mode}])	none	start logging channel activity
ch_open({address} [, {options}])	Channel	open a channel to {address}
ch_read({handle} [, {options}])	String	read from {handle}
ch_readblob({handle} [, {options}])	Blob	read Blob from {handle}
ch_readraw({handle} [, {options}])		

ch_sendexpr({handle}, {expr} [, {options}])	String	read raw from {handle}
ch_senddraw({handle}, {expr} [, {options}])	any	send {expr} over JSON {handle}
ch_setoptoptions({handle}, {options})	any	send {expr} over raw {handle}
ch_status({handle} [, {options}])	none	set options for {handle}
changenr()	String	status of channel {handle}
char2nr({expr} [, {utf8}])	Number	current change number
charclass({string})	Number	ASCII/UTF-8 value of first char in {expr}
charcol({expr} [, {winid}])	Number	character class of {string}
charidx({string}, {idx} [, {countcc} [, {utf16}]])	Number	column number of cursor or mark
chdir({dir})	Number	char index of byte {idx} in {string}
cindent({lnum})	String	change current working directory
clearmatches([win])	Number	C indent for line {lnum}
col({expr} [, {winid}])	none	clear all matches
complete({startcol}, {matches})	Number	column byte index of cursor or mark
complete_add({expr})	none	set Insert mode completion
complete_check()	Number	add completion match
complete_info([what])	Number	check for key typed during completion
confirm({msg} [, {choices} [, {default} [, {type}]]])	Dict	get current completion information
copy({expr})	Number	number of choice picked by user
cos({expr})	any	make a shallow copy of {expr}
cosh({expr})	Float	cosine of {expr}
count({comp}, {expr} [, {ic} [, {start}]])	Float	hyperbolic cosine of {expr}
cscope_connection([num], {dbpath} [, {prepend}])	Number	count how many {expr} are in {comp}
cursor({lnum}, {col} [, {off}])	Number	checks existence of cscope connection
cursor({list})	Number	move cursor to {lnum}, {col}, {off}
debugbreak({pid})	Number	move cursor to position in {list}
deepcopy({expr} [, {noref}])	Number	interrupt process being debugged
delete({fname} [, {flags}])	any	make a full copy of {expr}
deletebufline({buf}, {first} [, {last}])	Number	delete the file or directory {fname}
did_filetype()	Number	delete lines from buffer {buf}
diff({fromlist}, {tolist} [, {options}])	Number	TRUE if FileType autocmd event used
diff_filler({lnum})	List	diff two Lists of strings
diff_hlID({lnum}, {col})	Number	diff filler lines about {lnum}
digraph_get({chars})	Number	diff highlighting at {lnum}/{col}
digraph_getlist([listall])	String	get the digraph of {chars}
digraph_set({chars}, {digraph})	List	get all digraph s
digraph_setlist({digraphlist})	Boolean	register digraph
echoraw({expr})	Boolean	register multiple digraph s
empty({expr})	none	output {expr} as-is
environ()	Number	TRUE if {expr} is empty
err_teapot([{expr}])	Dict	return environment variables
escape({string}, {chars})	none	give E418, or E503 if {expr} is TRUE
	String	escape {chars} in {string} with '\'

eval({string})	any	evaluate {string} into its value
eventhandler()	Number	TRUE if inside an event handler
executable({expr})	Number	1 if executable {expr} exists
execute({command})	String	execute {command} and get the output
exepath({expr})	String	full path of the command {expr}
exists({expr})	Number	TRUE if {expr} exists
exists_compiled({expr})	Number	TRUE if {expr} exists at compile time
exp({expr})	Float	exponential of {expr}
expand({expr} [, {nosuf} [, {list}]])	any	expand special keywords in {expr}
expandcmd({string} [, {options}])	String	expand {string} like with `:edit`
extend({expr1}, {expr2} [, {expr3}])	List/Dict	insert items of {expr2} into {expr1}
extendnew({expr1}, {expr2} [, {expr3}])	List/Dict	like extend() but creates a new List or Dictionary
feedkeys({string} [, {mode}])	Number	add key sequence to typeahead buffer
filereadable({file})	Number	TRUE if {file} is a readable file
filewritable({file})	Number	TRUE if {file} is a writable file
filter({expr1}, {expr2})	List/Dict/Blob/String	remove items from {expr1} where {expr2} is 0
finddir({name} [, {path} [, {count}])	String	find directory {name} in {path}
findfile({name} [, {path} [, {count}])	String	find file {name} in {path}
flatten({list} [, {maxdepth}])	List	flatten {list} up to {maxdepth} levels
flattennew({list} [, {maxdepth}])	List	flatten a copy of {list}
float2nr({expr})	Number	convert Float {expr} to a Number
floor({expr})	Float	round {expr} down
fmod({expr1}, {expr2})	Float	remainder of {expr1} / {expr2}
fnameescape({fname})	String	escape special characters in {fname}
fnamemodify({fname}, {mods})	String	modify file name
foldclosed({lnum})	Number	first line of fold at {lnum} if closed
foldclosedend({lnum})	Number	last line of fold at {lnum} if closed
foldlevel({lnum})	Number	fold level at {lnum}
foldtext()	String	line displayed for closed fold
foldtextresult({lnum})	String	text for closed fold at {lnum}
foreach({expr1}, {expr2})	List/Dict/Blob/String	for each item in {expr1} call {expr2}
foreground()	Number	bring the Vim window to the foreground
fullcommand({name} [, {vim9}])	String	get full command from {name}
funcref({name} [, {arglist}] [, {dict}])	Funcref	reference to function {name}
function({name} [, {arglist}] [, {dict}])	Funcref	named reference to function {name}
garbagecollect([{:atexit}])	none	free memory, breaking cyclic references
get({list}, {idx} [, {def}])	any	get item {idx} from {list} or {def}
get({dict}, {key} [, {def}])	any	get item {key} from {dict} or {def}
get({func}, {what})	any	get property of funcref/partial {func}
getbufinfo([{:buf}])	List	information about buffers
getbuflines({buf}, {lnum} [, {end}])		

getbufoneline({buf}, {lnum})	List	lines {lnum} to {end} of buffer {buf}
getbufvar({buf}, {varname} [, {def}])	String	line {lnum} of buffer {buf}
getcellwidths()	any	variable {varname} in buffer {buf}
getchangelist([{buf}])	List	get character cell width overrides
getchar([expr])	List	list of change list items
getcharmod()	Number or String	get one character from the user
getcharpos({expr})	Number	modifiers for the last typed character
getcharsearch()	List	position of cursor, mark, etc.
getcharstr([expr])	Dict	last character search
getcmdcompltype()	String	get one character from the user
getcmdline()	String	return the type of the current command-line completion
getcmdpos()	String	return the current command-line
getcmdscreenpos()	Number	return cursor position in command-line
getcmdtype()	Number	return cursor screen position in command-line
getcmdwintype()	String	return current command-line type
getcompletion({pat}, {type} [, {filtered}])	String	return current command-line window type
getcurpos([{winnr}])	List	list of cmdline completion matches
getcursorcharpos([{winnr}])	List	list of cmdline completion matches
getcwd([{winnr} [, {tabnr}]])	List	position of the cursor
getenv({name})	List	character position of the cursor
getfontname([{name}])	String	get the current working directory
getfperm({fname})	String	return environment variable
getfsize({fname})	String	name of font being used
getftime({fname})	String	file permissions of file {fname}
getftype({fname})	Number	size in bytes of file {fname}
getimstatus()	Number	last modification time of file
getjumplist([{winnr} [, {tabnr}]])	String	description of type of file {fname}
getline({lnum})	Number	TRUE if the IME status is active
getline({lnum}, {end})	List	list of jump list items
getloclist({nr})	String	list of jump list items
getloclist({nr}, {what})	String	line {lnum} of current buffer
getmarklist([{buf}])	List	lines {lnum} to {end} of current buffer
getmatches([{win}])	List	list of location list items
getmousepos()	Dict	list of location list properties
getmouseshape()	Dict	get specific location list properties
getpid()	String	list of global/local marks
getpos({expr})	String	list of current matches
getqflist()	Dict	last known mouse position
getqflist({what})	String	current mouse shape name
getreg([{regname} [, 1 [, {list}]])	Number	process ID of Vim
getreginfo([{regname}])	Number	process ID of Vim
getregion({pos1}, {pos2} [, {opts}])	List	position of cursor, mark, etc.
getregtype([{regname}])	List	list of quickfix items
getscriptinfo([{opts}])	Dict	get specific quickfix list properties
gettabinfo([{expr}])	String or List	contents of a register
	Dict	information about a register
	List	get the text from {pos1} to {pos2}
	String	type of a register
	List	list of sourced scripts
	List	list of tab pages

gettabvar({nr}, {varname} [, {def}])	any	variable {varname} in tab {nr} or {def}
gettabwinvar({tabnr}, {winnr}, {name} [, {def}])	any	{name} in {winnr} in tab page {tabnr}
gettagstack([{nr}])	Dict	get the tag stack of window {nr}
gettext({text})	String	lookup translation of {text}
getwininfo([{winid}])	List	list of info about each window
getwinpos([{timeout}])	List	X and Y coord in pixels of Vim window
getwinposx()	Number	X coord in pixels of the Vim window
getwinposy()	Number	Y coord in pixels of the Vim window
getwinvar({nr}, {varname} [, {def}])	any	variable {varname} in window {nr}
glob({expr} [, {nosuf} [, {list} [, {alllinks}]]])	any	expand file wildcards in {expr}
glob2regpat({expr})	String	convert a glob pat into a search pat
globpath({path}, {expr} [, {nosuf} [, {list} [, {alllinks}]]])	String	do glob({expr}) for all dirs in {path}
has({feature} [, {check}])	Number	TRUE if feature {feature} supported
has_key({dict}, {key})	Number	TRUE if {dict} has entry {key}
haslocaldir([{winnr} [, {tabnr}]])	Number	TRUE if the window executed :lcd or :tcd
hasmapto({what} [, {mode} [, {abbr}]])	Number	TRUE if mapping to {what} exists
histadd({history}, {item})	Number	add an item to a history
histdel({history} [, {item}])	Number	remove an item from a history
histget({history} [, {index}])	String	get the item {index} from a history
histnr({history})	Number	highest index of a history
hlID({name})	Number	syntax ID of highlight group {name}
hlexists({name})	Number	TRUE if highlight group {name} exists
hlget([{name} [, {resolve}]])	List	get highlight group attributes
hlset({list})	Number	set highlight group attributes
hostname()	String	name of the machine Vim is running on
iconv({expr}, {from}, {to})	String	convert encoding of {expr}
indent({lnum})	Number	indent of line {lnum}
index({object}, {expr} [, {start} [, {ic}]])	Number	index in {object} where {expr} appears
indexof({object}, {expr} [, {opts}])	Number	index in {object} where {expr} is true
input({prompt} [, {text} [, {completion}]])	String	get input from the user
inputdialog({prompt} [, {text} [, {cancelreturn}]])	String	like input() but in a GUI dialog
inputlist({textlist})	Number	let the user pick from a choice list
inputrestore()	Number	restore typeahead
inputsave()	Number	save and clear typeahead
inputsecret({prompt} [, {text}])	String	like input() but hiding the text
insert({object}, {item} [, {idx}])	List	insert {item} in {object} [before {idx}]
instanceof({object}, {class})	Number	TRUE if {object} is an instance of {class}
interrupt()	none	interrupt script execution
invert({expr})	Number	bitwise invert
isabsolutepath({path})	Number	TRUE if {path} is an absolute path
isdirectory({directory})	Number	TRUE if {directory} is a directory
isinf({expr})	Number	determine if {expr} is infinity value

		(positive or negative)
islocked({expr})	Number	TRUE if {expr} is locked
isnan({expr})	Number	TRUE if {expr} is NaN
items({dict})	List	key-value pairs in {dict}
job_getchannel({job})	Channel	get the channel handle for {job}
job_info([{job}])	Dict	get information about {job}
job_setopts({job}, {options})	none	set options for {job}
job_start({command} [, {options}])	Job	start a job
job_status({job})	String	get the status of {job}
job_stop({job} [, {how}])	Number	stop {job}
join({list} [, {sep}])	String	join {list} items into one String
js_decode({string})	any	decode JS style JSON
js_encode({expr})	String	encode JS style JSON
json_decode({string})	any	decode JSON
json_encode({expr})	String	encode JSON
keys({dict})	List	keys in {dict}
keytrans({string})	String	translate internal keycodes to a form that can be used by :map
len({expr})	Number	the length of {expr}
libcall({lib}, {func}, {arg})	String	call {func} in library {lib} with {arg}
libcallnr({lib}, {func}, {arg})	Number	idem, but return a Number
line({expr} [, {winid}])	Number	line nr of cursor, last line or mark
line2byte({lnum})	Number	byte count of line {lnum}
lispindent({lnum})	Number	Lisp indent for line {lnum}
list2blob({list})	Blob	turn {list} of numbers into a Blob
list2str({list} [, {utf8}])	String	turn {list} of numbers into a String
listener_add({callback} [, {buf}])	Number	add a callback to listen to changes
listener_flush([{buf}])	none	invoke listener callbacks
listener_remove({id})	none	remove a listener callback
localtime()	Number	current time
log({expr})	Float	natural logarithm (base e) of {expr}
log10({expr})	Float	logarithm of Float {expr} to base 10
luaeval({expr} [, {expr}])	any	evaluate Lua expression
map({expr1}, {expr2})	List/Dict/Blob/String	change each item in {expr1} to {expr2}
maparg({name} [, {mode} [, {abbr} [, {dict}]]])	String or Dict	rhs of mapping {name} in mode {mode}
mapcheck({name} [, {mode} [, {abbr}])	String	check for mappings matching {name}
maplist([{abbr}])	List	list of all mappings, a dict for each
mapnew({expr1}, {expr2})	List/Dict/Blob/String	like map() but creates a new List or Dictionary
mapset({mode}, {abbr}, {dict})	none	restore mapping from maparg() result
match({expr}, {pat} [, {start} [, {count}])	Number	position where {pat} matches in {expr}
matchadd({group}, {pattern} [, {priority} [, {id} [, {dict}]]])	Number	highlight {pattern} with {group}
matchaddpos({group}, {pos} [, {priority} [, {id} [, {dict}]]])	Number	highlight positions with {group}
matcharg({nr})	List	arguments of :match

matchbuflines({buf}, {pat}, {lnum}, {end}, [, {dict}])	List	all the {pat} matches in buffer {buf}
matchdelete({id} [, {win}])	Number	delete match identified by {id}
matchend({expr}, {pat} [, {start} [, {count}]])	Number	position where {pat} ends in {expr}
matchfuzzy({list}, {str} [, {dict}])	List	fuzzy match {str} in {list}
matchfuzzypos({list}, {str} [, {dict}])	List	fuzzy match {str} in {list}
matchlist({expr}, {pat} [, {start} [, {count}]])	List	match and submatches of {pat} in {expr}
matchstr({expr}, {pat} [, {start} [, {count}]])	String	{count}'th match of {pat} in {expr}
matchstrlist({list}, {pat} [, {dict}])	List	all the {pat} matches in {list}
matchstrpos({expr}, {pat} [, {start} [, {count}]])	List	{count}'th match of {pat} in {expr}
max({expr})	Number	maximum value of items in {expr}
menu_info({name} [, {mode}])	Dict	get menu item information
min({expr})	Number	minimum value of items in {expr}
mkdir({name} [, {flags} [, {prot}]])	Number	create directory {name}
mode([expr])	String	current editing mode
mzeval({expr})	any	evaluate MzScheme expression
nextnonblank({lnum})	Number	line nr of non-blank line >= {lnum}
nr2char({expr} [, {utf8}])	String	single char with ASCII/UTF-8 value {expr}
or({expr}, {expr})	Number	bitwise OR
pathshorten({expr} [, {len}])	String	shorten directory names in a path
perlevel({expr})	any	evaluate Perl expression
popup_atcursor({what}, {options})	Number	create popup window near the cursor
popup_beval({what}, {options})	Number	create popup window for 'ballooneval'
popup_clear()	none	close all popup windows
popup_close({id} [, {result}])	none	close popup window {id}
popup_create({what}, {options})	Number	create a popup window
popup_dialog({what}, {options})	Number	create a popup window used as a dialog
popup_filter_menu({id}, {key})	Number	filter for a menu popup window
popup_filter_yesno({id}, {key})	Number	filter for a dialog popup window
popup_findecho()	Number	get window ID of popup for `:echowin`
popup_findinfo()	Number	get window ID of info popup window
popup_findpreview()	Number	get window ID of preview popup window
popup_getoptions({id})	Dict	get options of popup window {id}
popup_getpos({id})	Dict	get position of popup window {id}
popup_hide({id})	none	hide popup menu {id}
popup_list()	List	get a list of window IDs of all popups
popup_locate({row}, {col})	Number	get window ID of popup at position
popup_menu({what}, {options})	Number	create a popup window used as a menu
popup_move({id}, {options})	none	set position of popup window {id}
popup_notification({what}, {options})	Number	create a notification popup window
popup_setoptions({id}, {options})	none	set options for popup window {id}
popup_settext({id}, {text})	none	set the text of popup window {id}
popup_show({id})	none	unhide popup window {id}
pow({x}, {y})	Float	{x} to the power of {y}

prevnonblank({lnum})	Number	line nr of non-blank line <= {lnum}
printf({fmt}, {expr1}...)	String	format text
prompt_getprompt({buf})	String	get prompt text
prompt_setcallback({buf}, {expr})	none	set prompt callback function
prompt_setinterrupt({buf}, {text})	none	set prompt interrupt function
prompt_setprompt({buf}, {text})	none	set prompt text
prop_add({lnum}, {col}, {props})	none	add one text property
prop_add_list({props}, [[{lnum}, {col}, {end-lnum}, {end-col}], ...])	none	add multiple text properties
prop_clear({lnum} [, {lnum-end} [, {props}]])	none	remove all text properties
prop_find({props} [, {direction}])	Dict	search for a text property
prop_list({lnum} [, {props}])	List	text properties in {lnum}
prop_remove({props} [, {lnum} [, {lnum-end}]])	Number	remove a text property
prop_type_add({name}, {props})	none	define a new property type
prop_type_change({name}, {props})	none	change an existing property type
prop_type_delete({name} [, {props}])	none	delete a property type
prop_type_get({name} [, {props}])	Dict	get property type values
prop_type_list([{props}])	List	get list of property types
pum_getpos()	Dict	position and size of pum if visible
pumvisible()	Number	whether popup menu is visible
py3eval({expr})	any	evaluate python3 expression
pyeval({expr})	any	evaluate Python expression
pyxeval({expr})	any	evaluate python_x expression
rand([{expr}])	Number	get pseudo-random number
range({expr} [, {max} [, {stride}])	List	items from {expr} to {max}
readblob({fname} [, {offset} [, {size}])	Blob	read a Blob from {fname}
readdir({dir} [, {expr} [, {dict}])	List	file names in {dir} selected by {expr}
readdirrex({dir} [, {expr} [, {dict}])	List	file info in {dir} selected by {expr}
readfile({fname} [, {type} [, {max}])	List	get list of lines from file {fname}
reduce({object}, {func} [, {initial}])	any	reduce {object} using {func}
reg_executing()	String	get the executing register name
reg_recording()	String	get the recording register name
reltime([{start} [, {end}])	List	get time value
reltimefloat({time})	Float	turn the time value into a Float
reltimestr({time})	String	turn time value into a String
remote_expr({server}, {string} [, {idvar} [, {timeout}])	String	send expression
remote_foreground({server})	Number	bring Vim server to the foreground
remote_peek({serverid} [, {retvar}])	Number	check for reply string
remote_read({serverid} [, {timeout}])	String	read reply string

```

remote_send({server}, {string} [, {idvar}])
    String send key sequence
remote_startserver({name}) none become server {name}
remove({list}, {idx} [, {end}]) any/List
    remove items {idx}-{end} from {list}
remove({blob}, {idx} [, {end}]) Number/Blob
    remove bytes {idx}-{end} from {blob}
remove({dict}, {key}) any remove entry {key} from {dict}
rename({from}, {to}) Number rename (move) file from {from} to {to}
repeat({expr}, {count}) List/Blob/String
    repeat {expr} {count} times
resolve({filename}) String get filename a shortcut points to
reverse({obj}) List/Blob/String
    reverse {obj}
round({expr}) Float round off {expr}
rubyeval({expr}) any evaluate Ruby expression
screenattr({row}, {col}) Number attribute at screen position
screenchar({row}, {col}) Number character at screen position
screenchars({row}, {col}) List List of characters at screen position
screencol() Number current cursor column
screenpos({winid}, {lnum}, {col}) Dict screen row and col of a text character
screenrow() Number current cursor row
screenstring({row}, {col}) String characters at screen position
search({pattern} [, {flags} [, {stopline} [, {timeout} [, {skip}]]]])
    Number search for {pattern}
searchcount([options]) Dict get or update search stats
searchdecl({name} [, {global} [, {thisblock}]]])
    Number search for variable declaration
searchpair({start}, {middle}, {end} [, {flags} [, {skip} [...]]])
    Number search for other end of start/end pair
searchpairpos({start}, {middle}, {end} [, {flags} [, {skip} [...]]])
    List search for other end of start/end pair
searchpos({pattern} [, {flags} [, {stopline} [, {timeout} [, {skip}]]]])
    List search for {pattern}
server2client({clientid}, {string})
    Number send reply string
serverlist() String get a list of available servers
setbufline({expr}, {lnum}, {text})
    Number set line {lnum} to {text} in buffer {expr}
setbufvar({buf}, {varname}, {val})
    none set {varname} in buffer {buf} to {val}
setcellwidths({list}) none set character cell width overrides
setcharpos({expr}, {list}) Number set the {expr} position to {list}
setcharsearch({dict}) Dict set character search from {dict}
setcmdline({str} [, {pos}]) Number set command-line
setcmdpos({pos}) Number set cursor position in command-line
setcursorcharpos({list}) Number move cursor to position in {list}
setenv({name}, {val}) none set environment variable
setfperm({fname}, {mode}) Number set {fname} file permissions to {mode}
setline({lnum}, {line}) Number set line {lnum} to {line}
setloclist({nr}, {list} [, {action}])
    Number modify location list using {list}
setloclist({nr}, {list}, {action}, {what})

```


setmatches({list} [, {win}])	Number	modify specific location list props
setpos({expr}, {list})	Number	restore a list of matches
setqflist({list} [, {action}])	Number	set the {expr} position to {list}
setqflist({list}, {action}, {what})	Number	modify quickfix list using {list}
setreg({n}, {v} [, {opt}])	Number	modify specific quickfix list props
settabvar({nr}, {varname}, {val})	none	set register to value and type
settabwinvar({tabnr}, {winnr}, {varname}, {val})	none	set {varname} in tab page {nr} to {val}
	none	set {varname} in window {winnr} in tab page {tabnr} to {val}
settagstack({nr}, {dict} [, {action}])	Number	modify tag stack using {dict}
setwinvar({nr}, {varname}, {val})	none	set {varname} in window {nr} to {val}
sha256({string})	String	SHA256 checksum of {string}
shellescape({string} [, {special}])	String	escape {string} for use as shell command argument
shiftwidth([{col}])	Number	effective value of 'shiftwidth'
sign_define({name} [, {dict}])	Number	define or update a sign
sign_define({list})	List	define or update a list of signs
sign_getdefined([{name}])	List	get a list of defined signs
sign_getplaced([{buf} [, {dict}]])	List	get a list of placed signs
sign_jump({id}, {group}, {buf})	Number	jump to a sign
sign_place({id}, {group}, {name}, {buf} [, {dict}])	Number	place a sign
sign_placelist({list})	List	place a list of signs
sign_undefine([{name}])	Number	undefine a sign
sign_undefine({list})	List	undefine a list of signs
sign_unplace({group} [, {dict}])	Number	unplace a sign
sign_unplacelist({list})	List	unplace a list of signs
simplify({filename})	String	simplify filename as much as possible
sin({expr})	Float	sine of {expr}
sinh({expr})	Float	hyperbolic sine of {expr}
slice({expr}, {start} [, {end}])	String, List or Blob	slice of a String, List or Blob
sort({list} [, {how} [, {dict}]])	List	sort {list}, compare with {how}
sound_clear()	none	stop playing all sounds
sound_playevent({name} [, {callback}])	Number	play an event sound
sound_playfile({path} [, {callback}])	Number	play sound file {path}
sound_stop({id})	none	stop playing sound {id}
soundfold({word})	String	sound-fold {word}
spellbadword()	String	badly spelled word at cursor
spellsuggest({word} [, {max} [, {capital}]])	List	spelling suggestions
split({expr} [, {pat} [, {keepempty}]])	List	make List from {pat} separated {expr}
sqrt({expr})	Float	square root of {expr}

strand([<i>expr</i>])	List	get seed for <code>rand()</code>
state([<i>what</i>])	String	current state of Vim
str2float(<i>expr</i> [, <i>quoted</i>])	Float	convert String to Float
str2list(<i>expr</i> [, <i>utf8</i>])	List	convert each character of <i>expr</i> to ASCII/UTF-8 value
str2nr(<i>expr</i> [, <i>base</i> [, <i>quoted</i>]])	Number	convert String to Number
strcharlen(<i>expr</i>)	Number	character length of the String <i>expr</i>
strcharpart(<i>str</i> , <i>start</i> [, <i>len</i> [, <i>skipcc</i>]])	String	<i>len</i> characters of <i>str</i> at character <i>start</i>
strchars(<i>expr</i> [, <i>skipcc</i>])	Number	character count of the String <i>expr</i>
strdisplaywidth(<i>expr</i> [, <i>col</i>])	Number	display length of the String <i>expr</i>
strftime(<i>format</i> [, <i>time</i>])	String	format time with a specified format
strgetchar(<i>str</i> , <i>index</i>)	Number	get char <i>index</i> from <i>str</i>
stridx(<i>haystack</i> , <i>needle</i> [, <i>start</i>])	Number	index of <i>needle</i> in <i>haystack</i>
string(<i>expr</i>)	String	String representation of <i>expr</i> value
strlen(<i>expr</i>)	Number	length of the String <i>expr</i>
strpart(<i>str</i> , <i>start</i> [, <i>len</i> [, <i>chars</i>]])	String	<i>len</i> bytes/chars of <i>str</i> at byte <i>start</i>
strptime(<i>format</i> , <i>timestring</i>)	Number	Convert <i>timestring</i> to unix timestamp
strridx(<i>haystack</i> , <i>needle</i> [, <i>start</i>])	Number	last index of <i>needle</i> in <i>haystack</i>
strtrans(<i>expr</i>)	String	translate string to make it printable
strutf16len(<i>string</i> [, <i>countcc</i>])	Number	number of UTF-16 code units in <i>string</i>
strwidth(<i>expr</i>)	Number	display cell length of the String <i>expr</i>
submatch(<i>nr</i> [, <i>list</i>])	String or List	specific match in ":s" or substitute()
substitute(<i>expr</i> , <i>pat</i> , <i>sub</i> , <i>flags</i>)	String	all <i>pat</i> in <i>expr</i> replaced with <i>sub</i>
swapfilelist()	List	swap files found in 'directory'
swapinfo(<i>fname</i>)	Dict	information about swap file <i>fname</i>
swapname(<i>buf</i>)	String	swap file of buffer <i>buf</i>
synID(<i>lnum</i> , <i>col</i> , <i>trans</i>)	Number	syntax ID at <i>lnum</i> and <i>col</i>
synIDattr(<i>synID</i> , <i>what</i> [, <i>mode</i>])	String	attribute <i>what</i> of syntax ID <i>synID</i>
synIDtrans(<i>synID</i>)	Number	translated syntax ID of <i>synID</i>
synconcealed(<i>lnum</i> , <i>col</i>)	List	info about concealing
synstack(<i>lnum</i> , <i>col</i>)	List	stack of syntax IDs at <i>lnum</i> and <i>col</i>
system(<i>expr</i> [, <i>input</i>])	String	output of shell command/filter <i>expr</i>
systemlist(<i>expr</i> [, <i>input</i>])	List	output of shell command/filter <i>expr</i>
tabpagebuflist([<i>arg</i>])	List	list of buffer numbers in tab page
tabpagenr([<i>arg</i>])	Number	number of current or last tab page
tabpagewinnr(<i>tabarg</i> [, <i>arg</i>])	Number	number of current window in tab page
tagfiles()	List	tags files used
taglist(<i>expr</i> [, <i>filename</i>])	List	list of tags matching <i>expr</i>
tan(<i>expr</i>)	Float	tangent of <i>expr</i>
tanh(<i>expr</i>)	Float	hyperbolic tangent of <i>expr</i>
tempname()	String	name for a temporary file
term_dumpdiff(<i>filename</i> , <i>filename</i> [, <i>options</i>])		

term_dumpload({filename} [, {options}])	Number	display difference between two dumps
term_dumpwrite({buf}, {filename} [, {options}])	Number	displaying a screen dump
term_getaltscreen({buf})	none	dump terminal window contents
term_getansicolors({buf})	Number	get the alternate screen flag
term_getattr({attr}, {what})	List	get ANSI palette in GUI color mode
term_getcursor({buf})	Number	get the value of attribute {what}
term_getjob({buf})	List	get the cursor position of a terminal
term_getline({buf}, {row})	Job	get the job associated with a terminal
term_getscrolled({buf})	String	get a line of text from a terminal
term_getsize({buf})	Number	get the scroll count of a terminal
term_getstatus({buf})	List	get the size of a terminal
term_gettitle({buf})	String	get the status of a terminal
term_gettty({buf}, [{input}])	String	get the title of a terminal
term_list()	String	get the tty name of a terminal
term_scrape({buf}, {row})	List	get the list of terminal buffers
term_sendkeys({buf}, {keys})	List	get row of a terminal screen
term_setansicolors({buf}, {colors})	none	send keystrokes to a terminal
term_setapi({buf}, {expr})	none	set ANSI palette in GUI color mode
term_setkill({buf}, {how})	none	set <code>terminal-api</code> function name prefix
term_setrestore({buf}, {command})	none	set signal to stop job in terminal
term_setsize({buf}, {rows}, {cols})	none	set command to restore terminal
term_start({cmd} [, {options}])	none	set the size of a terminal
term_wait({buf} [, {time}])	Number	open a terminal window and run a job
terminalprops()	Number	wait for screen to be updated
test_alloc_fail({id}, {countdown}, {repeat})	Dict	properties of the terminal
test_autochdir()	none	make memory allocation fail
test_feedinput({string})	none	enable <code>'autochdir'</code> during startup
test_garbagecollect_now()	none	add key sequence to input buffer
test_garbagecollect_soon()	none	free memory right now for testing
test_getvalue({string})	none	free memory soon for testing
test_gui_event({event}, {args})	any	get value of an internal variable
test_ignore_error({expr})	bool	generate a GUI event for testing
test_mswin_event({event}, {args})	none	ignore a specific error
test_null_blob()	bool	generate MS-Windows event for testing
test_null_channel()	Blob	null value for testing
test_null_dict()	Channel	null value for testing
test_null_function()	Dict	null value for testing
test_null_job()	Funcref	null value for testing
test_null_list()	Job	null value for testing
test_null_partial()	List	null value for testing
test_null_string()	Funcref	null value for testing
test_option_not_set({name})	String	null value for testing
test_override({expr}, {val})	none	reset flag indicating option was set
test_refcount({expr})	none	test with Vim internal overrides
test_setmouse({row}, {col})	Number	get the reference count of {expr}
test_settime({expr})	none	set the mouse position for testing
test_srand_seed([seed])	none	set current time for testing
	none	set seed for testing srand()

test_unknown()	any	unknown value for testing
test_void()	any	void value for testing
timer_info([{id}])	List	information about timers
timer_pause({id} , {pause})	none	pause or unpause a timer
timer_start({time} , {callback} [, {options}])	Number	create a timer
timer_stop({timer})	none	stop a timer
timer_stopall()	none	stop all timers
tolower({expr})	String	the String {expr} switched to lowercase
toupper({expr})	String	the String {expr} switched to uppercase
tr({src} , {fromstr} , {tostr})	String	translate chars of {src} in {fromstr} to chars in {tostr}
trim({text} [, {mask} [, {dir}]])	String	trim characters in {mask} from {text}
trunc({expr})	Float	truncate Float {expr}
type({expr})	Number	type of value {expr}
typename({expr})	String	representation of the type of {expr}
undofile({name})	String	undo file name for {name}
undotree([{buf}])	List	undo file tree for buffer {buf}
uniq({list} [, {func} [, {dict}]])	List	remove adjacent duplicates from a list
utf16idx({string} , {idx} [, {countcc} [, {charidx}]])	Number	UTF-16 index of byte {idx} in {string}
values({dict})	List	values in {dict}
virtcol({expr} [, {list} [, {winid}])	Number or List	screen column of cursor or mark
virtcol2col({winid} , {lnum} , {col})	Number	byte index of a character on screen
visualmode([{expr}])	String	last visual mode used
wildmenu()	Number	whether 'wildmenu' mode is active
win_execute({id} , {command} [, {silent}])	String	execute {command} in window {id}
win_findbuf({bufnr})	List	find windows containing {bufnr}
win_getid([{win} [, {tab}]])	Number	get window ID for {win} in {tab}
win_gettype([{nr}])	String	type of window {nr}
win_gotoid({expr})	Number	go to window with ID {expr}
win_id2tabwin({expr})	List	get tab and window nr from window ID
win_id2win({expr})	Number	get window nr from window ID
win_move_separator({nr})	Number	move window vertical separator
win_move_statusline({nr})	Number	move window status line
win_screenpos({nr})	List	get screen position of window {nr}
win_splitmove({nr} , {target} [, {options}])	Number	move window {nr} to split of {target}
winbufnr({nr})	Number	buffer number of window {nr}
wincol()	Number	window column of the cursor
windowsversion()	String	MS-Windows OS version
winheight({nr})	Number	height of window {nr}
winlayout([{tabnr}])	List	layout of windows in tab {tabnr}
winline()	Number	window line of the cursor
winnr([{expr}])	Number	number of current window
winrestcmd()	String	returns command to restore window sizes
winrestview({dict})	none	restore view of current window
winsaveview()	Dict	save view of current window

winwidth({nr})	Number	width of window {nr}
wordcount()	Dict	get byte/char/word statistics
writefile({object}, {fname} [, {flags}])	Number	write Blob or List of lines to file
xor({expr}, {expr})	Number	bitwise XOR

2. Details

builtin-function-details

Not all functions are here, some have been moved to a help file covering the specific functionality.

abs({expr})

abs()

Return the absolute value of {expr}. When {expr} evaluates to a **Float** abs() returns a **Float**. When {expr} can be converted to a **Number** abs() returns a **Number**. Otherwise abs() gives an error message and returns -1.

Examples:

```
echo abs(1.456)
1.456
echo abs(-5.456)
5.456
echo abs(-4)
4
```

Can also be used as a **method** :

```
Compute()->abs()
```

acos({expr})

acos()

Return the arc cosine of {expr} measured in radians, as a **Float** in the range of [0, pi].

{expr} must evaluate to a **Float** or a **Number** in the range [-1, 1]. Otherwise acos() returns "nan".

Examples:

```
:echo acos(0)
1.570796
:echo acos(-0.5)
2.094395
```

Can also be used as a **method** :

```
Compute()->acos()
```

add({object}, {expr})

add()

Append the item {expr} to **List** or **Blob** {object}. Returns the resulting **List** or **Blob**. Examples:

```
:let alist = add([1, 2, 3], item)
:call add(mylist, "woodstock")
```

Note that when {expr} is a **List** it is appended as a single item. Use **extend()** to concatenate **Lists**.

When {object} is a **Blob** then {expr} must be a number.

Use **insert()** to add an item at another position.

Returns 1 if {object} is not a **List** or a **Blob**.

Can also be used as a `method` :
`mylist->add(val1)->add(val2)`

`and({expr}, {expr})` `and()`
Bitwise AND on the two arguments. The arguments are converted to a number. A List, Dict or Float argument causes an error. Also see ``or()`` and ``xor()``.
Example:

```
:let flag = and(bits, 0x80)
```

Can also be used as a `method` :
`:let flag = bits->and(0x80)`

`append({lnum}, {text})` `append()`
When `{text}` is a `List` : Append each item of the `List` as a text line below line `{lnum}` in the current buffer. Otherwise append `{text}` as one text line below line `{lnum}` in the current buffer.
Any type of item is accepted and converted to a String.
`{lnum}` can be zero to insert a line before the first one.
`{lnum}` is used like with `getline()` .
Returns 1 for failure (`{lnum}` out of range or out of memory), 0 for success. When `{text}` is an empty list zero is returned, no matter the value of `{lnum}`.
In `Vim9` script an invalid argument or negative number results in an error. Example:

```
:let failed = append(line('$'), "# THE END")  
:let failed = append(0, ["Chapter 1", "the beginning"])
```

Can also be used as a `method` after a List, the base is passed as the second argument:
`mylist->append(lnum)`

`appendbufline({buf}, {lnum}, {text})` `appendbufline()`
Like `append()` but append the text in buffer `{buf}`.

This function works only for loaded buffers. First call `bufload()` if needed.

For the use of `{buf}`, see `bufname()` .

`{lnum}` is the line number to append below. **Note** that using `line()` would use the current buffer, not the one appending to. Use "\$" to append at the end of the buffer. Other string values are not supported.

On success 0 is returned, on failure 1 is returned.
In `Vim9` script an error is given for an invalid `{lnum}`.

If `{buf}` is not a valid buffer or `{lnum}` is not valid, an error message is given. Example:

```
        :let failed = appendbufline(13, 0, "# THE START")
```

However, when `{text}` is an empty list then no error is given for an invalid `{lnum}`, since `{lnum}` isn't actually used.

Can also be used as a `method` after a List, the base is passed as the second argument:

```
mylist->appendbufline(buf, lnum)
```

`argc([winid])`

`argc()`

The result is the number of files in the argument list. See `arglist`.

If `{winid}` is not supplied, the argument list of the current window is used.

If `{winid}` is -1, the global argument list is used.

Otherwise `{winid}` specifies the window of which the argument list is used: either the window number or the window ID.

Returns -1 if the `{winid}` argument is invalid.

`argidx()`

`argidx()`

The result is the current index in the argument list. 0 is the first file. `argc() - 1` is the last one. See `arglist`.

`arglistid([winnr] [, {tabnr}])`

`arglistid()`

Return the argument list ID. This is a number which identifies the argument list being used. Zero is used for the global argument list. See `arglist`.

Returns -1 if the arguments are invalid.

Without arguments use the current window.

With `{winnr}` only use this window in the current tab page.

With `{winnr}` and `{tabnr}` use the window in the specified tab page.

`{winnr}` can be the window number or the `window-ID`.

`argv([nr] [, {winid}])`

`argv()`

The result is the `{nr}`th file in the argument list. See `arglist`. "argv(0)" is the first one. Example:

```
:let i = 0
:while i < argc()
: let f = escape(fnameescape(argv(i)), '.')
: exe 'amenu Arg.' .. f .. ' :e ' .. f .. '<CR>'
: let i = i + 1
:endwhile
```

Without the `{nr}` argument, or when `{nr}` is -1, a List with the whole `arglist` is returned.

The `{winid}` argument specifies the window ID, see `argc()`.

For the Vim command line arguments see `v:argv`.

Returns an empty string if `{nr}`th argument is not present in the argument list. Returns an empty List if the `{winid}`

argument is invalid.

`asin({expr})`

`asin()`

Return the arc sine of `{expr}` measured in radians, as a `Float` in the range of $[-\pi/2, \pi/2]$.

`{expr}` must evaluate to a `Float` or a `Number` in the range $[-1, 1]$.

Returns "nan" if `{expr}` is outside the range $[-1, 1]$. Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo asin(0.8)
0.927295
:echo asin(-0.5)
-0.523599
```

Can also be used as a `method` :

```
Compute()->asin()
```

`assert_` functions are documented here: [assert-functions-details](#)

`atan({expr})`

`atan()`

Return the principal value of the arc tangent of `{expr}`, in the range $[-\pi/2, +\pi/2]$ radians, as a `Float`.

`{expr}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo atan(100)
1.560797
:echo atan(-4.01)
-1.326405
```

Can also be used as a `method` :

```
Compute()->atan()
```

`atan2({expr1}, {expr2})`

`atan2()`

Return the arc tangent of `{expr1} / {expr2}`, measured in radians, as a `Float` in the range $[-\pi, \pi]$.

`{expr1}` and `{expr2}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr1}` or `{expr2}` is not a `Float` or a `Number`.

Examples:

```
:echo atan2(-1, 1)
-0.785398
:echo atan2(1, -1)
2.356194
```

Can also be used as a `method` :

```
Compute()->atan2(1)
```

`autocmd_add({acmds})` `autocmd_add()`

Adds a List of autocmds and autocmd groups.

The `{acmds}` argument is a List where each item is a Dict with the following optional items:

<code>bufnr</code>	buffer number to add a buffer-local autocmd. If this item is specified, then the "pattern" item is ignored.
<code>cmd</code> <code>event</code>	Ex command to execute for this autocmd event autocmd event name. Refer to autocmd-events . This can be either a String with a single event name or a List of event names.
<code>group</code>	autocmd group name. Refer to autocmd-groups . If this group doesn't exist then it is created. If not specified or empty, then the default group is used.
<code>nested</code>	boolean flag, set to <code>v:true</code> to add a nested autocmd. Refer to autocmd-nested .
<code>once</code>	boolean flag, set to <code>v:true</code> to add an autocmd which executes only once. Refer to autocmd-once .
<code>pattern</code>	autocmd pattern string. Refer to autocmd-patterns . If "bufnr" item is present, then this item is ignored. This can be a String with a single pattern or a List of patterns.
<code>replace</code>	boolean flag, set to <code>v:true</code> to remove all the commands associated with the specified autocmd event and group and add the <code>{cmd}</code> . This is useful to avoid adding the same command multiple times for an autocmd event in a group.

Returns `v:true` on success and `v:false` on failure.

Examples:

```
" Create a buffer-local autocmd for buffer 5
let acmd = {}
let acmd.group = 'MyGroup'
let acmd.event = 'BufEnter'
let acmd.bufnr = 5
let acmd.cmd = 'call BufEnterFunc()'
call autocmd_add([acmd])
```

Can also be used as a [method](#) :

```
GetAutocmdList()->autocmd_add()
```

`autocmd_delete({acmds})` `autocmd_delete()`

Deletes a List of autocmds and autocmd groups.

The `{acmds}` argument is a List where each item is a Dict with the following optional items:

<code>bufnr</code>	buffer number to delete a buffer-local autocmd. If this item is specified, then the "pattern" item is ignored.
<code>cmd</code>	Ex command for this autocmd event

event	autocmd event name. Refer to autocmd-events . If '*' then all the autocmd events in this group are deleted.
group	autocmd group name. Refer to autocmd-groups . If not specified or empty, then the default group is used.
nested	set to v:true for a nested autocmd. Refer to autocmd-nested .
once	set to v:true for an autocmd which executes only once. Refer to autocmd-once .
pattern	autocmd pattern string. Refer to autocmd-patterns . If "bufnr" item is present, then this item is ignored.

If only `{group}` is specified in a `{acmds}` entry and `{event}`, `{pattern}` and `{cmd}` are not specified, then that autocmd group is deleted.

Returns `v:true` on success and `v:false` on failure.

Examples:

```

" :autocmd! BufLeave *.vim
let acmd = #{event: 'BufLeave', pattern: '*.vim'}
call autocmd_delete([acmd])
" :autocmd! MyGroup1 BufLeave
let acmd = #{group: 'MyGroup1', event: 'BufLeave'}
call autocmd_delete([acmd])
" :autocmd! MyGroup2 BufEnter *.c
let acmd = #{group: 'MyGroup2', event: 'BufEnter',
            \ pattern: '*.c'}

" :autocmd! MyGroup2 * *.c
let acmd = #{group: 'MyGroup2', event: '*',
            \ pattern: '*.c'}

call autocmd_delete([acmd])
" :autocmd! MyGroup3
let acmd = #{group: 'MyGroup3'}
call autocmd_delete([acmd])

```

Can also be used as a [method](#) :

```
GetAutocmdList()->autocmd_delete()
```

`autocmd_get([opts])`

[autocmd_get\(\)](#)

Returns a [List](#) of autocmds. If `{opts}` is not supplied, then returns the autocmds for all the events in all the groups.

The optional `{opts}` Dict argument supports the following items:

group	Autocmd group name. If specified, returns only the autocmds defined in this group. If the specified group doesn't exist, results in an error message. If set to an empty string, then the default autocmd group is used.
event	Autocmd event name. If specified, returns only the autocmds defined for this event. If set to "*", then returns autocmds for all the

events. If the specified event doesn't exist, results in an error message.

`pattern` Autocmd pattern. If specified, returns only the autocmds defined for this pattern.

A combination of the above three times can be supplied in `{opts}`.

Each Dict in the returned List contains the following items:

`bufnr` For buffer-local autocmds, buffer number where the autocmd is defined.

`cmd` Command executed for this autocmd.

`event` Autocmd event name.

`group` Autocmd group name.

`nested` Boolean flag, set to `v:true` for a nested autocmd. See [autocmd-nested](#).

`once` Boolean flag, set to `v:true`, if the autocmd will be executed only once. See [autocmd-once](#).

`pattern` Autocmd pattern. For a buffer-local autocmd, this will be of the form "`<buffer=n>`".

If there are multiple commands for an autocmd event in a group, then separate items are returned for each command.

Returns an empty List if an autocmd with the specified group or event or pattern is not found.

Examples:

```
" :autocmd MyGroup
echo autocmd_get({group: 'Mygroup'})
" :autocmd G BufUnload
echo autocmd_get({group: 'G', event: 'BufUnload'})
" :autocmd G * *.ts
let acmd = {group: 'G', event: '*', pattern: '*.ts'}
echo autocmd_get(acmd)
" :autocmd Syntax
echo autocmd_get({event: 'Syntax'})
" :autocmd G BufEnter *.ts
let acmd = {group: 'G', event: 'BufEnter',
           \ pattern: '*.ts'}
echo autocmd_get(acmd)
```

Can also be used as a `method` :
`Getopts()->autocmd_get()`

`balloon_gettext()` `balloon_gettext()`
Return the current text in the balloon. Only for the string, not used for the List. Returns an empty string if balloon is not present.

`balloon_show({expr})` `balloon_show()`
Show `{expr}` inside the balloon. For the GUI `{expr}` is used as a string. For a terminal `{expr}` can be a list, which contains the lines of the balloon. If `{expr}` is not a list it will be split with `balloon_split()`.
If `{expr}` is an empty string any existing balloon is removed.

Example:

```
func GetBalloonContent()
    " ... initiate getting the content
    return ''
endfunc
set balloonexpr=GetBalloonContent()

func BalloonCallback(result)
    call balloon_show(a:result)
endfunc
```

Can also be used as a `method` :

```
GetText()->balloon_show()
```

The intended use is that fetching the content of the balloon is initiated from '`balloonexpr`'. It will invoke an asynchronous method, in which a callback invokes `balloon_show()`. The '`balloonexpr`' itself can return an empty string or a placeholder, e.g. "loading...".

When showing a balloon is not possible then nothing happens, no error message is given.

{only available when compiled with the `+balloon_eval` or `+balloon_eval_term` feature}

`balloon_split({msg})` `balloon_split()`
Split String `{msg}` into lines to be displayed in a balloon. The splits are made for the current window size and optimize to show debugger output.
Returns a `List` with the split lines. Returns an empty List on error.

Can also be used as a `method` :

```
GetText()->balloon_split()->balloon_show()
```

{only available when compiled with the `+balloon_eval_term` feature}

`blob2list({blob})` `blob2list()`
Return a List containing the number value of each byte in Blob `{blob}`. Examples:

```
blob2list(0z0102.0304) returns [1, 2, 3, 4]
blob2list(0z)           returns []
```

Returns an empty List on error. `list2blob()` does the opposite.

Can also be used as a `method` :

```
GetBlob()->blob2list()
```

`browse({save}, {title}, {initdir}, {default})` `browse()`
Put up a file requester. This only works when "has("browse")" returns `TRUE` (only in some GUI versions).
The input fields are:

```
{save} when TRUE , select file to write
```

```
{title}    title for the requester
{initdir}  directory to start browsing in
{default}  default file name
```

An empty string is returned when the "Cancel" button is hit, something went wrong, or browsing is not possible.

browsedir()

browsedir({title}, {initdir})

Put up a directory requester. This only works when "has("browse")" returns **TRUE** (only in some GUI versions). On systems where a directory browser is not supported a file browser is used. In that case: select a file in the directory to be used.

The input fields are:

```
{title}    title for the requester
{initdir}  directory to start browsing in
```

When the "Cancel" button is hit, something went wrong, or browsing is not possible, an empty string is returned.

bufadd({name})

bufadd()

Add a buffer to the buffer list with name {name} (must be a String).

If a buffer for file {name} already exists, return that buffer number. Otherwise return the buffer number of the newly created buffer. When {name} is an empty string then a new buffer is always created.

The buffer will not have **'buflisted'** set and not be loaded yet. To add some text to the buffer use this:

```
let bufnr = bufadd('someName')
call bufload(bufnr)
call setbufline(bufnr, 1, ['some', 'text'])
```

Returns 0 on error.

Can also be used as a **method** :

```
let bufnr = 'somename'->bufadd()
```

bufexists({buf})

bufexists()

The result is a Number, which is **TRUE** if a buffer called {buf} exists.

If the {buf} argument is a number, buffer numbers are used. Number zero is the alternate buffer for the current window.

If the {buf} argument is a string it must match a buffer name exactly. The name can be:

- Relative to the current directory.
- A full path.
- The name of a buffer with **'buftype'** set to "nofile".
- A URL name.

Unlisted buffers will be found.

Note that help files are listed by their short name in the output of **:buffers** , but bufexists() requires using their long name to be able to find them.

bufexists() may report a buffer exists, but to use the name with a **:buffer** command you may need to use **expand()** . Esp for MS-Windows 8.3 names in the form "c:\DOCUME~1"

Use "bufexists(0)" to test for the existence of an alternate file name.

Can also be used as a **method** :

```
let exists = 'somename'->bufexists()
```

Obsolete name: `buffer_exists()`. `buffer_exists()`

`buflisted({buf})` `buflisted()`

The result is a Number, which is **TRUE** if a buffer called `{buf}` exists and is listed (has the '**buflisted**' option set). The `{buf}` argument is used like with `bufexists()` .

Can also be used as a **method** :

```
let listed = 'somename'->buflisted()
```

`bufload({buf})` `bufload()`

Ensure the buffer `{buf}` is loaded. When the buffer name refers to an existing file then the file is read. Otherwise the buffer will be empty. If the buffer was already loaded then there is no change. If the buffer is not related to a file then no file is read (e.g., when '**buftype**' is "nofile"). If there is an existing swap file for the file of the buffer, there will be no dialog, the buffer will be loaded anyway. The `{buf}` argument is used like with `bufexists()` .

Can also be used as a **method** :

```
eval 'somename'->bufload()
```

`bufloaded({buf})` `bufloaded()`

The result is a Number, which is **TRUE** if a buffer called `{buf}` exists and is loaded (shown in a window or hidden). The `{buf}` argument is used like with `bufexists()` .

Can also be used as a **method** :

```
let loaded = 'somename'->bufloaded()
```

`bufname([{buf}])` `bufname()`

The result is the name of a buffer. Mostly as it is displayed by the `:ls` command, but not using special names such as "[No Name]".

If `{buf}` is omitted the current buffer is used.

If `{buf}` is a Number, that buffer number's name is given.

Number zero is the alternate buffer for the current window.

If `{buf}` is a String, it is used as a **file-pattern** to match with the buffer names. This is always done like '**magic**' is set and '**cptions**' is empty. When there is more than one match an empty string is returned.

" " or "%" can be used for the current buffer, "#" for the alternate buffer.

A full match is preferred, otherwise a match at the start, end or middle of the buffer name is accepted. If you only want a full match then put "^" at the start and "\$" at the end of the pattern.

Listed buffers are found first. If there is a single match with a listed buffer, that one is returned. Next unlisted buffers are searched for.

If the `{buf}` is a String, but you want to use it as a buffer number, force it to be a Number by adding zero to it:

```
:echo bufname("3" + 0)
```

Can also be used as a `method` :

```
echo bufnr->bufname()
```

If the buffer doesn't exist, or doesn't have a name, an empty string is returned.

```
bufname("#")      alternate buffer name
bufname(3)       name of buffer 3
bufname("%")     name of current buffer
bufname("file2") name of buffer where "file2" matches.
                                     buffer_name()
```

Obsolete name: `buffer_name()`.

`bufnr()`

```
bufnr([{buf} [, {create}]])
```

The result is the number of a buffer, as it is displayed by the `:ls` command. For the use of `{buf}`, see `bufname()` above.

If the buffer doesn't exist, `-1` is returned. Or, if the `{create}` argument is present and `TRUE`, a new, unlisted, buffer is created and its number is returned. Example:

```
let newbuf = bufnr('Scratch001', 1)
```

Using an empty name uses the current buffer. To create a new buffer with an empty name use `bufadd()` .

`bufnr("$")` is the last buffer:

```
:let last_buffer = bufnr("$")
```

The result is a Number, which is the highest buffer number of existing buffers. **Note** that not all buffers with a smaller number necessarily exist, because `":bwipeout"` may have removed them. Use `bufexists()` to test for the existence of a buffer.

Can also be used as a `method` :

```
echo bufref->bufnr()
```

Obsolete name: `buffer_number()`.

`buffer_number()`

`last_buffer_nr()`

Obsolete name for `bufnr("$")`: `last_buffer_nr()`.

```
bufwinid({buf})
```

`bufwinid()`

The result is a Number, which is the `window-ID` of the first window associated with buffer `{buf}`. For the use of `{buf}`, see `bufname()` above. If buffer `{buf}` doesn't exist or there is no such window, `-1` is returned. Example:

```
echo "A window containing buffer 1 is " .. (bufwinid(1))
```

Only deals with the current tab page. See `win_findbuf()` for

finding more.

Can also be used as a `method` :
`FindBuffer()->bufwinid()`

`bufwinnr({buf})`

`bufwinnr()`

Like `bufwinid()` but return the window number instead of the `window-ID` .

If buffer `{buf}` doesn't exist or there is no such window, `-1` is returned. Example:

```
echo "A window containing buffer 1 is " .. (bufwinnr(1))
```

The number can be used with `CTRL-W_w` and `":wincmd w"`
`:wincmd` .

Can also be used as a `method` :
`FindBuffer()->bufwinnr()`

`byte2line({byte})`

`byte2line()`

Return the line number that contains the character at byte count `{byte}` in the current buffer. This includes the end-of-line character, depending on the `'fileformat'` option for the current buffer. The first character has byte count one.

Also see `line2byte()` , `go` and `:goto` .

Returns `-1` if the `{byte}` value is invalid.

Can also be used as a `method` :
`GetOffset()->byte2line()`

{not available when compiled without the `+byte_offset` feature}

`byteidx({expr}, {nr} [, {utf16}])`

`byteidx()`

Return byte index of the `{nr}`'th character in the String `{expr}`. Use zero for the first character, it then returns zero.

If there are no multibyte characters the returned value is equal to `{nr}`.

Composing characters are not counted separately, their byte length is added to the preceding base character. See `byteidxcomp()` below for counting composing characters separately.

When `{utf16}` is present and TRUE, `{nr}` is used as the UTF-16 index in the String `{expr}` instead of as the character index. The UTF-16 index is the index in the string when it is encoded with 16-bit words. If the specified UTF-16 index is in the middle of a character (e.g. in a 4-byte character), then the byte index of the first byte in the character is returned. Refer to `string-offset-encoding` for more information.

Example :

```
echo matchstr(str, ".", byteidx(str, 3))
```

will display the fourth character. Another way to do the same:

```
let s = strpart(str, byteidx(str, 3))
echo strpart(s, 0, byteidx(s, 1))
```

Also see `strgetchar()` and `strcharpart()` .

If there are less than `{nr}` characters -1 is returned.

If there are exactly `{nr}` characters the length of the string in bytes is returned.

See `charidx()` and `utf16idx()` for getting the character and UTF-16 index respectively from the byte index.

Examples:

```
echo byteidx('a𐀀', 2) returns 5
echo byteidx('a𐀀', 2, 1) returns 1
echo byteidx('a𐀀', 3, 1) returns 5
```

Can also be used as a `method` :

```
GetName()->byteidx(idx)
```

`byteidxcomp({expr}, {nr} [, {utf16}])` `byteidxcomp()`

Like `byteidx()`, except that a composing character is counted as a separate character. Example:

```
let s = 'e' .. nr2char(0x301)
echo byteidx(s, 1)
echo byteidxcomp(s, 1)
echo byteidxcomp(s, 2)
```

The first and third echo result in 3 ('e' plus composing character is 3 bytes), the second echo results in 1 ('e' is one byte).

Only works differently from `byteidx()` when `'encoding'` is set to a Unicode encoding.

Can also be used as a `method` :

```
GetName()->byteidxcomp(idx)
```

`call({func}, {arglist} [, {dict}])` `call()` E699

Call function `{func}` with the items in `List {arglist}` as arguments.

`{func}` can either be a `Funcref` or the name of a function. `a:firstline` and `a:lastline` are set to the cursor line.

Returns the return value of the called function.

`{dict}` is for functions with the "dict" attribute. It will be used to set the local variable "self". `Dictionary-function`

Can also be used as a `method` :

```
GetFunc()->call([arg, arg], dict)
```

`ceil({expr})` `ceil()`

Return the smallest integral value greater than or equal to `{expr}` as a `Float` (round up).

`{expr}` must evaluate to a `Float` or a `Number` .

Examples:

```
echo ceil(1.456)
2.0
```

```
echo ceil(-5.456)
-5.0
echo ceil(4.0)
4.0
```

Returns 0.0 if `{expr}` is not a `Float` or a `Number` .

Can also be used as a `method` :
`Compute()->ceil()`

ch_ functions are documented here: [channel-functions-details](#)

`changenr()`

`changenr()`

Return the number of the most recent change. This is the same number as what is displayed with `:undolist` and can be used with the `:undo` command.
When a change was made it is the number of that change. After redo it is the number of the redone change. After undo it is one less than the number of the undone change.
Returns 0 if the undo list is empty.

`char2nr({string} [, {utf8}])`

`char2nr()`

Return Number value of the first char in `{string}`.

Examples:

```
char2nr(" ")           returns 32
char2nr("ABC")         returns 65
```

When `{utf8}` is omitted or zero, the current `'encoding'` is used.

Example for "utf-8":

```
char2nr("á")           returns 225
char2nr("á"[0])        returns 195
```

When `{utf8}` is TRUE, always treat as UTF-8 characters.

A combining character is a separate character.

`nr2char()` does the opposite.

To turn a string into a list of character numbers:

```
let str = "ABC"
let list = map(split(str, '\zs'), {_, val -> char2nr(val)})
```

Result: [65, 66, 67]

Returns 0 if `{string}` is not a `String` .

Can also be used as a `method` :
`GetChar()->char2nr()`

`charclass({string})`

`charclass()`

Return the character class of the first character in `{string}`.
The character class is one of:

```
0      blank
1      punctuation
2      word character
3      emoji
other  specific Unicode class
```

The class is used in patterns and word motions.

Returns 0 if `{string}` is not a `String` .

`charcol({expr} [, {winid}])` `charcol()`
Same as `col()` but returns the character index of the column position given with `{expr}` instead of the byte position.

Example:

With the cursor on 'Ø' in line 5 with text "ØØØØ":
`charcol('.')` returns 3
`col('.')` returns 7

Can also be used as a `method` :
`GetPos()->col()`

`charidx({string}, {idx} [, {countcc} [, {utf16}]])` `charidx()`
Return the character index of the byte at `{idx}` in `{string}`. The index of the first character is zero. If there are no multibyte characters the returned value is equal to `{idx}`.

When `{countcc}` is omitted or `FALSE` , then composing characters are not counted separately, their byte length is added to the preceding base character.

When `{countcc}` is `TRUE` , then composing characters are counted as separate characters.

When `{utf16}` is present and `TRUE`, `{idx}` is used as the UTF-16 index in the `String {expr}` instead of as the byte index.

Returns -1 if the arguments are invalid or if there are less than `{idx}` bytes. If there are exactly `{idx}` bytes the length of the string in characters is returned.

An error is given and -1 is returned if the first argument is not a string, the second argument is not a number or when the third argument is present and is not zero or one.

See `byteidx()` and `byteidxcomp()` for getting the byte index from the character index and `utf16idx()` for getting the UTF-16 index from the character index.

Refer to `string-offset-encoding` for more information.

Examples:

```
echo charidx('ábc', 3)           returns 1
echo charidx('ábc', 6, 1)       returns 4
echo charidx('ábc', 16)         returns -1
echo charidx('aØ', 4, 0, 1)     returns 2
```

Can also be used as a `method` :
`GetName()->charidx(idx)`

`chdir({dir})` `chdir()`
Change the current working directory to `{dir}`. The scope of

the directory change depends on the directory of the current window:

- If the current window has a window-local directory (`:lcd`), then changes the window local directory.
- Otherwise, if the current tabpage has a local directory (`:tcd`) then changes the tabpage local directory.
- Otherwise, changes the global directory.

`{dir}` must be a String.

If successful, returns the previous working directory. Pass this to another `chdir()` to restore the directory.

On failure, returns an empty string.

Example:

```
let save_dir = chdir(newdir)
if save_dir != ""
    ... do some work
    call chdir(save_dir)
endif
```

Can also be used as a `method` :

```
GetDir()->chdir()
```

`cindent({lnum})`

`cindent()`

Get the amount of indent for line `{lnum}` according the C indenting rules, as with '`cindent`'.

The indent is counted in spaces, the value of '`tabstop`' is relevant. `{lnum}` is used just like in `getline()` .

When `{lnum}` is invalid -1 is returned.

See `C-indenting` .

Can also be used as a `method` :

```
GetLnum()->cindent()
```

`clearmatches([win])`

`clearmatches()`

Clears all matches previously defined for the current window by `matchadd()` and the `:match` commands.

If `{win}` is specified, use the window with this number or window ID instead of the current window.

Can also be used as a `method` :

```
GetWin()->clearmatches()
```

`col({expr} [, winid])`

`col()`

The result is a Number, which is the byte index of the column position given with `{expr}`. The accepted positions are:

- . the cursor position
- \$ the end of the cursor line (the result is the number of bytes in the cursor line plus one)
- 'x' position of mark x (if the mark is not set, 0 is returned)
- v In Visual mode: the start of the Visual area (the cursor is the end). When not in Visual mode returns the cursor position. Differs from '<' in

that it's updated right away.
 Additionally `{expr}` can be `[lnum, col]`: a `List` with the line and column number. Most useful when the column is "\$", to get the last column of a specific line. When "lnum" or "col" is out of range then `col()` returns zero.
 With the optional `{winid}` argument the values are obtained for that window instead of the current window.
 To get the line number use `line()` . To get both use `getpos()` .
 For the screen column position use `virtcol()` . For the character position use `charcol()` .
Note that only marks in the current file can be used.
 Examples:

```
col(".")           column of cursor
col("$")          length of cursor line plus one
col("'t")         column of mark t
col("' " .. markname) column of mark markname
```

The first column is 1. Returns 0 if `{expr}` is invalid or when the window with ID `{winid}` is not found.

For an uppercase mark the column may actually be in another buffer.

For the cursor position, when **'virtualedit'** is active, the column is one higher if the cursor is after the end of the line. Also, when using a `<Cmd>` mapping the cursor isn't moved, this can be used to obtain the column in Insert mode:

```
:imap <F2> <Cmd>echowin col(".")<CR>
```

Can also be used as a `method` :

```
GetPos()->col()
```

`complete({startcol}, {matches})` `complete()` E785

Set the matches for Insert mode completion.

Can only be used in Insert mode. You need to use a mapping with **CTRL-R** = (see `i_CTRL-R`). It does not work after **CTRL-O** or with an expression mapping.

`{startcol}` is the byte offset in the line where the completed text start. The text up to the cursor is the original text that will be replaced by the matches. Use `col('.')` for an empty string. "`col('.') - 1`" will replace one character by a match.

`{matches}` must be a `List` . Each `List` item is one match. See `complete-items` for the kind of items that are possible. "longest" in **'completeopt'** is ignored.

Note that the after calling this function you need to avoid inserting anything that would cause completion to stop.

The match can be selected with **CTRL-N** and **CTRL-P** as usual with Insert mode completion. The popup menu will appear if specified, see `ins-completion-menu` .

Example:

```
inoremap <F5> <C-R>=ListMonths()<CR>
```

```
func ListMonths()
  call complete(col('.'), ['January', 'February', 'March',
```

```

        \ 'April', 'May', 'June', 'July', 'August', 'September',
        \ 'October', 'November', 'December'])
return ''
endfunc

```

This isn't very useful, but it shows how it works. [Note](#) that an empty string is returned to avoid a zero being inserted.

Can also be used as a [method](#) , the base is passed as the second argument:

```
GetMatches()->complete(col('.'))
```

`complete_add({expr})`

[complete_add\(\)](#)

Add `{expr}` to the list of matches. Only to be used by the function specified with the `'completefunc'` option. Returns 0 for failure (empty string or out of memory), 1 when the match was added, 2 when the match was already in the list.

See [complete-functions](#) for an explanation of `{expr}`. It is the same as one item in the list that `'omnifunc'` would return.

Can also be used as a [method](#) :

```
GetMoreMatches()->complete_add()
```

`complete_check()`

[complete_check\(\)](#)

Check for a key typed while looking for completion matches. This is to be used when looking for matches takes some time. Returns `TRUE` when searching for matches is to be aborted, zero otherwise.

Only to be used by the function specified with the `'completefunc'` option.

`complete_info([{what}])`

[complete_info\(\)](#)

Returns a [Dictionary](#) with information about Insert mode completion. See [ins-completion](#) .

The items are:

<code>mode</code>	Current completion mode name string. See complete_info_mode for the values.
<code>pum_visible</code>	<code>TRUE</code> if popup menu is visible. See pumvisible() .
<code>items</code>	List of completion matches. Each item is a dictionary containing the entries "word", "abbr", "menu", "kind", "info" and "user_data". See complete-items .
<code>selected</code>	Selected item index. First index is zero. Index is -1 if no item is selected (showing typed text only, or the last completion after no item is selected when using the <code><Up></code> or <code><Down></code> keys)
<code>inserted</code>	Inserted string. [NOT IMPLEMENTED YET]

[complete_info_mode](#)

mode values are:

```
""           Not in completion mode
```

"keyword"	Keyword completion <code>i_CTRL-X_CTRL-N</code>
"ctrl_x"	Just pressed CTRL-X <code>i_CTRL-X</code>
"scroll"	Scrolling with <code>i_CTRL-X_CTRL-E</code> or <code>i_CTRL-X_CTRL-Y</code>
"whole_line"	Whole lines <code>i_CTRL-X_CTRL-L</code>
"files"	File names <code>i_CTRL-X_CTRL-F</code>
"tags"	Tags <code>i_CTRL-X_CTRL-]</code>
"path_defines"	Definition completion <code>i_CTRL-X_CTRL-D</code>
"path_patterns"	Include completion <code>i_CTRL-X_CTRL-I</code>
"dictionary"	Dictionary <code>i_CTRL-X_CTRL-K</code>
"thesaurus"	Thesaurus <code>i_CTRL-X_CTRL-T</code>
"cmdline"	Vim Command line <code>i_CTRL-X_CTRL-V</code>
"function"	User defined completion <code>i_CTRL-X_CTRL-U</code>
"omni"	Omni completion <code>i_CTRL-X_CTRL-O</code>
"spell"	Spelling suggestions <code>i_CTRL-X_s</code>
"eval"	<code>complete()</code> completion
"unknown"	Other internal modes

If the optional `{what}` list argument is supplied, then only the items listed in `{what}` are returned. Unsupported items in `{what}` are silently ignored.

To get the position and size of the popup menu, see `pum_getpos()`. It's also available in `v:event` during the `CompleteChanged` event.

Returns an empty `Dictionary` on error.

Examples:

```
" Get all items
call complete_info()
" Get only 'mode'
call complete_info(['mode'])
" Get only 'mode' and 'pum_visible'
call complete_info(['mode', 'pum_visible'])
```

Can also be used as a `method` :

```
GetItems()->complete_info()
```

```
confirm({msg} [, {choices} [, {default} [, {type}]]])
```

`confirm()` offers the user a dialog, from which a choice can be made. It returns the number of the choice. For the first choice this is 1.

Note: `confirm()` is only supported when compiled with dialog support, see `+dialog_con` and `+dialog_gui`.

`{msg}` is displayed in a `dialog` with `{choices}` as the alternatives. When `{choices}` is missing or empty, "&OK" is used (and translated).

`{msg}` is a String, use `'\n'` to include a newline. Only on some systems the string is wrapped when it doesn't fit.

`{choices}` is a String, with the individual choices separated

by '\n', e.g.

```
confirm("Save changes?", "&Yes\n&No\n&Cancel")
```

The letter after the '&' is the shortcut key for that choice.

Thus you can type 'c' to select "Cancel". The shortcut does not need to be the first letter:

```
confirm("file has been modified", "&Save\nSave &All")
```

For the console, the first letter of each choice is used as the default shortcut key. Case is ignored.

The optional `{default}` argument is the number of the choice that is made if the user hits `<CR>`. Use 1 to make the first choice the default one. Use 0 to not set a default. If `{default}` is omitted, 1 is used.

The optional `{type}` String argument gives the type of dialog. This is only used for the icon of the GTK, Mac, Motif and Win32 GUI. It can be one of these values: "Error", "Question", "Info", "Warning" or "Generic". Only the first character is relevant. When `{type}` is omitted, "Generic" is used.

If the user aborts the dialog by pressing `<Esc>`, **CTRL-C**, or another valid interrupt key, `confirm()` returns 0.

An example:

```
let choice = confirm("What do you want?",
                    \ "&Apples\n&Oranges\n&Bananas", 2)
if choice == 0
    echo "make up your mind!"
elseif choice == 3
    echo "tasteful"
else
    echo "I prefer bananas myself."
endif
```

In a GUI dialog, buttons are used. The layout of the buttons depends on the 'v' flag in '**guioptions**'. If it is included, the buttons are always put vertically. Otherwise, `confirm()` tries to put the buttons in one horizontal line. If they don't fit, a vertical layout is used anyway. For some systems the horizontal layout is always used.

Can also be used as a **method** in:

```
BuildMessage()->confirm("&Yes\n&No")
```

copy()

`copy({expr})`

Make a copy of `{expr}`. For Numbers and Strings this isn't different from using `{expr}` directly.

When `{expr}` is a **List** a shallow copy is created. This means that the original **List** can be changed without changing the copy, and vice versa. But the items are identical, thus changing an item changes the contents of both **Lists**.

A **Dictionary** is copied in a similar way as a **List**.

Also see `deepcopy()`.

Can also be used as a **method** :

`mylist->copy()`

`cos({expr})`

`cos()`

Return the cosine of `{expr}`, measured in radians, as a `Float`.
`{expr}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo cos(100)
0.862319
:echo cos(-4.01)
-0.646043
```

Can also be used as a `method` :

`Compute()->cos()`

`cosh({expr})`

`cosh()`

Return the hyperbolic cosine of `{expr}` as a `Float` in the range
[1, inf].

`{expr}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo cosh(0.5)
1.127626
:echo cosh(-0.5)
-1.127626
```

Can also be used as a `method` :

`Compute()->cosh()`

`count({comp}, {expr} [, {ic} [, {start}]])`

`count()` E706

Return the number of times an item with value `{expr}` appears
in `String`, `List` or `Dictionary` `{comp}`.

If `{start}` is given then start with the item with this index.
`{start}` can only be used with a `List`.

When `{ic}` is given and it's `TRUE` then case is ignored.

When `{comp}` is a string then the number of not overlapping
occurrences of `{expr}` is returned. Zero is returned when
`{expr}` is an empty string.

Can also be used as a `method` :

`mylist->count(val)`

`cscope_connection([{num} , {dbpath} [, {prepend}]])`

`cscope_connection()`

Checks for the existence of a `cscope` connection. If no
parameters are specified, then the function returns:

```
0, if cscope was not available (not compiled in), or
   if there are no cscope connections;
1, if there is at least one cscope connection.
```

If parameters are specified, then the value of `{num}` determines how existence of a cscope connection is checked:

<code>{num}</code>	Description of existence check
0	Same as no parameters (e.g., "cscope_connection()").
1	Ignore <code>{prepend}</code> , and use partial string matches for <code>{dbpath}</code> .
2	Ignore <code>{prepend}</code> , and use exact string matches for <code>{dbpath}</code> .
3	Use <code>{prepend}</code> , use partial string matches for both <code>{dbpath}</code> and <code>{prepend}</code> .
4	Use <code>{prepend}</code> , use exact string matches for both <code>{dbpath}</code> and <code>{prepend}</code> .

Note: All string comparisons are case sensitive!

Examples. Suppose we had the following (from "cs show"):

```
# pid      database name      prepend path
0 27664    cscope.out         /usr/local
```

Invocation	Return Val
<code>cscope_connection()</code>	1
<code>cscope_connection(1, "out")</code>	1
<code>cscope_connection(2, "out")</code>	0
<code>cscope_connection(3, "out")</code>	0
<code>cscope_connection(3, "out", "local")</code>	1
<code>cscope_connection(4, "out")</code>	0
<code>cscope_connection(4, "out", "local")</code>	0
<code>cscope_connection(4, "cscope.out", "/usr/local")</code>	1

```
cursor({lnum}, {col} [, {off}])      cursor()
cursor({list})
```

Positions the cursor at the column (byte count) `{col}` in the line `{lnum}`. The first column is one.

When there is one argument `{list}` this is used as a `List` with two, three or four item:

```
[{lnum}, {col}]
[{lnum}, {col}, {off}]
[{lnum}, {col}, {off}, {curswant}]
```

This is like the return value of `getpos()` or `getcurpos()`, but without the first item.

To position the cursor using `{col}` as the character count, use `setcursorcharpos()`.

Does not change the jumplist.

`{lnum}` is used like with `getline()`, except that if `{lnum}` is zero, the cursor will stay in the current line.

If `{lnum}` is greater than the number of lines in the buffer,

the cursor will be positioned at the last line in the buffer. If `{col}` is greater than the number of bytes in the line, the cursor will be positioned at the last character in the line.

If `{col}` is zero, the cursor will stay in the current column. If `{curswant}` is given it is used to set the preferred column for vertical movement. Otherwise `{col}` is used.

When `'virtualedit'` is used `{off}` specifies the offset in screen columns from the start of the character. E.g., a position within a `<Tab>` or after the last character. Returns 0 when the position could be set, -1 otherwise.

Can also be used as a `method` :
`GetCursorPos()->cursor()`

`debugbreak({pid})` `debugbreak()`
Specifically used to interrupt a program being debugged. It will cause process `{pid}` to get a SIGTRAP. Behavior for other processes is undefined. See `terminal-debugger` .
`{only available on MS-Windows}`
Returns `TRUE` if successfully interrupted the program. Otherwise returns `FALSE` .

Can also be used as a `method` :
`GetPid()->debugbreak()`

`deepcopy({expr} [, {noref}])` `deepcopy()` E698
Make a copy of `{expr}`. For Numbers and Strings this isn't different from using `{expr}` directly.
When `{expr}` is a `List` a full copy is created. This means that the original `List` can be changed without changing the copy, and vice versa. When an item is a `List` or `Dictionary` , a copy for it is made, recursively. Thus changing an item in the copy does not change the contents of the original `List` .
A `Dictionary` is copied in a similar way as a `List` .
When `{noref}` is omitted or zero a contained `List` or `Dictionary` is only copied once. All references point to this single copy. With `{noref}` set to 1 every occurrence of a `List` or `Dictionary` results in a new copy. This also means that a cyclic reference causes `deepcopy()` to fail. E724
Nesting is possible up to 100 levels. When there is an item that refers back to a higher level making a deep copy with `{noref}` set to 1 will fail.
Also see `copy()` .

Can also be used as a `method` :
`GetObject()->deepcopy()`

`delete({fname} [, {flags}])` `delete()`

Without `{flags}` or with `{flags}` empty: Deletes the file by the name `{fname}`.

This also works when `{fname}` is a symbolic link. The symbolic link itself is deleted, not what it points to.

When `{flags}` is "d": Deletes the directory by the name `{fname}`. This fails when directory `{fname}` is not empty.

When `{flags}` is "rf": Deletes the directory by the name `{fname}` and everything in it, recursively. BE CAREFUL!

Note: on MS-Windows it is not possible to delete a directory that is being used.

The result is a Number, which is 0/false if the delete operation was successful and -1/true when the deletion failed or partly failed.

Use `remove()` to delete an item from a `List` .
To delete a line from the buffer use `:delete` or `deletebufline()` .

Can also be used as a `method` :
`GetName()->delete()`

`deletebufline({buf}, {first} [, {last}])` `deletebufline()`
Delete lines `{first}` to `{last}` (inclusive) from buffer `{buf}`.
If `{last}` is omitted then delete line `{first}` only.
On success 0 is returned, on failure 1 is returned.

This function works only for loaded buffers. First call `bufload()` if needed.

For the use of `{buf}`, see `bufname()` above.

`{first}` and `{last}` are used like with `getline()` . **Note** that when using `line()` this refers to the current buffer. Use "\$" to refer to the last line in buffer `{buf}`.

Can also be used as a `method` :
`GetBuffer()->deletebufline(1)`

`did_filetype()` `did_filetype()`
Returns `TRUE` when autocommands are being executed and the FileType event has been triggered at least once. Can be used to avoid triggering the FileType event again in the scripts that detect the file type. `FileType`
Returns `FALSE` when ``:setf FALLBACK`` was used.
When editing another file, the counter is reset, thus this really checks if the FileType event has been triggered for the current buffer. This allows an autocommand that starts editing another buffer to set `'filetype'` and load a syntax file.

`diff({fromlist}, {tolist} [, {options}])` `diff()`

Returns a String or a List containing the diff between the strings in `{fromlist}` and `{tolist}`. Uses the Vim internal diff library to compute the diff.

E106

The optional "output" item in `{options}` specifies the returned diff format. The following values are supported:

- `indices` Return a List of the starting and ending indices and a count of the strings in each diff hunk.
- `unified` Return the unified diff output as a String. This is the default.

If the "output" item in `{options}` is "indices", then a List is returned. Each List item contains a Dict with the following items for each diff hunk:

- `from_idx` start index in `{fromlist}` for this diff hunk.
- `from_count` number of strings in `{fromlist}` that are added/removed/modified in this diff hunk.
- `to_idx` start index in `{tolist}` for this diff hunk.
- `to_count` number of strings in `{tolist}` that are added/removed/modified in this diff hunk.

The `{options}` Dict argument also specifies diff options (similar to `'diffopt'`) and supports the following items:

- `algorithm` Dict specifying the diff algorithm to use. Supported boolean items are "myers", "minimal", "patience" and "histogram".
- `context` diff context length. Default is 0.
- `iblack` ignore changes where lines are all blank.
- `icase` ignore changes in case of text.
- `indent-heuristic` use the indent heuristic for the internal diff library.
- `iwhite` ignore changes in amount of white space.
- `iwhiteall` ignore all white space changes.
- `iwhiteeol` ignore white space changes at end of line.

For more information about these options, refer to `'diffopt'`.

To compute the unified diff, all the items in `{fromlist}` are concatenated into a string using a newline separator and the same for `{tolist}`. The unified diff output uses line numbers.

Returns an empty List or String if `{fromlist}` and `{tolist}` are identical.

Examples:

```
:echo diff(['abc'], ['xxx'])  
@@ -1 +1 @@  
-abc
```

```
+xxx
```

```
:echo diff(['abc'], ['xxx'], {'output': 'indices'})
  [{ 'from_idx': 0, 'from_count': 1, 'to_idx': 0, 'to_count': 1}]
:echo diff(readfile('oldfile'), readfile('newfile'))
:echo diff(getbufline(5, 1, '$'), getbufline(6, 1, '$'))
```

For more examples, refer to [diff-func-examples](#)

Can also be used as a [method](#) :
[GetFromList](#)->diff(to_list)

[diff_filler\({lnum}\)](#) [diff_filler\(\)](#)
Returns the number of filler lines above line [{lnum}](#).
These are the lines that were inserted at this point in
another diff'ed window. These filler lines are shown in the
display but don't exist in the buffer.
[{lnum}](#) is used like with [getline\(\)](#) . Thus "." is the current
line, "'m" mark m, etc.
Returns 0 if the current window is not in diff mode.

Can also be used as a [method](#) :
[GetLnum\(\)](#)->diff_filler()

[diff_hlID\({lnum}, {col}\)](#) [diff_hlID\(\)](#)
Returns the highlight ID for diff mode at line [{lnum}](#) column
[{col}](#) (byte index). When the current line does not have a
diff change zero is returned.
[{lnum}](#) is used like with [getline\(\)](#) . Thus "." is the current
line, "'m" mark m, etc.
[{col}](#) is 1 for the leftmost column, [{lnum}](#) is 1 for the first
line.
The highlight ID can be used with [synIDattr\(\)](#) to obtain
syntax information about the highlighting.

Can also be used as a [method](#) :
[GetLnum\(\)](#)->diff_hlID(col)

[digraph_get\({chars}\)](#) [digraph_get\(\)](#) [E1214](#)
Return the digraph of [{chars}](#). This should be a string with
exactly two characters. If [{chars}](#) are not just two
characters, or the digraph of [{chars}](#) does not exist, an error
is given and an empty string is returned.

The character will be converted from Unicode to ['encoding'](#)
when needed. This does require the conversion to be
available, it might fail.

Also see [digraph_getlist\(\)](#) .

Examples:

```
" Get a built-in digraph
:echo digraph_get('00')           " Returns '∞'
```

```
" Get a user-defined digraph
:call digraph_set('aa', '⌘')
:echo digraph_get('aa')           " Returns '⌘'
```

Can also be used as a `method` :

```
GetChars()->digraph_get()
```

This function works only when compiled with the `+digraphs` feature. If this feature is disabled, this function will display an error message.

```
digraph_getlist([listall])           digraph_getlist()
Return a list of digraphs. If the {listall} argument is given
and it is TRUE, return all digraphs, including the default
digraphs. Otherwise, return only user-defined digraphs.
```

The characters will be converted from Unicode to `'encoding'` when needed. This does require the conservation to be available, it might fail.

Also see `digraph_get()` .

Examples:

```
" Get user-defined digraphs
:echo digraph_getlist()
```

```
" Get all the digraphs, including default digraphs
:echo digraph_getlist(1)
```

Can also be used as a `method` :

```
GetNumber()->digraph_getlist()
```

This function works only when compiled with the `+digraphs` feature. If this feature is disabled, this function will display an error message.

```
digraph_set({chars}, {digraph})     digraph_set()
Add digraph {chars} to the list. {chars} must be a string
with two characters. {digraph} is a string with one UTF-8
encoded character. E1215
Be careful, composing characters are NOT ignored. This
function is similar to :digraphs command, but useful to add
digraphs start with a white space.
```

The function result is `v:true` if `digraph` is registered. If this fails an error message is given and `v:false` is returned.

If you want to define multiple digraphs at once, you can use `digraph_setlist()` .

Example:

```
call digraph_set(' ', '␣')
```

Can be used as a `method` :

```
GetString()->digraph_set('␣')
```

This function works only when compiled with the `+digraphs` feature. If this feature is disabled, this function will display an error message.

`digraph_setlist({digraphlist})` `digraph_setlist()`

Similar to `digraph_set()` but this function can add multiple digraphs at once. `{digraphlist}` is a list composed of lists, where each list contains two strings with `{chars}` and `{digraph}` as in `digraph_set()` . [E1216](#)

Example:

```
call digraph_setlist([[ 'aa', '␣' ], [ 'ii', '␣' ]])
```

It is similar to the following:

```
for [chars, digraph] in [[ 'aa', '␣' ], [ 'ii', '␣' ]]  
  call digraph_set(chars, digraph)  
endfor
```

Except that the function returns after the first error, following digraphs will not be added.

Can be used as a `method` :

```
GetList()->digraph_setlist()
```

This function works only when compiled with the `+digraphs` feature. If this feature is disabled, this function will display an error message.

`echoraw({string})` `echoraw()`

Output `{string}` as-is, including unprintable characters. This can be used to output a terminal code. For example, to disable `modifyOtherKeys`:

```
call echoraw(&t_TE)
```

and to enable it again:

```
call echoraw(&t_TI)
```

Use with care, you can mess up the terminal this way.

`empty({expr})` `empty()`

Return the Number 1 if `{expr}` is empty, zero otherwise.

- A `List` or `Dictionary` is empty when it does not have any items.
- A `String` is empty when its length is zero.
- A `Number` and `Float` are empty when their value is zero.
- `v:false` , `v:none` and `v:null` are empty, `v:true` is not.
- A `Job` is empty when it failed to start.
- A `Channel` is empty when it is closed.
- A `Blob` is empty when its length is zero.
- An `Object` is empty, when the `empty()` builtin method in

the object (if present) returns true.

For a long `List` this is much faster than comparing the length with zero.

Can also be used as a `method` :

```
mylist->empty()
```

`environ()`

`environ()`

Return all of environment variables as dictionary. You can check if an environment variable exists like this:

```
:echo has_key(environ(), 'HOME')
```

Note that the variable name may be CamelCase; to ignore case use this:

```
:echo index(keys(environ()), 'HOME', 0, 1) != -1
```

`err_teapot([{expr}])`

`err_teapot()`

Produce an error with number 418, needed for implementation of RFC 2324.

If `{expr}` is present and it is TRUE error 503 is given, indicating that coffee is temporarily not available.

If `{expr}` is present it must be a String.

`escape({string}, {chars})`

`escape()`

Escape the characters in `{chars}` that occur in `{string}` with a backslash. Example:

```
:echo escape('c:\program files\vim', ' \')
```

results in:

```
c:\\program\ files\\vim
```

Also see `shellescape()` and `fnameescape()` .

Can also be used as a `method` :

```
GetText()->escape(' \')
```

`eval({string})`

`eval()`

Evaluate `{string}` and return the result. Especially useful to turn the result of `string()` back into the original value. This works for Numbers, Floats, Strings, Blobs and composites of them. Also works for `Funcref` s that refer to existing functions.

Can also be used as a `method` :

```
argv->join()->eval()
```

`eventhandler()`

`eventhandler()`

Returns 1 when inside an event handler. That is that Vim got interrupted while waiting for the user to type a character, e.g., when dropping a file on Vim. This means interactive commands cannot be used. Otherwise zero is returned.

`executable({expr})`

`executable()`

This function checks if an executable with the name `{expr}`

exists. `{expr}` must be the name of the program without any arguments.

`executable()` uses the value of `$PATH` and/or the normal searchpath for programs.

`PATHTEXT`

On MS-Windows the ".exe", ".bat", etc. can optionally be included. Then the extensions in `$PATHEXT` are tried. Thus if "foo.exe" does not exist, "foo.exe.bat" can be found. If `$PATHEXT` is not set then ".com;.exe;.bat;.cmd" is used. A dot by itself can be used in `$PATHEXT` to try using the name without an extension. When '`shell`' looks like a Unix shell, then the name is also tried without adding an extension.

On MS-Windows it only checks if the file exists and is not a directory, not if it's really executable.

On MS-Windows an executable in the same directory as Vim is normally found. Since this directory is added to `$PATH` it should also work to execute it `win32-PATH`. This can be disabled by setting the `$NoDefaultCurrentDirectoryInExePath` environment variable.

`NoDefaultCurrentDirectoryInExePath`

The result is a Number:

```
1      exists
0      does not exist
-1     not implemented on this system
```

`exeopath()` can be used to get the full path of an executable.

Can also be used as a `method` :

```
GetCommand()->executable()
```

`execute({command} [, {silent}])` `execute()`

Execute an Ex command or commands and return the output as a string.

`{command}` can be a string or a List. In case of a List the lines are executed one by one.

This is more or less equivalent to:

```
redir => var
{command}
redir END
```

Except that line continuation in `{command}` is not recognized.

The optional `{silent}` argument can have these values:

```
""      no `:silent` used
"silent"  `:silent` used
"silent!" `:silent!` used
```

The default is "silent". Note that with "silent!", unlike `:redir`, error messages are dropped. When using an external command the screen may be messed up, use `system()` instead.

`E930`

It is not possible to use `:redir` anywhere in `{command}`.

To get a list of lines use `split()` on the result:

```
execute('args')->split("\n")
```

To execute a command in another window than the current one use `win_execute()`.

When used recursively the output of the recursive call is not included in the output of the higher level call.

Can also be used as a `method` :
`GetCommand()->execute()`

`exepath({expr})`

`exepath()`

If `{expr}` is an executable and is either an absolute path, a relative path or found in `$PATH`, return the full path.

Note that the current directory is used when `{expr}` starts with `"/"`, which may be a problem for Vim:

```
echo exepath(v:progbath)
```

If `{expr}` cannot be found in `$PATH` or is not executable then an empty string is returned.

Can also be used as a `method` :
`GetCommand()->exepath()`

`exists({expr})`

`exists()`

The result is a Number, which is `TRUE` if `{expr}` is defined, zero otherwise.

Note: In a compiled `:def` function the evaluation is done at runtime. Use ``exists_compiled()`` to evaluate the expression at compile time.

For checking for a supported feature use `has()` .

For checking if a file exists use `filereadable()` .

The `{expr}` argument is a string, which contains one of these:

<code>varname</code>	internal variable (see
<code>dict.key</code>	<code>internal-variables</code>). Also works
<code>list[i]</code>	for <code>curly-braces-names</code> , <code>Dictionary</code>
<code>import.Func</code>	entries, <code>List</code> items, class and
<code>class.Func</code>	object methods, imported items, etc.
<code>object.Func</code>	Does not work for local variables in a
<code>class.varname</code>	compiled <code>:def</code> function.
<code>object.varname</code>	Also works for a function in <code>Vim9</code>

script, since it can be used as a function reference.
Beware that evaluating an index may cause an error message for an invalid expression. E.g.:

```
:let l = [1, 2, 3]
:echo exists("l[5]")
```

```
0
```

```
:echo exists("l[xx]")
```

```
E121: Undefined variable: xx
```

```
0
```

<code>&option-name</code>	Vim option (only checks if it exists, not if it really works)
-------------------------------	---

<code>+option-name</code>	Vim option that works.
---------------------------	------------------------

<code>\$ENVNAME</code>	environment variable (could also be done by comparing with an empty
------------------------	---

	string)
*funcname	built-in function (see <code>functions</code>) or user defined function (see <code>user-functions</code>) that is implemented. Also works for a variable that is a Funcref.
?funcname	built-in function that could be implemented; to be used to check if "funcname" is valid
:cmdname	Ex command: built-in command, user command or command modifier <code>:command</code> . Returns: 1 for match with start of a command 2 full match with a command 3 matches several user commands To check for a supported command always check the return value to be 2.
:2match	The <code>:2match</code> command.
:3match	The <code>:3match</code> command (but you probably should not use it, it is reserved for internal usage)
#event	autocommand defined for this event
#event#pattern	autocommand defined for this event and pattern (the pattern is taken literally and compared to the autocommand patterns character by character)
#group	autocommand group exists
#group#event	autocommand defined for this group and event.
#group#event#pattern	autocommand defined for this group, event and pattern.
##event	autocommand for this event is supported.

Examples:

```
exists("&shortname")
exists("$HOSTNAME")
exists("*strftime")
exists("s:MyFunc")      " only for legacy script
exists("*MyFunc")
exists("bufcount")
exists(":Make")
exists("#CursorHold")
exists("#BufReadPre#*.gz")
exists("#filetypeindent")
exists("#filetypeindent#FileType")
exists("#filetypeindent#FileType#*")
exists("##ColorScheme")
```

There must be no space between the symbol (&/\$/*/#) and the name.

There must be no extra characters after the name, although in a few cases this is ignored. That may become stricter in the

future, thus don't count on it!
Working example:
 exists(":make")
NOT working example:
 exists(":make install")

Note that the argument must be a string, not the name of the variable itself. For example:

```
exists(bufcount)
```

This doesn't check for existence of the "bufcount" variable, but gets the value of "bufcount", and checks if that exists.

Can also be used as a **method** :

```
Varname()->exists()
```

exists_compiled({expr}) exists_compiled()

Like `exists()` but evaluated at compile time. This is useful to skip a block where a function is used that would otherwise give an error:

```
if exists_compiled('*ThatFunction')
    ThatFunction('works')
endif
```

If `exists()` were used then a compilation error would be given if `ThatFunction()` is not defined.

`{expr}` must be a literal string. **E1232**

Can only be used in a `:def` function. **E1233**

This does not work to check for arguments or local variables.

exp({expr}) exp()

Return the exponential of `{expr}` as a **Float** in the range `[0, inf]`.

`{expr}` must evaluate to a **Float** or a **Number**.

Returns 0.0 if `{expr}` is not a **Float** or a **Number**.

Examples:

```
:echo exp(2)
7.389056
:echo exp(-1)
0.367879
```

Can also be used as a **method** :

```
Compute()->exp()
```

expand({string} [, {nosuf} [, {list}]]) expand()

Expand wildcards and the following special keywords in `{string}`. **'wildignorecase'** applies.

If `{list}` is given and it is **TRUE**, a List will be returned. Otherwise the result is a String and when there are several matches, they are separated by `<NL>` characters. [Note: in version 5.0 a space was used, which caused problems when a

file name contains a space]

If the expansion fails, the result is an empty string. A name for a non-existing file is not included, unless `{string}` does not start with '%', '#', or '<', see below.

For a `:terminal` window '%' expands to a '!' followed by the command or shell that is run `terminal-bufname`

When `{string}` starts with '%', '#', or '<', the expansion is done like for the `cmdline-special` variables with their associated modifiers. Here is a short overview:

<code>%</code>	current file name
<code>#</code>	alternate file name
<code>#n</code>	alternate file name n
<code><cfile></code>	file name under the cursor
<code><afile></code>	autocmd file name
<code><abuf></code>	autocmd buffer number (as a String!)
<code><amatch></code>	autocmd matched name
<code><cexpr></code>	C expression under the cursor
<code><sfile></code>	sourced script file or function name
<code><slnum></code>	sourced script line number or function line number
<code><sflnum></code>	script file line number, also when in a function
<code><SID></code>	"<SNR>123_" where "123" is the current script ID <code><SID></code>
<code><script></code>	sourced script file, or script file where the current function was defined
<code><stack></code>	call stack
<code><word></code>	word under the cursor
<code><cWORD></code>	WORD under the cursor
<code><client></code>	the <code>{clientid}</code> of the last received message <code>server2client()</code>

Modifiers:

<code>:p</code>	expand to full path
<code>:h</code>	head (last path component removed)
<code>:t</code>	tail (last path component only)
<code>:r</code>	root (one extension removed)
<code>:e</code>	extension only

Example:

```
:let &tags = expand("%:p:h") .. "/tags"
```

Note that when expanding a string that starts with '%', '#', or '<', any following text is ignored. This does NOT work:

```
:let doesntwork = expand("%:h.bak")
```

Use this:

```
:let doeswork = expand("%:h") .. ".bak"
```

Also **note** that expanding "`<cfile>`" and others only returns the referenced file name without further expansion. If "`<cfile>`" is "`~/.`cshrc", you need to do another `expand()` to have the "`~/`" expanded into the path of the home directory:

```
:echo expand(expand("<cfile>"))
```

There cannot be white space between the variables and the following modifier. The `fnamemodify()` function can be used to modify normal file names.

When using '%' or '#', and the current or alternate file name is not defined, an empty string is used. Using "%:p" in a buffer with no name, results in the current directory, with a '/' added.

When **'verbose'** is set then expanding '%', '#' and '<>' items will result in an error message if the argument cannot be expanded.

When `{string}` does not start with '%', '#' or '<', it is expanded like a file name is expanded on the command line. **'suffixes'** and **'wildignore'** are used, unless the optional `{nosuf}` argument is given and it is `TRUE`.

Names for non-existing files are included. The "*" item can be used to search in a directory tree. For example, to find all "README" files in the current directory and below:

```
:echo expand("**/README")
```

`expand()` can also be used to expand variables and environment variables that are only known in a shell. But this can be slow, because a shell may be used to do the expansion. See `expr-env-expand`.

The expanded variable is still handled like a list of file names. When an environment variable cannot be expanded, it is left unchanged. Thus `:echo expand('$FOOBAR')` results in `"$FOOBAR"`.

See `glob()` for finding existing files. See `system()` for getting the raw output of an external command.

Can also be used as a `method` :
`Getpattern()->expand()`

```
expandcmd({string} [, {options}]) expandcmd()  
Expand special items in String {string} like what is done for an Ex command such as :edit. This expands special keywords, like with expand(), and environment variables, anywhere in {string}. "~user" and "~/path" are only expanded at the start.
```

The following items are supported in the `{options}` Dict argument:

<code>errmsg</code>	If set to <code>TRUE</code> , error messages are displayed if an error is encountered during expansion. By default, error messages are not displayed.
---------------------	---

Returns the expanded string. If an error is encountered during expansion, the unmodified `{string}` is returned.

Example:

```
:echo expandcmd('make %<.o')
make /path/runtime/doc/builtin.o
:echo expandcmd('make %<.o', {'errmsg': v:true})
```

Can also be used as a `method` :

```
GetCommand()->expandcmd()
```

```
extend({expr1}, {expr2} [, {expr3}]) extend()
{expr1} and {expr2} must be both Lists or both
Dictionaries .
```

If they are `Lists` : Append {expr2} to {expr1}.
If {expr3} is given insert the items of {expr2} before the
item with index {expr3} in {expr1}. When {expr3} is zero
insert before the first item. When {expr3} is equal to
len({expr1}) then {expr2} is appended.

Examples:

```
:echo sort(extend(mylist, [7, 5]))
:call extend(mylist, [2, 3], 1)
```

When {expr1} is the same List as {expr2} then the number of
items copied is equal to the original length of the List.
E.g., when {expr3} is 1 you get N new copies of the first item
(where N is the original length of the List).

Use `add()` to concatenate one item to a list. To concatenate
two lists into a new list use the + operator:

```
:let newlist = [1, 2, 3] + [4, 5]
```

If they are `Dictionaries` :

Add all entries from {expr2} to {expr1}.

If a key exists in both {expr1} and {expr2} then {expr3} is
used to decide what to do:

```
{expr3} = "keep": keep the value of {expr1}
```

```
{expr3} = "force": use the value of {expr2}
```

```
{expr3} = "error": give an error message
```

E737

When {expr3} is omitted then "force" is assumed.

{expr1} is changed when {expr2} is not empty. If necessary
make a copy of {expr1} first.

{expr2} remains unchanged.

When {expr1} is locked and {expr2} is not empty the operation
fails.

Returns {expr1}. Returns 0 on error.

Can also be used as a `method` :

```
mylist->extend(otherlist)
```

```
extendnew({expr1}, {expr2} [, {expr3}]) extendnew()
Like extend() but instead of adding items to {expr1} a new
List or Dictionary is created and returned. {expr1} remains
unchanged.
```

```
feedkeys({string} [, {mode}]) feedkeys()
```

Characters in `{string}` are queued for processing as if they come from a mapping or were typed by the user.

By default the string is added to the end of the typeahead buffer, thus if a mapping is still being executed the characters come after them. Use the 'i' flag to insert before other characters, they will be executed next, before any characters from a mapping.

The function does not wait for processing of keys contained in `{string}`.

To include special keys into `{string}`, use double-quotes and "\..." notation `expr-quote`. For example, `feedkeys("\<CR>")` simulates pressing of the `<Enter>` key. But `feedkeys('\<CR>')` pushes 5 characters. A special code that might be useful is `<Ignore>`, it exits the wait for a character without doing anything. `<Ignore>`

`{mode}` is a String, which can contain these character flags:

- 'm' Remap keys. This is default. If `{mode}` is absent, keys are remapped.
- 'n' Do not remap keys.
- 't' Handle keys as if typed; otherwise they are handled as if coming from a mapping. This matters for undo, opening folds, etc.
- 'L' Lowlevel input. Only works for Unix or when using the GUI. Keys are used as if they were coming from the terminal. Other flags are not used. [E980](#)
When a **CTRL-C** interrupts and 't' is included it sets the internal "got_int" flag.
- 'i' Insert the string instead of appending (see above).
- 'x' Execute commands until typeahead is empty. This is similar to using ":normal!". You can call `feedkeys()` several times without 'x' and then one time with 'x' (possibly with an empty `{string}`) to execute all the typeahead. **Note** that when Vim ends in Insert mode it will behave as if `<Esc>` is typed, to avoid getting stuck, waiting for a character to be typed before the script continues.
Note that if you manage to call `feedkeys()` while executing commands, thus calling it recursively, then all typeahead will be consumed by the last call.
- 'c' Remove any script context when executing, so that legacy script syntax applies, "s:var" does not work, etc. **Note** that if the string being fed sets a script context this still applies.
- !' When used with 'x' will not end Insert mode. Can be used in a test when a timer is set to exit Insert mode a little later. Useful for testing `CursorHoldI`.

Return value is always 0.

Can also be used as a [method](#) :

GetInput()->feedkeys()

filereadable({file})

filereadable()

The result is a Number, which is `TRUE` when a file with the name `{file}` exists, and can be read. If `{file}` doesn't exist, or is a directory, the result is `FALSE`. `{file}` is any expression, which is used as a String.

If you don't care about the file being readable you can use `glob()`.

`{file}` is used as-is, you may want to expand wildcards first:

```
echo filereadable('~/.vimrc')
```

```
0
```

```
echo filereadable(expand('~/.vimrc'))
```

```
1
```

Can also be used as a `method` :

```
GetName()->filereadable()
```

file_readable()

Obsolete name: `file_readable()`.

filewritable({file})

filewritable()

The result is a Number, which is 1 when a file with the name `{file}` exists, and can be written. If `{file}` doesn't exist, or is not writable, the result is 0. If `{file}` is a directory, and we can write to it, the result is 2.

Can also be used as a `method` :

```
GetName()->filewritable()
```

filter({expr1}, {expr2})

filter()

`{expr1}` must be a `List`, `String`, `Blob` or `Dictionary`. For each item in `{expr1}` evaluate `{expr2}` and when the result is zero or false remove the item from the `List` or `Dictionary`. Similarly for each byte in a `Blob` and each character in a `String`.

`{expr2}` must be a `string` or `Funcref`.

If `{expr2}` is a `string`, inside `{expr2}` `v:val` has the value of the current item. For a `Dictionary` `v:key` has the key of the current item and for a `List` `v:key` has the index of the current item. For a `Blob` `v:key` has the index of the current byte. For a `String` `v:key` has the index of the current character.

Examples:

```
call filter(mylist, 'v:val !~ "OLD"')
```

Removes the items where "OLD" appears.

```
call filter(mydict, 'v:key >= 8')
```

Removes the items with a key below 8.

```
call filter(var, 0)
```

Removes all the items, thus clears the `List` or `Dictionary`.

Note that `{expr2}` is the result of expression and is then used as an expression again. Often it is good to use a `literal-string` to avoid having to double backslashes.

If `{expr2}` is a `Funcref` it must take two arguments:

1. the key or the index of the current item.
2. the value of the current item.

The function must return `TRUE` if the item should be kept.

Example that keeps the odd items of a list:

```
func Odd(idx, val)
    return a:idx % 2 == 1
endfunc
call filter(mylist, function('Odd'))
```

It is shorter when using a `lambda`. In `Vim9` syntax:

```
call filter(myList, (idx, val) => idx * val <= 42)
```

In legacy script syntax:

```
call filter(myList, {idx, val -> idx * val <= 42})
```

If you do not use "val" you can leave it out:

```
call filter(myList, {idx -> idx % 2 == 1})
```

In `Vim9` script the result must be true, false, zero or one. Other values will result in a type error.

For a `List` and a `Dictionary` the operation is done in-place. If you want it to remain unmodified make a copy first:

```
:let l = filter(copy(mylist), 'v:val =~ "KEEP"')
```

Returns `{expr1}`, the `List` or `Dictionary` that was filtered, or a new `Blob` or `String`.

When an error is encountered while evaluating `{expr2}` no further items in `{expr1}` are processed.

When `{expr2}` is a `Funcref` errors inside a function are ignored, unless it was defined with the "abort" flag.

Can also be used as a `method` :

```
mylist->filter(expr2)
```

`finddir({name} [, {path} [, {count}]])` `finddir()`

Find directory `{name}` in `{path}`. Supports both downwards and upwards recursive directory searches. See `file-searching` for the syntax of `{path}`.

Returns the path of the first found match. When the found directory is below the current directory a relative path is returned. Otherwise a full path is returned.

If `{path}` is omitted or empty then `'path'` is used.

If the optional `{count}` is given, find `{count}`'s occurrence of `{name}` in `{path}` instead of the first one.

When `{count}` is negative return all the matches in a `List`.

Returns an empty string if the directory is not found.

This is quite similar to the ex-command `:find``.

Can also be used as a `method` :
`GetName()->finddir()`

`findfile({name} [, {path} [, {count}]])` `findfile()`
Just like `finddir()` , but find a file instead of a directory.
Uses `'suffixesadd'`.

Example:

```
:echo findfile("tags.vim", ".;")
```

Searches from the directory of the current file upwards until it finds the file "tags.vim".

Can also be used as a `method` :
`GetName()->findfile()`

`flatten({list} [, {maxdepth}])` `flatten()`
Flatten `{list}` up to `{maxdepth}` levels. Without `{maxdepth}` the result is a `List` without nesting, as if `{maxdepth}` is a very large number.
The `{list}` is changed in place, use `flattennew()` if you do not want that.
In Vim9 script `flatten()` cannot be used, you must always use `flattennew()` .

E900

`{maxdepth}` means how deep in nested lists changes are made.
`{list}` is not modified when `{maxdepth}` is 0.
`{maxdepth}` must be positive number.

If there is an error the number zero is returned.

Example:

```
:echo flatten([1, [2, [3, 4]], 5])  
[1, 2, 3, 4, 5]  
:echo flatten([1, [2, [3, 4]], 5], 1)  
[1, 2, [3, 4], 5]
```

Can also be used as a `method` :
`mylist->flatten()`

`flattennew({list} [, {maxdepth}])` `flattennew()`
Like `flatten()` but first make a copy of `{list}`.

`float2nr({expr})` `float2nr()`
Convert `{expr}` to a Number by omitting the part after the decimal point.
`{expr}` must evaluate to a `Float` or a `Number` .
Returns 0 if `{expr}` is not a `Float` or a `Number` .
When the value of `{expr}` is out of range for a `Number` the result is truncated to `0x7fffffff` or `-0x7fffffff` (or when 64-bit Number support is enabled, `0x7fffffffffffffff` or `-0x7fffffffffffffff`). NaN results in `-0x80000000` (or when 64-bit Number support is enabled, `-0x8000000000000000`).

Examples:

```
echo float2nr(3.95)
3
echo float2nr(-23.45)
-23
echo float2nr(1.0e100)
2147483647 (or 9223372036854775807)
echo float2nr(-1.0e150)
-2147483647 (or -9223372036854775807)
echo float2nr(1.0e-100)
0
```

Can also be used as a `method` :

```
Compute()->float2nr()
```

`floor({expr})`

`floor()`

Return the largest integral value less than or equal to `{expr}` as a `Float` (round down).

`{expr}` must evaluate to a `Float` or a `Number` .

Returns 0.0 if `{expr}` is not a `Float` or a `Number` .

Examples:

```
echo floor(1.856)
1.0
echo floor(-5.456)
-6.0
echo floor(4.0)
4.0
```

Can also be used as a `method` :

```
Compute()->floor()
```

`fmod({expr1}, {expr2})`

`fmod()`

Return the remainder of `{expr1} / {expr2}`, even if the division is not representable. Returns `{expr1} - i * {expr2}` for some integer `i` such that if `{expr2}` is non-zero, the result has the same sign as `{expr1}` and magnitude less than the magnitude of `{expr2}`. If `{expr2}` is zero, the value returned is zero. The value returned is a `Float` .

`{expr1}` and `{expr2}` must evaluate to a `Float` or a `Number` .

Returns 0.0 if `{expr1}` or `{expr2}` is not a `Float` or a

`Number` .

Examples:

```
:echo fmod(12.33, 1.22)
0.13
:echo fmod(-12.33, 1.22)
-0.13
```

Can also be used as a `method` :

```
Compute()->fmod(1.22)
```

`fnameescape({string})`

`fnameescape()`

Escape `{string}` for use as file name command argument. All characters that have a special meaning, such as '%' and '|' are escaped with a backslash.

For most systems the characters escaped are `" \t\n*?[\`$\%#\`" |!<"`. For systems where a backslash appears in a filename, it depends on the value of `'isfname'`. A leading '+' and '>' is also escaped (special after `:edit` and `:write`). And a "-" by itself (special after `:cd`). Returns an empty string on error.

Example:

```
:let fname = '+some str%nge|name'
:exe "edit " .. fnameescape(fname)
```

results in executing:

```
edit \+some\ str\%nge\|name
```

Can also be used as a `method` :

```
GetName()->fnameescape()
```

`fnamemodify({fname}, {mods})` `fnamemodify()`

Modify file name `{fname}` according to `{mods}`. `{mods}` is a string of characters like it is used for file names on the command line. See `filename-modifiers`.

Example:

```
:echo fnamemodify("main.c", ":p:h")
```

results in:

```
/home/user/vim/vim/src
```

If `{mods}` is empty or an unsupported modifier is used then `{fname}` is returned.

When `{fname}` is empty then with `{mods} ":h"` returns ".", so that `:cd`` can be used with it. This is different from `expand('%:h')` without a buffer name, which returns an empty string.

Note: Environment variables don't work in `{fname}`, use `expand()` first then.

Can also be used as a `method` :

```
GetName()->fnamemodify(':p:h')
```

`foldclosed({lnum})` `foldclosed()`

The result is a Number. If the line `{lnum}` is in a closed fold, the result is the number of the first line in that fold. If the line `{lnum}` is not in a closed fold, -1 is returned. `{lnum}` is used like with `getline()`. Thus "." is the current line, "'m" mark m, etc.

Can also be used as a `method` :

```
GetLnum()->foldclosed()
```

`foldclosedend({lnum})` `foldclosedend()`

The result is a Number. If the line `{lnum}` is in a closed fold, the result is the number of the last line in that fold. If the line `{lnum}` is not in a closed fold, -1 is returned. `{lnum}` is used like with `getline()`. Thus "." is the current line, "'m" mark m, etc.

Can also be used as a `method` :
`GetLnum()->foldclosedend()`

`foldlevel({lnum})` `foldlevel()`
The result is a Number, which is the foldlevel of line `{lnum}` in the current buffer. For nested folds the deepest level is returned. If there is no fold at line `{lnum}`, zero is returned. It doesn't matter if the folds are open or closed. When used while updating folds (from `'foldexpr'`) -1 is returned for lines where folds are still to be updated and the foldlevel is unknown. As a special case the level of the previous line is usually available. `{lnum}` is used like with `getline()`. Thus "." is the current line, "'m" mark m, etc.

Can also be used as a `method` :
`GetLnum()->foldlevel()`

`foldtext()` `foldtext()`
Returns a String, to be displayed for a closed fold. This is the default function used for the `'foldtext'` option and should only be called from evaluating `'foldtext'`. It uses the `v:foldstart`, `v:foldend` and `v:folddashes` variables. The returned string looks like this:

```
+-- 45 lines: abcdef
```

The number of leading dashes depends on the foldlevel. The "45" is the number of lines in the fold. "abcdef" is the text in the first non-blank line of the fold. Leading white space, `"/"` or `"/*` and the text from the `'foldmarker'` and `'commentstring'` options is removed.

When used to draw the actual foldtext, the rest of the line will be filled with the fold char from the `'fillchars'` setting.

Returns an empty string when there is no fold.

`{not available when compiled without the |+folding| feature}`

`foldtextresult({lnum})` `foldtextresult()`
Returns the text that is displayed for the closed fold at line `{lnum}`. Evaluates `'foldtext'` in the appropriate context. When there is no closed fold at `{lnum}` an empty string is returned.

`{lnum}` is used like with `getline()`. Thus "." is the current line, "'m" mark m, etc.

Useful when exporting folded text, e.g., to HTML.

`{not available when compiled without the |+folding| feature}`

Can also be used as a `method` :
`GetLnum()->foldtextresult()`

`foreach({expr1}, {expr2})` `foreach()`
`{expr1}` must be a `List`, `String`, `Blob` or `Dictionary`. For each item in `{expr1}` execute `{expr2}`. `{expr1}` is not

modified; its values may be, as with `:lockvar` 1. E741
See `map()` and `filter()` to modify `{expr1}`.

`{expr2}` must be a `string` or `Funcref` .

If `{expr2}` is a `string` , inside `{expr2}` `v:val` has the value of the current item. For a `Dictionary` `v:key` has the key of the current item and for a `List` `v:key` has the index of the current item. For a `Blob` `v:key` has the index of the current byte. For a `String` `v:key` has the index of the current character.

Examples:

```
call foreach(mylist, 'used[v:val] = true')
```

This records the items that are in the `{expr1}` list.

Note that `{expr2}` is the result of expression and is then used as a command. Often it is good to use a `literal-string` to avoid having to double backslashes.

If `{expr2}` is a `Funcref` it must take two arguments:

1. the key or the index of the current item.
2. the value of the current item.

With a legacy script lambda you don't get an error if it only accepts one argument, but with a Vim9 lambda you get "E1106: One argument too many", the number of arguments must match. If the function returns a value, it is ignored.

Returns `{expr1}` in all cases.

When an error is encountered while executing `{expr2}` no further items in `{expr1}` are processed.

When `{expr2}` is a `Funcref` errors inside a function are ignored, unless it was defined with the "abort" flag.

Can also be used as a `method` :

```
mylist->foreach(expr2)
```

`foreground()` `foreground()`
Move the Vim window to the foreground. Useful when sent from a client to a Vim server. `remote_send()`
On Win32 systems this might not work, the OS does not always allow a window to bring itself to the foreground. Use `remote_foreground()` instead.
{only in the Win32, Motif and GTK GUI versions and the Win32 console version}

`fullcommand({name} [, {vim9}])` `fullcommand()`
Get the full command name from a short abbreviated command name; see 20.2 for details on command abbreviations.

The string argument `{name}` may start with a ``:`` and can include a `[range]`, these are skipped and not returned. Returns an empty string if a command doesn't exist, if it's ambiguous (for user-defined commands) or cannot be shortened this way. `vim9-no-shorten`

Without the `{vim9}` argument uses the current script version. If `{vim9}` is present and FALSE then legacy script rules are used. When `{vim9}` is present and TRUE then Vim9 rules are used, e.g. "en" is not a short form of "endif".

For example ``fullcommand('s')``, ``fullcommand('sub')``, ``fullcommand(':%substitute')`` all return "substitute".

Can also be used as a `method` :
`GetName()->fullcommand()`

`funcref()`

`funcref({name} [, {arglist}] [, {dict}])`

Just like `function()` , but the returned Funcref will lookup the function by reference, not by name. This matters when the function `{name}` is redefined later.

Unlike `function()` , `{name}` must be an existing user function. It only works for an autoloading function if it has already been loaded (to avoid mistakenly loading the autoloading script when only intending to use the function name, use `function()` instead). `{name}` cannot be a builtin function. Returns 0 on error.

Can also be used as a `method` :
`GetFuncname()->funcref([arg])`

`function()` `partial` `E700` `E923`

`function({name} [, {arglist}] [, {dict}])`

Return a `Funcref` variable that refers to function `{name}`. `{name}` can be the name of a user defined function or an internal function.

`{name}` can also be a Funcref or a partial. When it is a partial the dict stored in it will be used and the `{dict}` argument is not allowed. E.g.:

```
let FuncWithArg = function(dict.Func, [arg])
let Broken = function(dict.Func, [arg], dict)
```

When using the Funcref the function will be found by `{name}`, also when it was redefined later. Use `funcref()` to keep the same function.

When `{arglist}` or `{dict}` is present this creates a partial. That means the argument list and/or the dictionary is stored in the Funcref and will be used when the Funcref is called.

The arguments are passed to the function in front of other arguments, but after any argument from `method` . Example:

```
func Callback(arg1, arg2, name)
...
let Partial = function('Callback', ['one', 'two'])
...
```

```
    call Partial('name')
Invokes the function as with:
    call Callback('one', 'two', 'name')
```

```
With a method :
    func Callback(one, two, three)
    ...
    let Partial = function('Callback', ['two'])
    ...
    eval 'one' -> Partial('three')
Invokes the function as with:
    call Callback('one', 'two', 'three')
```

The function() call can be nested to add more arguments to the Funcref. The extra arguments are appended to the list of arguments. Example:

```
    func Callback(arg1, arg2, name)
    "...
    let Func = function('Callback', ['one'])
    let Func2 = function(Func, ['two'])
    "...
    call Func2('name')
Invokes the function as with:
    call Callback('one', 'two', 'name')
```

The Dictionary is only useful when calling a "dict" function.

In that case the {dict} is passed in as "self". Example:

```
    function Callback() dict
        echo "called for " .. self.name
    endfunction
    "...
    let context = {"name": "example"}
    let Func = function('Callback', context)
    "...
    call Func()      " will echo: called for example
```

The use of function() is not needed when there are no extra arguments, these two are equivalent, if Callback() is defined as context.Callback():

```
    let Func = function('Callback', context)
    let Func = context.Callback
```

The argument list and the Dictionary can be combined:

```
    function Callback(arg1, count) dict
    "...
    let context = {"name": "example"}
    let Func = function('Callback', ['one'], context)
    "...
    call Func(500)
```

Invokes the function as with:
 call context.Callback('one', 500)

Returns 0 on error.

Can also be used as a method :

GetFuncname()->function([arg])

garbagecollect([{atexit}]) garbagecollect()
Cleanup unused Lists , Dictionaries , Channels and Jobs that have circular references.

There is hardly ever a need to invoke this function, as it is automatically done when Vim runs out of memory or is waiting for the user to press a key after 'updatetime'. Items without circular references are always freed when they become unused. This is useful if you have deleted a very big List and/or Dictionary with circular references in a script that runs for a long time.

When the optional {atexit} argument is one, garbage collection will also be done when exiting Vim, if it wasn't done before. This is useful when checking for memory leaks.

The garbage collection is not done immediately but only when it's safe to perform. This is when waiting for the user to type a character. To force garbage collection immediately use test_garbagecollect_now() .

get({list}, {idx} [, {default}]) get()
Get item {idx} from List {list}. When this item is not available return {default}. Return zero when {default} is omitted.

Preferably used as a method :
mylist->get(idx)

get({blob}, {idx} [, {default}])
Get byte {idx} from Blob {blob}. When this byte is not available return {default}. Return -1 when {default} is omitted.

Preferably used as a method :
myblob->get(idx)

get({dict}, {key} [, {default}])
Get item with key {key} from Dictionary {dict}. When this item is not available return {default}. Return zero when {default} is omitted. Useful example:

```
let val = get(g:, 'var_name', 'default')
```

This gets the value of g:var_name if it exists, and uses 'default' when it does not exist.

Preferably used as a method :
mydict->get(key)

get({func}, {what})
Get item {what} from Funcref {func}. Possible values for {what} are:

- "name" The function name
- "func" The function
- "dict" The dictionary
- "args" The list with arguments

Returns zero on error.

Preferably used as a method :

myfunc->get(what)

getbufinfo()

```
getbufinfo([buf])
getbufinfo([dict])
```

Get information about buffers as a List of Dictionaries.

Without an argument information about all the buffers is returned.

When the argument is a Dictionary only the buffers matching the specified criteria are returned. The following keys can be specified in {dict}:

buflisted	include only listed buffers.
bufloaded	include only loaded buffers.
bufmodified	include only modified buffers.

Otherwise, {buf} specifies a particular buffer to return information for. For the use of {buf}, see [bufname\(\)](#) above. If the buffer is found the returned List has one item. Otherwise the result is an empty list.

Each returned List item is a dictionary with the following entries:

bufnr	Buffer number.
changed	TRUE if the buffer is modified.
changedtick	Number of changes made to the buffer.
command	TRUE if the buffer belongs to the command-line window cmdwin .
hidden	TRUE if the buffer is hidden.
lastused	Timestamp in seconds, like localtime() , when the buffer was last used.
listed	{only with the +viminfo feature} TRUE if the buffer is listed.
lnum	Line number used for the buffer when opened in the current window. Only valid if the buffer has been displayed in the window in the past. If you want the line number of the last known cursor position in a given window, use line() : <code>:echo line('.', {winid})</code>
linecount	Number of lines in the buffer (only valid when loaded)
loaded	TRUE if the buffer is loaded.
name	Full path to the file in the buffer.
signs	List of signs placed in the buffer. Each list item is a dictionary with the following fields: id sign identifier lnum line number name sign name

variables	A reference to the dictionary with buffer-local variables.
windows	List of <code>window-ID</code> s that display this buffer
popups	List of popup <code>window-ID</code> s that display this buffer

Examples:

```

for buf in getbufinfo()
  echo buf.name
endfor
for buf in getbufinfo({'buflisted':1})
  if buf.changed
    ....
  endif
endfor

```

To get buffer-local options use:

```
getbufvar({bufnr}, '&option_name')
```

Can also be used as a `method` :

```
GetBufnr()->getbufinfo()
```

getbufline()

```
getbufline({buf}, {lnum} [, {end}])
```

Return a `List` with the lines starting from `{lnum}` to `{end}` (inclusive) in the buffer `{buf}`. If `{end}` is omitted, a `List` with only the line `{lnum}` is returned. See `'getbufoneline()'` for only getting the line.

For the use of `{buf}`, see `bufname()` above.

For `{lnum}` and `{end}` "\$" can be used for the last line of the buffer. Otherwise a number must be used.

When `{lnum}` is smaller than 1 or bigger than the number of lines in the buffer, an empty `List` is returned.

When `{end}` is greater than the number of lines in the buffer, it is treated as `{end}` is set to the number of lines in the buffer. When `{end}` is before `{lnum}` an empty `List` is returned.

This function works only for loaded buffers. For unloaded and non-existing buffers, an empty `List` is returned.

Example:

```
:let lines = getbufline(bufnr("myfile"), 1, "$")
```

Can also be used as a `method` :

```
GetBufnr()->getbufline(lnum)
```

getbufoneline()

`getbufoneline({buf}, {lnum})`
Just like `getbufline()` but only get one line and return it as a string.

`getbufvar({buf}, {varname} [, {def}])` `getbufvar()`
The result is the value of option or local buffer variable `{varname}` in buffer `{buf}`. Note that the name without "b:" must be used.
The `{varname}` argument is a string.
When `{varname}` is empty returns a `Dictionary` with all the buffer-local variables.
When `{varname}` is equal to "&" returns a `Dictionary` with all the buffer-local options.
Otherwise, when `{varname}` starts with "&" returns the value of a buffer-local option.
This also works for a global or buffer-local option, but it doesn't work for a global variable, window-local variable or window-local option.
For the use of `{buf}`, see `bufname()` above.
When the buffer or variable doesn't exist `{def}` or an empty string is returned, there is no error message.
Examples:

```
:let bufmodified = getbufvar(1, "&mod")
:echo "todo myvar = " .. getbufvar("todo", "myvar")
```

Can also be used as a `method` :
`GetBufnr()->getbufvar(varname)`

`getcellwidths()` `getcellwidths()`
Returns a `List` of cell widths of character ranges overridden by `setcellwidths()`. The format is equal to the argument of `setcellwidths()`. If no character ranges have their cell widths overridden, an empty `List` is returned.

`getchangelist([{buf}])` `getchangelist()`
Returns the `changelist` for the buffer `{buf}`. For the use of `{buf}`, see `bufname()` above. If buffer `{buf}` doesn't exist, an empty list is returned.

The returned list contains two entries: a list with the change locations and the current position in the list. Each entry in the change list is a dictionary with the following entries:

<code>col</code>	column number
<code>coladd</code>	column offset for 'virtualedit'
<code>lnum</code>	line number

If buffer `{buf}` is the current buffer, then the current position refers to the position in the list. For other buffers, it is set to the length of the list.

Can also be used as a `method` :
`GetBufnr()->getchangelist()`

getchar([expr])

getchar()

Get a single character from the user or input stream.
If [expr] is omitted, wait until a character is available.
If [expr] is 0, only get a character when one is available.
Return zero otherwise.
If [expr] is 1, only check if a character is available, it is not consumed. Return zero if no character available.
If you prefer always getting a string use [getcharstr\(\)](#) .

Without [expr] and when [expr] is 0 a whole character or special key is returned. If it is a single character, the result is a Number. Use [nr2char\(\)](#) to convert it to a String. Otherwise a String is returned with the encoded character. For a special key it's a String with a sequence of bytes starting with 0x80 (decimal: 128). This is the same value as the String "\<Key>", e.g., "\<Left>". The returned value is also a String when a modifier (shift, control, alt) was used that is not included in the character.

When [expr] is 0 and Esc is typed, there will be a short delay while Vim waits to see if this is the start of an escape sequence.

When [expr] is 1 only the first byte is returned. For a one-byte character it is the character itself as a number. Use [nr2char\(\)](#) to convert it to a String.

Use [getcharmod\(\)](#) to obtain any additional modifiers.

When the user clicks a mouse button, the mouse event will be returned. The position can then be found in [v:mouse_col](#) , [v:mouse_lnum](#) , [v:mouse_winid](#) and [v:mouse_win](#) . [getmousepos\(\)](#) can also be used. Mouse move events will be ignored.

This example positions the mouse as it would normally happen:

```
let c = getchar()
if c == "\<LeftMouse>" && v:mouse_win > 0
  exe v:mouse_win .. "wincmd w"
  exe v:mouse_lnum
  exe "normal " .. v:mouse_col .. "|"
endif
```

When using bracketed paste only the first character is returned, the rest of the pasted text is dropped.

[xterm-bracketed-paste](#) .

There is no prompt, you will somehow have to make clear to the user that a character has to be typed. The screen is not redrawn, e.g. when resizing the window. When using a popup window it should work better with a [popup-filter](#) .

There is no mapping for the character.

Key codes are replaced, thus when the user presses the [](#) key you get the code for the [](#) key, not the raw character

sequence. Examples:

```
getchar() == "\<Del>"
getchar() == "\<S-Left>"
```

This example redefines "f" to ignore case:

```
:nmap f :call FindChar(<CR>
:function FindChar()
: let c = nr2char(getchar())
: while col('.') < col('$') - 1
:   normal l
:   if getline('.')[col('.') - 1] ==? c
:     break
:   endif
: endwhile
:endfunction
```

You may also receive synthetic characters, such as `<CursorHold>`. Often you will want to ignore this and get another character:

```
:function GetKey()
: let c = getchar()
: while c == "\<CursorHold>"
:   let c = getchar()
: endwhile
: return c
:endfunction
```

`getcharmod()`

`getcharmod()`

The result is a Number which is the state of the modifiers for the last obtained character with `getchar()` or in another way. These values are added together:

```
2      shift
4      control
8      alt (meta)
16     meta (when it's different from ALT)
32     mouse double click
64     mouse triple click
96     mouse quadruple click (== 32 + 64)
128    command (Mac) or super (GTK)
```

Only the modifiers that have not been included in the character itself are obtained. Thus Shift-a results in "A" without a modifier. Returns 0 if no modifiers are used.

`getcharpos({expr})`

`getcharpos()`

Get the position for String `{expr}`. Same as `getpos()` but the column number in the returned List is a character index instead of a byte index.

If `getpos()` returns a very large column number, equal to `v:maxcol`, then `getcharpos()` will return the character index of the last character.

Example:

With the cursor on 'x' in line 5 with text "xxxxx":

```
getcharpos('.') returns [0, 5, 3, 0]
```

`getpos('.')` returns `[0, 5, 7, 0]`

Can also be used as a `method` :
`GetMark()->getcharpos()`

`getcharsearch()` `getcharsearch()`
Return the current character search information as a `{dict}` with the following entries:

<code>char</code>	character previously used for a character search (<code>t</code> , <code>f</code> , <code>T</code> , or <code>F</code>); empty string if no character search has been performed
<code>forward</code>	direction of character search; 1 for forward, 0 for backward
<code>until</code>	type of character search; 1 for a <code>t</code> or <code>T</code> character search, 0 for an <code>f</code> or <code>F</code> character search

This can be useful to always have `;` and `,` search forward/backward regardless of the direction of the previous character search:

```
:nnoemap <expr> ; getcharsearch().forward ? ';' : ','  
:nnoemap <expr> , getcharsearch().forward ? ',' : ';' ;
```

Also see `setcharsearch()` .

`getcharstr([expr])` `getcharstr()`
Get a single character from the user or input stream as a string.
If `[expr]` is omitted, wait until a character is available.
If `[expr]` is 0 or false, only get a character when one is available. Return an empty string otherwise.
If `[expr]` is 1 or true, only check if a character is available, it is not consumed. Return an empty string if no character is available.
Otherwise this works like `getchar()` , except that a number result is converted to a string.

`getcmdcomplttype()` `getcmdcomplttype()`
Return the type of the current command-line completion. Only works when the command line is being edited, thus requires use of `c_CTRL-_e` or `c_CTRL-R_=` .
See `:command-completion` for the return string.
Also see `getcmdtype()` , `setcmdpos()` , `getcmdline()` and `setcmdline()` .
Returns an empty string when completion is not defined.

`getcmdline()` `getcmdline()`
Return the current command-line. Only works when the command line is being edited, thus requires use of `c_CTRL-_e` or `c_CTRL-R_=` .
Example:

```
:cmap <F7> <C-\>eescape(getcmdline(), ' \')<CR>
```

Also see `getcmdtype()` , `getcmdpos()` , `setcmdpos()` and

`setcmdline()` .
Returns an empty string when entering a password or using `inputsecret()` .

`getcmdpos()` `getcmdpos()`
Return the position of the cursor in the command line as a byte count. The first column is 1. Only works when editing the command line, thus requires use of `c_CTRL-_e` or `c_CTRL-R_=` or an expression mapping. Returns 0 otherwise. Also see `getcmdtype()` , `setcmdpos()` , `getcmdline()` and `setcmdline()` .

`getcmdscreenpos()` `getcmdscreenpos()`
Return the screen position of the cursor in the command line as a byte count. The first column is 1. Instead of `getcmdpos()` , it adds the prompt position. Only works when editing the command line, thus requires use of `c_CTRL-_e` or `c_CTRL-R_=` or an expression mapping. Returns 0 otherwise. Also see `getcmdpos()` , `setcmdpos()` , `getcmdline()` and `setcmdline()` .

`getcmdtype()` `getcmdtype()`
Return the current command-line type. Possible return values are:
: normal Ex command
> debug mode command `debug-mode`
/ forward search command
? backward search command
@ `input()` command
- `:insert` or `:append` command
= `i_CTRL-R_=`
Only works when editing the command line, thus requires use of `c_CTRL-_e` or `c_CTRL-R_=` or an expression mapping. Returns an empty string otherwise. Also see `getcmdpos()` , `setcmdpos()` and `getcmdline()` .

`getcmdwintype()` `getcmdwintype()`
Return the current `command-line-window` type. Possible return values are the same as `getcmdtype()` . Returns an empty string when not in the command-line window.

`getcompletion({pat}, {type} [, {filtered}])` `getcompletion()`
Return a list of command-line completion matches. The String `{type}` argument specifies what for. The following completion types are supported:

<code>arglist</code>	file names in argument list
<code>augroup</code>	autocmd groups
<code>buffer</code>	buffer names
<code>behave</code>	<code>:behave</code> suboptions
<code>breakpoint</code>	<code>:breakadd</code> and <code>:breakdel</code> suboptions
<code>color</code>	color schemes

command	Ex command
cmdline	<code>cmdline-completion</code> result
compiler	compilers
cscope	<code>:cscope</code> suboptions
custom,{func}	custom completion, defined via {func}
customlist,{func}	custom completion, defined via {func}
diff_buffer	<code>:diffget</code> and <code>:diffput</code> completion
dir	directory names
environment	environment variable names
event	autocommand events
expression	Vim expression
file	file and directory names
file_in_path	file and directory names in 'path'
filetype	filetype names 'filetype'
function	function name
help	help subjects
highlight	highlight groups
history	<code>:history</code> suboptions
keymap	keyboard mappings
locale	locale names (as output of locale -a)
mapclear	buffer argument
mapping	mapping name
menu	menus
messages	<code>:messages</code> suboptions
option	options
packadd	optional package <code>pack-add</code> names
runtime	<code>:runtime</code> completion
scriptnames	sourced script names <code>:scriptnames</code>
shellcmd	Shell command
sign	<code>:sign</code> suboptions
syntax	syntax file names 'syntax'
syntime	<code>:syntime</code> suboptions
tag	tags
tag_listfiles	tags, file names
user	user names
var	user variables

If {pat} is an empty string, then all the matches are returned. Otherwise only items matching {pat} are returned. See `wildcards` for the use of special characters in {pat}.

If the optional {filtered} flag is set to 1, then `'wildignore'` is applied to filter the results. Otherwise all the matches are returned. The `'wildignorecase'` option always applies.

If the `'wildoptions'` option contains `'fuzzy'`, then fuzzy matching is used to get the completion matches. Otherwise regular expression matching is used. Thus this function follows the user preference, what happens on the command line. If you do not want this you can make `'wildoptions'` empty before calling `getcompletion()` and restore it afterwards.

If {type} is "cmdline", then the `cmdline-completion` result is returned. For example, to complete the possible values after

a `":call"` command:
`echo getcompletion('call ', 'cmdline')`

If there are no matches, an empty list is returned. An invalid value for `{type}` produces an error.

Can also be used as a `method` :
`GetPattern()->getcompletion('color')`

`getcurpos([winid])` `getcurpos()`
Get the position of the cursor. This is like `getpos('.')`, but includes an extra "curswant" item in the list:
`[0, lnum, col, off, curswant]`
The "curswant" number is the preferred column when moving the cursor vertically. After `$` command it will be a very large number equal to `v:maxcol` . Also see `getcursorcharpos()` and `getpos()` .
The first "bufnum" item is always zero. The byte position of the cursor is returned in `'col'`. To get the character position, use `getcursorcharpos()` .

The optional `{winid}` argument can specify the window. It can be the window number or the `window-ID` . The last known cursor position is returned, this may be invalid for the current value of the buffer if it is not the current window. If `{winid}` is invalid a list with zeroes is returned.

This can be used to save and restore the cursor position:

```
let save_cursor = getcurpos()
MoveTheCursorAround
call setpos('.', save_cursor)
```

Note that this only works within the window. See `winrestview()` for restoring more state.

Can also be used as a `method` :
`GetWinid()->getcurpos()`

`getcursorcharpos([winid])` `getcursorcharpos()`
Same as `getcurpos()` but the column number in the returned list is a character index instead of a byte index.

Example:

With the cursor on 'x' in line 3 with text "xxx":

```
getcursorcharpos() returns [0, 3, 2, 0, 3]
getcurpos()       returns [0, 3, 4, 0, 3]
```

Can also be used as a `method` :
`GetWinid()->getcursorcharpos()`

`getcwd([winnr] [, {tabnr}])` `getcwd()`
The result is a String, which is the name of the current

working directory. `'autochdir'` is ignored.

With `{winnr}` return the local current directory of this window in the current tab page. `{winnr}` can be the window number or the `window-ID` .

If `{winnr}` is `-1` return the name of the global working directory. See also `haslocaldir()` .

With `{winnr}` and `{tabnr}` return the local current directory of the window in the specified tab page. If `{winnr}` is `-1` return the working directory of the tabpage.

If `{winnr}` is zero use the current window, if `{tabnr}` is zero use the current tabpage.

Without any arguments, return the actual working directory of the current window.

Return an empty string if the arguments are invalid.

Examples:

```
" Get the working directory of the current window
:echo getcwd()
:echo getcwd(0)
:echo getcwd(0, 0)
" Get the working directory of window 3 in tabpage 2
:echo getcwd(3, 2)
" Get the global working directory
:echo getcwd(-1)
" Get the working directory of tabpage 3
:echo getcwd(-1, 3)
" Get the working directory of current tabpage
:echo getcwd(-1, 0)
```

Can also be used as a `method` :

```
GetWinnr()->getcwd()
```

`getenv({name})`

`getenv()`

Return the value of environment variable `{name}`. The `{name}` argument is a string, without a leading '\$'. Example:

```
myHome = getenv('HOME')
```

When the variable does not exist `v:null` is returned. That is different from a variable set to an empty string, although some systems interpret the empty value as the variable being deleted. See also `expr-env` .

Can also be used as a `method` :

```
GetVarname()->getenv()
```

`getfontname([name])`

`getfontname()`

Without an argument returns the name of the normal font being used. Like what is used for the Normal highlight group `hl-Normal` .

With an argument a check is done whether String `{name}` is a valid font name. If not then an empty string is returned. Otherwise the actual font name is returned, or `{name}` if the

GUI does not support obtaining the real name.
Only works when the GUI is running, thus not in your vimrc or gvimrc file. Use the `GUIEnter` autocommand to use this function just after the GUI has started.
Note that the GTK GUI accepts any font name, thus checking for a valid name does not work.

`getfperm({fname})` `getfperm()`
The result is a String, which is the read, write, and execute permissions of the given file `{fname}`.
If `{fname}` does not exist or its directory cannot be read, an empty string is returned.
The result is of the form "rwxrwxrwx", where each group of "rwx" flags represent, in turn, the permissions of the owner of the file, the group the file belongs to, and other users. If a user does not have a given permission the flag for this is replaced with the string "-". Examples:

```
:echo getfperm("/etc/passwd")  
:echo getfperm(expand("~/vimrc"))
```


This will hopefully (from a security point of view) display the string "rw-r--r--" or even "rw-----".

Can also be used as a `method` :
`GetFilename()->getfperm()`

For setting permissions use `setfperm()` .

`getfsize({fname})` `getfsize()`
The result is a Number, which is the size in bytes of the given file `{fname}`.
If `{fname}` is a directory, 0 is returned.
If the file `{fname}` can't be found, -1 is returned.
If the size of `{fname}` is too big to fit in a Number then -2 is returned.

Can also be used as a `method` :
`GetFilename()->getfsize()`

`getftime({fname})` `getftime()`
The result is a Number, which is the last modification time of the given file `{fname}`. The value is measured as seconds since 1st Jan 1970, and may be passed to `strftime()`. See also `localtime()` and `strftime()` .
If the file `{fname}` can't be found -1 is returned.

Can also be used as a `method` :
`GetFilename()->getftime()`

`getftype({fname})` `getftype()`
The result is a String, which is a description of the kind of file of the given file `{fname}`.
If `{fname}` does not exist an empty string is returned.
Here is a table over different kinds of files and their results:

Normal file	"file"
Directory	"dir"
Symbolic link	"link"
Block device	"bdev"
Character device	"cdev"
Socket	"socket"
FIFO	"fifo"
All other	"other"

Example:

```
getftype("/home")
```

Note that a type such as "link" will only be returned on systems that support it. On some systems only "dir" and "file" are returned. On MS-Windows a symbolic link to a directory returns "dir" instead of "link".

Can also be used as a **method** :

```
GetFilename()->getftype()
```

getimstatus()

getimstatus()

The result is a Number, which is **TRUE** when the IME status is active and **FALSE** otherwise.

See **'imstatusfunc'**.

getjumplist([{winnr} [, {tabnr}]])

getjumplist()

Returns the **jumplist** for the specified window.

Without arguments use the current window.

With {winnr} only use this window in the current tab page.

{winnr} can also be a **window-ID** .

With {winnr} and {tabnr} use the window in the specified tab page. If {winnr} or {tabnr} is invalid, an empty list is returned.

The returned list contains two entries: a list with the jump locations and the last used jump position number in the list. Each entry in the jump location list is a dictionary with the following entries:

bufnr	buffer number
col	column number
coladd	column offset for 'virtualedit'
filename	filename if available
lnum	line number

Can also be used as a **method** :

```
GetWinnr()->getjumplist()
```

getline()

getline({lnum} [, {end}])

Without {end} the result is a String, which is line {lnum} from the current buffer. Example:

```
getline(1)
```

When {lnum} is a String that doesn't start with a digit, **line()** is called to translate the String into a Number. To get the line under the cursor:

```
getline(". ")
```

When `{lnum}` is a number smaller than 1 or bigger than the number of lines in the buffer, an empty string is returned.

When `{end}` is given the result is a `List` where each item is a line from the current buffer in the range `{lnum}` to `{end}`, including line `{end}`.

`{end}` is used in the same way as `{lnum}`.

Non-existing lines are silently omitted.

When `{end}` is before `{lnum}` an empty `List` is returned.

Example:

```
:let start = line('.')
:let end = search("^\$") - 1
:let lines = getline(start, end)
```

Can also be used as a `method` :

```
ComputeLnum()->getline()
```

To get lines from another buffer see `getbufline()` and `getbufoneline()`

```
getloclist({nr} [, {what}])
```

```
getloclist()
```

Returns a `List` with all the entries in the location list for window `{nr}`. `{nr}` can be the window number or the `window-ID`. When `{nr}` is zero the current window is used.

For a location list window, the displayed location list is returned. For an invalid window number `{nr}`, an empty list is returned. Otherwise, same as `getqflist()`.

If the optional `{what}` dictionary argument is supplied, then returns the items listed in `{what}` as a dictionary. Refer to `getqflist()` for the supported items in `{what}`.

In addition to the items supported by `getqflist()` in `{what}`, the following item is supported by `getloclist()` :

<code>filewinid</code>	id of the window used to display files from the location list. This field is applicable only when called from a location list window. See <code>location-list-file-window</code> for more details.
------------------------	--

Returns a `Dictionary` with default values if there is no location list for the window `{nr}`.

Returns an empty Dictionary if window `{nr}` does not exist.

Examples (See also `getqflist-examples`):

```
:echo getloclist(3, {'all': 0})
:echo getloclist(5, {'filewinid': 0})
```

```
getmarklist([ {buf} ])
```

```
getmarklist()
```

Without the `{buf}` argument returns a `List` with information about all the global marks. `mark`

If the optional `{buf}` argument is specified, returns the local marks defined in buffer `{buf}`. For the use of `{buf}`, see `bufname()` . If `{buf}` is invalid, an empty list is returned.

Each item in the returned `List` is a `Dict` with the following:

```
mark  name of the mark prefixed by ""
pos   a List with the position of the mark:
      [bufnum, lnum, col, off]
      Refer to getpos() for more information.
file  file name
```

Refer to `getpos()` for getting information about a specific mark.

Can also be used as a `method` :
`GetBufnr()->getmarklist()`

`getmatches([win])`

`getmatches()`

Returns a `List` with all matches previously defined for the current window by `matchadd()` and the `:match` commands.

`getmatches()` is useful in combination with `setmatches()` , as `setmatches()` can restore a list of matches saved by `getmatches()` .

If `{win}` is specified, use the window with this number or window ID instead of the current window. If `{win}` is invalid, an empty list is returned.

Example:

```
:echo getmatches()
[{'group': 'MyGroup1', 'pattern': 'TODO',
 'priority': 10, 'id': 1}, {'group': 'MyGroup2',
 'pattern': 'FIXME', 'priority': 10, 'id': 2}]
:let m = getmatches()
:call clearmatches()
:echo getmatches()
[]
:call setmatches(m)
:echo getmatches()
[{'group': 'MyGroup1', 'pattern': 'TODO',
 'priority': 10, 'id': 1}, {'group': 'MyGroup2',
 'pattern': 'FIXME', 'priority': 10, 'id': 2}]
:unlet m
```

`getmousepos()`

`getmousepos()`

Returns a `Dictionary` with the last known position of the mouse. This can be used in a mapping for a mouse click or in a filter of a popup window. The items are:

```
screenrow    screen row
screencol    screen column
winid        Window ID of the click
winrow       row inside "winid"
```

wincol	column inside "winid"
line	text line inside "winid"
column	text column inside "winid"
coladd	offset (in screen columns) from the start of the clicked char

All numbers are 1-based.

If not over a window, e.g. when in the command line, then only "screenrow" and "screencol" are valid, the others are zero.

When on the status line below a window or the vertical separator right of a window, the "line" and "column" values are zero.

When the position is after the text then "column" is the length of the text in bytes plus one.

If the mouse is over a popup window then that window is used.

When using `getchar()` the Vim variables `v:mouse_lnum`, `v:mouse_col` and `v:mouse_winid` also provide these values.

`getmouseshape()` `getmouseshape()`
 Returns the name of the currently showing mouse pointer. When the `+mouseshape` feature is not supported or the shape is unknown an empty string is returned. This function is mainly intended for testing.

`getpid()` `getpid()`
 Return a Number which is the process ID of the Vim process. On Unix and MS-Windows this is a unique number, until Vim exits.

`getpos({expr})` `getpos()`
 Get the position for String `{expr}`. For possible values of `{expr}` see `line()`. For getting the cursor position see `getcurpos()`.
 The result is a List with four numbers:
 [bufnum, lnum, col, off]
 "bufnum" is zero, unless a mark like '0 or 'A is used, then it is the buffer number of the mark.
 "lnum" and "col" are the position in the buffer. The first column is 1.
 The "off" number is zero, unless `'virtualedit'` is used. Then it is the offset in screen columns from the start of the character. E.g., a position within a `<Tab>` or after the last character.
Note that for '< and '> Visual mode matters: when it is "V" (visual line mode) the column of '< is zero and the column of '> is a large number equal to `v:maxcol`.
 The column number in the returned List is the byte position within the line. To get the character position in the line, use `getcharpos()`.
 A very large column number equal to `v:maxcol` can be returned,

in which case it means "after the end of the line".
If {expr} is invalid, returns a list with all zeros.
This can be used to save and restore the position of a mark:

```
let save_a_mark = getpos("'a")
...
call setpos("'a", save_a_mark)
```

Also see `getcharpos()` , `getcurpos()` and `setpos()` .

Can also be used as a `method` :
`GetMark()->getpos()`

`getqflist([{what}])`

`getqflist()`

Returns a `List` with all the current quickfix errors. Each list item is a dictionary with these entries:

```
bufnr    number of buffer that has the file name, use
         bufname() to get the name
module   module name
lnum     line number in the buffer (first line is 1)
end_lnum end of line number if the item is multiline
col      column number (first column is 1)
end_col  end of column number if the item has range
vcol     TRUE : "col" is visual column
         FALSE : "col" is byte index
nr       error number
pattern  search pattern used to locate the error
text     description of the error
type     type of the error, 'E', '1', etc.
valid    TRUE : recognized error message
user_data custom data associated with the item, can be
         any type.
```

When there is no error list or it's empty, an empty list is returned. Quickfix list entries with a non-existing buffer number are returned with "bufnr" set to zero (Note: some functions accept buffer number zero for the alternate buffer, you may need to explicitly check for zero).

Useful application: Find pattern matches in multiple files and do something with them:

```
:vimgrep /theword/jg *.c
:for d in getqflist()
:  echo bufname(d.bufnr) ':' d.lnum '=' d.text
:endfor
```

If the optional {what} dictionary argument is supplied, then returns only the items listed in {what} as a dictionary. The following string items are supported in {what}:

```
changedtick  get the total number of changes made
              to the list quickfix-changedtick
context      get the quickfix-context
efm          errorformat to use when parsing "lines". If
              not present, then the 'errorformat' option
```

value is used.

`id` get information for the quickfix list with `quickfix-ID` ; zero means the id for the current list or the list specified by "nr"

`idx` get information for the quickfix entry at this index in the list specified by 'id' or 'nr'. If set to zero, then uses the current entry. See `quickfix-index`

`items` quickfix list entries

`lines` parse a list of lines using 'efm' and return the resulting entries. Only a `List` type is accepted. The current quickfix list is not modified. See `quickfix-parse` .

`nr` get information for this quickfix list; zero means the current quickfix list and "\$" means the last quickfix list

`qfbufnr` number of the buffer displayed in the quickfix window. Returns 0 if the quickfix buffer is not present. See `quickfix-buffer` .

`size` number of entries in the quickfix list

`title` get the list title `quickfix-title`

`winid` get the quickfix `window-ID`

`all` all of the above quickfix properties

Non-string items in `{what}` are ignored. To get the value of a particular item, set it to zero.

If "nr" is not present then the current quickfix list is used. If both "nr" and a non-zero "id" are specified, then the list specified by "id" is used.

To get the number of lists in the quickfix stack, set "nr" to "\$" in `{what}`. The "nr" value in the returned dictionary contains the quickfix stack size.

When "lines" is specified, all the other items except "efm" are ignored. The returned dictionary contains the entry "items" with the list of entries.

The returned dictionary contains the following entries:

`changedtick` total number of changes made to the list `quickfix-changedtick`

`context` quickfix list context. See `quickfix-context`
If not present, set to "".

`id` quickfix list ID `quickfix-ID` . If not present, set to 0.

`idx` index of the quickfix entry in the list. If not present, set to 0.

`items` quickfix list entries. If not present, set to an empty list.

`nr` quickfix list number. If not present, set to 0

`qfbufnr` number of the buffer displayed in the quickfix window. If not present, set to 0.

`size` number of entries in the quickfix list. If not present, set to 0.

`title` quickfix list title text. If not present, set to "".

`winid` quickfix `window-ID` . If not present, set to 0

Examples (See also [getqflist-examples](#)):

```
:echo getqflist({'all': 1})
:echo getqflist({'nr': 2, 'title': 1})
:echo getqflist({'lines' : ["F1:10:L10"]})
```

`getreg([{regname} [, 1 [, {list}]]])` `getreg()`

The result is a String, which is the contents of register `{regname}`. Example:

```
:let cliptext = getreg('*')
```

When register `{regname}` was not set the result is an empty string.

The `{regname}` argument must be a string. E1162

`getreg('=')` returns the last evaluated value of the expression register. (For use in maps.)

`getreg(=', 1)` returns the expression itself, so that it can be restored with `setreg()` . For other registers the extra argument is ignored, thus you can always give it.

If `{list}` is present and `TRUE` , the result type is changed to `List` . Each list item is one text line. Use it if you care about zero bytes possibly present inside register: without third argument both NLs and zero bytes are represented as NLs (see [NL-used-for-Nul](#)).

When the register was not set an empty list is returned.

If `{regname}` is "", the unnamed register "" is used.

If `{regname}` is not specified, `v:register` is used.

In [Vim9-script](#) `{regname}` must be one character.

Can also be used as a [method](#) :

```
GetRegname()->getreg()
```

`getreginfo([{regname}])` `getreginfo()`

Returns detailed information about register `{regname}` as a Dictionary with the following entries:

<code>regcontents</code>	List of lines contained in register <code>{regname}</code> , like <code>getreg ({regname}, 1, 1)</code> .
<code>regtype</code>	the type of register <code>{regname}</code> , as in <code>getregtype()</code> .
<code>isunnamed</code>	Boolean flag, <code>v:true</code> if this register is currently pointed to by the unnamed register.
<code>points_to</code>	for the unnamed register, gives the single letter name of the register currently pointed to (see quotequote). For example, after deleting a line with <code>`dd`</code> , this field will be "1", which is the register that got the deleted text.

The `{regname}` argument is a string. If `{regname}` is invalid

or not set, an empty Dictionary will be returned.
 If `{regname}` is "" or "@", the unnamed register "" is used.
 If `{regname}` is not specified, `v:register` is used.
 The returned Dictionary can be passed to `setreg()` .
 In `Vim9-script` `{regname}` must be one character.

Can also be used as a `method` :
`GetRegname()->getreginfo()`

`getregion({pos1}, {pos2} [, {opts}])` `getregion()`
 Returns the list of strings from `{pos1}` to `{pos2}` from a buffer.

`{pos1}` and `{pos2}` must both be `List` s with four numbers.
 See `getpos()` for the format of the list. It's possible to specify positions from a different buffer, but please [note](#) the limitations at `getregion-notes` .

The optional argument `{opts}` is a Dict and supports the following items:

<code>type</code>	Specify the region's selection type (default: "v"):
<code>"v"</code>	for <code>characterwise</code> mode
<code>"V"</code>	for <code>linewise</code> mode
<code>"<CTRL-V>"</code>	for <code>blockwise-visual</code> mode
<code>exclusive</code>	If <code>TRUE</code> , use exclusive selection for the end position (default: follow <code>'selection'</code>)

You can get the last selection type by `visualmode()` .
 If Visual mode is active, use `mode()` to get the Visual mode (e.g., in a `:vmap`).
 This function is useful to get text starting and ending in different columns, such as a `characterwise-visual` selection.

`getregion-notes`

Note that:

- Order of `{pos1}` and `{pos2}` doesn't matter, it will always return content from the upper left position to the lower right position.
- If `'virtualedit'` is enabled and the region is past the end of the lines, resulting lines are padded with spaces.
- If the region is blockwise and it starts or ends in the middle of a multi-cell character, it is not included but its selected part is substituted with spaces.
- If `{pos1}` and `{pos2}` are not in the same buffer, an empty list is returned.
- `{pos1}` and `{pos2}` must belong to a `bufloaded()` buffer.
- It is evaluated in current window context, which makes a difference if the buffer is displayed in a window with different `'virtualedit'` or `'list'` values.

Examples:

```
:xnoremap <CR>
\ <Cmd>echow getregion(
\ getpos('v'), getpos('.'), #{ type: mode() })<CR>
```

Can also be used as a `method` :

```
getpos('.')->getregion(getpos("'a"))
```

`getregtype([{{regname}}])`

`getregtype()`

The result is a String, which is type of register `{{regname}}`.

The value will be one of:

```
"v"           for characterwise text
"V"           for linewise text
"<CTRL-V>{{width}}" for blockwise-visual text
""           for an empty or unknown register
```

`<CTRL-V>` is one character with value `0x16`.

The `{{regname}}` argument is a string. If `{{regname}}` is "", the unnamed register "" is used. If `{{regname}}` is not specified, `v:register` is used.

In `Vim9-script` `{{regname}}` must be one character.

Can also be used as a `method` :

```
GetRegname()->getregtype()
```

`getscriptinfo([{{opts}}])`

`getscriptinfo()`

Returns a `List` with information about all the sourced Vim scripts in the order they were sourced, like what `:scriptnames` shows.

The optional Dict argument `{{opts}}` supports the following optional items:

```
name      Script name match pattern. If specified,
          and "sid" is not specified, information about
          scripts with a name that match the pattern
          "name" are returned.
sid       Script ID <SID> . If specified, only
          information about the script with ID "sid" is
          returned and "name" is ignored.
```

Each item in the returned List is a `Dict` with the following items:

```
autoload  Set to TRUE for a script that was used with
          `import autoload` but was not actually sourced
          yet (see import-autoload ).
functions List of script-local function names defined in
          the script. Present only when a particular
          script is specified using the "sid" item in
          {{opts}}.
name      Vim script file name.
sid       Script ID <SID> .
sourced   Script ID of the actually sourced script that
          this script name links to, if any, otherwise
          zero
variables A dictionary with the script-local variables.
```

Present only when a particular script is specified using the "sid" item in {opts}.
Note that this is a copy, the value of script-local variables cannot be changed using this dictionary.

version Vim script version (`scriptversion`)

Examples:

```
:echo getscripinfo({'name': 'myscript'})  
:echo getscripinfo({'sid': 15}).variables
```

`gettabinfo([{tabnr}])`

`gettabinfo()`

If {tabnr} is not specified, then information about all the tab pages is returned as a `List`. Each List item is a `Dictionary`. Otherwise, {tabnr} specifies the tab page number and information about that one is returned. If the tab page does not exist an empty List is returned.

Each List item is a `Dictionary` with the following entries:

tabnr	tab page number.
variables	a reference to the dictionary with tabpage-local variables
windows	List of <code>window-ID</code> s in the tab page.

Can also be used as a `method` :

`GetTabnr()->gettabinfo()`

`gettabvar({tabnr}, {varname} [, {def}])`

`gettabvar()`

Get the value of a tab-local variable {varname} in tab page {tabnr}. `t:var`

Tabs are numbered starting with one.

The {varname} argument is a string. When {varname} is empty a dictionary with all tab-local variables is returned.

Note that the name without "t:" must be used.

When the tab or variable doesn't exist {def} or an empty string is returned, there is no error message.

Can also be used as a `method` :

`GetTabnr()->gettabvar(varname)`

`gettabwinvar({tabnr}, {winnr}, {varname} [, {def}])`

`gettabwinvar()`

Get the value of window-local variable {varname} in window {winnr} in tab page {tabnr}.

The {varname} argument is a string. When {varname} is empty a dictionary with all window-local variables is returned.

When {varname} is equal to "&" get the values of all window-local options in a `Dictionary`.

Otherwise, when {varname} starts with "&" get the value of a window-local option.

Note that {varname} must be the name without "w:".

Tabs are numbered starting with one. For the current tabpage use `getwinvar()`.

{winnr} can be the window number or the `window-ID`.

When {winnr} is zero the current window is used.

This also works for a global option, buffer-local option and window-local option, but it doesn't work for a global variable or buffer-local variable.

When the tab, window or variable doesn't exist `{def}` or an empty string is returned, there is no error message.

Examples:

```
:let list_is_on = gettabwinvar(1, 2, '&list')
:echo "myvar = " .. gettabwinvar(3, 1, 'myvar')
```

To obtain all window-local variables use:

```
gettabwinvar({tabnr}, {winnr}, '&')
```

Can also be used as a [method](#) :

```
GetTabnr()->gettabwinvar(winnr, varname)
```

`gettagstack([{winnr}])`

`gettagstack()`

The result is a Dict, which is the tag stack of window `{winnr}`.

`{winnr}` can be the window number or the [window-ID](#) .

When `{winnr}` is not specified, the current window is used.

When window `{winnr}` doesn't exist, an empty Dict is returned.

The returned dictionary contains the following entries:

<code>curidx</code>	Current index in the stack. When at top of the stack, set to (length + 1). Index of bottom of the stack is 1.
<code>items</code>	List of items in the stack. Each item is a dictionary containing the entries described below.
<code>length</code>	Number of entries in the stack.

Each item in the stack is a dictionary with the following entries:

<code>bufnr</code>	buffer number of the current jump
<code>from</code>	cursor position before the tag jump. See getpos() for the format of the returned list.
<code>matchnr</code>	current matching tag number. Used when multiple matching tags are found for a name.
<code>tagname</code>	name of the tag

See [tagstack](#) for more information about the tag stack.

Can also be used as a [method](#) :

```
GetWinnr()->gettagstack()
```

`gettext({text})`

`gettext()`

Translate String `{text}` if possible.

This is mainly for use in the distributed Vim scripts. When generating message translations the `{text}` is extracted by `xgettext`, the translator can add the translated message in the `.po` file and Vim will lookup the translation when `gettext()` is called.

For `{text}` double quoted strings are preferred, because `xgettext` does not understand escaping in single quoted strings.

`getwininfo([winid])`

`getwininfo()`

Returns information about windows as a `List` with Dictionaries.

If `{winid}` is given Information about the window with that ID is returned, as a `List` with one item. If the window does not exist the result is an empty list.

Without `{winid}` information about all the windows in all the tab pages is returned.

Each List item is a `Dictionary` with the following entries:

<code>botline</code>	last complete displayed buffer line
<code>bufnr</code>	number of buffer in the window
<code>height</code>	window height (excluding winbar)
<code>loclist</code>	1 if showing a location list {only with the <code>+quickfix</code> feature}
<code>quickfix</code>	1 if quickfix or location list window {only with the <code>+quickfix</code> feature}
<code>terminal</code>	1 if a terminal window {only with the <code>+terminal</code> feature}
<code>tabnr</code>	tab page number
<code>topline</code>	first displayed buffer line
<code>variables</code>	a reference to the dictionary with window-local variables
<code>width</code>	window width
<code>winbar</code>	1 if the window has a toolbar, 0 otherwise
<code>wincol</code>	leftmost screen column of the window; "col" from <code>win_screenpos()</code>
<code>textoff</code>	number of columns occupied by any ' <code>foldcolumn</code> ', ' <code>signcolumn</code> ' and line number in front of the text
<code>winid</code>	<code>window-ID</code>
<code>winnr</code>	window number
<code>winrow</code>	topmost screen line of the window; "row" from <code>win_screenpos()</code>

Can also be used as a `method` :

`GetWinnr()->getwininfo()`

`getwinpos([timeout])`

`getwinpos()`

The result is a `List` with two numbers, the result of `getwinposx()` and `getwinposy()` combined:

`[x-pos, y-pos]`

`{timeout}` can be used to specify how long to wait in msec for a response from the terminal. When omitted 100 msec is used. Use a longer time for a remote terminal.

When using a value less than 10 and no response is received within that time, a previously reported position is returned,

if available. This can be used to poll for the position and do some work in the meantime:

```
while 1
  let res = getwinpos(1)
  if res[0] >= 0
    break
  endif
  " Do some work here
endwhile
```

Can also be used as a `method` :

```
GetTimeout()->getwinpos()
```

`getwinposx()` `getwinposx()`
The result is a Number, which is the X coordinate in pixels of the left hand side of the GUI Vim window. Also works for an xterm (uses a timeout of 100 msec). The result will be -1 if the information is not available (e.g. on the Wayland backend). The value can be used with `:winpos``.

`getwinposy()` `getwinposy()`
The result is a Number, which is the Y coordinate in pixels of the top of the GUI Vim window. Also works for an xterm (uses a timeout of 100 msec). The result will be -1 if the information is not available (e.g. on the Wayland backend). The value can be used with `:winpos``.

`getwinvar({winnr}, {varname} [, {def}])` `getwinvar()`
Like `gettabwinvar()` for the current tabpage.
Examples:

```
:let list_is_on = getwinvar(2, '&list')
:echo "myvar = " .. getwinvar(1, 'myvar')
```

Can also be used as a `method` :

```
GetWinnr()->getwinvar(varname)
```

`glob({expr} [, {nosuf} [, {list} [, {alllinks}]]])` `glob()`
Expand the file wildcards in `{expr}`. See `wildcards` for the use of special characters.

Unless the optional `{nosuf}` argument is given and is `TRUE`, the `'suffixes'` and `'wildignore'` options apply: Names matching one of the patterns in `'wildignore'` will be skipped and `'suffixes'` affect the ordering of matches. `'wildignorecase'` always applies.

When `{list}` is present and it is `TRUE` the result is a `List` with all matching files. The advantage of using a `List` is, you also get filenames containing newlines correctly. Otherwise the result is a `String` and when there are several matches, they are separated by `<NL>` characters.

If the expansion fails, the result is an empty String or List.

You can also use `readdir()` if you need to do complicated things, such as limiting the number of matches.

A name for a non-existing file is not included. A symbolic link is only included if it points to an existing file. However, when the `{alllinks}` argument is present and it is `TRUE` then all symbolic links are included.

For most systems backticks can be used to get files names from any external command. Example:

```
:let tagfiles = glob("`find . -name tags -print`")
:let &tags = substitute(tagfiles, "\n", ",", "g")
```

The result of the program inside the backticks should be one item per line. Spaces inside an item are allowed.

See `expand()` for expanding special Vim variables. See `system()` for getting the raw output of an external command.

Can also be used as a `method` :

```
GetExpr()->glob()
```

`glob2regpat({string})`

`glob2regpat()`

Convert a file pattern, as used by `glob()`, into a search pattern. The result can be used to match with a string that is a file name. E.g.

```
if filename =~ glob2regpat('Make*.mak')
```

This is equivalent to:

```
if filename =~ '^Make.*\mak$'
```

When `{string}` is an empty string the result is `^$`, match an empty string.

Note that the result depends on the system. On MS-Windows a backslash usually means a path separator.

Can also be used as a `method` :

```
GetExpr()->glob2regpat()
```

`globpath()`

`globpath({path}, {expr} [, {nosuf} [, {list} [, {alllinks}]]])`

Perform `glob()` for String `{expr}` on all directories in `{path}` and concatenate the results. Example:

```
:echo globpath(&rtp, "syntax/c.vim")
```

`{path}` is a comma-separated list of directory names. Each directory name is prepended to `{expr}` and expanded like with `glob()`. A path separator is inserted when needed.

To add a comma inside a directory name escape it with a backslash. **Note** that on MS-Windows a directory may have a trailing backslash, remove it if you put a comma after it. If the expansion fails for one of the directories, there is no error message.

Unless the optional `{nosuf}` argument is given and is `TRUE`,

the **'suffixes'** and **'wildignore'** options apply: Names matching one of the patterns in **'wildignore'** will be skipped and **'suffixes'** affect the ordering of matches.

When `{list}` is present and it is `TRUE` the result is a `List` with all matching files. The advantage of using a `List` is, you also get filenames containing newlines correctly. Otherwise the result is a `String` and when there are several matches, they are separated by `<NL>` characters. Example:

```
:echo globpath(&rtp, "syntax/c.vim", 0, 1)
```

`{alllinks}` is used as with `glob()` .

The `"**"` item can be used to search in a directory tree. For example, to find all `"README.txt"` files in the directories in **'runtimepath'** and below:

```
:echo globpath(&rtp, "**/README.txt")
```

Upwards search and limiting the depth of `"**"` is not supported, thus using **'path'** will not always work properly.

Can also be used as a `method` , the base is passed as the second argument:

```
GetExpr()->globpath(&rtp)
```

`has()`

`has({feature} [, {check}])`

When `{check}` is omitted or is zero: The result is a `Number`, which is 1 if the feature `{feature}` is supported, zero otherwise. The `{feature}` argument is a string, case is ignored. See `feature-list` below.

When `{check}` is present and not zero: The result is a `Number`, which is 1 if the feature `{feature}` could ever be supported, zero otherwise. This is useful to check for a typo in `{feature}` and to detect dead code. Keep in mind that an older `Vim version` will not know about a feature added later and features that have been abandoned will not be known by the current `Vim version`.

Also see `exists()` and `exists_compiled()` .

Note that to skip code that has a syntax error when the feature is not available, `Vim` may skip the rest of the line and miss a following ``endif``. Therefore put the ``endif`` on a separate line:

```
if has('feature')
  let x = this->breaks->without->the->feature
endif
```

If the ``endif`` would be moved to the second line as `"| endif"` it would not be found.

`has_key({dict}, {key})`

`has_key()`

The result is a `Number`, which is `TRUE` if `Dictionary {dict}`

has an entry with key `{key}`. FALSE otherwise.
The `{key}` argument is a string. In Vim9 script a number is also accepted (and converted to a string) but no other types. In legacy script the usual automatic conversion to string is done.

Can also be used as a `method` :
`mydict->has_key(key)`

```
haslocaldir([{winnr} [, {tabnr}]]]) haslocaldir()  
The result is a Number:  
1 when the window has set a local directory via :lcd  
2 when the tab-page has set a local directory via :tcd  
0 otherwise.
```

Without arguments use the current window.
With `{winnr}` use this window in the current tab page.
With `{winnr}` and `{tabnr}` use the window in the specified tab page.
`{winnr}` can be the window number or the `window-ID` .
If `{winnr}` is -1 it is ignored and only the tabpage is used.
Return 0 if the arguments are invalid.

Examples:

```
if haslocaldir() == 1  
  " window local directory case  
elseif haslocaldir() == 2  
  " tab-local directory case  
else  
  " global directory case  
endif  
  
" current window  
:echo haslocaldir()  
:echo haslocaldir(0)  
:echo haslocaldir(0, 0)  
" window n in current tab page  
:echo haslocaldir(n)  
:echo haslocaldir(n, 0)  
" window n in tab page m  
:echo haslocaldir(n, m)  
" tab page m  
:echo haslocaldir(-1, m)
```

Can also be used as a `method` :
`GetWinnr()->haslocaldir()`

```
hasmapto({what} [, {mode} [, {abbr}]]]) hasmapto()  
The result is a Number, which is TRUE if there is a mapping that contains {what} in somewhere in the rhs (what it is mapped to) and this mapping exists in one of the modes indicated by {mode}.  
The arguments {what} and {mode} are strings.  
When {abbr} is there and it is TRUE use abbreviations instead of mappings. Don't forget to specify Insert and/or
```

Command-line mode.

Both the global mappings and the mappings local to the current buffer are checked for a match.

If no matching mapping is found FALSE is returned.

The following characters are recognized in `{mode}`:

n	Normal mode
v	Visual and Select mode
x	Visual mode
s	Select mode
o	Operator-pending mode
i	Insert mode
l	Language-Argument ("r", "f", "t", etc.)
c	Command-line mode

When `{mode}` is omitted, "nvo" is used.

This function is useful to check if a mapping already exists to a function in a Vim script. Example:

```
:if !hasmapto('\ABCdoit')
:  map <Leader>d \ABCdoit
:endif
```

This installs the mapping to "\ABCdoit" only if there isn't already a mapping to "\ABCdoit".

Can also be used as a `method` :

```
GetRHS()->hasmapto()
```

`histadd({history}, {item})`

`histadd()`

Add the String `{item}` to the history `{history}` which can be one of:

`hist-names`

"cmd"	or ":"	command line history
"search"	or "/"	search pattern history
"expr"	or "="	typed expression history
"input"	or "@"	input line history
"debug"	or ">"	debug command history
empty		the current or last used history

The `{history}` string does not need to be the whole name, one character is sufficient.

If `{item}` does already exist in the history, it will be shifted to become the newest entry.

The result is a Number: TRUE if the operation was successful, otherwise FALSE is returned.

Example:

```
:call histadd("input", strftime("%Y %b %d"))
:let date=input("Enter date: ")
```

This function is not available in the `sandbox` .

Can also be used as a `method` , the base is passed as the second argument:

```
GetHistory()->histadd('search')
```

`histdel({history} [, {item}])`

`histdel()`

Clear `{history}`, i.e. delete all its entries. See `hist-names` for the possible values of `{history}`.

If the parameter `{item}` evaluates to a String, it is used as a regular expression. All entries matching that expression will be removed from the history (if there are any).

Upper/lowercase must match, unless `"\c"` is used `/\c`.

If `{item}` evaluates to a Number, it will be interpreted as an index, see `:history-indexing`. The respective entry will be removed if it exists.

The result is TRUE for a successful operation, otherwise FALSE is returned.

Examples:

Clear expression register history:

```
:call histdel("expr")
```

Remove all entries starting with "*" from the search history:

```
:call histdel("/", '^\\*')
```

The following three are equivalent:

```
:call histdel("search", histnr("search"))
```

```
:call histdel("search", -1)
```

```
:call histdel("search", '^' .. histget("search", -1) .. '$')
```

To delete the last search pattern and use the last-but-one for the "n" command and `'hlsearch'`:

```
:call histdel("search", -1)
```

```
:let @/ = histget("search", -1)
```

Can also be used as a `method` :

```
GetHistory()->histdel()
```

`histget({history} [, {index}])` `histget()`

The result is a String, the entry with Number `{index}` from `{history}`. See `hist-names` for the possible values of `{history}`, and `:history-indexing` for `{index}`. If there is no such entry, an empty String is returned. When `{index}` is omitted, the most recent item from the history is used.

Examples:

Redo the second last search from history.

```
:execute '/' .. histget("search", -2)
```

Define an Ex command `":H {num}"` that supports re-execution of the `{num}`th entry from the output of `:history`.

```
:command -nargs=1 H execute histget("cmd", 0+<args>)
```

Can also be used as a `method` :

```
GetHistory()->histget()
```

`histnr({history})` `histnr()`

The result is the Number of the current entry in `{history}`.

See `hist-names` for the possible values of `{history}`.

If an error occurred, -1 is returned.

Example:

```
:let inp_index = histnr("expr")
```

Can also be used as a `method` :
`GetHistory()->histnr()`

`hlexists({name})`

`hlexists()`

The result is a Number, which is TRUE if a highlight group called `{name}` exists. This is when the group has been defined in some way. Not necessarily when highlighting has been defined for it, it may also have been used for a syntax item.

`highlight_exists()`

Obsolete name: `highlight_exists()`.

Can also be used as a `method` :
`GetName()->hlexists()`

`hlget([{name} [, {resolve}]])`

`hlget()`

Returns a List of all the highlight group attributes. If the optional `{name}` is specified, then returns a List with only the attributes of the specified highlight group. Returns an empty List if the highlight group `{name}` is not present.

If the optional `{resolve}` argument is set to `v:true` and the highlight group `{name}` is linked to another group, then the link is resolved recursively and the attributes of the resolved highlight group are returned.

Each entry in the returned List is a Dictionary with the following items:

- `cleared` boolean flag, set to `v:true` if the highlight group attributes are cleared or not yet specified. See `highlight-clear`.
- `cterm` cterm attributes. See `highlight-cterm`.
- `ctermbg` cterm background color. See `highlight-ctermbg`.
- `ctermfg` cterm foreground color. See `highlight-ctermfg`.
- `ctermul` cterm underline color. See `highlight-ctermul`.
- `default` boolean flag, set to `v:true` if the highlight group link is a default link. See `highlight-default`.
- `font` highlight group font. See `highlight-font`.
- `gui` gui attributes. See `highlight-gui`.
- `guibg` gui background color. See `highlight-guibg`.
- `guifg` gui foreground color. See `highlight-guifg`.
- `guisp` gui special color. See `highlight-guisp`.
- `id` highlight group ID.
- `linksto` linked highlight group name. See `:highlight-link`.
- `name` highlight group name. See `group-name`.
- `start` start terminal keycode. See `highlight-start`.

stop stop terminal keycode. See [highlight-stop](#) .
term term attributes. See [highlight-term](#) .

The **'term'**, **'cterm'** and **'gui'** items in the above Dictionary have a dictionary value with the following optional boolean items: **'bold'**, **'standout'**, **'underline'**, **'undercurl'**, **'italic'**, **'reverse'**, **'inverse'** and **'strikethrough'**.

Example(s):

```
:echo hlget()  
:echo hlget('ModeMsg')  
:echo hlget('Number', v:true)
```

Can also be used as a [method](#) :

```
GetName()->hlget()
```

hlset({list})

hlset()

Creates or modifies the attributes of a List of highlight groups. Each item in `{list}` is a dictionary containing the attributes of a highlight group. See [hlget\(\)](#) for the list of supported items in this dictionary.

In addition to the items described in [hlget\(\)](#) , the following additional items are supported in the dictionary:

force boolean flag to force the creation of a link for an existing highlight group with attributes.

The highlight group is identified using the **'name'** item and the **'id'** item (if supplied) is ignored. If a highlight group with a specified name doesn't exist, then it is created. Otherwise the attributes of an existing highlight group are modified.

If an empty dictionary value is used for the **'term'** or **'cterm'** or **'gui'** entries, then the corresponding attributes are cleared. If the **'cleared'** item is set to `v:true`, then all the attributes of the highlight group are cleared.

The **'linksto'** item can be used to link a highlight group to another highlight group. See [:highlight-link](#) .

Returns zero for success, -1 for failure.

Example(s):

```
" add bold attribute to the Visual highlight group  
:call hlset([#{name: 'Visual',  
              \ term: #{reverse: 1 , bold: 1}}])  
:call hlset([#{name: 'Type', guifg: 'DarkGreen'}])  
:let l = hlget()  
:call hlset(l)  
" clear the Search highlight group  
:call hlset([#{name: 'Search', cleared: v:true}])
```



```

" clear the 'term' attributes for a highlight group
:call hlset([#{name: 'Title', term: {}}])
" create the MyHlg group linking it to DiffAdd
:call hlset([#{name: 'MyHlg', linksto: 'DiffAdd'}])
" remove the MyHlg group link
:call hlset([#{name: 'MyHlg', linksto: 'NONE'}])
" clear the attributes and a link
:call hlset([#{name: 'MyHlg', cleared: v:true,
\ linksto: 'NONE'}])

```

Can also be used as a `method` :
`GetAttrList()->hlset()`

`hlID({name})` `hlID()`
The result is a Number, which is the ID of the highlight group with name `{name}`. When the highlight group doesn't exist, zero is returned.
This can be used to retrieve information about the highlight group. For example, to get the background color of the "Comment" group:

```

:echo synIDattr(synIDtrans(hlID("Comment")), "bg")
highlightID()

```

Obsolete name: `highlightID()`.

Can also be used as a `method` :
`GetName()->hlID()`

`hostname()` `hostname()`
The result is a String, which is the name of the machine on which Vim is currently running. Machine names greater than 256 characters long are truncated.

`iconv({string}, {from}, {to})` `iconv()`
The result is a String, which is the text `{string}` converted from encoding `{from}` to encoding `{to}`.
When the conversion completely fails an empty string is returned. When some characters could not be converted they are replaced with "?".
The encoding names are whatever the `iconv()` library function can accept, see `:!man 3 iconv`.
Most conversions require Vim to be compiled with the `+iconv` feature. Otherwise only UTF-8 to latin1 conversion and back can be done.
This can be used to display messages with special characters, no matter what `'encoding'` is set to. Write the message in UTF-8 and use:

```

echo iconv(utf8_str, "utf-8", &enc)

```

Note that Vim uses UTF-8 for all Unicode encodings, conversion from/to UCS-2 is automatically changed to use UTF-8. You cannot use UCS-2 in a string anyway, because of the NUL bytes.

Can also be used as a `method` :
`GetText()->iconv('latin1', 'utf-8')`

`indent({lnum})` `indent()`
The result is a Number, which is indent of line `{lnum}` in the current buffer. The indent is counted in spaces, the value of `'tabstop'` is relevant. `{lnum}` is used just like in `getline()`.
When `{lnum}` is invalid -1 is returned. In Vim9 script an error is given.

Can also be used as a method :
`GetLnum()->indent()`

`index({object}, {expr} [, {start} [, {ic}]])` `index()`
Find `{expr}` in `{object}` and return its index. See `indexof()` for using a lambda to select the item.

If `{object}` is a `List` return the lowest index where the item has a value equal to `{expr}`. There is no automatic conversion, so the String "4" is different from the Number 4. And the number 4 is different from the Float 4.0. The value of `'ignorecase'` is not used here, case matters as indicated by the `{ic}` argument.

If `{object}` is a `Blob` return the lowest index where the byte value is equal to `{expr}`.

If `{start}` is given then start looking at the item with index `{start}` (may be negative for an item relative to the end).

When `{ic}` is given and it is `TRUE`, ignore case. Otherwise case must match.

-1 is returned when `{expr}` is not found in `{object}`.

Example:

```
:let idx = index(words, "the")  
:if index(numbers, 123) >= 0
```

Can also be used as a method :
`GetObject()->index(what)`

`indexof({object}, {expr} [, {opts}])` `indexof()`
Returns the index of an item in `{object}` where `{expr}` is v:true. `{object}` must be a `List` or a `Blob`.

If `{object}` is a `List`, evaluate `{expr}` for each item in the List until the expression is v:true and return the index of this item.

If `{object}` is a `Blob` evaluate `{expr}` for each byte in the Blob until the expression is v:true and return the index of this byte.

`{expr}` must be a `string` or `Funcref`.

If `{expr}` is a `string`: If `{object}` is a `List`, inside

`{expr}` `v:key` has the index of the current List item and `v:val` has the value of the item. If `{object}` is a `Blob`, inside `{expr}` `v:key` has the index of the current byte and `v:val` has the byte value.

If `{expr}` is a `Funcref` it must take two arguments:

1. the key or the index of the current item.
2. the value of the current item.

The function must return `TRUE` if the item is found and the search should stop.

The optional argument `{opts}` is a Dict and supports the following items:

`startidx` start evaluating `{expr}` at the item with this index; may be negative for an item relative to the end

Returns -1 when `{expr}` evaluates to `v:false` for all the items.

Example:

```
:let l = [#n: 10}, #{n: 20}, #{n: 30}]
:echo indexof(l, "v:val.n == 20")
:echo indexof(l, {i, v -> v.n == 30})
:echo indexof(l, "v:val.n == 20", #{startidx: 1})
```

Can also be used as a `method` :

```
mylist->indexof(expr)
```

`input({prompt} [, {text} [, {completion}]])` `input()`

The result is a String, which is whatever the user typed on the command-line. The `{prompt}` argument is either a prompt string, or a blank string (for no prompt). A `'\n'` can be used in the prompt to start a new line.

The highlighting set with `:echohl` is used for the prompt. The input is entered just like a command-line, with the same editing commands and mappings. There is a separate history for lines typed for `input()`.

Example:

```
:if input("Coffee or beer? ") == "beer"
: echo "Cheers!"
:endif
```

If the optional `{text}` argument is present and not empty, this is used for the default reply, as if the user typed this.

Example:

```
:let color = input("Color? ", "white")
```

The optional `{completion}` argument specifies the type of completion supported for the input. Without it completion is not performed. The supported completion types are the same as that can be supplied to a user-defined command using the `"-complete="` argument. Refer to `:command-completion` for more information. Example:

```
let fname = input("File: ", "", "file")
```

NOTE: This function must not be used in a startup file, for

the versions that only run in GUI mode (e.g., the Win32 GUI).
Note: When `input()` is called from within a mapping it will consume remaining characters from that mapping, because a mapping is handled like the characters were typed. Use `inputsave()` before `input()` and `inputrestore()` after `input()` to avoid that. Another solution is to avoid that further characters follow in the mapping, e.g., by using `:execute` or `:normal`.

Example with a mapping:
`:nmap \x :call GetFoo()<CR>:exe "/" .. Foo<CR>`
`:function GetFoo()`
`: call inputsave()`
`: let g:Foo = input("enter search pattern: ")`
`: call inputrestore()`
`:endfunction`

Can also be used as a `method` :
`GetPrompt()->input()`

`inputdialog({prompt} [, {text} [, {cancelreturn}]])` `inputdialog()`

Like `input()`, but when the GUI is running and text dialogs are supported, a dialog window pops up to input the text.

Example:

```
:let n = inputdialog("value for shiftwidth", shiftwidth())
:if n != ""
: let &sw = n
:endif
```

When the dialog is cancelled `{cancelreturn}` is returned. When omitted an empty string is returned.

Hitting `<Enter>` works like pressing the OK button. Hitting `<Esc>` works like pressing the Cancel button.

NOTE: Command-line completion is not supported.

Can also be used as a `method` :
`GetPrompt()->inputdialog()`

`inputlist({textlist})` `inputlist()`

`{textlist}` must be a `List` of strings. This `List` is displayed, one string per line. The user will be prompted to enter a number, which is returned.

The user can also select an item by clicking on it with the mouse, if the mouse is enabled in the command line ('mouse' is "a" or includes "c"). For the first string 0 is returned.

When clicking above the first item a negative number is returned. When clicking on the prompt one more than the length of `{textlist}` is returned.

Make sure `{textlist}` has less than '`lines`' entries, otherwise it won't work. It's a good idea to put the entry number at the start of the string. And put a prompt in the first item.

Example:

```
let color = inputlist(['Select color:', '1. red',
\ '2. green', '3. blue'])
```

Can also be used as a `method` :
`GetChoices()->inputlist()`

`inputrestore()` `inputrestore()`
Restore typeahead that was saved with a previous `inputsave()` .
Should be called the same number of times `inputsave()` is called. Calling it more often is harmless though.
Returns TRUE when there is nothing to restore, FALSE otherwise.

`inputsave()` `inputsave()`
Preserve typeahead (also from mappings) and clear it, so that a following prompt gets input from the user. Should be followed by a matching `inputrestore()` after the prompt. Can be used several times, in which case there must be just as many `inputrestore()` calls.
Returns TRUE when out of memory, FALSE otherwise.

`inputsecret({prompt} [, {text}])` `inputsecret()`
This function acts much like the `input()` function with but two exceptions:
a) the user's response will be displayed as a sequence of asterisks ("*") thereby keeping the entry secret, and
b) the user's response will not be recorded on the `input history` stack.
The result is a String, which is whatever the user actually typed on the command-line in response to the issued prompt.
NOTE: Command-line completion is not supported.

Can also be used as a `method` :
`GetPrompt()->inputsecret()`

`insert({object}, {item} [, {idx}])` `insert()`
When `{object}` is a `List` or a `Blob` insert `{item}` at the start of it.

If `{idx}` is specified insert `{item}` before the item with index `{idx}`. If `{idx}` is zero it goes before the first item, just like omitting `{idx}`. A negative `{idx}` is also possible, see `list-index` . -1 inserts just before the last item.

Returns the resulting `List` or `Blob` . Examples:
`:let mylist = insert([2, 3, 5], 1)`
`:call insert(mylist, 4, -1)`
`:call insert(mylist, 6, len(mylist))`

The last example can be done simpler with `add()` .
Note that when `{item}` is a `List` it is inserted as a single item. Use `extend()` to concatenate `Lists` .

Can also be used as a `method` :
`mylist->insert(item)`

`instanceof({object}, {class})` `instanceof()` `E614` `E616` `E693`
The result is a Number, which is `TRUE` when the `{object}`

argument is a direct or indirect instance of a `Class` , `Interface` , or class `:type` alias specified by `{class}`. If `{class}` is varargs, the function returns `TRUE` when `{object}` is an instance of any of the specified classes. Example:

```
instanceof(animal, Dog, Cat)
```

Can also be used as a `method` :

```
myobj->instanceof(mytype)
```

`interrupt()`

`interrupt()`

Interrupt script execution. It works more or less like the user typing **CTRL-C**, most commands won't execute and control returns to the user. This is useful to abort execution from lower down, e.g. in an autocommand. Example:

```
:function s:check_typoname(file)
:   if fnamemodify(a:file, ':t') == '['
:       echomsg 'Maybe typo'
:       call interrupt()
:   endif
:endifunction
:au BufWritePre * call s:check_typoname(expand('<amatch>'))
```

`invert({expr})`

`invert()`

Bitwise invert. The argument is converted to a number. A List, Dict or Float argument causes an error. Example:

```
:let bits = invert(bits)
```

Can also be used as a `method` :

```
:let bits = bits->invert()
```

`isabsolutepath({path})`

`isabsolutepath()`

The result is a Number, which is `TRUE` when `{path}` is an absolute path.

On Unix, a path is considered absolute when it starts with `/'`. On MS-Windows, it is considered absolute when it starts with an optional drive prefix and is followed by a `'\'` or `/'`. UNC paths are always absolute.

Example:

```
echo isabsolutepath('/usr/share/')      " 1
echo isabsolutepath('./foobar')        " 0
echo isabsolutepath('C:\Windows')      " 1
echo isabsolutepath('foobar')          " 0
echo isabsolutepath('\\remote\file')    " 1
```

Can also be used as a `method` :

```
GetName()->isabsolutepath()
```

`isdirectory({directory})`

`isdirectory()`

The result is a Number, which is `TRUE` when a directory with the name `{directory}` exists. If `{directory}` doesn't exist, or isn't a directory, the result is `FALSE` . `{directory}` is any expression, which is used as a String.

Can also be used as a `method` :
`GetName()->isdirectory()`

`isinf({expr})`

`isinf()`

Return 1 if `{expr}` is a positive infinity, or -1 a negative infinity, otherwise 0.

```
:echo isinf(1.0 / 0.0)
1
:echo isinf(-1.0 / 0.0)
-1
```

Can also be used as a `method` :
`Compute()->isinf()`

`islocked({expr})`

`islocked()` E786

The result is a Number, which is `TRUE` when `{expr}` is the name of a locked variable.

The string argument `{expr}` must be the name of a variable, `List` item or `Dictionary` entry, not the variable itself!

Example:

```
:let alist = [0, ['a', 'b'], 2, 3]
:lockvar 1 alist
:echo islocked('alist')          " 1
:echo islocked('alist[1]')      " 0
```

When `{expr}` is a variable that does not exist -1 is returned. If `{expr}` uses a range, list or dict index that is out of range or does not exist you get an error message. Use `exists()` to check for existence. In Vim9 script it does not work for local function variables.

Can also be used as a `method` :
`GetName()->islocked()`

`isnan({expr})`

`isnan()`

Return `TRUE` if `{expr}` is a float with value NaN.

```
:echo isnan(0.0 / 0.0)
1
```

Can also be used as a `method` :
`Compute()->isnan()`

`items({dict})`

`items()`

Return a `List` with all the key-value pairs of `{dict}`. Each `List` item is a list with two items: the key of a `{dict}` entry and the value of this entry. The `List` is in arbitrary order. Also see `keys()` and `values()` .

Example:

```
for [key, value] in items(mydict)
  echo key .. ': ' .. value
endfor
```

A `List` or a `String` argument is also supported. In these cases, `items()` returns a `List` with the index and the value at

the index.

Can also be used as a `method` :
`mydict->items()`

`job_` functions are documented here: [job-functions-details](#)

`join({list} [, {sep}])` `join()`
Join the items in `{list}` together into one String.
When `{sep}` is specified it is put in between the items. If `{sep}` is omitted a single space is used.
Note that `{sep}` is not added at the end. You might want to add it there too:

```
let lines = join(mylist, "\n") .. "\n"
```

String items are used as-is. `Lists` and `Dictionaries` are converted into a string like with `string()` .
The opposite function is `split()` .

Can also be used as a `method` :
`mylist->join()`

`js_decode({string})` `js_decode()`
This is similar to `json_decode()` with these differences:
- Object key names do not have to be in quotes.
- Strings can be in single quotes.
- Empty items in an array (between two commas) are allowed and result in `v:none` items.

Can also be used as a `method` :
`ReadObject()->js_decode()`

`js_encode({expr})` `js_encode()`
This is similar to `json_encode()` with these differences:
- Object key names are not in quotes.
- `v:none` items in an array result in an empty item between commas.

For example, the Vim object:

```
[1,v:none,{"one":1},v:none]
```

Will be encoded as:

```
[1,,{one:1},,]
```

While `json_encode()` would produce:

```
[1,null,{"one":1},null]
```

This encoding is valid for JavaScript. It is more efficient than JSON, especially when using an array with optional items.

Can also be used as a `method` :
`GetObject()->js_encode()`

`json_decode({string})` `json_decode()` **E491**
This parses a JSON formatted string and returns the equivalent in Vim values. See `json_encode()` for the relation between JSON and Vim values.
The decoding is permissive:

- A trailing comma in an array and object is ignored, e.g. "[1, 2,]" is the same as "[1, 2]".
- Integer keys are accepted in objects, e.g. {1:2} is the same as {"1":2}.
- More floating point numbers are recognized, e.g. "1." for "1.0", or "001.2" for "1.2". Special floating point values "Infinity", "-Infinity" and "NaN" (capitalization ignored) are accepted.
- Leading zeroes in integer numbers are ignored, e.g. "012" for "12" or "-012" for "-12".
- Capitalization is ignored in literal names null, true or false, e.g. "NULL" for "null", "True" for "true".
- Control characters U+0000 through U+001F which are not escaped in strings are accepted, e.g. " " (tab character in string) for "\t".
- An empty JSON expression or made of only spaces is accepted and results in v:none.
- Backslash in an invalid 2-character sequence escape is ignored, e.g. "\a" is decoded as "a".
- A correct surrogate pair in JSON strings should normally be a 12 character sequence such as "\uD834\uDD1E", but json_decode() silently accepts truncated surrogate pairs such as "\uD834" or "\uD834\u"

E938

A duplicate key in an object, valid in rfc7159, is not accepted by json_decode() as the result must be a valid Vim type, e.g. this fails: {"a":"b", "a":"c"}

Can also be used as a `method` :
`ReadObject()->json_decode()`

`json_encode({expr})`

`json_encode()`

Encode {expr} as JSON and return this as a string.

The encoding is specified in:

<https://tools.ietf.org/html/rfc7159.html>

Vim values are converted as follows: E1161

<code>Number</code>	decimal number
<code>Float</code>	floating point number
<code>Float nan</code>	"NaN"
<code>Float inf</code>	"Infinity"
<code>Float -inf</code>	"-Infinity"
<code>String</code>	in double quotes (possibly null)
<code>Funcref</code>	not possible, error
<code>List</code>	as an array (possibly null); when used recursively: []
<code>Dict</code>	as an object (possibly null); when used recursively: {}
<code>Blob</code>	as an array of the individual bytes
<code>v:false</code>	"false"
<code>v:true</code>	"true"
<code>v:none</code>	"null"
<code>v:null</code>	"null"

Note that NaN and Infinity are passed on as values. This is missing in the JSON standard, but several implementations do

allow it. If not then you will get an error.
If a string contains an illegal character then the replacement character 0xfffd is used.

Can also be used as a `method` :
`GetObject()->json_encode()`

`keys({dict})` `keys()`
Return a `List` with all the keys of `{dict}`. The `List` is in arbitrary order. Also see `items()` and `values()` .

Can also be used as a `method` :
`mydict->keys()`

`keytrans({string})` `keytrans()`
Turn the internal byte representation of keys into a form that can be used for `:map` . E.g.
`:let xx = "\<C-Home>"`
`:echo keytrans(xx)`
`<C-Home>`

Can also be used as a `method` :
`"\<C-Home>"->keytrans()`

`len({expr})` `len()` E701
The result is a Number, which is the length of the argument. When `{expr}` is a String or a Number the length in bytes is used, as with `strlen()` .
When `{expr}` is a `List` the number of items in the `List` is returned.
When `{expr}` is a `Blob` the number of bytes is returned.
When `{expr}` is a `Dictionary` the number of entries in the `Dictionary` is returned.
When `{expr}` is an `Object` , invokes the `len()` method in the object (if present) to get the length. Otherwise returns zero.

Can also be used as a `method` :
`mylist->len()`

`libcall({libname}, {funcname}, {argument})` `libcall()` E364 E368
Call function `{funcname}` in the run-time library `{libname}` with single argument `{argument}`.
This is useful to call functions in a library that you especially made to be used with Vim. Since only one argument is possible, calling standard library functions is rather limited.
The result is the String returned by the function. If the function returns NULL, this will appear as an empty string "" to Vim.
If the function returns a number, use `libcallnr()`!
If `{argument}` is a number, it is passed to the function as an int; if `{argument}` is a string, it is passed as a

null-terminated string.
This function will fail in `restricted-mode` .

`libcall()` allows you to write your own 'plug-in' extensions to Vim without having to recompile the program. It is NOT a means to call system functions! If you try to do so Vim will very probably crash.

For Win32, the functions you write must be placed in a DLL and use the normal C calling convention (NOT Pascal which is used in Windows System DLLs). The function must take exactly one parameter, either a character pointer or a long integer, and must return a character pointer or NULL. The character pointer returned must point to memory that will remain valid after the function has returned (e.g. in static data in the DLL). If it points to allocated memory, that memory will leak away. Using a static buffer in the function should work, it's then freed when the DLL is unloaded.

WARNING: If the function returns a non-valid pointer, Vim may crash! This also happens if the function returns a number, because Vim thinks it's a pointer.
For Win32 systems, `{libname}` should be the filename of the DLL without the ".DLL" suffix. A full path is only required if the DLL is not in the usual places.
For Unix: When compiling your own plugins, remember that the object code must be compiled as position-independent ('PIC').
{only in Win32 and some Unix versions, when the `+libcall` feature is present}

Examples:

```
:echo libcall("libc.so", "getenv", "HOME")
```

Can also be used as a `method` , the base is passed as the third argument:

```
GetValue()->libcall("libc.so", "getenv")
```

`libcallnr()`

`libcallnr({libname}, {funcname}, {argument})`

Just like `libcall()` , but used for a function that returns an int instead of a string.

{only in Win32 on some Unix versions, when the `+libcall` feature is present}

Examples:

```
:echo libcallnr("/usr/lib/libc.so", "getpid", "")
:call libcallnr("libc.so", "printf", "Hello World!\n")
:call libcallnr("libc.so", "sleep", 10)
```

Can also be used as a `method` , the base is passed as the third argument:

```
GetValue()->libcallnr("libc.so", "printf")
```

`line({expr} [, {winid}])`

`line()`

The result is a Number, which is the line number of the file

position given with `{expr}`. The `{expr}` argument is a string. The accepted positions are: E1209

.	the cursor position
\$	the last line in the current buffer
'x	position of mark x (if the mark is not set, 0 is returned)
w0	first line visible in current window (one if the display isn't updated, e.g. in silent Ex mode)
w\$	last line visible in current window (this is one less than "w0" if no lines are visible)
v	In Visual mode: the start of the Visual area (the cursor is the end). When not in Visual mode returns the cursor position. Differs from '<' in that it's updated right away.

Note that a mark in another file can be used. The line number then applies to another buffer.

To get the column number use `col()`. To get both use `getpos()`.

With the optional `{winid}` argument the values are obtained for that window instead of the current window.

Returns 0 for invalid values of `{expr}` and `{winid}`.

Examples:

<code>line(".")</code>	line number of the cursor
<code>line(".", winid)</code>	idem, in window "winid"
<code>line("'t")</code>	line number of mark t
<code>line("' " .. marker)</code>	line number of mark marker

To jump to the last known position when opening a file see [last-position-jump](#).

Can also be used as a **method** :

`GetValue()->line()`

`line2byte({lnum})`

`line2byte()`

Return the byte count from the start of the buffer for line `{lnum}`. This includes the end-of-line character, depending on the **'fileformat'** option for the current buffer. The first line returns 1. **'encoding'** matters, **'fileencoding'** is ignored. This can also be used to get the byte count for the line just below the last line:

`line2byte(line("$") + 1)`

This is the buffer size plus one. If **'fileencoding'** is empty it is the file size plus one. `{lnum}` is used like with `getline()`. When `{lnum}` is invalid, or the **+byte_offset** feature has been disabled at compile time, -1 is returned. Also see `byte2line()`, `go` and `:goto`.

Can also be used as a **method** :

`GetLnum()->line2byte()`

`lispindent({lnum})`

`lispindent()`

Get the amount of indent for line `{lnum}` according the lisp indenting rules, as with **'lisp'**.

The indent is counted in spaces, the value of **'tabstop'** is

relevant. `{lnum}` is used just like in `getline()` .
When `{lnum}` is invalid -1 is returned. In Vim9 script an error is given.

Can also be used as a `method` :
`GetLnum()->lispindent()`

`list2blob({list})` `list2blob()`
Return a Blob concatenating all the number values in `{list}`.
Examples:
`list2blob([1, 2, 3, 4])` returns `0z01020304`
`list2blob([])` returns `0z`
Returns an empty Blob on error. If one of the numbers is negative or more than 255 error `E1239` is given.

`blob2list()` does the opposite.

Can also be used as a `method` :
`GetList()->list2blob()`

`list2str({list} [, {utf8}])` `list2str()`
Convert each number in `{list}` to a character string and concatenate them all. Examples:
`list2str([32])` returns `" "`
`list2str([65, 66, 67])` returns `"ABC"`
The same can be done (slowly) with:
`join(map(list, {nr, val -> nr2char(val)}), '')`
`str2list()` does the opposite.

When `{utf8}` is omitted or zero, the current `'encoding'` is used.
When `{utf8}` is TRUE, always return UTF-8 characters.
With UTF-8 composing characters work as expected:
`list2str([97, 769])` returns `"á"`

Returns an empty string on error.

Can also be used as a `method` :
`GetList()->list2str()`

`listener_add({callback} [, {buf}])` `listener_add()`
Add a callback function that will be invoked when changes have been made to buffer `{buf}`.
`{buf}` refers to a buffer name or number. For the accepted values, see `bufname()` . When `{buf}` is omitted the current buffer is used.
Returns a unique ID that can be passed to `listener_remove()` .

The `{callback}` is invoked with five arguments:
bufnr the buffer that was changed
start first changed line number
end first line number below the change
added number of lines added, negative if lines were deleted
changes a List of items with details about the changes

```

Example:
func Listener(bufnr, start, end, added, changes)
  echo 'lines ' .. a:start .. ' until ' .. a:end .. ' changed'
endfunc
call listener_add('Listener', bufnr)

```

The List cannot be changed. Each item in "changes" is a dictionary with these entries:

lnum	the first line number of the change
end	the first line below the change
added	number of lines added; negative if lines were deleted
col	first column in "lnum" that was affected by the change; one if unknown or the whole line was affected; this is a byte index, first character has a value of one.

When lines are inserted (not when a line is split, e.g. by typing CR in Insert mode) the values are:

lnum	line above which the new line is added
end	equal to "lnum"
added	number of lines inserted
col	1

When lines are deleted the values are:

lnum	the first deleted line
end	the line below the first deleted line, before the deletion was done
added	negative, number of lines deleted
col	1

When lines are changed:

lnum	the first changed line
end	the line below the last changed line
added	0
col	first column with a change or 1

The entries are in the order the changes were made, thus the most recent change is at the end. The line numbers are valid when the callback is invoked, but later changes may make them invalid, thus keeping a copy for later might not work.

The `{callback}` is invoked just before the screen is updated, when `listener_flush()` is called or when a change is being made that changes the line count in a way it causes a line number in the list of changes to become invalid.

The `{callback}` is invoked with the text locked, see `textlock`. If you do need to make changes to the buffer, use a timer to do this later `timer_start()`.

The `{callback}` is not invoked when the buffer is first loaded. Use the `BufReadPost` autocmd event to handle the initial text of a buffer.

The `{callback}` is also not invoked when the buffer is unloaded, use the `BufUnload` autocmd event for that.

Returns zero if `{callback}` or `{buf}` is invalid.

Can also be used as a `method` , the base is passed as the second argument:

```
GetBuffer()->listener_add(callback)
```

`listener_flush([{buf}])` `listener_flush()`

Invoke listener callbacks for buffer `{buf}`. If there are no pending changes then no callbacks are invoked.

`{buf}` refers to a buffer name or number. For the accepted values, see `bufname()` . When `{buf}` is omitted the current buffer is used.

Can also be used as a `method` :

```
GetBuffer()->listener_flush()
```

`listener_remove({id})` `listener_remove()`

Remove a listener previously added with `listener_add()`. Returns FALSE when `{id}` could not be found, TRUE when `{id}` was removed.

Can also be used as a `method` :

```
GetListenerId()->listener_remove()
```

`localtime()` `localtime()`

Return the current time, measured as seconds since 1st Jan 1970. See also `strftime()` , `strptime()` and `getftime()` .

`log({expr})` `log()`

Return the natural logarithm (base e) of `{expr}` as a `Float` . `{expr}` must evaluate to a `Float` or a `Number` in the range `(0, inf]`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number` .

Examples:

```
:echo log(10)
2.302585
:echo log(exp(5))
5.0
```

Can also be used as a `method` :

```
Compute()->log()
```

`log10({expr})` `log10()`

Return the logarithm of `Float` `{expr}` to base 10 as a `Float` . `{expr}` must evaluate to a `Float` or a `Number` .

Returns 0.0 if `{expr}` is not a `Float` or a `Number` .

Examples:

```
:echo log10(1000)
3.0
:echo log10(0.01)
```

-2.0

Can also be used as a `method` :
`Compute()->log10()`

`luaeval({expr} [, {expr}])` `luaeval()`
Evaluate Lua expression `{expr}` and return its result converted to Vim data structures. Second `{expr}` may hold additional argument accessible as `_A` inside first `{expr}`.
Strings are returned as they are.
Boolean objects are converted to numbers.
Numbers are converted to `Float` values.
Dictionaries and lists obtained by `vim.eval()` are returned as-is.
Other objects are returned as zero without any errors.
See `lua-luaeval` for more details.
Note that in a `:def` function local variables are not visible to {expr}.`

Can also be used as a `method` :
`GetExpr()->luaeval()`

`{only available when compiled with the |+lua| feature}`

`map({expr1}, {expr2})` `map()`
`{expr1}` must be a `List` , `String` , `Blob` or `Dictionary` .
When `{expr1}` is a `List` or `Dictionary` , replace each item in `{expr1}` with the result of evaluating `{expr2}`.
For a `Blob` each byte is replaced.
For a `String` , each character, including composing characters, is replaced.
If the item type changes you may want to use `mapnew()` to create a new `List` or `Dictionary`. This is required when using Vim9 script.

`{expr2}` must be a `String` or `Funcref` .

If `{expr2}` is a `String` , inside `{expr2}` `v:val` has the value of the current item. For a `Dictionary` `v:key` has the key of the current item and for a `List` `v:key` has the index of the current item. For a `Blob` `v:key` has the index of the current byte. For a `String` `v:key` has the index of the current character.

Example:

```
:call map(mylist, '> " .. v:val .. " <")
```

This puts "> " before and " <" after each item in "mylist".

Note that `{expr2}` is the result of an expression and is then used as an expression again. Often it is good to use a `literal-string` to avoid having to double backslashes. You still have to double ' quotes

If `{expr2}` is a `Funcref` it is called with two arguments:
1. The key or the index of the current item.

2. the value of the current item.

With a legacy script lambda you don't get an error if it only accepts one argument, but with a Vim9 lambda you get "E1106: One argument too many", the number of arguments must match.

The function must return the new value of the item. Example that changes each value by "key-value":

```
func KeyValue(key, val)
    return a:key .. '-' .. a:val
endfunc
call map(myDict, function('KeyValue'))
```

It is shorter when using a `lambda` :

```
call map(myDict, {key, val -> key .. '-' .. val})
```

If you do not use "val" you can leave it out:

```
call map(myDict, {key -> 'item: ' .. key})
```

If you do not use "key" you can use a short name:

```
call map(myDict, {_, val -> 'item: ' .. val})
```

The operation is done in-place for a `List` and `Dictionary` .

If you want it to remain unmodified make a copy first:

```
:let tlist = map(copy(mylist), ' v:val .. "\t")
```

Returns `{expr1}`, the `List` or `Dictionary` that was filtered, or a new `Blob` or `String` .

When an error is encountered while evaluating `{expr2}` no further items in `{expr1}` are processed.

When `{expr2}` is a Funcref errors inside a function are ignored, unless it was defined with the "abort" flag.

Can also be used as a `method` :

```
mylist->map(expr2)
```

`maparg({name} [, {mode} [, {abbr} [, {dict}]]])` `maparg()`

When `{dict}` is omitted or zero: Return the rhs of mapping `{name}` in mode `{mode}`. The returned String has special characters translated like in the output of the `":map"` command listing. When `{dict}` is TRUE a dictionary is returned, see below. To get a list of all mappings see `maplist()` .

When there is no mapping for `{name}`, an empty String is returned if `{dict}` is FALSE, otherwise returns an empty Dict. When the mapping for `{name}` is empty, then "`<Nop>`" is returned.

The `{name}` can have special key names, like in the `":map"` command.

`{mode}` can be one of these strings:

```
"n"      Normal
"v"      Visual (including Select)
"o"      Operator-pending
"i"      Insert
"c"      Cmd-line
```

```

"s"      Select
"x"      Visual
"l"      langmap language-mapping
"t"      Terminal-Job
""       Normal, Visual and Operator-pending

```

When `{mode}` is omitted, the modes for "" are used.

When `{abbr}` is there and it is `TRUE` use abbreviations instead of mappings.

When `{dict}` is there and it is `TRUE` return a dictionary containing all the information of the mapping with the following items: [mapping-dict](#)

```

"lhs"      The {lhs} of the mapping as it would be typed
"lhsraw"   The {lhs} of the mapping as raw bytes
"lhsrawalt" The {lhs} of the mapping as raw bytes, alternate
           form, only present when it differs from "lhsraw"
"rhs"      The {rhs} of the mapping as typed.
"silent"   1 for a :map-silent mapping, else 0.
"noremap"  1 if the {rhs} of the mapping is not remappable.
"script"   1 if mapping was defined with <script>.
"expr"     1 for an expression mapping ( :map-<expr> ).
"buffer"   1 for a buffer local mapping ( :map-local ).
"mode"     Modes for which the mapping is defined. In
           addition to the modes mentioned above, these
           characters will be used:
           " "      Normal, Visual and Operator-pending
           "!"     Insert and Commandline mode
           ( mapmode-ic )
"sid"      The script local ID, used for <sid> mappings
           ( <SID> ). Negative for special contexts.
"scriptversion" The version of the script. 999999 for
           Vim9 script.
"lnum"     The line number in "sid", zero if unknown.
"nowait"   Do not wait for other, longer mappings.
           ( :map-<nowait> ).
"abbr"     True if this is an abbreviation abbreviations .
"mode_bits" Vim's internal binary representation of "mode".
           mapset\(\) ignores this; only "mode" is used.
           See maplist\(\) for usage examples. The values
           are from src/vim.h and may change in the future.

```

The dictionary can be used to restore a mapping with [mapset\(\)](#) .

The mappings local to the current buffer are checked first, then the global mappings.

This function can be used to map a key even when it's already mapped, and have it do the original mapping too. Sketch:

```

exe 'nnoremap <Tab> ==' .. maparg('<Tab>', 'n')

```

Can also be used as a [method](#) :

```

GetKey()->maparg('n')

```

`mapcheck({name} [, {mode} [, {abbr}]])` `mapcheck()`
 Check if there is a mapping that matches with `{name}` in mode `{mode}`. See `maparg()` for `{mode}` and special names in `{name}`.
 When `{abbr}` is there and it is `TRUE` use abbreviations instead of mappings.
 A match happens with a mapping that starts with `{name}` and with a mapping which is equal to the start of `{name}`.

	matches	mapping	"a"	"ab"	"abc"
<code>mapcheck("a")</code>	yes	yes	yes	yes	yes
<code>mapcheck("abc")</code>	yes	yes	yes	yes	yes
<code>mapcheck("ax")</code>	yes	no	no	no	no
<code>mapcheck("b")</code>	no	no	no	no	no

The difference with `maparg()` is that `mapcheck()` finds a mapping that matches with `{name}`, while `maparg()` only finds a mapping for `{name}` exactly.

When there is no mapping that starts with `{name}`, an empty String is returned. If there is one, the RHS of that mapping is returned. If there are several mappings that start with `{name}`, the RHS of one of them is returned. This will be `"<Nop>"` if the RHS is empty.

The mappings local to the current buffer are checked first, then the global mappings.

This function can be used to check if a mapping can be added without being ambiguous. Example:

```
:if mapcheck("_vv") == ""
:  map _vv :set guifont=7x13<CR>
:endif
```

This avoids adding the `"_vv"` mapping when there already is a mapping for `"_v"` or for `"_vvv"`.

Can also be used as a `method` :

```
GetKey()->mapcheck('n')
```

`maplist([{abbr}])` `maplist()`
 Returns a `List` of all mappings. Each List item is a `Dict`, the same as what is returned by `maparg()`, see `mapping-dict`. When `{abbr}` is there and it is `TRUE` use abbreviations instead of mappings.

Example to show all mappings with 'MultiMatch' in rhs:

```
vim9script
echo maplist()->filter(
  (_, m) => match(m.rhs, 'MultiMatch') >= 0)
```

It can be tricky to find mappings for particular `:map-modes`. `mapping-dict`'s `"mode_bits"` can simplify this. For example, the `mode_bits` for Normal, Insert or Command-line modes are `0x19`. To find all the mappings available in those modes you can do:

```
vim9script
var saved_maps = []
```

```

    for m in maplist()
        if and(m.mode_bits, 0x19) != 0
            saved_maps->add(m)
        endif
    endfor
    echo saved_maps->mapnew(('_', m) => m.lhs)

```

The values of the mode_bits are defined in Vim's src/vim.h file and they can be discovered at runtime using `:map-commands` and `"maplist()"`. Example:

```

vim9script
omap xyzyy <Nop>
var op_bit = maplist()->filter(
    (_, m) => m.lhs == 'xyzyy')[0].mode_bits
ounmap xyzyy
echo printf("Operator-pending mode bit: 0x%x", op_bit)

```

`mapnew({expr1}, {expr2})` mapnew()
 Like `map()` but instead of replacing items in `{expr1}` a new List or Dictionary is created and returned. `{expr1}` remains unchanged. Items can still be changed by `{expr2}`, if you don't want that use `deepcopy()` first.

`mapset({mode}, {abbr}, {dict})` mapset()
`mapset({dict})`
 Restore a mapping from a dictionary, possibly returned by `maparg()` or `maplist()`. A buffer mapping, when `dict.buffer` is true, is set on the current buffer; it is up to the caller to ensure that the intended buffer is the current buffer. This feature allows copying mappings from one buffer to another. The `dict.mode` value may restore a single mapping that covers more than one mode, like with mode values of `!'`, `' '`, `'nox'`, or `'v'`. [E1276](#)

In the first form, `{mode}` and `{abbr}` should be the same as for the call to `maparg()`. [E460](#)
`{mode}` is used to define the mode in which the mapping is set, not the "mode" entry in `{dict}`.

Example for saving and restoring a mapping:

```

let save_map = maparg('K', 'n', 0, 1)
nnoemap K somethingelse
...
call mapset('n', 0, save_map)

```

Note that if you are going to replace a map in several modes, e.g. with ``:map!``, you need to save/restore the mapping for all of them, when they might differ.

In the second form, with `{dict}` as the only argument, mode and abbr are taken from the dict.

Example:

```

vim9script
var save_maps = maplist()->filter(
    (_, m) => m.lhs == 'K')

```

```

nnoemap K somethingelse
cnoemap K somethingelse2
# ...
unmap K
for d in save_maps
  mapset(d)
endfor

```

`match({expr}, {pat} [, {start} [, {count}]]])` match()

When `{expr}` is a `List` then this returns the index of the first item where `{pat}` matches. Each item is used as a String, `Lists` and `Dictionaries` are used as echoed.

Otherwise, `{expr}` is used as a String. The result is a Number, which gives the index (byte offset) in `{expr}` where `{pat}` matches.

A match at the first character or `List` item returns zero. If there is no match -1 is returned.

For getting submatches see `matchlist()` .

Example:

```

:echo match("testing", "ing") " results in 4
:echo match([1, 'x'], '\a') " results in 1

```

See `string-match` for how `{pat}` is used.

strpbrk()

Vim doesn't have a `strpbrk()` function. But you can do:

```
:let sepidx = match(line, '[.,;: \t]')
```

strcasestr()

Vim doesn't have a `strcasestr()` function. But you can add `"\c"` to the pattern to ignore case:

```
:let idx = match(haystack, '\cneedle')
```

If `{start}` is given, the search starts from byte index `{start}` in a String or item `{start}` in a `List` .

The result, however, is still the index counted from the first character/item. Example:

```
:echo match("testing", "ing", 2)
```

result is again "4".

```
:echo match("testing", "ing", 4)
```

result is again "4".

```
:echo match("testing", "t", 2)
```

result is "3".

For a String, if `{start} > 0` then it is like the string starts `{start}` bytes later, thus `"^"` will match at `{start}`. Except when `{count}` is given, then it's like matches before the `{start}` byte are ignored (this is a bit complicated to keep it backwards compatible).

For a String, if `{start} < 0`, it will be set to 0. For a list the index is counted from the end.

If `{start}` is out of range (`{start} > strlen({expr})` for a String or `{start} > len({expr})` for a `List`) -1 is returned.

When `{count}` is given use the `{count}`'th match. When a match is found in a String the search for the next one starts one character further. Thus this example results in 1:

```
echo match("testing", "..", 0, 2)
```

In a `List` the search continues in the next item.

Note that when `{count}` is added the way `{start}` works changes, see above.

match-pattern

See `pattern` for the patterns that are accepted.

The `'ignorecase'` option is used to set the ignore-caseness of the pattern. `'smartcase'` is NOT used. The matching is always done like `'magic'` is set and `'coptions'` is empty.

Note that a match at the start is preferred, thus when the pattern is using `"*` (any number of matches) it tends to find zero matches at the start instead of a number of matches further down in the text.

Can also be used as a `method` :

```
GetText()->match('word')
GetList()->match('word')
```

```
matchadd({group}, {pattern} [, {priority} [, {id} [, {dict}]]])
```

Defines a pattern to be highlighted in the current window (a "match"). It will be highlighted with `{group}`. Returns an identification number (ID), which can be used to delete the match using `matchdelete()`. The ID is bound to the window. Matching is case sensitive and magic, unless case sensitivity or magicness are explicitly overridden in `{pattern}`. The `'magic'`, `'smartcase'` and `'ignorecase'` options are not used. The "Conceal" value is special, it causes the match to be concealed.

The optional `{priority}` argument assigns a priority to the match. A match with a high priority will have its highlighting overrule that of a match with a lower priority. A priority is specified as an integer (negative numbers are no exception). If the `{priority}` argument is not specified, the default priority is 10. The priority of `'hlsearch'` is zero, hence all matches with a priority greater than zero will overrule it. Syntax highlighting (see `'syntax'`) is a separate mechanism, and regardless of the chosen priority a match will always overrule syntax highlighting.

The optional `{id}` argument allows the request for a specific match ID. If a specified ID is already taken, an error message will appear and the match will not be added. An ID is specified as a positive integer (zero excluded). IDs 1, 2 and 3 are reserved for `:match`, `:2match` and `:3match`, respectively. 3 is reserved for use by the `matchparen` plugin.

If the `{id}` argument is not specified or -1, `matchadd()` automatically chooses a free ID, which is at least 1000.

The optional `{dict}` argument allows for further custom values. Currently this is used to specify a match specific conceal character that will be shown for `hl-Conceal` highlighted matches. The dict can have the following members:

<code>conceal</code>	Special character to show instead of the match (only for <code>hl-Conceal</code> highlighted matches, see <code>:syn-cchar</code>)
<code>window</code>	Instead of the current window use the window with this number or window ID.

The number of matches is not limited, as it is the case with the `:match` commands.

Returns -1 on error.

Example:

```
:highlight MyGroup ctermbg=green guibg=green
:let m = matchadd("MyGroup", "TODO")
```

Deletion of the pattern:

```
:call matchdelete(m)
```

A list of matches defined by `matchadd()` and `:match` are available from `getmatches()` . All matches can be deleted in one operation by `clearmatches()` .

Can also be used as a `method` :

```
GetGroup()->matchadd('TODO')
```

`matchaddpos({group}, {pos} [, {priority} [, {id} [, {dict}]]])` `matchaddpos()`
Same as `matchadd()` , but requires a list of positions `{pos}` instead of a pattern. This command is faster than `matchadd()` because it does not require to handle regular expressions and sets buffer line boundaries to redraw screen. It is supposed to be used when fast match additions and deletions are required, for example to highlight matching parentheses.

`{pos}` is a list of positions. Each position can be one of these:

- A number. This whole line will be highlighted. The first line has number 1.
- A list with one number, e.g., `[23]`. The whole line with this number will be highlighted.
- A list with two numbers, e.g., `[23, 11]`. The first number is the line number, the second one is the column number (first column is 1, the value must correspond to the byte index as `col()` would return). The character at this position will be highlighted.
- A list with three numbers, e.g., `[23, 11, 3]`. As above, but the third number gives the length of the highlight in bytes.

Returns -1 on error.

Example:

```
:highlight MyGroup ctermbg=green guibg=green
:let m = matchaddpos("MyGroup", [[23, 24], 34])
```

Deletion of the pattern:

```
:call matchdelete(m)
```

Matches added by `matchaddpos()` are returned by `getmatches()` .

Can also be used as a `method` :

```
GetGroup()->matchaddpos([23, 11])
```

`matcharg({nr})`

`matcharg()`

Selects the `{nr}` match item, as set with a `:match` ,
`:2match` or `:3match` command.

Return a `List` with two elements:

- The name of the highlight group used
- The pattern used.

When `{nr}` is not 1, 2 or 3 returns an empty `List` .

When there is no match item set returns `['', '']`.

This is useful to save and restore a `:match` .

Highlighting matches using the `:match` commands are limited to three matches. `matchadd()` does not have this limitation.

Can also be used as a `method` :

```
GetMatch()->matcharg()
```

`matchbufline()`

`matchbufline({buf}, {pat}, {lnum}, {end}, [, {dict}])`

Returns the `List` of matches in lines from `{lnum}` to `{end}` in buffer `{buf}` where `{pat}` matches.

`{lnum}` and `{end}` can either be a line number or the string "\$" to refer to the last line in `{buf}`.

The `{dict}` argument supports following items:

submatches include submatch information (`/\()`)

For each match, a `Dict` with the following items is returned:

byteidx	starting byte index of the match
lnum	line number where there is a match
text	matched string

Note that there can be multiple matches in a single line.

This function works only for loaded buffers. First call

`bufload()` if needed.

See `match-pattern` for information about the effect of some option settings on the pattern.

When `{buf}` is not a valid buffer, the buffer is not loaded or `{lnum}` or `{end}` is not valid then an error is given and an empty `List` is returned.

Examples:

```
" Assuming line 3 in buffer 5 contains "a"
:echo matchbufline(5, '\<\k\>', 3, 3)
[{'lnum': 3, 'byteidx': 0, 'text': 'a'}]
" Assuming line 4 in buffer 10 contains "tik tok"
:echo matchbufline(10, '\<\k\>', 1, 4)
[{'lnum': 4, 'byteidx': 0, 'text': 'tik'}, {'lnum': 4, 'byteidx': 4, 'text': 'tok'}]
```

If `{submatch}` is present and is `v:true`, then submatches like `"\1"`, `"\2"`, etc. are also returned. Example:

```
" Assuming line 2 in buffer 2 contains "acd"
:echo matchbufline(2, '\(a\)?\(b\)?\(c\)?\(.*\)')
\ {'submatches': v:true}
[{'lnum': 2, 'byteidx': 0, 'text': 'acd', 'submatches': ['a', '', 'c', 'd']}]
```

The "submatches" List always contains 9 items. If a submatch is not found, then an empty string is returned for that submatch.

Can also be used as a `method` :

```
GetBuffer()->matchbufline('mypat', 1, '$')
```

`matchdelete({id} [, {win}])` `matchdelete()` E802 E803
Deletes a match with ID `{id}` previously defined by `matchadd()` or one of the `:match` commands. Returns 0 if successful, otherwise -1. See example for `matchadd()` . All matches can be deleted in one operation by `clearmatches()` .
If `{win}` is specified, use the window with this number or window ID instead of the current window.

Can also be used as a `method` :

```
GetMatch()->matchdelete()
```

`matchend({expr}, {pat} [, {start} [, {count}]])` `matchend()`
Same as `match()` , but return the index of first character after the match. Example:
`:echo matchend("testing", "ing")`
results in "7".

`strspn()` `strcspn()`

Vim doesn't have a `strspn()` or `strcspn()` function, but you can do it with `matchend()`:

```
:let span = matchend(line, '[a-zA-Z]')
:let span = matchend(line, '^a-zA-Z')
```

Except that -1 is returned when there are no matches.

The `{start}`, if given, has the same meaning as for `match()` .
`:echo matchend("testing", "ing", 2)`
results in "7".
`:echo matchend("testing", "ing", 5)`
result is "-1".

When `{expr}` is a `List` the result is equal to `match()` .

Can also be used as a `method` :

```
GetText()->matchend('word')
```

`matchfuzzy({list}, {str} [, {dict}])`

`matchfuzzy()`

If `{list}` is a list of strings, then returns a `List` with all the strings in `{list}` that fuzzy match `{str}`. The strings in the returned list are sorted based on the matching score.

The optional `{dict}` argument always supports the following items:

- `matchseq` When this item is present return only matches that contain the characters in `{str}` in the given sequence.
- `limit` Maximum number of matches in `{list}` to be returned. Zero means no limit.

If `{list}` is a list of dictionaries, then the optional `{dict}` argument supports the following additional items:

- `key` Key of the item which is fuzzy matched against `{str}`. The value of this item should be a string.
- `text_cb` `Funcref` that will be called for every item in `{list}` to get the text for fuzzy matching. This should accept a dictionary item as the argument and return the text for that item to use for fuzzy matching.

`{str}` is treated as a literal string and regular expression matching is NOT supported. The maximum supported `{str}` length is 256.

When `{str}` has multiple words each separated by white space, then the list of strings that have all the words is returned.

If there are no matching strings or there is an error, then an empty list is returned. If length of `{str}` is greater than 256, then returns an empty list.

When `{limit}` is given, `matchfuzzy()` will find up to this number of matches in `{list}` and return them in sorted order.

Refer to [fuzzy-matching](#) for more information about fuzzy matching strings.

Example:

```
:echo matchfuzzy(["clay", "crow"], "cay")
```

results in ["clay"].

```
:echo getbufinfo()->map({_, v -> v.name})->matchfuzzy("ndl")
```

results in a list of buffer names fuzzy matching "ndl".

```
:echo getbufinfo()->matchfuzzy("ndl", {'key' : 'name'})
```

results in a list of buffer information dicts with buffer names fuzzy matching "ndl".

```
:echo getbufinfo()->matchfuzzy("spl",  
    \ {'text_cb' : {v -> v.name}})
```

results in a list of buffer information dicts with buffer

```

names fuzzy matching "spl".
:echo v:oldfiles->matchfuzzy("test")
results in a list of file names fuzzy matching "test".
:let l = readfile("buffer.c")->matchfuzzy("str")
results in a list of lines in "buffer.c" fuzzy matching "str".
:echo ['one two', 'two one']->matchfuzzy('two one')
results in ['two one', 'one two'].
:echo ['one two', 'two one']->matchfuzzy('two one',
\ {'matchseq': 1})
results in ['two one'].

```

`matchfuzzypos({list}, {str} [, {dict}])` `matchfuzzypos()`
 Same as `matchfuzzy()`, but returns the list of matched strings, the list of character positions where characters in `{str}` matches and a list of matching scores. You can use `byteidx()` to convert a character position to a byte position.

If `{str}` matches multiple times in a string, then only the positions for the best match is returned.

If there are no matching strings or there is an error, then a list with three empty list items is returned.

Example:

```

:echo matchfuzzypos(['testing'], 'tsg')
results in [['testing'], [[0, 2, 6]], [99]]
:echo matchfuzzypos(['clay', 'lacy'], 'la')
results in [['lacy', 'clay'], [[0, 1], [1, 2]], [153, 133]]
:echo [{'text': 'hello', 'id' : 10}]->matchfuzzypos('ll', {'key' :
results in [[{'id': 10, 'text': 'hello'}], [[2, 3]], [127]]

```

`matchlist({expr}, {pat} [, {start} [, {count}])` `matchlist()`
 Same as `match()`, but return a `List`. The first item in the list is the matched string, same as what `matchstr()` would return. Following items are submatches, like "\1", "\2", etc. in `:substitute`. When an optional submatch didn't match an empty string is used. Example:

```

:echo matchlist('acd', '\(a\)?\(b\)?\(c\)?\(.*\)')
Results in: ['acd', 'a', '', 'c', 'd', '', '', '', '', '']
When there is no match an empty list is returned.

```

You can pass in a `List`, but that is not very useful.

Can also be used as a `method` :

```

GetText()->matchlist('word')

```

`matchstrlist({list}, {pat} [, {dict}])` `matchstrlist()`
 Returns the `List` of matches in `{list}` where `{pat}` matches. `{list}` is a `List` of strings. `{pat}` is matched against each string in `{list}`.

The `{dict}` argument supports following items:

submatches include submatch information (/ \ ()

For each match, a `Dict` with the following items is returned:

- `byteidx` starting byte index of the match.
- `idx` index in `{list}` of the match.
- `text` matched string
- `submatches` a List of submatches. Present only if "submatches" is set to `v:true` in `{dict}`.

See `match-pattern` for information about the effect of some option settings on the pattern.

Example:

```
:echo matchstrlist(['tik tok'], '\<k\+>')
[{'idx': 0, 'byteidx': 0, 'text': 'tik'}, {'idx': 0, 'byteidx': 4, 'text': 'tok'}]
:echo matchstrlist(['a', 'b'], '\<k\+>')
[{'idx': 0, 'byteidx': 0, 'text': 'a'}, {'idx': 1, 'byteidx': 0, 'text': 'b'}]
```

If "submatches" is present and is `v:true`, then submatches like `"\1"`, `"\2"`, etc. are also returned. Example:

```
:echo matchstrlist(['acd'], '\(a\)?\(b\)?\(c\)?\(.*\)')
\ # {submatches: v:true}
[{'idx': 0, 'byteidx': 0, 'text': 'acd', 'submatches': ['a', '', 'c', 'd']}]
```

The "submatches" List always contains 9 items. If a submatch is not found, then an empty string is returned for that submatch.

Can also be used as a `method` :

```
GetListOfStrings()->matchstrlist('mypat')
```

`matchstr({expr}, {pat} [, {start} [, {count}]])` `matchstr()`

Same as `match()` , but return the matched string. Example:

```
:echo matchstr("testing", "ing")
```

results in "ing".

When there is no match "" is returned.

The `{start}`, if given, has the same meaning as for `match()` .

```
:echo matchstr("testing", "ing", 2)
```

results in "ing".

```
:echo matchstr("testing", "ing", 5)
```

result is "".

When `{expr}` is a `List` then the matching item is returned.

The type isn't changed, it's not necessarily a String.

Can also be used as a `method` :

```
GetText()->matchstr('word')
```

`matchstrpos({expr}, {pat} [, {start} [, {count}]])` `matchstrpos()`

Same as `matchstr()` , but return the matched string, the start position and the end position of the match. Example:

```
:echo matchstrpos("testing", "ing")
```

results in ["ing", 4, 7].

When there is no match ["", -1, -1] is returned.

The `{start}`, if given, has the same meaning as for `match()` .

```
:echo matchstrpos("testing", "ing", 2)
```

results in ["ing", 4, 7].

```
:echo matchstrpos("testing", "ing", 5)
```

result is ["", -1, -1].

When `{expr}` is a `List` then the matching item, the index of first item where `{pat}` matches, the start position and the end position of the match are returned.

```
:echo matchstrpos([1, 'x'], '\a')
```

result is ["x", 1, 2, 3].

The type isn't changed, it's not necessarily a String.

Can also be used as a `method` :

```
GetText()->matchstrpos('word')
```

`max({expr})` `max()`
Return the maximum value of all items in `{expr}`. Example:

```
echo max([apples, pears, oranges])
```

`{expr}` can be a `List` or a `Dictionary` . For a `Dictionary`, it returns the maximum of all values in the `Dictionary`. If `{expr}` is neither a `List` nor a `Dictionary`, or one of the items in `{expr}` cannot be used as a `Number` this results in an error. An empty `List` or `Dictionary` results in zero.

Can also be used as a `method` :

```
mylist->max()
```

`menu_info({name} [, {mode}])` `menu_info()`
Return information about the specified menu `{name}` in mode `{mode}`. The menu name should be specified without the shortcut character ('&'). If `{name}` is "", then the top-level menu names are returned.

`{mode}` can be one of these strings:

```
"n" Normal
"v" Visual (including Select)
"o" Operator-pending
"i" Insert
"c" Cmd-line
"s" Select
"x" Visual
"t" Terminal-Job
"" Normal, Visual and Operator-pending
"!" Insert and Cmd-line
```

When `{mode}` is omitted, the modes for "" are used.

Returns a `Dictionary` containing the following items:

```
accel menu item accelerator text menu-text
display display name (name without '&')
enabled v:true if this menu item is enabled
Refer to :menu-enable
icon name of the icon file (for toolbar)
toolbar-icon
```

iconidx	index of a built-in icon
modes	modes for which the menu is defined. In addition to the modes mentioned above, these characters will be used: " " Normal, Visual and Operator-pending
name	menu item name.
noremenu	v:true if the {rhs} of the menu item is not remappable else v:false.
priority	menu order priority menu-priority
rhs	right-hand-side of the menu item. The returned string has special characters translated like in the output of the ":menu" command listing. When the {rhs} of a menu item is empty, then "<Nop>" is returned.
script	v:true if script-local remapping of {rhs} is allowed else v:false. See :menu-script .
shortcut	shortcut key (character after '&' in the menu name) menu-shortcut
silent	v:true if the menu item is created with <silent> argument :menu-silent
submenus	List containing the names of all the submenus. Present only if the menu item has submenus.

Returns an empty dictionary if the menu item is not found.

Examples:

```

:echo menu_info('Edit.Cut')
:echo menu_info('File.Save', 'n')

" Display the entire menu hierarchy in a buffer
func ShowMenu(name, pfx)
  let m = menu_info(a:name)
  call append(line('$'), a:pfx .. m.display)
  for child in m->get('submenus', [])
    call ShowMenu(a:name .. '.' .. escape(child, '.'),
                  \ a:pfx .. '  ')
  endfor
endfunc
new
for topmenu in menu_info('').submenus
  call ShowMenu(topmenu, '')
endfor

```

Can also be used as a [method](#) :

```
GetMenuName()->menu_info('v')
```

`min({expr})` [min\(\)](#)
Return the minimum value of all items in {expr}. Example:
`echo min([apples, pears, oranges])`

{expr} can be a [List](#) or a [Dictionary](#). For a Dictionary, it returns the minimum of all values in the Dictionary.

If `{expr}` is neither a List nor a Dictionary, or one of the items in `{expr}` cannot be used as a Number this results in an error. An empty List or Dictionary results in zero.

Can also be used as a method :
`mylist->min()`

`mkdir()` E739

`mkdir({name} [, {flags} [, {prot}]])`
Create directory `{name}`.

When `{flags}` is present it must be a string. An empty string has no effect.

If `{flags}` contains "p" then intermediate directories are created as necessary.

If `{flags}` contains "D" then `{name}` is deleted at the end of the current function, as with:
`defer delete({name}, 'd')`

If `{flags}` contains "R" then `{name}` is deleted recursively at the end of the current function, as with:
`defer delete({name}, 'rf')`

Note that when `{name}` has more than one part and "p" is used some directories may already exist. Only the first one that is created and what it contains is scheduled to be deleted. E.g. when using:

```
call mkdir('subdir/tmp/autoload', 'pR')
```

and "subdir" already exists then "subdir/tmp" will be scheduled for deletion, like with:

```
defer delete('subdir/tmp', 'rf')
```

Note that if scheduling the defer fails the directory is not deleted. This should only happen when out of memory.

If `{prot}` is given it is used to set the protection bits of the new directory. The default is 0o755 (rwxr-xr-x: r/w for the user, readable for others). Use 0o700 to make it unreadable for others. This is only used for the last part of `{name}`. Thus if you create /tmp/foo/bar then /tmp/foo will be created with 0o755.

Example:

```
:call mkdir($HOME .. "/tmp/foo/bar", "p", 0o700)
```

This function is not available in the `sandbox` .

There is no error if the directory already exists and the "p" flag is passed (since patch 8.0.1708). However, without the "p" option the call will fail.

The function result is a Number, which is TRUE if the call was successful or FALSE if the directory creation failed or partly failed.

Not available on all systems. To check use:
`:if exists("*mkdir")`

Can also be used as a `method` :
`GetName()->mkdir()`

`mode([expr])` `mode()`
Return a string that indicates the current mode.
If `[expr]` is supplied and it evaluates to a non-zero Number or a non-empty String (`non-zero-arg`), then the full mode is returned, otherwise only the first letter is returned.
Also see `state()` .

<code>n</code>	Normal
<code>no</code>	Operator-pending
<code>nov</code>	Operator-pending (forced characterwise <code>o_v</code>)
<code>noV</code>	Operator-pending (forced linewise <code>o_V</code>)
<code>noCTRL-V</code>	Operator-pending (forced blockwise <code>o_CTRL-V</code>); CTRL-V is one character
<code>niI</code>	Normal using <code>i_CTRL-0</code> in <code>Insert-mode</code>
<code>niR</code>	Normal using <code>i_CTRL-0</code> in <code>Replace-mode</code>
<code>niV</code>	Normal using <code>i_CTRL-0</code> in <code>Virtual-Replace-mode</code>
<code>nt</code>	Terminal-Normal (insert goes to Terminal-Job mode)
<code>v</code>	Visual by character
<code>vs</code>	Visual by character using <code>v_CTRL-0</code> in Select mode
<code>V</code>	Visual by line
<code>Vs</code>	Visual by line using <code>v_CTRL-0</code> in Select mode
CTRL-V	Visual blockwise
CTRL-Vs	Visual blockwise using <code>v_CTRL-0</code> in Select mode
<code>s</code>	Select by character
<code>S</code>	Select by line
CTRL-S	Select blockwise
<code>i</code>	Insert
<code>ic</code>	Insert mode completion <code>compl-generic</code>
<code>ix</code>	Insert mode <code>i_CTRL-X</code> completion
<code>R</code>	Replace <code>R</code>
<code>Rc</code>	Replace mode completion <code>compl-generic</code>
<code>Rx</code>	Replace mode <code>i_CTRL-X</code> completion
<code>Rv</code>	Virtual Replace <code>gR</code>
<code>Rvc</code>	Virtual Replace mode completion <code>compl-generic</code>
<code>Rvx</code>	Virtual Replace mode <code>i_CTRL-X</code> completion
<code>c</code>	Command-line editing
<code>ct</code>	Command-line editing via Terminal-Job mode
<code>cr</code>	Command-line editing overstrike mode <code>c_<Insert></code>
<code>cv</code>	Vim Ex mode <code>gQ</code>
<code>cvr</code>	Vim Ex mode while in overstrike mode <code>c_<Insert></code>
<code>ce</code>	Normal Ex mode <code>Q</code>
<code>r</code>	Hit-enter prompt
<code>rm</code>	The -- more -- prompt
<code>r?</code>	A <code>:confirm</code> query of some sort
<code>!</code>	Shell or external command is executing
<code>t</code>	Terminal-Job mode: keys go to the job

This is useful in the `'statusline'` option or when used

with `remote_expr()` In most other places it always returns "c" or "n".

Note that in the future more modes and more specific modes may be added. It's better not to compare the whole string but only the leading character(s).

Also see `visualmode()` .

Can also be used as a **method** :
`DoFull()->mode()`

`mzeval({expr})`

`mzeval()`

Evaluate MzScheme expression `{expr}` and return its result converted to Vim data structures.

Numbers and strings are returned as they are.

Pairs (including lists and improper lists) and vectors are returned as Vim **Lists** .

Hash tables are represented as Vim **Dictionary** type with keys converted to strings.

All other types are converted to string with display function.

Examples:

```
:mz (define l (list 1 2 3))
:mz (define h (make-hash)) (hash-set! h "list" l)
:echo mzeval("l")
:echo mzeval("h")
```

Note that in a `:def` function local variables are not visible to {expr}.`

Can also be used as a **method** :
`GetExpr()->mzeval()`

`{only available when compiled with the |+mzscheme| feature}`

`nextnonblank({lnum})`

`nextnonblank()`

Return the line number of the first line at or below `{lnum}` that is not blank. Example:

```
if getline(nextnonblank(1)) =~ "Java"
```

When `{lnum}` is invalid or there is no non-blank line at or below it, zero is returned.

`{lnum}` is used like with `getline()` .

See also `prevnonblank()` .

Can also be used as a **method** :
`GetLnum()->nextnonblank()`

`nr2char({expr} [, {utf8}])`

`nr2char()`

Return a string with a single character, which has the number value `{expr}`. Examples:

```
nr2char(64)           returns "@"
nr2char(32)           returns " "
```

When `{utf8}` is omitted or zero, the current **'encoding'** is used. Example for "utf-8":

```
nr2char(300)          returns I with bow character
```

When `{utf8}` is TRUE, always return UTF-8 characters.

Note that a NUL character in the file is specified with `nr2char(10)`, because NULs are represented with newline characters. `nr2char(0)` is a real NUL and terminates the string, thus results in an empty string.

To turn a list of character numbers into a string:

```
let list = [65, 66, 67]
let str = join(map(list, {_, val -> nr2char(val)}), '')
Result: "ABC"
```

Can also be used as a **method** :

```
GetNumber()->nr2char()
```

`or({expr}, {expr})` **or()**

Bitwise OR on the two arguments. The arguments are converted to a number. A List, Dict or Float argument causes an error. Also see ``and()`` and ``xor()``.

Example:

```
:let bits = or(bits, 0x80)
```

Can also be used as a **method** :

```
:let bits = bits->or(0x80)
```

Rationale: The reason this is a function and not using the `"|"` character like many languages, is that Vi has always used `"|"` to separate commands. In many places it would not be clear if `"|"` is an operator or a command separator.

`pathshorten({path} [, {len}])` **pathshorten()**

Shorten directory names in the path `{path}` and return the result. The tail, the file name, is kept as-is. The other components in the path are reduced to `{len}` letters in length. If `{len}` is omitted or smaller than 1 then 1 is used (single letters). Leading `'~'` and `'.'` characters are kept. Examples:

```
:echo pathshorten('~/.vim/autoload/myfile.vim')
~/.v/a/myfile.vim
```

```
:echo pathshorten('~/.vim/autoload/myfile.vim', 2)
~/.vi/au/myfile.vim
```

It doesn't matter if the path exists or not.

Returns an empty string on error.

Can also be used as a **method** :

```
GetDirectories()->pathshorten()
```

`perlevel({expr})` **perlevel()**

Evaluate Perl expression `{expr}` in scalar context and return its result converted to Vim data structures. If value can't be converted, it is returned as a string Perl representation.

Note: If you want an array or hash, `{expr}` must return a reference to it.

Example:

```
:echo perlevel('[1 .. 4]')
[1, 2, 3, 4]
```

Note that in a `:def` function local variables are not visible to {expr}.`

Can also be used as a `method` :
`GetExpr()->perlevel()`

{only available when compiled with the `|+perl|` feature}

`popup_` functions are documented here: [popup-functions](#)

`pow({x}, {y})`

`pow()`

Return the power of `{x}` to the exponent `{y}` as a `Float` .
`{x}` and `{y}` must evaluate to a `Float` or a `Number` .
Returns 0.0 if `{x}` or `{y}` is not a `Float` or a `Number` .
Examples:

```
:echo pow(3, 3)
27.0
:echo pow(2, 16)
65536.0
:echo pow(32, 0.20)
2.0
```

Can also be used as a `method` :
`Compute()->pow(3)`

`prevnonblank({lnum})`

`prevnonblank()`

Return the line number of the first line at or above `{lnum}` that is not blank. Example:

```
let ind = indent(prevnonblank(v:lnum - 1))
```

When `{lnum}` is invalid or there is no non-blank line at or above it, zero is returned.

`{lnum}` is used like with `getline()` .

Also see `nextnonblank()` .

Can also be used as a `method` :

`GetLnum()->prevnonblank()`

`printf({fmt}, {expr1} ...)`

`printf()`

Return a String with `{fmt}`, where `"%"` items are replaced by the formatted form of their respective arguments. Example:

```
printf("%4d: E%d %.30s", lnum, errno, msg)
```

May result in:

```
" 99: E42 asdfasdfasdfasdfasdfasdfas"
```

When used as a `method` the base is passed as the second argument:

```
Compute()->printf("result: %d")
```

You can use ``call()`` to pass the items as a list.

Often used items are:

```
%s    string
```

```

%6S  string right-aligned in 6 display cells
%6s  string right-aligned in 6 bytes
%.9s string truncated to 9 bytes
%c   single byte
%d   decimal number
%5d  decimal number padded with spaces to 5 characters
%x   hex number
%04x hex number padded with zeros to at least 4 characters
%X   hex number using upper case letters
%o   octal number
%08b binary number padded with zeros to at least 8 chars
%f   floating point number as 12.23, inf, -inf or nan
%F   floating point number as 12.23, INF, -INF or NAN
%e   floating point number as 1.23e3, inf, -inf or nan
%E   floating point number as 1.23E3, INF, -INF or NAN
%g   floating point number, as %f or %e depending on value
%G   floating point number, as %F or %E depending on value
%%   the % character itself

```

Conversion specifications start with '%' and end with the conversion type. All other characters are copied unchanged to the result.

The "%" starts a conversion specification. The following arguments appear in sequence:

```
% [pos-argument] [flags] [field-width] [.precision] type
```

pos-argument

At most one positional argument specifier. These take the form {n\$}, where n is >= 1.

flags

Zero or more of the following flags:

- # The value should be converted to an "alternate form". For c, d, and s conversions, this option has no effect. For o conversions, the precision of the number is increased to force the first character of the output string to a zero (except if a zero value is printed with an explicit precision of zero). For b and B conversions, a non-zero result has the string "0b" (or "0B" for B conversions) prepended to it. For x and X conversions, a non-zero result has the string "0x" (or "0X" for X conversions) prepended to it.
- 0 (zero) Zero padding. For all conversions the converted value is padded on the left with zeros rather than blanks. If a precision is given with a numeric conversion (d, b, B, o, x, and X), the 0 flag is ignored.

- A negative field width flag; the converted value is to be left adjusted on the field boundary. The converted value is padded on the right with blanks, rather than on the left with blanks or zeros. A - overrides a 0 if both are given.
- ' ' (space) A blank should be left before a positive number produced by a signed conversion (d).
- + A sign must always be placed before a number produced by a signed conversion. A + overrides a space if both are used.

field-width

An optional decimal digit string specifying a minimum field width. If the converted value has fewer bytes than the field width, it will be padded with spaces on the left (or right, if the left-adjustment flag has been given) to fill out the field width. For the S conversion the count is in cells.

.precision

An optional precision, in the form of a period '.' followed by an optional digit string. If the digit string is omitted, the precision is taken as zero. This gives the minimum number of digits to appear for d, o, x, and X conversions, the maximum number of bytes to be printed from a string for s conversions, or the maximum number of cells to be printed from a string for S conversions. For floating point it is the number of digits after the decimal point.

type

A character that specifies the type of conversion to be applied, see below.

A field width or precision, or both, may be indicated by an asterisk '*' instead of a digit string. In this case, a Number argument supplies the field width or precision. A negative field width is treated as a left adjustment flag followed by a positive field width; a negative precision is treated as though it were missing. Example:

```
:echo printf("%d: %.*s", nr, width, line)
```

This limits the length of the text used from "line" to "width" bytes.

If the argument to be formatted is specified using a positional argument specifier, and a '*' is used to indicate that a number argument is to be used to specify the width or precision, the argument(s) to be used must also be specified using a {n\$} positional argument specifier. See [printf-\\$](#) .

The conversion specifiers and their meanings are:

	<code>printf-d</code>	<code>printf-b</code>	<code>printf-B</code>	<code>printf-o</code>	<code>printf-x</code>	<code>printf-X</code>
<code>dbBoxX</code>	The Number argument is converted to signed decimal (d), unsigned binary (b and B), unsigned octal (o), or unsigned hexadecimal (x and X) notation. The letters "abcdef" are used for x conversions; the letters "ABCDEF" are used for X conversions. The precision, if any, gives the minimum number of digits that must appear; if the converted value requires fewer digits, it is padded on the left with zeros. In no case does a non-existent or small field width cause truncation of a numeric field; if the result of a conversion is wider than the field width, the field is expanded to contain the conversion result. The 'h' modifier indicates the argument is 16 bits. The 'l' modifier indicates the argument is a long integer. The size will be 32 bits or 64 bits depending on your platform. The "ll" modifier indicates the argument is 64 bits. The b and B conversion specifiers never take a width modifier and always assume their argument is a 64 bit integer. Generally, these modifiers are not useful. They are ignored when type is known from the argument.					
<code>i</code>	alias for d					
<code>D</code>	alias for ld					
<code>U</code>	alias for lu					
<code>O</code>	alias for lo					
			<code>printf-c</code>			
<code>c</code>	The Number argument is converted to a byte, and the resulting character is written.					
			<code>printf-s</code>			
<code>s</code>	The text of the String argument is used. If a precision is specified, no more bytes than the number specified are used. If the argument is not a String type, it is automatically converted to text with the same format as ":echo".					
			<code>printf-S</code>			
<code>S</code>	The text of the String argument is used. If a precision is specified, no more display cells than the number specified are used.					
			<code>printf-f</code>	<code>E807</code>		
<code>f F</code>	The Float argument is converted into a string of the form 123.456. The precision specifies the number of digits after the decimal point. When the precision is zero the decimal point is omitted. When the precision					

is not specified 6 is used. A really big number (out of range or dividing by zero) results in "inf" or "-inf" with %f (INF or -INF with %F). "0.0 / 0.0" results in "nan" with %f (NAN with %F). Example:

```
echo printf("%.2f", 12.115)
12.12
```

Note that roundoff depends on the system libraries. Use `round()` when in doubt.

- `printf-e` `printf-E`

e E The Float argument is converted into a string of the form 1.234e+03 or 1.234E+03 when using 'E'. The precision specifies the number of digits after the decimal point, like with 'f'.
- `printf-g` `printf-G`

g G The Float argument is converted like with 'f' if the value is between 0.001 (inclusive) and 10000000.0 (exclusive). Otherwise 'e' is used for 'g' and 'E' for 'G'. When no precision is specified superfluous zeroes and '+' signs are removed, except for the zero immediately after the decimal point. Thus 10000000.0 results in 1.0e7.
- `printf-%`

% A '%' is written. No argument is converted. The complete conversion specification is "%%".

When a Number argument is expected a String argument is also accepted and automatically converted.
When a Float or String argument is expected a Number argument is also accepted and automatically converted.
Any other argument type results in an error message.

`E766` `E767`

The number of `{exprN}` arguments must exactly match the number of "%" items. If there are not sufficient or too many arguments an error is given. Up to 18 arguments can be used.

`printf-$`

In certain languages, error and informative messages are more readable when the order of words is different from the corresponding message in English. To accommodate translations having a different word order, positional arguments may be used to indicate this. For instance:

```
#, c-format
msgid "%s returning %s"
msgstr "waarde %2$s komt terug van %1$s"
```

In this example, the sentence has its 2 string arguments reversed in the output.

```
echo printf(
    "In The Netherlands, vim's creator's name is: %1$s %2$s",
    "Bram", "Moolenaar")
In The Netherlands, vim's creator's name is: Bram Moolenaar
```

```
echo printf(
    "In Belgium, vim's creator's name is: %2$s %1$s",
    "Bram", "Moolenaar")
In Belgium, vim's creator's name is: Moolenaar Bram
```

Width (and precision) can be specified using the '*' specifier. In this case, you must specify the field width position in the argument list.

```
echo printf("%1$*2$.*3$d", 1, 2, 3)
001
echo printf("%2$*3$.*1$d", 1, 2, 3)
2
echo printf("%3$*1$.*2$d", 1, 2, 3)
03
echo printf("%1$*2$.*3$g", 1.4142, 2, 3)
1.414
```

You can mix specifying the width and/or precision directly and via positional arguments:

```
echo printf("%1$4.*2$f", 1.4142135, 6)
1.414214
echo printf("%1$*2$.4f", 1.4142135, 6)
1.4142
echo printf("%1$*2$.*3$f", 1.4142135, 6, 2)
1.41
```

E1500

You cannot mix positional and non-positional arguments:

```
echo printf("%s%1$s", "One", "Two")
E1500: Cannot mix positional and non-positional arguments:
%s%1$s
```

E1501

You cannot skip a positional argument in a format string:

```
echo printf("%3$s%1$s", "One", "Two", "Three")
E1501: format argument 2 unused in $-style format:
%3$s%1$s
```

E1502

You can re-use a [field-width] (or [precision]) argument:

```
echo printf("%1$d at width %2$d is: %01$*2$d", 1, 2)
1 at width 2 is: 01
```

However, you can't use it as a different type:

```
echo printf("%1$d at width %2$ld is: %01$*2$d", 1, 2)
E1502: Positional argument 2 used as field width reused as
different type: long int/int
```


E1503

When a positional argument is used, but not the correct number or arguments is given, an error is raised:

```
echo printf("%1$d at width %2$d is: %01$*2$.*3$d", 1, 2)
E1503: Positional argument 3 out of bounds: %1$d at width
%2$d is: %01$*2$.*3$d
```

Only the first error is reported:

```
echo printf("%01$*2$.*3$d %4$d", 1, 2)
E1503: Positional argument 3 out of bounds: %01$*2$.*3$d
%4$d
```

E1504

A positional argument can be used more than once:

```
echo printf("%1$s %2$s %1$s", "One", "Two")
One Two One
```

However, you can't use a different type the second time:

```
echo printf("%1$s %2$s %1$d", "One", "Two")
E1504: Positional argument 1 type used inconsistently:
int/string
```

E1505

Various other errors that lead to a format string being wrongly formatted lead to:

```
echo printf("%1$d at width %2$d is: %01$*2$.3$d", 1, 2)
E1505: Invalid format specifier: %1$d at width %2$d is:
%01$*2$.3$d
```

E1507

This internal error indicates that the logic to parse a positional format argument ran into a problem that couldn't be otherwise reported. Please file a bug against Vim if you run into this, copying the exact format string and parameters that were used.

`prompt_getprompt({buf})`

`prompt_getprompt()`

Returns the effective prompt text for buffer `{buf}`. `{buf}` can be a buffer name or number. See [prompt-buffer](#).

If the buffer doesn't exist or isn't a prompt buffer, an empty string is returned.

Can also be used as a [method](#) :

```
GetBuffer()->prompt_getprompt()
```

{only available when compiled with the `|+channel|` feature}

`prompt_setcallback({buf}, {expr})`

`prompt_setcallback()`

Set prompt callback for buffer `{buf}` to `{expr}`. When `{expr}` is an empty string the callback is removed. This has only

effect if `{buf}` has `'buftype'` set to "prompt".

The callback is invoked when pressing Enter. The current buffer will always be the prompt buffer. A new line for a prompt is added before invoking the callback, thus the prompt for which the callback was invoked will be in the last but one line.

If the callback wants to add text to the buffer, it must insert it above the last line, since that is where the current prompt is. This can also be done asynchronously.

The callback is invoked with one argument, which is the text that was entered at the prompt. This can be an empty string if the user only typed Enter.

Example:

```
func s:TextEntered(text)
  if a:text == 'exit' || a:text == 'quit'
    stopinsert
    " Reset 'modified' to allow the buffer to be closed.
    " We assume there is nothing useful to be saved.
    set nomodified
    close
  else
    " Do something useful with "a:text". In this example
    " we just repeat it.
    call append(line('$') - 1, 'Entered: "' .. a:text .. "')
  endif
endfunc
call prompt_setcallback(bufnr(), function('s:TextEntered'))
```

Can also be used as a `method` :

```
GetBuffer()->prompt_setcallback(callback)
```

{only available when compiled with the |+channel| feature}

`prompt_setinterrupt({buf}, {expr})` `prompt_setinterrupt()`
Set a callback for buffer `{buf}` to `{expr}`. When `{expr}` is an empty string the callback is removed. This has only effect if `{buf}` has `'buftype'` set to "prompt".

This callback will be invoked when pressing **CTRL-C** in Insert mode. Without setting a callback Vim will exit Insert mode, as in any buffer.

Can also be used as a `method` :

```
GetBuffer()->prompt_setinterrupt(callback)
```

{only available when compiled with the |+channel| feature}

`prompt_setprompt({buf}, {text})` `prompt_setprompt()`
Set prompt for buffer `{buf}` to `{text}`. You most likely want `{text}` to end in a space.
The result is only visible if `{buf}` has `'buftype'` set to "prompt". Example:

```
call prompt_setprompt(bufnr(), 'command: ')
```

Can also be used as a [method](#) :
`GetBuffer()->prompt_setprompt('command: ')`

{only available when compiled with the |+channel| feature}

prop_ functions are documented here: [text-prop-functions](#)

`pum_getpos()` [pum_getpos\(\)](#)
If the popup menu (see [ins-completion-menu](#)) is not visible, returns an empty [Dictionary](#) , otherwise, returns a [Dictionary](#) with the following keys:

height	nr of items visible
width	screen cells
row	top screen row (0 first row)
col	leftmost screen column (0 first col)
size	total nr of items
scrollbar	<code>TRUE</code> if scrollbar is visible

The values are the same as in `v:event` during `CompleteChanged` .

`pumvisible()` [pumvisible\(\)](#)
Returns non-zero when the popup menu is visible, zero otherwise. See [ins-completion-menu](#) .
This can be used to avoid some things that would remove the popup menu.

`py3eval({expr})` [py3eval\(\)](#)
Evaluate Python expression `{expr}` and return its result converted to Vim data structures.
Numbers and strings are returned as they are (strings are copied though, Unicode strings are additionally converted to `'encoding'`).
Lists are represented as Vim [List](#) type.
Dictionaries are represented as Vim [Dictionary](#) type with keys converted to strings.
Note that in a `:def` function local variables are not visible to `{expr}` .

Can also be used as a [method](#) :
`GetExpr()->py3eval()`

{only available when compiled with the |+python3| feature}

`pyeval({expr})` [E858](#) [E859](#)
[pyeval\(\)](#)
Evaluate Python expression `{expr}` and return its result converted to Vim data structures.
Numbers and strings are returned as they are (strings are copied though).
Lists are represented as Vim [List](#) type.
Dictionaries are represented as Vim [Dictionary](#) type, non-string keys result in error.

Note that in a `:def` function local variables are not visible to `{expr}`.

Can also be used as a `method` :
`GetExpr()->pyeval()`

{only available when compiled with the `|+python|` feature}

`pyxeval({expr})` `pyxeval()`
Evaluate Python expression `{expr}` and return its result converted to Vim data structures.
Uses Python 2 or 3, see `python_x` and `'pyxversion'`.
See also: `pyeval()` , `py3eval()`

Can also be used as a `method` :
`GetExpr()->pyxeval()`

{only available when compiled with the `+python` or the `+python3` feature}

`rand([{expr}])` `rand()` `random`
Return a pseudo-random Number generated with an xoshiro128** algorithm using seed `{expr}`. The returned number is 32 bits, also on 64 bits systems, for consistency.
`{expr}` can be initialized by `srand()` and will be updated by `rand()`. If `{expr}` is omitted, an internal seed value is used and updated.
Returns -1 if `{expr}` is invalid.

Examples:

```
:echo rand()
:let seed = srand()
:echo rand(seed)
:echo rand(seed) % 16 " random number 0 - 15
```

`range({expr} [, {max} [, {stride}])` `E726` `E727`
`range()`
Returns a `List` with Numbers:
- If only `{expr}` is specified: `[0, 1, ..., {expr} - 1]`
- If `{max}` is specified: `[{expr}, {expr} + 1, ..., {max}]`
- If `{stride}` is specified: `[{expr}, {expr} + {stride}, ..., {max}]` (increasing `{expr}` with `{stride}` each time, not producing a value past `{max}`).

When the maximum is one before the start the result is an empty list. When the maximum is more than one before the start this is an error.

Examples:

```
range(4) " [0, 1, 2, 3]
range(2, 4) " [2, 3, 4]
range(2, 9, 3) " [2, 5, 8]
range(2, -2, -1) " [2, 1, 0, -1, -2]
range(0) " []
range(2, 0) " error!
```

Can also be used as a `method` :
`GetExpr()->range()`

`readblob({fname} [, {offset} [, {size}]])` `readblob()`

Read file `{fname}` in binary mode and return a `Blob` .

If `{offset}` is specified, read the file from the specified offset. If it is a negative value, it is used as an offset from the end of the file. E.g., to read the last 12 bytes:

```
readblob('file.bin', -12)
```

If `{size}` is specified, only the specified size will be read.

E.g. to read the first 100 bytes of a file:

```
readblob('file.bin', 0, 100)
```

If `{size}` is -1 or omitted, the whole data starting from `{offset}` will be read.

This can be also used to read the data from a character device on Unix when `{size}` is explicitly set. Only if the device supports seeking `{offset}` can be used. Otherwise it should be zero. E.g. to read 10 bytes from a serial console:

```
readblob('/dev/ttyS0', 0, 10)
```

When the file can't be opened an error message is given and the result is an empty `Blob` .

When the offset is beyond the end of the file the result is an empty blob.

When trying to read more bytes than are available the result is truncated.

Also see `readfile()` and `writefile()` .

`readdir({directory} [, {expr} [, {dict}]])` `readdir()`

Return a list with file and directory names in `{directory}`.

You can also use `glob()` if you don't need to do complicated things, such as limiting the number of matches.

The list will be sorted (case sensitive), see the `{dict}` argument below for changing the sort order.

When `{expr}` is omitted all entries are included.

When `{expr}` is given, it is evaluated to check what to do:

If `{expr}` results in -1 then no further entries will be handled.

If `{expr}` results in 0 then this entry will not be added to the list.

If `{expr}` results in 1 then this entry will be added to the list.

The entries "." and ".." are always excluded.

Each time `{expr}` is evaluated `v:val` is set to the entry name.

When `{expr}` is a function the name is passed as the argument.

For example, to get a list of files ending in ".txt":

```
readdir(dirname, {n -> n =~ '.txt$'})
```

To skip hidden and backup files:

```
readdir(dirname, {n -> n !~ '^\.|\~$'})
```

E857

The optional `{dict}` argument allows for further custom

values. Currently this is used to specify if and how sorting should be performed. The dict can have the following members:

```
sort    How to sort the result returned from the system.
        Valid values are:
            "none"    do not sort (fastest method)
            "case"    sort case sensitive (byte value of
                    each character, technically, using
                    strcmp()) (default)
            "icase"   sort case insensitive (technically
                    using strcasecmp())
            "collate" sort using the collation order
                    of the "POSIX" or "C" locale
                    (technically using strcoll())
        Other values are silently ignored.
```

For example, to get a list of all files in the current directory without sorting the individual entries:

```
readdir('.', '1', #{sort: 'none'})
```

If you want to get a directory tree:

```
function! s:tree(dir)
    return {a:dir : map(readdir(a:dir),
        \ {_, x -> isdirectory(x) ?
        \      {x : s:tree(a:dir .. '/' .. x)} : x)}}
endfunction
echo s:tree(".")
```

Returns an empty List on error.

Can also be used as a `method` :
`GetDirName()->readdir()`

```
readdir({directory} [, {expr} [, {dict}]]] readdir()
```

Extended version of `readdir()` .

Return a list of Dictionaries with file and directory information in `{directory}`.

This is useful if you want to get the attributes of file and directory at the same time as getting a list of a directory.

This is much faster than calling `readdir()` then calling `getfperm()` , `getfsize()` , `getftime()` and `getftype()` for each file and directory especially on MS-Windows.

The list will by default be sorted by name (case sensitive), the sorting can be changed by using the optional `{dict}` argument, see `readdir()` .

The Dictionary for file and directory information has the following items:

```
group  Group name of the entry. (Only on Unix)
name   Name of the entry.
perm   Permissions of the entry. See getfperm() .
size   Size of the entry. See getfsize() .
time   Timestamp of the entry. See getftime() .
type   Type of the entry.
       On Unix, almost same as getftype() except:
```

Symlink to a dir	"linkd"
Other symlink	"link"
On MS-Windows:	
Normal file	"file"
Directory	"dir"
Junction	"junction"
Symlink to a dir	"linkd"
Other symlink	"link"
Other reparse point	"reparse"

user User name of the entry's owner. (Only on Unix)

On Unix, if the entry is a symlink, the Dictionary includes the information of the target (except the "type" item).

On MS-Windows, it includes the information of the symlink itself because of performance reasons.

When `{expr}` is omitted all entries are included.

When `{expr}` is given, it is evaluated to check what to do:

If `{expr}` results in -1 then no further entries will be handled.

If `{expr}` results in 0 then this entry will not be added to the list.

If `{expr}` results in 1 then this entry will be added to the list.

The entries "." and ".." are always excluded.

Each time `{expr}` is evaluated `v:val` is set to a Dictionary of the entry.

When `{expr}` is a function the entry is passed as the argument.

For example, to get a list of files ending in ".txt":

```
readdirx(dirname, {e -> e.name =~ '.txt$'})
```

For example, to get a list of all files in the current directory without sorting the individual entries:

```
readdirx(dirname, '1', #{sort: 'none'})
```

Can also be used as a `method` :

```
GetDirName()->readdirx()
```

`readfile()`

```
readfile({fname} [, {type} [, {max}]])
```

Read file `{fname}` and return a `List`, each line of the file as an item. Lines are broken at NL characters. Macintosh files separated with CR will result in a single long line (unless a NL appears somewhere).

All NUL characters are replaced with a NL character.

When `{type}` contains "b" binary mode is used:

- When the last line ends in a NL an extra empty list item is added.

- No CR characters are removed.

Otherwise:

- CR characters that appear before a NL are removed.

- Whether the last line ends in a NL or not does not matter.

- When `'encoding'` is Unicode any UTF-8 byte order mark is removed from the text.

When `{max}` is given this specifies the maximum number of lines to be read. Useful if you only want to check the first ten lines of a file:

```
:for line in readfile(fname, '', 10)
: if line =~ 'Date' | echo line | endif
:endifor
```

When `{max}` is negative `-{max}` lines from the end of the file are returned, or as many as there are.

When `{max}` is zero the result is an empty list.

Note that without `{max}` the whole file is read into memory. Also **note** that there is no recognition of encoding. Read a file into a buffer if you need to.

Deprecated (use `readblob()` instead): When `{type}` contains "B" a `Blob` is returned with the binary data of the file unmodified.

When the file can't be opened an error message is given and the result is an empty list.

Also see `writefile()` .

Can also be used as a `method` :

```
GetFileName()->readfile()
```

`reduce({object}, {func} [, {initial}])` `reduce()` E998

`{func}` is called for every item in `{object}`, which can be a `String` , `List` or a `Blob` . `{func}` is called with two arguments: the result so far and current item. After processing all items the result is returned. E1132

`{initial}` is the initial result. When omitted, the first item in `{object}` is used and `{func}` is first called for the second item. If `{initial}` is not given and `{object}` is empty no result can be computed, an E998 error is given.

Examples:

```
echo reduce([1, 3, 5], { acc, val -> acc + val })
echo reduce(['x', 'y'], { acc, val -> acc .. val }, 'a')
echo reduce(0z1122, { acc, val -> 2 * acc + val })
echo reduce('xyz', { acc, val -> acc .. ',' .. val })
```

Can also be used as a `method` :

```
echo mylist->reduce({ acc, val -> acc + val }, 0)
```

`reg_executing()` `reg_executing()`

Returns the single letter name of the register being executed. Returns an empty string when no register is being executed.

See `@` .

`reg_recording()` `reg_recording()`

Returns the single letter name of the register being recorded.

Returns an empty string when not recording. See `q` .

`reltime()`

`reltime({start})`

`reltime({start}, {end})` `reltime()`

Return an item that represents a time value. The item is a list with items that depend on the system. In Vim 9 script the type `list<any>` can be used. The item can be passed to `reltimestr()` to convert it to a string or `reltimefloat()` to convert to a Float. For example, to see the time spent in function `Work()`:

```
var startTime = reltime()
Work()
echo startTime->reltime()->reltimestr()
```

Without an argument `reltime()` returns the current time (the representation is system-dependent, it cannot be used as the wall-clock time, see `localtime()` for that). With one argument it returns the time passed since the time specified in the argument. With two arguments it returns the time passed between `{start}` and `{end}`.

The `{start}` and `{end}` arguments must be values returned by `reltime()`. If there is an error an empty List is returned in legacy script, in Vim9 script an error is given.

Can also be used as a `method` :

```
GetStart()->reltime()
```

{only available when compiled with the `|+reltime|` feature}

`reltimefloat({time})` `reltimefloat()`

Return a Float that represents the time value of `{time}`. Example:

```
let start = reltime()
call MyFunction()
let seconds = reltimefloat(reltime(start))
```

See the [note](#) of `reltimestr()` about overhead. Also see [profiling](#) .

If there is an error 0.0 is returned in legacy script, in Vim9 script an error is given.

Can also be used as a `method` :

```
reltime(start)->reltimefloat()
```

{only available when compiled with the `|+reltime|` feature}

`reltimestr({time})` `reltimestr()`

Return a String that represents the time value of `{time}`. This is the number of seconds, a dot and the number of microseconds. Example:

```
let start = reltime()
call MyFunction()
echo reltimestr(reltime(start))
```

Note that overhead for the commands will be added to the time. The accuracy depends on the system. Use `reltimefloat()` for the greatest accuracy which is nanoseconds on some systems.

Leading spaces are used to make the string align nicely. You can use `split()` to remove it.

```
echo split(reltimestr(reftime(start)))[0]
```

Also see [profiling](#) .

If there is an error an empty string is returned in legacy script, in Vim9 script an error is given.

Can also be used as a [method](#) :

```
reftime(start)->reltimestr()
```

{only available when compiled with the |+reftime| feature}

[remote_expr\(\)](#) E449

```
remote_expr({server}, {string} [, {idvar} [, {timeout}]])
```

Send the `{string}` to `{server}`. The `{server}` argument is a string, also see `{server}` .

The string is sent as an expression and the result is returned after evaluation. The result must be a String or a [List](#) . A [List](#) is turned into a String by joining the items with a line break in between (not at the end), like with `join(expr, "\n")`.

If `{idvar}` is present and not empty, it is taken as the name of a variable and a `{serverid}` for later use with `remote_read()` is stored there.

If `{timeout}` is given the read times out after this many seconds. Otherwise a timeout of 600 seconds is used.

See also [clientserver RemoteReply](#) .

This function is not available in the [sandbox](#) .

{only available when compiled with the |+clientserver| feature}

Note: Any errors will cause a local error message to be issued and the result will be the empty string.

Variables will be evaluated in the global namespace, independent of a function currently being active. Except when in debug mode, then local function variables and arguments can be evaluated.

Examples:

```
:echo remote_expr("gvim", "2+2")
:echo remote_expr("gvim1", "b:current_syntax")
```

Can also be used as a [method](#) :

```
ServerName()->remote_expr(expr)
```

```
remote_foreground({server}) remote\_foreground\(\)
```

Move the Vim server with the name `{server}` to the foreground.

The `{server}` argument is a string, also see `{server}` .

This works like:

```
remote_expr({server}, "foreground()")
```

Except that on Win32 systems the client does the work, to work

around the problem that the OS doesn't always allow the server to bring itself to the foreground.

Note: This does not restore the window if it was minimized, like `foreground()` does.

This function is not available in the `sandbox` .

Can also be used as a `method` :

```
ServerName()->remote_foreground()
```

{only in the Win32, Motif and GTK GUI versions and the Win32 console version}

```
remote_peek({serverid} [, {retvar}])           remote_peek()
Returns a positive number if there are available strings
from {serverid}. Copies any reply string into the variable
{retvar} if specified. {retvar} must be a string with the
name of a variable.
Returns zero if none are available.
Returns -1 if something is wrong.
See also clientserver .
This function is not available in the sandbox .
{only available when compiled with the |+clientserver| feature}
Examples:
```

```
:let repl = ""
:echo "PEEK: " .. remote_peek(id, "repl") .. ": " .. repl
```

Can also be used as a `method` :

```
ServerId()->remote_peek()
```

```
remote_read({serverid}, [{timeout}])          remote_read()
Return the oldest available reply from {serverid} and consume
it. Unless a {timeout} in seconds is given, it blocks until a
reply is available. Returns an empty string, if a reply is
not available or on error.
See also clientserver .
This function is not available in the sandbox .
{only available when compiled with the |+clientserver| feature}
Example:
```

```
:echo remote_read(id)
```

Can also be used as a `method` :

```
ServerId()->remote_read()
```

```
remote_send({server}, {string} [, {idvar}])   remote_send() E241
Send the {string} to {server}. The {server} argument is a
string, also see {server} .
```

The string is sent as input keys and the function returns immediately. At the Vim server the keys are not mapped
`:map` .

If `{idvar}` is present, it is taken as the name of a variable

and a `{serverid}` for later use with `remote_read()` is stored there.

See also `clientserver RemoteReply` .

This function is not available in the `sandbox` .

`{only available when compiled with the |+clientserver| feature}`

Note: Any errors will be reported in the server and may mess up the display.

Examples:

```
:echo remote_send("gvim", ":DropAndReply " .. file, "serverid") ..  
  \ remote_read(serverid)
```

```
:autocmd NONE RemoteReply *  
  \ echo remote_read(expand("<amatch>"))  
:echo remote_send("gvim", ":sleep 10 | echo " ..  
  \ 'server2client(expand("<client>"), "HELLO")<CR>')
```

Can also be used as a `method` :

```
ServerName()->remote_send(keys)
```

`remote_startserver()` E941 E942

`remote_startserver({name})`

Become the server `{name}`. `{name}` must be a non-empty string. This fails if already running as a server, when `v:servername` is not empty.

Can also be used as a `method` :

```
ServerName()->remote_startserver()
```

`{only available when compiled with the |+clientserver| feature}`

`remove({list}, {idx})`

`remove({list}, {idx}, {end})` `remove()`

Without `{end}`: Remove the item at `{idx}` from `List {list}` and return the item.

With `{end}`: Remove items from `{idx}` to `{end}` (inclusive) and return a `List` with these items. When `{idx}` points to the same item as `{end}` a list with one item is returned. When `{end}` points to an item before `{idx}` this is an error.

See `list-index` for possible values of `{idx}` and `{end}`.

Returns zero on error.

Example:

```
:echo "last item: " .. remove(mylist, -1)  
:call remove(mylist, 0, 9)
```

Use `delete()` to remove a file.

Can also be used as a `method` :

```
mylist->remove(idx)
```

`remove({blob}, {idx})`

`remove({blob}, {idx}, {end})`

Without `{end}`: Remove the byte at `{idx}` from `Blob {blob}` and

return the byte.

With `{end}`: Remove bytes from `{idx}` to `{end}` (inclusive) and return a `Blob` with these bytes. When `{idx}` points to the same byte as `{end}` a `Blob` with one byte is returned. When `{end}` points to a byte before `{idx}` this is an error.

Returns zero on error.

Example:

```
:echo "last byte: " .. remove(myblob, -1)
:call remove(mylist, 0, 9)
```

`remove({dict}, {key})`

Remove the entry from `{dict}` with key `{key}` and return it.

Example:

```
:echo "removed " .. remove(dict, "one")
```

If there is no `{key}` in `{dict}` this is an error.

Returns zero on error.

`rename({from}, {to})`

`rename()`

Rename the file by the name `{from}` to the name `{to}`. This should also work to move files across file systems. The result is a Number, which is 0 if the file was renamed successfully, and non-zero when the renaming failed.

NOTE: If `{to}` exists it is overwritten without warning.

This function is not available in the `sandbox`.

Can also be used as a `method` :

```
GetOldName()->rename(newname)
```

`repeat({expr}, {count})`

`repeat()`

Repeat `{expr}` `{count}` times and return the concatenated result. Example:

```
:let separator = repeat('-', 80)
```

When `{count}` is zero or negative the result is empty.

When `{expr}` is a `List` or a `Blob` the result is `{expr}` concatenated `{count}` times. Example:

```
:let longlist = repeat(['a', 'b'], 3)
```

Results in `['a', 'b', 'a', 'b', 'a', 'b']`.

Can also be used as a `method` :

```
mylist->repeat(count)
```

`resolve({filename})`

`resolve()` E655

On MS-Windows, when `{filename}` is a shortcut (a `.lnk` file), returns the path the shortcut points to in a simplified form.

When `{filename}` is a symbolic link or junction point, return the full path to the target. If the target of junction is removed, return `{filename}`.

On Unix, repeat resolving symbolic links in all path components of `{filename}` and return the simplified result.

To cope with link cycles, resolving of symbolic links is stopped after 100 iterations.

On other systems, return the simplified `{filename}`.

The simplification step is done as by `simplify()`.

`resolve()` keeps a leading path component specifying the

current directory (provided the result is still a relative path name) and also keeps a trailing path separator.

Can also be used as a `method` :
`GetName()->resolve()`

`reverse({object})` `reverse()`
Reverse the order of items in `{object}`. `{object}` can be a `List`, a `Blob` or a `String`. For a `List` and a `Blob` the items are reversed in-place and `{object}` is returned. For a `String` a new `String` is returned. Returns zero if `{object}` is not a `List`, `Blob` or a `String`. If you want a `List` or `Blob` to remain unmodified make a copy first:

```
:let revlist = reverse(copy(mylist))
```

Can also be used as a `method` :
`mylist->reverse()`

`round({expr})` `round()`
Round off `{expr}` to the nearest integral value and return it as a `Float`. If `{expr}` lies halfway between two integral values, then use the larger one (away from zero). `{expr}` must evaluate to a `Float` or a `Number`. Returns 0.0 if `{expr}` is not a `Float` or a `Number`.
Examples:

```
echo round(0.456)
0.0
echo round(4.5)
5.0
echo round(-4.5)
-5.0
```

Can also be used as a `method` :
`Compute()->round()`

`rubyeval({expr})` `rubyeval()`
Evaluate Ruby expression `{expr}` and return its result converted to Vim data structures. Numbers, floats and strings are returned as they are (strings are copied though). Arrays are represented as Vim `List` type. Hashes are represented as Vim `Dictionary` type. Other objects are represented as strings resulted from their "Object#to_s" method.
Note that in a ``:def`` function local variables are not visible to `{expr}`.

Can also be used as a `method` :
`GetRubyExpr()->rubyeval()`

{only available when compiled with the |+ruby| feature}

`screenattr({row}, {col})` `screenattr()`
Like `screenchar()`, but return the attribute. This is a rather

arbitrary number that can only be used to compare to the attribute at other positions.
Returns -1 when row or col is out of range.

Can also be used as a [method](#) :
`GetRow()->screenattr(col)`

`screenchar({row}, {col})` [screenchar\(\)](#)

The result is a Number, which is the character at position [row, col] on the screen. This works for every possible screen position, also status lines, window separators and the command line. The top left position is row one, column one. The character excludes composing characters. For double-byte encodings it may only be the first byte. This is mainly to be used for testing.
Returns -1 when row or col is out of range.

Can also be used as a [method](#) :
`GetRow()->screenchar(col)`

`screenchars({row}, {col})` [screenchars\(\)](#)

The result is a [List](#) of Numbers. The first number is the same as what [screenchar\(\)](#) returns. Further numbers are composing characters on top of the base character. This is mainly to be used for testing.
Returns an empty List when row or col is out of range.

Can also be used as a [method](#) :
`GetRow()->screenchars(col)`

`screencol()` [screencol\(\)](#)

The result is a Number, which is the current screen column of the cursor. The leftmost column has number 1. This function is mainly used for testing.

Note: Always returns the current screen column, thus if used in a command (e.g. `":echo screencol()`") it will return the column inside the command line, which is 1 when the command is executed. To get the cursor position in the file use one of the following mappings:

```
noremap <expr> GG ":echom " .. screencol() .. "\n"  
noremap <silent> GG :echom screencol()<CR>  
noremap GG <Cmd>echom screencol()<CR>
```

`screenpos({winid}, {lnum}, {col})` [screenpos\(\)](#)

The result is a Dict with the screen position of the text character in window {winid} at buffer line {lnum} and column {col}. {col} is a one-based byte index.

The Dict has these members:

```
row      screen row  
col      first screen column  
endcol   last screen column  
curscol  cursor screen column
```

If the specified position is not visible, all values are zero.

The "endcol" value differs from "col" when the character occupies more than one screen cell. E.g. for a Tab "col" can be 1 and "endcol" can be 8.

The "curscol" value is where the cursor would be placed. For a Tab it would be the same as "endcol", while for a double width character it would be the same as "col".

The `conceal` feature is ignored here, the column numbers are as if '`conceallevel`' is zero. You can set the cursor to the right position and use `screencol()` to get the value with `conceal` taken into account.

If the position is in a closed fold the screen position of the first character is returned, `{col}` is not used.

Returns an empty Dict if `{winid}` is invalid.

Can also be used as a `method` :

```
GetWinid()->screenpos(lnum, col)
```

`screenrow()`

`screenrow()`

The result is a Number, which is the current screen row of the cursor. The top line has number one.

This function is mainly used for testing.

Alternatively you can use `winline()` .

Note: Same restrictions as with `screencol()` .

`screenstring({row}, {col})`

`screenstring()`

The result is a String that contains the base character and any composing characters at position `[row, col]` on the screen. This is like `screenchars()` but returning a String with the characters.

This is mainly to be used for testing.

Returns an empty String when `row` or `col` is out of range.

Can also be used as a `method` :

```
GetRow()->screenstring(col)
```

`search({pattern} [, {flags} [, {stoptline} [, {timeout} [, {skip}]]]])`

`search()`

Search for regexp pattern `{pattern}`. The search starts at the cursor position (you can use `cursor()` to set it).

When a match has been found its line number is returned.

If there is no match a 0 is returned and the cursor doesn't move. No error message is given.

To get the matched string, use `matchbufline()` .

`{flags}` is a String, which can contain these character flags:

'b'	search Backward instead of forward
'c'	accept a match at the Cursor position
'e'	move to the End of the match
'n'	do Not move the cursor
'p'	return number of matching sub-Pattern (see below)
's'	Set the ' mark at the previous location of the cursor
'w'	Wrap around the end of the file

'W' don't Wrap around the end of the file
'z' start searching at the cursor column instead of zero
If neither 'w' or 'W' is given, the **'wrapscan'** option applies.

If the 's' flag is supplied, the ' mark is set, only if the cursor is moved. The 's' flag cannot be combined with the 'n' flag.

'ignorecase', **'smartcase'** and **'magic'** are used.

When the 'z' flag is not given, forward searching always starts in column zero and then matches before the cursor are skipped. When the 'c' flag is present in **'cpo'** the next search starts after the match. Without the 'c' flag the next search starts one column after the start of the match. This matters for overlapping matches. See **cpo-c** . You can also insert "\ze" to change where the match ends, see **/\ze** .

When searching backwards and the 'z' flag is given then the search starts in column zero, thus no match in the current line will be found (unless wrapping around the end of the file).

When the `{stoptline}` argument is given then the search stops after searching this line. This is useful to restrict the search to a range of lines. Examples:

```
let match = search('(', 'b', line("w0"))
let end = search('END', '', line("w$"))
```

When `{stoptline}` is used and it is not zero this also implies that the search does not wrap around the end of the file. A zero value is equal to not giving the argument.

E1285 **E1286** **E1287** **E1288** **E1289**

When the `{timeout}` argument is given the search stops when more than this many milliseconds have passed. Thus when `{timeout}` is 500 the search stops after half a second. The value must not be negative. A zero value is like not giving the argument.

`{only available when compiled with the |+retime| feature}`

If the `{skip}` expression is given it is evaluated with the cursor positioned on the start of a match. If it evaluates to non-zero this match is skipped. This can be used, for example, to skip a match in a comment or a string.

`{skip}` can be a string, which is evaluated as an expression, a function reference or a lambda.

When `{skip}` is omitted or empty, every match is accepted.

When evaluating `{skip}` causes an error the search is aborted and -1 returned.

search()-sub-match

With the 'p' flag the returned value is one more than the first sub-match in `\(\)`. One if none of them matched but the whole pattern did match.

To get the column number too use **searchpos()** .

The cursor will be positioned at the match, unless the 'n' flag is used.

Example (goes over all files in the argument list):

```
:let n = 1
:while n <= argc()      " loop over all files in arglist
:  exe "argument " .. n
:  " start at the last char in the file and wrap for the
:  " first search to find match at start of file
:  normal G$
:  let flags = "w"
:  while search("foo", flags) > 0
:    s/foo/bar/g
:    let flags = "W"
:  endwhile
:  update                " write the file if modified
:  let n = n + 1
:endwhile
```

Example for using some flags:

```
:echo search('\<if\|\(else\)\|\(endif\)', 'ncpe')
```

This will search for the keywords "if", "else", and "endif" under or after the cursor. Because of the 'p' flag, it returns 1, 2, or 3 depending on which keyword is found, or 0 if the search fails. With the cursor on the first word of the line:

```
if (foo == 0) | let foo = foo + 1 | endif
```

the function returns 1. Without the 'c' flag, the function finds the "endif" and returns 3. The same thing happens without the 'e' flag if the cursor is on the "f" of "if". The 'n' flag tells the function not to move the cursor.

Can also be used as a [method](#) :

```
GetPattern()->search()
```

`searchcount([options])`

[searchcount\(\)](#)

Get or update the last search count, like what is displayed without the "S" flag in '[shortmess](#)'. This works even if '[shortmess](#)' does contain the "S" flag.

This returns a [Dictionary](#) . The dictionary is empty if the previous pattern was not set and "pattern" was not specified.

key	type	meaning
current	Number	current position of match; 0 if the cursor position is before the first match
exact_match	Boolean	1 if "current" is matched on "pos", otherwise 0
total	Number	total count of matches found
incomplete	Number	0: search was fully completed 1: recomputing was timed out 2: max count exceeded

For `{options}` see further down.

To get the last search count when `n` or `N` was pressed, call this function with ``recompute: 0``. This sometimes returns wrong information because `n` and `N`'s maximum count is 99. If it exceeded 99 the result must be max count + 1 (100). If you want to get correct information, specify ``recompute: 1``:

```
" result == maxcount + 1 (100) when many matches
let result = searchcount({recompute: 0})

" Below returns correct result (recompute defaults
" to 1)
let result = searchcount()
```

The function is useful to add the count to `'statusline'`:

```
function! LastSearchCount() abort
  let result = searchcount({recompute: 0})
  if empty(result)
    return ''
  endif
  if result.incomplete ==# 1      " timed out
    return printf(' /%s [?/??]', @/)
  elseif result.incomplete ==# 2 " max count exceeded
    if result.total > result.maxcount &&
      \ result.current > result.maxcount
      return printf(' /%s [>%d/>%d]', @/,
        \          result.current, result.total)
    elseif result.total > result.maxcount
      return printf(' /%s [%d/>%d]', @/,
        \          result.current, result.total)
    endif
  endif
  return printf(' /%s [%d/%d]', @/,
    \          result.current, result.total)
endfunction
let &statusline .= '%{LastSearchCount()}'

" Or if you want to show the count only when
" 'hlsearch' was on
" let &statusline .=
" \   '%{v:hlsearch ? LastSearchCount() : ""}'
```

You can also update the search count, which can be useful in a `CursorMoved` or `CursorMovedI` autocommand:

```
autocmd CursorMoved,CursorMovedI *
  \ let s:searchcount_timer = timer_start(
  \   200, function('s:update_searchcount'))
function! s:update_searchcount(timer) abort
  if a:timer ==# s:searchcount_timer
    call searchcount({
      \ recompute: 1, maxcount: 0, timeout: 100})
    redrawstatus
```

```

endif
endfunction

```

This can also be used to count matched texts with specified pattern in the current buffer using "pattern":

```

" Count '\<foo\>' in this buffer
" (Note that it also updates search count)
let result = searchcount({pattern: '\<foo\>'})

" To restore old search count by old pattern,
" search again
call searchcount()

```

{options} must be a **Dictionary** . It can contain:

key	type	meaning
recompute	Boolean	if TRUE , recompute the count like n or N was executed. otherwise returns the last computed result (when n or N was used when "S" is not in ' shortmess ', or this function was called). (default: TRUE)
pattern	String	recompute if this was given and different with @/ . this works as same as the below command is executed before calling this function <pre>let @/ = pattern</pre> (default: @/)
timeout	Number	0 or negative number is no timeout. timeout milliseconds for recomputing the result (default: 0)
maxcount	Number	0 or negative number is no limit. max count of matched text while recomputing the result. if search exceeded total count, "total" value becomes `maxcount + 1` (default: 99)
pos	List	`[lnum, col, off]` value when recomputing the result. this changes "current" result value. see cursor() , getpos() (default: cursor's position)

Can also be used as a **method** :

```
GetSearchOpts()->searchcount()
```

```
searchdecl({name} [, {global} [, {thisblock}]]) searchdecl()
Search for the declaration of {name}.
```

With a non-zero `{global}` argument it works like `gD`, find first match in the file. Otherwise it works like `gd`, find first match in the function.

With a non-zero `{thisblock}` argument matches in a `{}` block that ends before the cursor position are ignored. Avoids finding variable declarations only valid in another scope.

Moves the cursor to the found match.

Returns zero for success, non-zero for failure.

Example:

```
if searchdecl('myvar') == 0
    echo getline('.')
endif
```

Can also be used as a `method` :

```
GetName()->searchdecl()
```

`searchpair()`

```
searchpair({start}, {middle}, {end} [, {flags} [, {skip}
            [, {stoptline} [, {timeout}]]]])
```

Search for the match of a nested start-end pair. This can be used to find the "endif" that matches an "if", while other if/endif pairs in between are ignored.

The search starts at the cursor. The default is to search forward, include 'b' in `{flags}` to search backward.

If a match is found, the cursor is positioned at it and the line number is returned. If no match is found 0 or -1 is returned and the cursor doesn't move. No error message is given.

`{start}`, `{middle}` and `{end}` are patterns, see `pattern`. They must not contain `\(\)` pairs. Use of `\%(\)` is allowed. When `{middle}` is not empty, it is found when searching from either direction, but only when not in a nested start-end pair. A typical use is:

```
searchpair('\<if>', '\<else>', '\<endif>')
```

By leaving `{middle}` empty the "else" is skipped.

`{flags}` 'b', 'c', 'n', 's', 'w' and 'W' are used like with `search()`. Additionally:

'r' Repeat until no more matches found; will find the outer pair. Implies the 'W' flag.

'm' Return number of matches instead of line number with the match; will be > 1 when 'r' is used.

Note: it's nearly always a good idea to use the 'W' flag, to avoid wrapping around the end of the file.

When a match for `{start}`, `{middle}` or `{end}` is found, the `{skip}` expression is evaluated with the cursor positioned on the start of the match. It should return non-zero if this match is to be skipped. E.g., because it is inside a comment or a string.

When `{skip}` is omitted or empty, every match is accepted. When evaluating `{skip}` causes an error the search is aborted and -1 returned.

`{skip}` can be a string, a lambda, a funcref or a partial. Anything else makes the function fail.

In a `:def`` function when the `{skip}` argument is a string constant it is compiled into instructions.

For `{stopline}` and `{timeout}` see `search()` .

The value of `'ignorecase'` is used. `'magic'` is ignored, the patterns are used like it's on.

The search starts exactly at the cursor. A match with `{start}`, `{middle}` or `{end}` at the next character, in the direction of searching, is the first one found. Example:

```
if 1
  if 2
    endif 2
  endif 1
```

When starting at the "if 2", with the cursor on the "i", and searching forwards, the "endif 2" is found. When starting on the character just before the "if 2", the "endif 1" will be found. That's because the "if 2" will be found first, and then this is considered to be a nested if/endif from "if 2" to "endif 2".

When searching backwards and `{end}` is more than one character, it may be useful to put `"\zs"` at the end of the pattern, so that when the cursor is inside a match with the end it finds the matching start.

Example, to find the "endif" command in a Vim script:

```
:echo searchpair('\<if\>', '\<el\%[seif]\>', '\<en\%[dif]\>', 'W',
  \ 'getline(".") =~ "^\\s*\\\"")
```

The cursor must be at or after the "if" for which a match is to be found. **Note** that single-quote strings are used to avoid having to double the backslashes. The skip expression only catches comments at the start of a line, not after a command. Also, a word "en" or "if" halfway a line is considered a match.

Another example, to search for the matching "{" of a "}":

```
:echo searchpair('{', '', '}', 'bW')
```

This works when the cursor is at or before the "}" for which a match is to be found. To reject matches that syntax highlighting recognized as strings:

```
:echo searchpair('{', '', '}', 'bW',
  \ 'synIDattr(synID(line("."), col("."), 0), "name") =~? "string")
```

`searchpairpos()`

searchpairpos({start}, {middle}, {end} [, {flags} [, {skip} [, {stopline} [, {timeout}]]]])

Same as `searchpair()` , but returns a `List` with the line and column position of the match. The first element of the `List` is the line number and the second element is the byte index of the column position of the match. If no match is found, returns `[0, 0]`.

```
:let [lnum,col] = searchpairpos('{', ' ', '}', 'n')
```

See `match-parens` for a bigger and more useful example.

searchpos({pattern} [, {flags} [, {stopline} [, {timeout} [, {skip}]]]]) searchpos()

Same as `search()` , but returns a `List` with the line and column position of the match. The first element of the `List` is the line number and the second element is the byte index of the column position of the match. If no match is found, returns `[0, 0]`.

Example:

```
:let [lnum, col] = searchpos('mypattern', 'n')
```

When the 'p' flag is given then there is an extra item with the sub-pattern match number `search()-sub-match` . Example:

```
:let [lnum, col, submatch] = searchpos('\(\l\)\|\(\u\)', 'np')
```

In this example "submatch" is 2 when a lowercase letter is found `/\l` , 3 when an uppercase letter is found `/\u` .

Can also be used as a `method` :

```
GetPattern()->searchpos()
```

server2client({clientid}, {string}) server2client()

Send a reply string to {clientid}. The most recent {clientid} that sent a string can be retrieved with `expand("<client>")`.
{only available when compiled with the |+clientserver| feature}
Returns zero for success, -1 for failure.

Note:

This id has to be stored before the next command can be received. I.e. before returning from the received command and before calling any commands that waits for input.

See also `clientserver` .

Example:

```
:echo server2client(expand("<client>"), "HELLO")
```

Can also be used as a `method` :

```
GetClientId()->server2client(string)
```

serverlist() serverlist()

Return a list of available server names, one per line.

When there are no servers or the information is not available an empty string is returned. See also `clientserver` .

{only available when compiled with the |+clientserver| feature}

Example:

```
:echo serverlist()
```

`setbufline({buf}, {lnum}, {text})` `setbufline()`
Set line `{lnum}` to `{text}` in buffer `{buf}`. This works like `setline()` for the specified buffer.

This function works only for loaded buffers. First call `bufload()` if needed.

To insert lines use `appendbufline()`.
Any text properties in `{lnum}` are cleared.

`{text}` can be a string to set one line, or a List of strings to set multiple lines. If the List extends below the last line then those lines are added. If the List is empty then nothing is changed and zero is returned.

For the use of `{buf}`, see `bufname()` above.

`{lnum}` is used like with `setline()`.
Use "\$" to refer to the last line in buffer `{buf}`.
When `{lnum}` is just below the last line the `{text}` will be added below the last line.

When `{buf}` is not a valid buffer, the buffer is not loaded or `{lnum}` is not valid then 1 is returned. In Vim9 script an error is given.
On success 0 is returned.

Can also be used as a `method`, the base is passed as the third argument:

```
GetText()->setbufline(buf, lnum)
```

`setbufvar({buf}, {varname}, {val})` `setbufvar()`
Set option or local variable `{varname}` in buffer `{buf}` to `{val}`.

This also works for a global or local window option, but it doesn't work for a global or local window variable.

For a local window option the global value is unchanged.

For the use of `{buf}`, see `bufname()` above.

The `{varname}` argument is a string.

Note that the variable name without "b:" must be used.

Examples:

```
:call setbufvar(1, "&mod", 1)  
:call setbufvar("todo", "myvar", "foobar")
```

This function is not available in the `sandbox`.

Can also be used as a `method`, the base is passed as the third argument:

```
GetValue()->setbufvar(buf, varname)
```

`setcellwidths({list})` `setcellwidths()`
Specify overrides for cell widths of character ranges. This tells Vim how wide characters are when displayed in the

terminal, counted in screen cells. The values override **'ambiwidth'**. Example:

```
call setcellwidths([
    \ [0x111, 0x111, 1],
    \ [0x2194, 0x2199, 2],
    \ ])
```

The `{list}` argument is a List of Lists with each three numbers: `[{low}, {high}, {width}]`. E1109 E1110
`{low}` and `{high}` can be the same, in which case this refers to one character. Otherwise it is the range of characters from `{low}` to `{high}` (inclusive). E1111 E1114
Only characters with value 0x80 and higher can be used.

`{width}` must be either 1 or 2, indicating the character width in screen cells. E1112

An error is given if the argument is invalid, also when a range overlaps with another. E1113

If the new value causes **'fillchars'** or **'listchars'** to become invalid it is rejected and an error is given.

To clear the overrides pass an empty `{list}`:
`setcellwidths([]);`

You can use the script `$VIMRUNTIME/tools/emoji_list.vim` to see the effect for known emoji characters. Move the cursor through the text to check if the cell widths of your terminal match with what Vim knows about each emoji. If it doesn't look right you need to adjust the `{list}` argument.

`setcharpos({expr}, {list})` setcharpos()
Same as `setpos()` but uses the specified column number as the character index instead of the byte index in the line.

Example:

With the text "⚡⚡⚡" in line 8:

```
call setcharpos('.', [0, 8, 4, 0])
positions the cursor on the fourth character '⚡'.
call setpos('.', [0, 8, 4, 0])
positions the cursor on the second character '⚡'.
```

Can also be used as a `method` :

```
GetPosition()->setcharpos('.')
```

`setcharesearch({dict})` setcharesearch()
Set the current character search information to `{dict}`, which contains one or more of the following entries:

char	character which will be used for a subsequent , or ; command; an empty string clears the character search
forward	direction of character search; 1 for forward,

until 0 for backward
type of character search; 1 for a t or T
character search, 0 for an f or F
character search

This can be useful to save/restore a user's character search from a script:

```
:let prevsearch = getcharsearch()  
:" Perform a command which clobbers user's search  
:call setcharsearch(prevsearch)
```

Also see `getcharsearch()` .

Can also be used as a `method` :
`SavedSearch()->setcharsearch()`

`setcmdline({str} [, {pos}])` `setcmdline()`

Set the command line to `{str}` and set the cursor position to `{pos}`.
If `{pos}` is omitted, the cursor is positioned after the text.
Returns 0 when successful, 1 when not editing the command line.

Can also be used as a `method` :
`GetText()->setcmdline()`

`setcmdpos({pos})` `setcmdpos()`

Set the cursor position in the command line to byte position `{pos}`. The first position is 1.
Use `getcmdpos()` to obtain the current position.
Only works while editing the command line, thus you must use `c_CTRL-_e` , `c_CTRL-R_=` or `c_CTRL-R_CTRL-R` with '='. For `c_CTRL-_e` and `c_CTRL-R_CTRL-R` with '=' the position is set after the command line is set to the expression. For `c_CTRL-R_=` it is set after evaluating the expression but before inserting the resulting text.
When the number is too big the cursor is put at the end of the line. A number smaller than one has undefined results.
Returns 0 when successful, 1 when not editing the command line.

Can also be used as a `method` :
`GetPos()->setcmdpos()`

`setcursorcharpos({lnum}, {col} [, {off}])` `setcursorcharpos()`

`setcursorcharpos({list})`

Same as `cursor()` but uses the specified column number as the character index instead of the byte index in the line.

Example:

With the text "␣␣␣" in line 4:

```
call setcursorcharpos(4, 3)
```

positions the cursor on the third character '␣'.

```
call cursor(4, 3)
```

positions the cursor on the first character '␣'.

Can also be used as a `method` :
`GetCursorPos()->setcursorcharpos()`

`setenv({name}, {val})` `setenv()`
Set environment variable `{name}` to `{val}`. Example:
`call setenv('HOME', '/home/myhome')`

When `{val}` is `v:null` the environment variable is deleted.
See also `expr-env` .

Can also be used as a `method` , the base is passed as the second argument:
`GetPath()->setenv('PATH')`

`setfperm({fname}, {mode})` `setfperm()` `chmod`
Set the file permissions for `{fname}` to `{mode}`.
`{mode}` must be a string with 9 characters. It is of the form "rwxrwxrwx", where each group of "rwx" flags represent, in turn, the permissions of the owner of the file, the group the file belongs to, and other users. A '-' character means the permission is off, any other character means on. Multi-byte characters are not supported.

For example "rw-r-----" means read-write for the user, readable by the group, not accessible by others. "xx-x-----" would do the same thing.

Returns non-zero for success, zero for failure.

Can also be used as a `method` :
`GetFilename()->setfperm(mode)`

To read permissions see `getfperm()` .

`setline({lnum}, {text})` `setline()`
Set line `{lnum}` of the current buffer to `{text}`. To insert lines use `append()` . To set lines in another buffer use `setbufline()` . Any text properties in `{lnum}` are cleared.

`{lnum}` is used like with `getline()` .

When `{lnum}` is just below the last line the `{text}` will be added below the last line.

`{text}` can be any type or a List of any type, each item is converted to a String. When `{text}` is an empty List then nothing is changed and FALSE is returned.

If this succeeds, FALSE is returned. If this fails (most likely because `{lnum}` is invalid) TRUE is returned.

In `Vim9` script an error is given if `{lnum}` is invalid.

Example:

```
:call setline(5, strftime("%c"))
```

When `{text}` is a `List` then line `{lnum}` and following lines will be set to the items in the list. Example:

```
:call setline(5, ['aaa', 'bbb', 'ccc'])
```

This is equivalent to:

```
:for [n, l] in [[5, 'aaa'], [6, 'bbb'], [7, 'ccc']]  
: call setline(n, l)  
:endfor
```

Note: The '[' and ']' marks are not set.

Can also be used as a `method`, the base is passed as the second argument:

```
GetText()->setline(lnum)
```

`setloclist({nr}, {list} [, {action} [, {what}]])` `setloclist()`
Create or replace or add to the location list for window `{nr}`.
`{nr}` can be the window number or the `window-ID`.
When `{nr}` is zero the current window is used.

For a location list window, the displayed location list is modified. For an invalid window number `{nr}`, -1 is returned. Otherwise, same as `setqflist()`. Also see `location-list`.

For `{action}` see `setqflist-action`.

If the optional `{what}` dictionary argument is supplied, then only the items listed in `{what}` are set. Refer to `setqflist()` for the list of supported keys in `{what}`.

Can also be used as a `method`, the base is passed as the second argument:

```
GetLoclist()->setloclist(winnr)
```

`setmatches({list} [, {win}])` `setmatches()`
Restores a list of matches saved by `getmatches()` for the current window. Returns 0 if successful, otherwise -1. All current matches are cleared before the list is restored. See example for `getmatches()`.
If `{win}` is specified, use the window with this number or window ID instead of the current window.

Can also be used as a `method`:

```
GetMatches()->setmatches()
```

`setpos({expr}, {list})` `setpos()`
Set the position for String `{expr}`. Possible values:
. the cursor
'x mark x

`{list}` must be a `List` with four or five numbers:

[bufnum, lnum, col, off]
[bufnum, lnum, col, off, curswant]

"bufnum" is the buffer number. Zero can be used for the current buffer. When setting an uppercase mark "bufnum" is used for the mark position. For other marks it specifies the buffer to set the mark in. You can use the `bufnr()` function to turn a file name into a buffer number. For setting the cursor and the ' mark "bufnum" is ignored, since these are associated with a window, not a buffer. Does not change the jumplist.

"lnum" and "col" are the position in the buffer. The first column is 1. Use a zero "lnum" to delete a mark. If "col" is smaller than 1 then 1 is used. To use the character count instead of the byte count, use `setcharpos()` .

The "off" number is only used when **'virtualedit'** is set. Then it is the offset in screen columns from the start of the character. E.g., a position within a `<Tab>` or after the last character.

The "curswant" number is only used when setting the cursor position. It sets the preferred column for when moving the cursor vertically. When the "curswant" number is missing the preferred column is not set. When it is present and setting a mark position it is not used.

Note that for '<' and '>' changing the line number may result in the marks to be effectively be swapped, so that '<' is always before '>'.

Returns 0 when the position could be set, -1 otherwise. An error message is given if `{expr}` is invalid.

Also see `setcharpos()` , `getpos()` and `getcurpos()` .

This does not restore the preferred column for moving vertically; if you set the cursor position with this, `j` and `k` motions will jump to previous columns! Use `cursor()` to also set the preferred column. Also see the "curswant" key in `winrestview()` .

Can also be used as a **method** :
`GetPosition()->setpos('.')`

`setqflist({list} [, {action} [, {what}]])` `setqflist()`
Create or replace or add to the quickfix list.

If the optional `{what}` dictionary argument is supplied, then only the items listed in `{what}` are set. The first `{list}` argument is ignored. See below for the supported items in `{what}`.

`setqflist-what`

When `{what}` is not present, the items in `{list}` are used. Each item must be a dictionary. Non-dictionary items in `{list}` are ignored. Each dictionary item can contain the following entries:

<code>bufnr</code>	buffer number; must be the number of a valid buffer
<code>filename</code>	name of a file; only used when "bufnr" is not present or it is invalid.
<code>module</code>	name of a module; if given it will be used in quickfix error window instead of the filename.
<code>lnum</code>	line number in the file
<code>end_lnum</code>	end of lines, if the item spans multiple lines
<code>pattern</code>	search pattern used to locate the error
<code>col</code>	column number
<code>vcol</code>	when non-zero: "col" is visual column when zero: "col" is byte index
<code>end_col</code>	end column, if the item spans multiple columns
<code>nr</code>	error number
<code>text</code>	description of the error
<code>type</code>	single-character error type, 'E', 'W', etc.
<code>valid</code>	recognized error message
<code>user_data</code>	custom data associated with the item, can be any type.

The "col", "vcol", "nr", "type" and "text" entries are optional. Either "lnum" or "pattern" entry can be used to locate a matching error line.

If the "filename" and "bufnr" entries are not present or neither the "lnum" or "pattern" entries are present, then the item will not be handled as an error line.

If both "pattern" and "lnum" are present then "pattern" will be used.

If the "valid" entry is not supplied, then the valid flag is set when "bufnr" is a valid buffer or "filename" exists.

If you supply an empty `{list}`, the quickfix list will be cleared.

Note that the list is not exactly the same as what `getqflist()` returns.

`{action}` values: `setqflist-action` E927

'a' The items from `{list}` are added to the existing quickfix list. If there is no existing list, then a new list is created.

'r' The items from the current quickfix list are replaced with the items from `{list}`. This can also be used to clear the list:

`:call setqflist([], 'r')`

'f' All the quickfix lists in the quickfix stack are freed.

If `{action}` is not present or is set to ' ', then a new list

is created. The new quickfix list is added after the current quickfix list in the stack and all the following lists are freed. To add a new quickfix list at the end of the stack, set "nr" in `{what}` to "\$".

The following items can be specified in dictionary `{what}`:

context	quickfix list context. See quickfix-context
efm	errorformat to use when parsing text from "lines". If this is not present, then the 'errorformat' option value is used. See quickfix-parse
id	quickfix list identifier quickfix-ID
idx	index of the current entry in the quickfix list specified by 'id' or 'nr' . If set to '\$', then the last entry in the list is set as the current entry. See quickfix-index
items	list of quickfix entries. Same as the <code>{list}</code> argument.
lines	use 'errorformat' to parse a list of lines and add the resulting entries to the quickfix list <code>{nr}</code> or <code>{id}</code> . Only a <code>List</code> value is supported. See quickfix-parse
nr	list number in the quickfix stack; zero means the current quickfix list and "\$" means the last quickfix list.
quickfixtextfunc	function to get the text to display in the quickfix window. The value can be the name of a function or a funcref or a lambda. Refer to quickfix-window-function for an explanation of how to write the function and an example.
title	quickfix list title text. See quickfix-title

Unsupported keys in `{what}` are ignored.

If the "nr" item is not present, then the current quickfix list is modified. When creating a new quickfix list, "nr" can be set to a value one greater than the quickfix stack size.

When modifying a quickfix list, to guarantee that the correct list is modified, "id" should be used instead of "nr" to specify the list.

Examples (See also [setqflist-examples](#)):

```
:call setqflist([], 'r', {'title': 'My search'})
:call setqflist([], 'r', {'nr': 2, 'title': 'Errors'})
:call setqflist([], 'a', {'id': qfid, 'lines': ["F1:10:L10"]})
```

Returns zero for success, -1 for failure.

This function can be used to create a quickfix list independent of the **'errorformat'** setting. Use a command like ``:cc 1`` to jump to the first position.

Can also be used as a [method](#), the base is passed as the second argument:

```
GetErrorlist()->setqflist()
```

setreg()

setreg({regname}, {value} [, {options}])

Set the register {regname} to {value}.

If {regname} is "" or "@", the unnamed register "" is used.

The {regname} argument is a string. In Vim9-script

{regname} must be one character.

{value} may be any value returned by `getreg()` or `getreginfo()`, including a `List` or `Dict`.

If {options} contains "a" or {regname} is upper case, then the value is appended.

{options} can also contain a register type specification:

"c" or "v" characterwise mode

"l" or "V" linewise mode

"b" or "<CTRL-V>" blockwise-visual mode

If a number immediately follows "b" or "<CTRL-V>" then this is used as the width of the selection - if it is not specified then the width of the block is set to the number of characters in the longest line (counting a <Tab> as 1 character).

If {options} contains no register settings, then the default is to use character mode unless {value} ends in a <NL> for string {value} and linewise mode for list {value}. Blockwise mode is never selected automatically.

Returns zero for success, non-zero for failure.

E883

Note: you may not use `List` containing more than one item to set search and expression registers. Lists containing no items act like empty strings.

Examples:

```
:call setreg(v:register, @*)
:call setreg('*', @%, 'ac')
:call setreg('a', "1\n2\n3", 'b5')
:call setreg('"'', { 'points_to': 'a'})
```

This example shows using the functions to save and restore a register:

```
:let var_a = getreginfo()
:call setreg('a', var_a)
```

or:

```
:let var_a = getreg('a', 1, 1)
:let var_amode = getregtype('a')
....
:call setreg('a', var_a, var_amode)
```

Note: you may not reliably restore register value without using the third argument to `getreg()` as without it newlines are represented as newlines AND Nul bytes are represented as newlines as well, see `NL-used-for-Nul`.

You can also change the type of a register by appending

nothing:
:call setreg('a', '', 'al')

Can also be used as a `method` , the base is passed as the second argument:

GetText()->setreg('a')

settabvar({tabnr}, {varname}, {val}) `settabvar()`
Set tab-local variable {varname} to {val} in tab page {tabnr}.
t:var

The {varname} argument is a string.

Note that autocommands are blocked, side effects may not be triggered, e.g. when setting **'filetype'**.

Note that the variable name without "t:" must be used.

Tabs are numbered starting with one.

This function is not available in the `sandbox` .

Can also be used as a `method` , the base is passed as the third argument:

GetValue()->settabvar(tab, name)

settabwinvar({tabnr}, {winnr}, {varname}, {val}) `settabwinvar()`
Set option or local variable {varname} in window {winnr} to {val}.

Tabs are numbered starting with one. For the current tabpage use `setwinvar()` .

{winnr} can be the window number or the `window-ID` .

When {winnr} is zero the current window is used.

Note that autocommands are blocked, side effects may not be triggered, e.g. when setting **'filetype'** or **'syntax'**.

This also works for a global or local buffer option, but it doesn't work for a global or local buffer variable.

For a local buffer option the global value is unchanged.

Note that the variable name without "w:" must be used.

Examples:

:call settabwinvar(1, 1, "&list", 0)

:call settabwinvar(3, 2, "myvar", "foobar")

This function is not available in the `sandbox` .

Can also be used as a `method` , the base is passed as the fourth argument:

GetValue()->settabwinvar(tab, winnr, name)

settagstack({nr}, {dict} [, {action}]) `settagstack()`
Modify the tag stack of the window {nr} using {dict}.
{nr} can be the window number or the `window-ID` .

For a list of supported items in {dict}, refer to

`gettagstack()` . "curidx" takes effect before changing the tag stack.

E962

How the tag stack is modified depends on the {action} argument:

- If {action} is not present or is set to 'r', then the tag

- stack is replaced.
- If `{action}` is set to 'a', then new entries from `{dict}` are pushed (added) onto the tag stack.
 - If `{action}` is set to 't', then all the entries from the current entry in the tag stack or "curidx" in `{dict}` are removed and then new entries are pushed to the stack.

The current index is set to one after the length of the tag stack after the modification.

Returns zero for success, -1 for failure.

Examples (for more examples see [tagstack-examples](#)):

```
Empty the tag stack of window 3:
    call settagstack(3, {'items' : []})
```

```
Save and restore the tag stack:
    let stack = gettagstack(1003)
    " do something else
    call settagstack(1003, stack)
    unlet stack
```

Can also be used as a `method`, the base is passed as the second argument:

```
GetStack()->settagstack(winnr)
```

`setwinvar({winnr}, {varname}, {val})` `setwinvar()`

Like `settabwinvar()` for the current tab page.

Examples:

```
:call setwinvar(1, "&list", 0)
:call setwinvar(2, "myvar", "foobar")
```

Can also be used as a `method`, the base is passed as the third argument:

```
GetValue()->setwinvar(winnr, name)
```

`sha256({string})` `sha256()`

Returns a String with 64 hex characters, which is the SHA256 checksum of `{string}`.

Can also be used as a `method`:

```
GetText()->sha256()
```

`{only available when compiled with the |+cryptv| feature}`

`shellescape({string} [, {special}])` `shellescape()`

Escape `{string}` for use as a shell command argument.

When the `'shell'` contains powershell (MS-Windows) or pwsh (MS-Windows, Linux, and macOS) then it will enclose `{string}` in single quotes and will double up all internal single quotes.

On MS-Windows, when `'shellslash'` is not set, it will enclose `{string}` in double quotes and double all double quotes within `{string}`.

Otherwise it will enclose `{string}` in single quotes and replace all `"` with `"\"`.

When the `{special}` argument is present and it's a non-zero Number or a non-empty String (`non-zero-arg`), then special items such as `!`, `%`, `#` and `<cword>` will be preceded by a backslash. This backslash will be removed again by the `:! command`.

The `!` character will be escaped (again with a `non-zero-arg {special}`) when `'shell'` contains `csh` in the tail. That is because for `csh` and `tcsh` `!` is used for history replacement even when inside single quotes.

With a `non-zero-arg {special}` the `<NL>` character is also escaped. When `'shell'` containing `csh` in the tail it's escaped a second time.

The `\` character will be escaped when `'shell'` contains `fish` in the tail. That is because for `fish` `\` is used as an escape character inside single quotes.

Example of use with a `:! command`:

```
:exe '!dir ' .. shellescape(expand('<cfile>'), 1)
```

This results in a directory listing for the file under the cursor. Example of use with `system()` :

```
:call system("chmod +w -- " .. shellescape(expand("%"))) )
```

See also `::S .`

Can also be used as a `method` :

```
GetCommand()->shellescape()
```

`shiftwidth([{col}])` `shiftwidth()`

Returns the effective value of `'shiftwidth'`. This is the `'shiftwidth'` value unless it is zero, in which case it is the `'tabstop'` value. This function was introduced with patch 7.3.694 in 2012, everybody should have it by now (however it did not allow for the optional `{col}` argument until 8.1.542).

When there is one argument `{col}` this is used as column number for which to return the `'shiftwidth'` value. This matters for the `'vartabstop'` feature. If the `'vartabstop'` setting is enabled and no `{col}` argument is given, column 1 will be assumed.

Can also be used as a `method` :

```
GetColumn()->shiftwidth()
```

`sign_ functions` are documented here: [sign-functions-details](#)

`simplify({filename})` `simplify()`

Simplify the file name as much as possible without changing the meaning. Shortcuts (on MS-Windows) or symbolic links (on Unix) are not resolved. If the first path component in

`{filename}` designates the current directory, this will be valid for the result as well. A trailing path separator is not removed either. On Unix `"/path"` is unchanged, but `"/path"` is simplified to `"/path"` (this follows the Posix standard).

Example:

```
simplify("./dir/../../file/") == "./file/"
```

Note: The combination `"dir/.."` is only removed if `"dir"` is a searchable directory or does not exist. On Unix, it is also removed when `"dir"` is a symbolic link within the same directory. In order to resolve all the involved symbolic links before simplifying the path name, use `resolve()`.

Can also be used as a `method` :

```
GetName()->simplify()
```

`sin({expr})`

`sin()`

Return the sine of `{expr}`, measured in radians, as a `Float`. `{expr}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo sin(100)
-0.506366
:echo sin(-4.01)
0.763301
```

Can also be used as a `method` :

```
Compute()->sin()
```

`sinh({expr})`

`sinh()`

Return the hyperbolic sine of `{expr}` as a `Float` in the range `[-inf, inf]`.

`{expr}` must evaluate to a `Float` or a `Number`.

Returns 0.0 if `{expr}` is not a `Float` or a `Number`.

Examples:

```
:echo sinh(0.5)
0.521095
:echo sinh(-0.9)
-1.026517
```

Can also be used as a `method` :

```
Compute()->sinh()
```

`slice({expr}, {start} [, {end}])`

`slice()`

Similar to using a `slice` `"expr[start : end]"`, but `"end"` is used exclusive. And for a string the indexes are used as character indexes instead of byte indexes, like in `vim9script`.

Also, composing characters are not counted.

When `{end}` is omitted the slice continues to the last item.

When `{end}` is `-1` the last item is omitted.

Returns an empty value if `{start}` or `{end}` are invalid.

Can also be used as a `method` :
`GetList()->slice(offset)`

```
sort({list} [, {how} [, {dict}]])) sort() E702  
Sort the items in {list} in-place. Returns {list}.
```

If you want a list to remain unmodified make a copy first:
`:let sortedlist = sort(copy(mylist))`

When {how} is omitted or is a string, then sort() uses the string representation of each item to sort on. Numbers sort after Strings, Lists after Numbers. For sorting text in the current buffer use `:sort` .

When {how} is given and it is 'i' then case is ignored. In legacy script, for backwards compatibility, the value one can be used to ignore case. Zero means to not ignore case.

When {how} is given and it is 'l' then the current collation locale is used for ordering. Implementation details: strcoll() is used to compare strings. See `:language` check or set the collation locale. `v:collate` can also be used to check the current locale. Sorting using the locale typically ignores case. Example:

```
" ö is sorted similarly to o with English locale.  
:language collate en_US.UTF8  
:echo sort(['n', 'o', '0', 'ö', 'p', 'z'], 'l')  
['n', 'o', '0', 'ö', 'p', 'z']
```

```
" ö is sorted after z with Swedish locale.  
:language collate sv_SE.UTF8  
:echo sort(['n', 'o', '0', 'ö', 'p', 'z'], 'l')  
['n', 'o', '0', 'p', 'z', 'ö']
```

This does not work properly on Mac.

When {how} is given and it is 'n' then all items will be sorted numerical (Implementation detail: this uses the strtod() function to parse numbers. Strings, Lists, Dicts and Funcrefs will be considered as being 0). **Note** that this won't sort a list of strings with numbers!

When {how} is given and it is 'N' then all items will be sorted numerical. This is like 'n' but a string containing digits will be used as the number they represent.

When {how} is given and it is 'f' then all items will be sorted numerical. All values must be a Number or a Float.

When {how} is a `Funcref` or a function name, this function is called to compare items. The function is invoked with two items as argument and must return zero if they are equal, 1 or bigger if the first one sorts after the second one, -1 or smaller if the first one sorts before the second one.

`{dict}` is for functions with the "dict" attribute. It will be used to set the local variable "self". [Dictionary-function](#)

The sort is stable, items which compare equal (as number or as string) will keep their relative position. E.g., when sorting on numbers, text strings will sort next to each other, in the same order as they were originally.

Can also be used as a [method](#) :
`mylist->sort()`

Also see [uniq\(\)](#) .

Example:

```
func MyCompare(i1, i2)
    return a:i1 == a:i2 ? 0 : a:i1 > a:i2 ? 1 : -1
endfunc
eval mylist->sort("MyCompare")
```

A shorter compare version for this specific simple case, which ignores overflow:

```
func MyCompare(i1, i2)
    return a:i1 - a:i2
endfunc
```

For a simple expression you can use a lambda:

```
eval mylist->sort({i1, i2 -> i1 - i2})
```

`sound_clear()`

[sound_clear\(\)](#)

Stop playing all sounds.

On some Linux systems you may need the libcanberra-pulse package, otherwise sound may not stop.

[{only available when compiled with the |+sound| feature}](#)

[sound_playevent\(\)](#)

`sound_playevent({name} [, {callback}])`

Play a sound identified by `{name}`. Which event names are supported depends on the system. Often the XDG sound names are used. On Ubuntu they may be found in `/usr/share/sounds/freedesktop/stereo`. Example:

```
call sound_playevent('bell')
```

On MS-Windows, `{name}` can be SystemAsterisk, SystemDefault, SystemExclamation, SystemExit, SystemHand, SystemQuestion, SystemStart, SystemWelcome, etc.

On macOS, `{name}` refers to files located in `/System/Library/Sounds` (e.g. "Tink"). It will also work for custom installed sounds in folders like `~/Library/Sounds`.

When `{callback}` is specified it is invoked when the sound is finished. The first argument is the sound ID, the second argument is the status:

```
0      sound was played to the end
1      sound was interrupted
```

2 error occurred after sound started

Example:

```
func Callback(id, status)
  echomsg "sound " .. a:id .. " finished with " .. a:status
endfunc
call sound_playevent('bell', 'Callback')
```

MS-Windows: `{callback}` doesn't work for this function.

Returns the sound ID, which can be passed to ``sound_stop()``.
Returns zero if the sound could not be played.

Can also be used as a **method** :

```
GetSoundName()->sound_playevent()
```

{only available when compiled with the `|+sound|` feature}

sound_playfile()

`sound_playfile({path} [, {callback}])`

Like ``sound_playevent()`` but play sound file `{path}`. `{path}` must be a full path. On Ubuntu you may find files to play with this command:

```
:!find /usr/share/sounds -type f | grep -v index.theme
```

Can also be used as a **method** :

```
GetSoundPath()->sound_playfile()
```

{only available when compiled with the `|+sound|` feature}

`sound_stop({id})`

sound_stop()

Stop playing sound `{id}`. `{id}` must be previously returned by ``sound_playevent()`` or ``sound_playfile()``.

On some Linux systems you may need the `libcanberra-pulse` package, otherwise sound may not stop.

On MS-Windows, this does not work for event sound started by ``sound_playevent()``. To stop event sounds, use ``sound_clear()``.

Can also be used as a **method** :

```
soundid->sound_stop()
```

{only available when compiled with the `|+sound|` feature}

soundfold()

`soundfold({word})`

Return the sound-folded equivalent of `{word}`. Uses the first language in `'spelllang'` for the current window that supports soundfolding. `'spell'` must be set. When no sound folding is possible the `{word}` is returned unmodified.

This can be used for making spelling suggestions. **Note** that the method can be quite slow.

Can also be used as a `method` :
`GetWord()->soundfold()`

`spellbadword()`

`spellbadword([sentence])`

Without argument: The result is the badly spelled word under or after the cursor. The cursor is moved to the start of the bad word. When no bad word is found in the cursor line the result is an empty string and the cursor doesn't move.

With argument: The result is the first word in `{sentence}` that is badly spelled. If there are no spelling mistakes the result is an empty string.

The return value is a list with two items:

- The badly spelled word or an empty string.
- The type of the spelling error:

"bad"	spelling mistake
"rare"	rare word
"local"	word only valid in another region
"caps"	word should start with Capital

Example:

```
echo spellbadword("the quik brown fox")
['quik', 'bad']
```

The spelling information for the current window and the value of `'spelllang'` are used.

Can also be used as a `method` :
`GetText()->spellbadword()`

`spellsuggest()`

`spellsuggest({word} [, {max} [, {capital}]])`

Return a `List` with spelling suggestions to replace `{word}`. When `{max}` is given up to this number of suggestions are returned. Otherwise up to 25 suggestions are returned.

When the `{capital}` argument is given and it's non-zero only suggestions with a leading capital will be given. Use this after a match with `'spellcapcheck'`.

`{word}` can be a badly spelled word followed by other text. This allows for joining two words that were split. The suggestions also include the following text, thus you can replace a line.

`{word}` may also be a good word. Similar words will then be returned. `{word}` itself is not included in the suggestions, although it may appear capitalized.

The spelling information for the current window is used. The values of `'spelllang'` and `'spellsuggest'` are used.

Can also be used as a `method` :

GetWord()->spellsuggest()

`split({string} [, {pattern} [, {keepempty}]])` `split()`

Make a `List` out of `{string}`. When `{pattern}` is omitted or empty each white-separated sequence of characters becomes an item.

Otherwise the string is split where `{pattern}` matches, removing the matched characters. `'ignorecase'` is not used here, add `\c` to ignore case. `/\c`

When the first or last item is empty it is omitted, unless the `{keepempty}` argument is given and it's non-zero.

Other empty items are kept when `{pattern}` matches at least one character or when `{keepempty}` is non-zero.

Example:

```
:let words = split(getline('.'), '\W\+')
```

To split a string in individual characters:

```
:for c in split(mystring, '\zs')
```

If you want to keep the separator you can also use `\zs` at the end of the pattern:

```
:echo split('abc:def:ghi', ':\zs')  
['abc:', 'def:', 'ghi']
```

Splitting a table where the first element can be empty:

```
:let items = split(line, ':', 1)
```

The opposite function is `join()` .

Can also be used as a `method` :

`GetString()->split()`

`sqrt({expr})` `sqrt()`

Return the non-negative square root of `Float {expr}` as a `Float` .

`{expr}` must evaluate to a `Float` or a `Number` . When `{expr}` is negative the result is NaN (Not a Number). Returns 0.0 if `{expr}` is not a `Float` or a `Number` .

Examples:

```
:echo sqrt(100)  
10.0  
:echo sqrt(-4.01)  
nan
```

"nan" may be different, it depends on system libraries.

Can also be used as a `method` :

`Compute()->sqrt()`

`srand([{expr}])` `srand()`

Initialize seed used by `rand()` :

- If `{expr}` is not given, seed values are initialized by reading from `/dev/urandom`, if possible, or using `time(NULL)` a.k.a. epoch time otherwise; this only has second accuracy.
- If `{expr}` is given it must be a `Number`. It is used to initialize the seed values. This is useful for testing or when a predictable sequence is intended.

Examples:

```
:let seed = srand()
:let seed = srand(userinput)
:echo rand(seed)
```

`state([{what}])`

`state()`

Return a string which contains characters indicating the current state. Mostly useful in callbacks that want to do work that may not always be safe. Roughly this works like:

- callback uses `state()` to check if work is safe to do.

Yes: then do it right away.

No: add to work queue and add a `SafeState` and/or `SafeStateAgain` autocommand (`SafeState` triggers at toplevel, `SafeStateAgain` triggers after handling messages and callbacks).

- When `SafeState` or `SafeStateAgain` is triggered and executes your autocommand, check with `state()` if the work can be done now, and if yes remove it from the queue and execute.

Remove the autocommand if the queue is now empty.

Also see `mode()` .

When `{what}` is given only characters in this string will be added. E.g, this checks if the screen has scrolled:

```
if state('s') == ''
    " screen has not scrolled
```

These characters indicate the state, generally indicating that something is busy:

```
m  halfway a mapping, :normal command, feedkeys() or
    stuffed command
o  operator pending, e.g. after d
a  Insert mode autocomplete active
x  executing an autocommand
w  blocked on waiting, e.g. ch_evalexpr(), ch_read() and
    ch_readraw() when reading json
S  not triggering SafeState or SafeStateAgain, e.g. after
    f or a count
c  callback invoked, including timer (repeats for
    recursiveness up to "ccc")
s  screen has scrolled for messages
```

`str2float({string} [, {quoted}])`

`str2float()`

Convert String `{string}` to a Float. This mostly works the same as when using a floating point number in an expression, see `floating-point-format` . But it's a bit more permissive. E.g., "1e40" is accepted, while in an expression you need to write "1.0e40". The hexadecimal form "0x123" is also accepted, but not others, like binary or octal.

When `{quoted}` is present and non-zero then embedded single quotes before the dot are ignored, thus "1'000.0" is a thousand.

Text after the number is silently ignored.

The decimal point is always '.', no matter what the locale is set to. A comma ends the number: "12,345.67" is converted to

12.0. You can strip out thousands separators with
`substitute()` :
`let f = str2float(substitute(text, ',', ' ', 'g'))`

Returns 0.0 if the conversion fails.

Can also be used as a `method` :
`let f = text->substitute(',', ' ', 'g')->str2float()`

`str2list({string} [, {utf8}])` `str2list()`
Return a list containing the number values which represent each character in String `{string}`. Examples:
`str2list(" ")` returns [32]
`str2list("ABC")` returns [65, 66, 67]
`list2str()` does the opposite.

When `{utf8}` is omitted or zero, the current `'encoding'` is used. When `{utf8}` is TRUE, always treat the String as UTF-8 characters. With UTF-8 composing characters are handled properly:

`str2list("á")` returns [97, 769]

Can also be used as a `method` :
`GetString()->str2list()`

`str2nr({string} [, {base} [, {quoted}]])` `str2nr()`
Convert string `{string}` to a number.
`{base}` is the conversion base, it can be 2, 8, 10 or 16.
When `{quoted}` is present and non-zero then embedded single quotes are ignored, thus `"1'000'000"` is a million.

When `{base}` is omitted base 10 is used. This also means that a leading zero doesn't cause octal conversion to be used, as with the default String to Number conversion. Example:

`let nr = str2nr('0123')`

When `{base}` is 16 a leading `"0x"` or `"0X"` is ignored. With a different base the result will be zero. Similarly, when `{base}` is 8 a leading `"0"`, `"0o"` or `"0O"` is ignored, and when `{base}` is 2 a leading `"0b"` or `"0B"` is ignored. Text after the number is silently ignored.

Returns 0 if `{string}` is empty or on error.

Can also be used as a `method` :
`GetText()->str2nr()`

`strcharlen({string})` `strcharlen()`
The result is a Number, which is the number of characters in String `{string}`. Composing characters are ignored.
`strchars()` can count the number of characters, counting composing characters separately.

Returns 0 if `{string}` is empty or on error.

Also see `strlen()` , `strdisplaywidth()` and `strwidth()` .

Can also be used as a `method` :

```
GetText()->strcharlen()
```

`strcharpart({src}, {start} [, {len} [, {skipcc}]])` `strcharpart()`

Like `strpart()` but using character index and length instead of byte index and length.

When `{skipcc}` is omitted or zero, composing characters are counted separately.

When `{skipcc}` set to 1, Composing characters are ignored, similar to `slice()` .

When a character index is used where a character does not exist it is omitted and counted as one character. For example:

```
strcharpart('abc', -1, 2)
```

results in 'a'.

Returns an empty string on error.

Can also be used as a `method` :

```
GetText()->strcharpart(5)
```

`strchars({string} [, {skipcc}])` `strchars()`

The result is a Number, which is the number of characters in String `{string}`.

When `{skipcc}` is omitted or zero, composing characters are counted separately.

When `{skipcc}` set to 1, Composing characters are ignored.

`strcharlen()` always does this.

Returns zero on error.

Also see `strlen()` , `strdisplaywidth()` and `strwidth()` .

`{skipcc}` is only available after 7.4.755. For backward compatibility, you can define a wrapper function:

```
if has("patch-7.4.755")
  function s:strchars(str, skipcc)
    return strchars(a:str, a:skipcc)
  endfunction
else
  function s:strchars(str, skipcc)
    if a:skipcc
      return strlen(substitute(a:str, ".", "x", "g"))
    else
      return strchars(a:str)
    endif
  endfunction
```

endif

Can also be used as a `method` :
`GetText()->strchars()`

`strdisplaywidth({string} [, {col}])` `strdisplaywidth()`
The result is a Number, which is the number of display cells String `{string}` occupies on the screen when it starts at `{col}` (first column is zero). When `{col}` is omitted zero is used. Otherwise it is the screen column where to start. This matters for Tab characters.
The option settings of the current window are used. This matters for anything that's displayed differently, such as `'tabstop'` and `'display'`.
When `{string}` contains characters with East Asian Width Class Ambiguous, this function's return value depends on `'ambiwidth'`. Returns zero on error.
Also see `strlen()` , `strwidth()` and `strchars()` .

Can also be used as a `method` :
`GetText()->strdisplaywidth()`

`strftime({format} [, {time}])` `strftime()`
The result is a String, which is a formatted date and time, as specified by the `{format}` string. The given `{time}` is used, or the current time if no time is given. The accepted `{format}` depends on your system, thus this is not portable! See the manual page of the C function `strftime()` for the format. The maximum length of the result is 80 characters. See also `localtime()` , `getftime()` and `strptime()` .
The language can be changed with the `:language` command.
Examples:

```
:echo strftime("%c")           Sun Apr 27 11:49:23 1997
:echo strftime("%Y %b %d %X")  1997 Apr 27 11:53:25
:echo strftime("%y%m%d %T")    970427 11:53:55
:echo strftime("%H:%M")       11:55
:echo strftime("%c", getftime("file.c"))
                               Show mod time of file.c.
```

Not available on all systems. To check use:
`:if exists("*strftime")`

Can also be used as a `method` :
`GetFormat()->strftime()`

`strgetchar({str}, {index})` `strgetchar()`
Get a Number corresponding to the character at `{index}` in `{str}`. This uses a zero-based character index, not a byte index. Composing characters are considered separate characters here. Use `nr2char()` to convert the Number to a String.
Returns -1 if `{index}` is invalid.
Also see `strcharpart()` and `strchars()` .

Can also be used as a `method` :

GetText()->strgetchar(5)

stridx({haystack}, {needle} [, {start}]) stridx()

The result is a Number, which gives the byte index in {haystack} of the first occurrence of the String {needle}. If {start} is specified, the search starts at index {start}. This can be used to find a second match:

```
:let colon1 = stridx(line, ":")
:let colon2 = stridx(line, ":", colon1 + 1)
```

The search is done case-sensitive.

For pattern searches use `match()` .

-1 is returned if the {needle} does not occur in {haystack}.

See also `stridx()` .

Examples:

```
:echo stridx("An Example", "Example")      3
:echo stridx("Starting point", "Start")    0
:echo stridx("Starting point", "start")    -1
```

`strstr()` `strchr()`

stridx() works similar to the C function strstr(). When used with a single character it works similar to strchr().

Can also be used as a `method` :

GetHaystack()->stridx(needle)

string({expr}) string()
Return {expr} converted to a String. If {expr} is a Number, Float, String, Blob or a composition of them, then the result can be parsed back with `eval()` .

{expr} type	result
String	'string' (single quotes are doubled)
Number	123
Float	123.123456 or 1.123456e8
Funcref	function('name')
Blob	0z00112233.44556677.8899
List	[item, item]
Dictionary	{key: value, key: value}
Class	class SomeName
Object	object of SomeName {lnum: 1, col: 3}

When a `List` or `Dictionary` has a recursive reference it is replaced by "[...]" or "{...}". Using `eval()` on the result will then fail.

For an object, invokes the `string()` method to get a textual representation of the object. If the method is not present, then the default representation is used.

Can also be used as a `method` :

mylist->string()

Also see `strtrans()` .

strlen({string}) strlen()

The result is a Number, which is the length of the String `{string}` in bytes.
If the argument is a Number it is first converted to a String.
For other types an error is given and zero is returned.
If you want to count the number of multibyte characters use `strchars()` .
Also see `len()` , `strdisplaywidth()` and `strwidth()` .

Can also be used as a `method` :
`GetString()->strlen()`

`strpart({src}, {start} [, {len} [, {chars}]])` `strpart()`
The result is a String, which is part of `{src}`, starting from byte `{start}`, with the byte length `{len}`.
When `{chars}` is present and TRUE then `{len}` is the number of characters positions (composing characters are not counted separately, thus "1" means one base character and any following composing characters).
To count `{start}` as characters instead of bytes use `strcharpart()` .

When bytes are selected which do not exist, this doesn't result in an error, the bytes are simply omitted.
If `{len}` is missing, the copy continues from `{start}` till the end of the `{src}`.

```
strpart("abcdefg", 3, 2)    == "de"  
strpart("abcdefg", -2, 4) == "ab"  
strpart("abcdefg", 5, 4)  == "fg"  
strpart("abcdefg", 3)     == "defg"
```

Note: To get the first character, `{start}` must be 0. For example, to get the character under the cursor:

```
strpart(getline("."), col(".") - 1, 1, v:true)
```

Returns an empty string on error.

Can also be used as a `method` :
`GetText()->strpart(5)`

`strptime({format}, {timestring})` `strptime()`
The result is a Number, which is a unix timestamp representing the date and time in `{timestring}`, which is expected to match the format specified in `{format}`.

The accepted `{format}` depends on your system, thus this is not portable! See the manual page of the C function `strptime()` for the format. Especially avoid "%c". The value of `$TZ` also matters.

If the `{timestring}` cannot be parsed with `{format}` zero is returned. If you do not know the format of `{timestring}` you can try different `{format}` values until you get a non-zero result.

See also `strftime()` .

Examples:

```
:echo strftime("%Y %b %d %X", "1997 Apr 27 11:49:23")
862156163
:echo strftime("%c", strftime("%y%m%d %T", "970427 11:53:55"))
Sun Apr 27 11:53:55 1997
:echo strftime("%c", strftime("%Y%m%d%H%M%S", "19970427115355") + 3600)
Sun Apr 27 12:53:55 1997
```

Can also be used as a `method` :

```
GetFormat()->strftime(timestring)
```

Not available on all systems. To check use:

```
:if exists("*strftime")
```

`stridx({haystack}, {needle} [, {start}])` `stridx()`

The result is a Number, which gives the byte index in `{haystack}` of the last occurrence of the String `{needle}`. When `{start}` is specified, matches beyond this index are ignored. This can be used to find a match before a previous match:

```
:let lastcomma = stridx(line, ",")
:let comma2 = stridx(line, ",", lastcomma - 1)
```

The search is done case-sensitive.

For pattern searches use `match()` .

-1 is returned if the `{needle}` does not occur in `{haystack}`.

If the `{needle}` is empty the length of `{haystack}` is returned.

See also `stridx()` . Examples:

```
:echo stridx("an angry armadillo", "an") 3
strchr()
```

When used with a single character it works similar to the C function `strchr()`.

Can also be used as a `method` :

```
GetHaystack()->stridx(needle)
```

`strtrans({string})` `strtrans()`

The result is a String, which is `{string}` with all unprintable characters translated into printable characters `'isprint'` . Like they are shown in a window. Example:

```
echo strtrans(@a)
```

This displays a newline in register a as `"^@"` instead of starting a new line.

Returns an empty string on error.

Can also be used as a `method` :

```
GetString()->strtrans()
```

`strutf16len({string} [, {countcc}])` `strutf16len()`

The result is a Number, which is the number of UTF-16 code units in String `{string}` (after converting it to UTF-16).

When `{countcc}` is TRUE, composing characters are counted

separately.

When `{countcc}` is omitted or FALSE, composing characters are ignored.

Returns zero on error.

Also see `strlen()` and `strcharlen()` .

Examples:

```
echo strutf16len('a')           returns 1
echo strutf16len('@')           returns 1
echo strutf16len('☒')           returns 2
echo strutf16len('á')           returns 1
echo strutf16len('á', v:true)   returns 3
```

Can also be used as a `method` :

```
GetText()->strutf16len()
```

`strwidth({string})`

`strwidth()`

The result is a Number, which is the number of display cells String `{string}` occupies. A Tab character is counted as one cell, alternatively use `strdisplaywidth()` .

When `{string}` contains characters with East Asian Width Class Ambiguous, this function's return value depends on `'ambiwidth'`.

Returns zero on error.

Also see `strlen()` , `strdisplaywidth()` and `strchars()` .

Can also be used as a `method` :

```
GetString()->strwidth()
```

`submatch({nr} [, {list}])`

`submatch()` E935

Only for an expression in a `:substitute` command or `substitute()` function.

Returns the `{nr}`'th submatch of the matched text. When `{nr}` is 0 the whole matched text is returned.

Note that a NL in the string can stand for a line break of a multi-line match or a NUL character in the text.

Also see `sub-replace-expression` .

If `{list}` is present and non-zero then `submatch()` returns a list of strings, similar to `getline()` with two arguments. NL characters in the text represent NUL characters in the text.

Only returns more than one item for `:substitute` , inside `substitute()` this list will always contain one or zero items, since there are no real line breaks.

When `substitute()` is used recursively only the submatches in the current (deepest) call can be obtained.

Returns an empty string or list on error.

Examples:

```
:s/\d+/\=submatch(0) + 1/
:echo substitute(text, '\d+', '\=submatch(0) + 1', '')
```

This finds the first number in the line and adds one to it.
A line break is included as a newline character.

Can also be used as a [method](#) :
`GetNr()->submatch()`

`substitute({string}, {pat}, {sub}, {flags})` [substitute\(\)](#)
The result is a String, which is a copy of `{string}`, in which the first match of `{pat}` is replaced with `{sub}`.
When `{flags}` is "g", all matches of `{pat}` in `{string}` are replaced. Otherwise `{flags}` should be "".

This works like the ":substitute" command (without any flags). But the matching with `{pat}` is always done like the '[magic](#)' option is set and '[coptions](#)' is empty (to make scripts portable). '[ignorecase](#)' is still relevant, use `/\c` or `/\C` if you want to ignore or match case and ignore '[ignorecase](#)'. '[smartcase](#)' is not used. See [string-match](#) for how `{pat}` is used.

A "~" in `{sub}` is not replaced with the previous `{sub}`.
Note that some codes in `{sub}` have a special meaning [sub-replace-special](#) . For example, to replace something with "\n" (two characters), use "\\n" or '\n'.

When `{pat}` does not match in `{string}`, `{string}` is returned unmodified.

Example:

```
:let &path = substitute(&path, "\\=[^,]*$", "", "")
```

This removes the last component of the '[path](#)' option.

```
:echo substitute("testing", ".*", "\\U\\@", "")
```

results in "TESTING".

When `{sub}` starts with "\=", the remainder is interpreted as an expression. See [sub-replace-expression](#) . Example:

```
:echo substitute(s, '%\(\x\x\)',  
  \ '\=nr2char("0x" .. submatch(1))', 'g')
```

When `{sub}` is a Funcref that function is called, with one optional argument. Example:

```
:echo substitute(s, '%\(\x\x\)', SubNr, 'g')
```

The optional argument is a list which contains the whole matched string and up to nine submatches, like what

[submatch\(\)](#) returns. Example:

```
:echo substitute(s, '%\(\x\x\)', {m -> '0x' .. m[1]}, 'g')
```

Returns an empty string on error.

Can also be used as a [method](#) :

```
GetString()->substitute(pat, sub, flags)
```

`swapfilelist()` [swapfilelist\(\)](#)

Returns a list of swap file names, like what "vim -r" shows.

See the `-r` command argument. The **'directory'** option is used for the directories to inspect. If you only want to get a list of swap files in the current directory then temporarily set **'directory'** to a dot:

```
let save_dir = &directory
let &directory = '.'
let swapfiles = swapfilelist()
let &directory = save_dir
```

`swapinfo({fname})`

`swapinfo()`

The result is a dictionary, which holds information about the swapfile `{fname}`. The available fields are:

```
version Vim version
user      user name
host      host name
fname     original file name
pid       PID of the Vim process that created the swap
          file
mtime     last modification time in seconds
inode     Optional: INODE number of the file
dirty     1 if file was modified, 0 if not
```

Note that "user" and "host" are truncated to at most 39 bytes.

In case of failure an "error" item is added with the reason:

```
Cannot open file: file not found or in accessible
Cannot read file: cannot read first block
Not a swap file: does not contain correct block ID
Magic number mismatch: Info in first block is invalid
```

Can also be used as a **method** :

```
GetFilename()->swapinfo()
```

`swapname({buf})`

`swapname()`

The result is the swap file path of the buffer `{buf}`.

For the use of `{buf}`, see `bufname()` above.

If buffer `{buf}` is the current buffer, the result is equal to `:swapname` (unless there is no swap file).

If buffer `{buf}` has no swap file, returns an empty string.

Can also be used as a **method** :

```
GetBufname()->swapname()
```

`synID({lnum}, {col}, {trans})`

`synID()`

The result is a Number, which is the syntax ID at the position `{lnum}` and `{col}` in the current window.

The syntax ID can be used with `synIDattr()` and `synIDtrans()` to obtain syntax information about text.

`{col}` is 1 for the leftmost column, `{lnum}` is 1 for the first line. **'synmaxcol'** applies, in a longer line zero is returned.

Note that when the position is after the last character, that's where the cursor can be in Insert mode, `synID()` returns zero. `{lnum}` is used like with `getline()`.

When `{trans}` is **TRUE**, transparent items are reduced to the

item that they reveal. This is useful when wanting to know the effective color. When `{trans}` is `FALSE`, the transparent item is returned. This is useful when wanting to know which syntax item is effective (e.g. inside parens).

Warning: This function can be very slow. Best speed is obtained by going through the file in forward direction.

Returns zero on error.

Example (echoes the name of the syntax item under the cursor):
`:echo synIDattr(synID(line("."), col("."), 1), "name")`

`synIDattr({synID}, {what} [, {mode}])`

`synIDattr()`

The result is a String, which is the `{what}` attribute of syntax ID `{synID}`. This can be used to obtain information about a syntax item.

`{mode}` can be "gui", "cterm" or "term", to get the attributes for that mode. When `{mode}` is omitted, or an invalid value is used, the attributes for the currently active highlighting are used (GUI, cterm or term).

Use `synIDtrans()` to follow linked highlight groups.

<code>{what}</code>	result
"name"	the name of the syntax item
"fg"	foreground color (GUI: color name used to set the color, cterm: color number as a string, term: empty string)
"bg"	background color (as with "fg")
"font"	font name (only available in the GUI) <code>highlight-font</code>
"sp"	special color for the GUI (as with "fg") <code>highlight-guisp</code>
"ul"	underline color for cterm: number as a string
"fg#"	like "fg", but for the GUI and the GUI is running the name in "#RRGGBB" form
"bg#"	like "fg#" for "bg"
"sp#"	like "fg#" for "sp"
"bold"	"1" if bold
"italic"	"1" if italic
"reverse"	"1" if reverse
"inverse"	"1" if inverse (= reverse)
"standout"	"1" if standout
"underline"	"1" if underlined
"undercurl"	"1" if undercurled
"strike"	"1" if strikethrough
"nocombine"	"1" if nocombine

Returns an empty string on error.

Example (echoes the color of the syntax item under the cursor):

`:echo synIDattr(synIDtrans(synID(line("."), col("."), 1)), "fg")`

Can also be used as a `method` :

```
:echo synID(line("."), col("."), 1)->synIDtrans()->synIDattr("fg")
```

synIDtrans({synID})

synIDtrans()

The result is a Number, which is the translated syntax ID of {synID}. This is the syntax group ID of what is being used to highlight the character. Highlight links given with ":highlight link" are followed.

Returns zero on error.

Can also be used as a method :

```
:echo synID(line("."), col("."), 1)->synIDtrans()->synIDattr("fg")
```

synconcealed({lnum}, {col})

synconcealed()

The result is a List with currently three items:

1. The first item in the list is 0 if the character at the position {lnum} and {col} is not part of a concealable region, 1 if it is. {lnum} is used like with `getline()`.
2. The second item in the list is a string. If the first item is 1, the second item contains the text which will be displayed in place of the concealed text, depending on the current setting of 'conceallevel' and 'listchars'.
3. The third and final item in the list is a number representing the specific syntax region matched in the line. When the character is not concealed the value is zero. This allows detection of the beginning of a new concealable region if there are two consecutive regions with the same replacement character. For an example, if the text is "123456" and both "23" and "45" are concealed and replaced by the character "X", then:

call	returns
synconcealed(lnum, 1)	[0, '', 0]
synconcealed(lnum, 2)	[1, 'X', 1]
synconcealed(lnum, 3)	[1, 'X', 1]
synconcealed(lnum, 4)	[1, 'X', 2]
synconcealed(lnum, 5)	[1, 'X', 2]
synconcealed(lnum, 6)	[0, '', 0]

synstack({lnum}, {col})

synstack()

Return a List, which is the stack of syntax items at the position {lnum} and {col} in the current window. {lnum} is used like with `getline()`. Each item in the List is an ID like what `synID()` returns.

The first item in the List is the outer region, following are items contained in that one. The last one is what `synID()` returns, unless not the whole item is highlighted or it is a transparent item.

This function is useful for debugging a syntax file.

Example that shows the syntax stack under the cursor:

```
for id in synstack(line("."), col("."))
  echo synIDattr(id, "name")
endfor
```

When the position specified with `{lnum}` and `{col}` is invalid an empty List is returned. The position just after the last character in a line and the first column in an empty line are valid positions.

`system({expr} [, {input}])` system() E677
Get the output of the shell command `{expr}` as a `String`. See `systemlist()` to get the output as a `List`.

When `{input}` is given and is a `String` this string is written to a file and passed as stdin to the command. The string is written as-is, you need to take care of using the correct line separators yourself.

If `{input}` is given and is a `List` it is written to the file in a way `writefile()` does with `{binary}` set to "b" (i.e. with a newline between each list item with newlines inside list items converted to NULs).

When `{input}` is given and is a number that is a valid id for an existing buffer then the content of the buffer is written to the file line by line, each line terminated by a NL and NULs characters where the text has a NL.

Pipes are not used, the `'shelltemp'` option is not used.

When prepended by `:silent` the terminal will not be set to cooked mode. This is meant to be used for commands that do not need the user to type. It avoids stray characters showing up on the screen which require `CTRL-L` to remove.

```
:silent let f = system('ls *.vim')
```

Note: Use `shellescape()` or `::S` with `expand()` or `fnamemodify()` to escape special characters in a command argument. Newlines in `{expr}` may cause the command to fail. The characters in `'shellquote'` and `'shellxquote'` may also cause trouble.

This is not to be used for interactive commands.

The result is a `String`. Example:

```
:let files = system('ls ' .. shellescape(expand('%:h')))
:let files = system('ls ' .. expand('%:h:S'))
```

To make the result more system-independent, the shell output is filtered to replace `<CR>` with `<NL>` for Macintosh, and `<CR><NL>` with `<NL>` for DOS-like systems.

To avoid the string being truncated at a NUL, all NUL characters are replaced with SOH (0x01).

The command executed is constructed using several options:

`'shell'` `'shellcmdflag'` `'shellxquote'` `{expr}` `'shellredir'` `{tmp}` `'shellxquote'`
(`{tmp}` is an automatically generated file name).

For Unix, braces are put around `{expr}` to allow for concatenated commands.

The command will be executed in "cooked" mode, so that a

CTRL-C will interrupt the command (on Unix at least).

The resulting error code can be found in `v:shell_error` .
This function will fail in `restricted-mode` .

Note that any wrong value in the options mentioned above may make the function fail. It has also been reported to fail when using a security agent application.
Unlike `:!cmd` there is no automatic check for changed files.
Use `:checktime` to force a check.

Can also be used as a `method` :
`:echo GetCmd()->system()`

`systemlist({expr} [, {input}])` `systemlist()`
Same as `system()` , but returns a `List` with lines (parts of output separated by NL) with NULs transformed into NLs. Output is the same as `readfile()` will output with `{binary}` argument set to "b", except that there is no extra empty item when the result ends in a NL.
Note that on MS-Windows you may get trailing CR characters.

To see the difference between "echo hello" and "echo -n hello" use `system()` and `split()` :
`echo system('echo hello')->split('\n', 1)`

Returns an empty string on error.

Can also be used as a `method` :
`:echo GetCmd()->systemlist()`

`tabpagebuflist([{arg}])` `tabpagebuflist()`
The result is a `List` , where each item is the number of the buffer associated with each window in the current tab page. `{arg}` specifies the number of the tab page to be used. When omitted the current tab page is used.
When `{arg}` is invalid the number zero is returned.
To get a list of all buffers in all tabs use this:
`let buflist = []
for i in range(tabpagenr('$'))
 call extend(buflist, tabpagebuflist(i + 1))
endfor`

Note that a buffer may appear in more than one window.

Can also be used as a `method` :
`GetTabPage()->tabpagebuflist()`

`tabpagenr([{arg}])` `tabpagenr()`
The result is a `Number`, which is the number of the current tab page. The first tab page has number 1.

The optional argument `{arg}` supports the following values:

\$ the number of the last tab page (the tab page count).
the number of the last accessed tab page (where `g<Tab>` goes to). if there is no previous tab page 0 is returned.
The number can be used with the `:tab` command.

Returns zero on error.

`tabpagewinnr({tabarg} [, {arg}])` `tabpagewinnr()`
Like `winnr()` but for tab page `{tabarg}`.
`{tabarg}` specifies the number of tab page to be used.
`{arg}` is used like with `winnr()` :
- When omitted the current window number is returned. This is the window which will be used when going to this tab page.
- When "\$" the number of windows is returned.
- When "#" the previous window nr is returned.
Useful examples:
`tabpagewinnr(1)` " current window of tab page 1
`tabpagewinnr(4, '$')` " number of windows in tab page 4
When `{tabarg}` is invalid zero is returned.

Can also be used as a `method` :
`GetTabPage()->tabpagewinnr()`

`tagfiles()` `tagfiles()`
Returns a `List` with the file names used to search for tags for the current buffer. This is the `'tags'` option expanded.

`taglist({expr} [, {filename}])` `taglist()`
Returns a `List` of tags matching the regular expression `{expr}`.
If `{filename}` is passed it is used to prioritize the results in the same way that `:tselect` does. See `tag-priority`.
`{filename}` should be the full path of the file.

Each list item is a dictionary with at least the following entries:

<code>name</code>	Name of the tag.
<code>filename</code>	Name of the file where the tag is defined. It is either relative to the current directory or a full path.
<code>cmd</code>	Ex command used to locate the tag in the file.
<code>kind</code>	Type of the tag. The value for this entry depends on the language specific kind values. Only available when using a tags file generated by Universal/Exuberant ctags or hdrtag.
<code>static</code>	A file specific tag. Refer to <code>static-tag</code> for more information.

More entries may be present, depending on the content of the

tags file: access, implementation, inherits and signature. Refer to the ctags documentation for information about these fields. For C code the fields "struct", "class" and "enum" may appear, they give the name of the entity the tag is contained in.

The ex-command "cmd" can be either an ex search pattern, a line number or a line number followed by a byte number.

If there are no matching tags, then an empty list is returned.

To get an exact tag match, the anchors '^' and '\$' should be used in {expr}. This also make the function work faster. Refer to [tag-regex](#) for more information about the tag search regular expression pattern.

Refer to ['tags'](#) for information about how the tags file is located by Vim. Refer to [tags-file-format](#) for the format of the tags file generated by the different ctags tools.

Can also be used as a [method](#) :
[GetTagpattern\(\)->taglist\(\)](#)

tan({expr})

tan()

Return the tangent of {expr}, measured in radians, as a [Float](#) in the range [-inf, inf].

{expr} must evaluate to a [Float](#) or a [Number](#) .

Returns 0.0 if {expr} is not a [Float](#) or a [Number](#) .

Examples:

```
:echo tan(10)
0.648361
:echo tan(-4.01)
-1.181502
```

Can also be used as a [method](#) :
[Compute\(\)->tan\(\)](#)

tanh({expr})

tanh()

Return the hyperbolic tangent of {expr} as a [Float](#) in the range [-1, 1].

{expr} must evaluate to a [Float](#) or a [Number](#) .

Returns 0.0 if {expr} is not a [Float](#) or a [Number](#) .

Examples:

```
:echo tanh(0.5)
0.462117
:echo tanh(-1)
-0.761594
```

Can also be used as a [method](#) :
[Compute\(\)->tanh\(\)](#)

tempname()

tempname() temp-file-name

The result is a String, which is the name of a file that doesn't exist. It can be used for a temporary file. The name is different for at least 26 consecutive calls. Example:

```
:let tmpfile = tempname()
:exe "redir > " .. tmpfile
```

For Unix, the file will be in a private directory `tmpfile`. For MS-Windows forward slashes are used when the `'shellslash'` option is set, or when `'shellcmdflag'` starts with '-' and `'shell'` does not contain powershell or pwsh.

term_ functions are documented here: [terminal-function-details](#)

terminalprops()

terminalprops()

Returns a [Dictionary](#) with properties of the terminal that Vim detected from the response to `t_RV` request. See [v:termresponse](#) for the response itself. If [v:termresponse](#) is empty most values here will be 'u' for unknown.

cursor_style	whether sending <code>t_RS</code> works	**
cursor_blink_mode	whether sending <code>t_RC</code> works	**
underline_rgb	whether <code>t_8u</code> works	**
mouse	mouse type supported	
kitty	whether Kitty terminal was detected	

** value 'u' for unknown, 'y' for yes, 'n' for no

If the [+termresponse](#) feature is missing then the result is an empty dictionary.

If "cursor_style" is 'y' then `t_RS` will be sent to request the current cursor style.

If "cursor_blink_mode" is 'y' then `t_RC` will be sent to request the cursor blink status.

"cursor_style" and "cursor_blink_mode" are also set if `t_u7` is not empty, Vim will detect the working of sending `t_RS` and `t_RC` on startup.

When "underline_rgb" is not 'y', then `t_8u` will be made empty. This avoids sending it to xterm, which would clear the colors.

For "mouse" the value 'u' is unknown

Also see:

- `'ambiwidth'` - detected by using `t_u7`.
- [v:termstyleresp](#) and [v:termblinkresp](#) for the response to `t_RS` and `t_RC`.

test_ functions are documented here: [test-functions-details](#)

timer_info([id])

timer_info()

Return a list with information about timers.
When `{id}` is given only information about this timer is returned. When timer `{id}` does not exist an empty list is returned.
When `{id}` is omitted information about all timers is returned.

For each timer the information is stored in a [Dictionary](#) with these items:

"id"	the timer ID
"time"	time the timer was started with
"remaining"	time until the timer fires
"repeat"	number of times the timer will still fire; -1 means forever
"callback"	the callback
"paused"	1 if the timer is paused, 0 otherwise

Can also be used as a [method](#) :
`GetTimer()->timer_info()`

{only available when compiled with the `|+timers|` feature}

`timer_pause({timer}, {paused})` [timer_pause\(\)](#)
Pause or unpause a timer. A paused timer does not invoke its callback when its time expires. Unpausing a timer may cause the callback to be invoked almost immediately if enough time has passed.

Pausing a timer is useful to avoid the callback to be called for a short time.

If `{paused}` evaluates to a non-zero Number or a non-empty String, then the timer is paused, otherwise it is unpaused. See [non-zero-arg](#) .

Can also be used as a [method](#) :
`GetTimer()->timer_pause(1)`

{only available when compiled with the `|+timers|` feature}

`timer_start({time}, {callback} [, {options}])` [timer_start\(\)](#) [timer](#) [timers](#)
Create a timer and return the timer ID.

`{time}` is the waiting time in milliseconds. This is the minimum time before invoking the callback. When the system is busy or Vim is not waiting for input the time will be longer. Zero can be used to execute the callback when Vim is back in the main loop.

`{callback}` is the function to call. It can be the name of a function or a [Funcref](#) . It is called with one argument, which is the timer ID. The callback is only invoked when Vim is waiting for input.

If you want to show a message look at [popup_notification\(\)](#)

to avoid interfering with what the user is doing.

`{options}` is a dictionary. Supported entries:

"repeat" Number of times to repeat calling the callback. -1 means forever. When not present the callback will be called once. If the timer causes an error three times in a row the repeat is cancelled. This avoids that Vim becomes unusable because of all the error messages.

Returns -1 on error.

Example:

```
func MyHandler(timer)
    echo 'Handler called'
endfunc
let timer = timer_start(500, 'MyHandler',
    \ {'repeat': 3})
```

This will invoke `MyHandler()` three times at 500 msec intervals.

Can also be used as a `method` :

```
GetMsec()->timer_start(callback)
```

Not available in the `sandbox` .

`{only available when compiled with the |+timers| feature}`

`timer_stop({timer})` `timer_stop()`
Stop a timer. The timer callback will no longer be invoked. `{timer}` is an ID returned by `timer_start()`, thus it must be a Number. If `{timer}` does not exist there is no error.

Can also be used as a `method` :

```
GetTimer()->timer_stop()
```

`{only available when compiled with the |+timers| feature}`

`timer_stopall()` `timer_stopall()`
Stop all timers. The timer callbacks will no longer be invoked. Useful if a timer is misbehaving. If there are no timers there is no error.

`{only available when compiled with the |+timers| feature}`

`tolower({expr})` `tolower()`
The result is a copy of the String given, with all uppercase characters turned into lowercase (just like applying `gu` to the string). Returns an empty string on error.

Can also be used as a `method` :

```
GetText()->tolower()
```

`toupper({expr})` `toupper()`

The result is a copy of the String given, with all lowercase characters turned into uppercase (just like applying `gU` to the string). Returns an empty string on error.

Can also be used as a `method` :
`GetText()->toupper()`

`tr({src}, {fromstr}, {tostr})` `tr()`
The result is a copy of the `{src}` string with all characters which appear in `{fromstr}` replaced by the character in that position in the `{tostr}` string. Thus the first character in `{fromstr}` is translated into the first character in `{tostr}` and so on. Exactly like the unix "tr" command. This code also deals with multibyte characters properly.

Returns an empty string on error.

Examples:

```
echo tr("hello there", "ht", "HT")
returns "Hello THere"
echo tr("<blob>", "<>", "{}")
returns "{blob}"
```

Can also be used as a `method` :
`GetText()->tr(from, to)`

`trim({text} [, {mask} [, {dir}]])` `trim()`
Return `{text}` as a String where any character in `{mask}` is removed from the beginning and/or end of `{text}`.

If `{mask}` is not given, or is an empty string, `{mask}` is all characters up to `0x20`, which includes Tab, space, NL and CR, plus the non-breaking space character `0xa0`.

The optional `{dir}` argument specifies where to remove the characters:

0	remove from the beginning and end of <code>{text}</code>
1	remove only at the beginning of <code>{text}</code>
2	remove only at the end of <code>{text}</code>

When omitted both ends are trimmed.

This function deals with multibyte characters properly.
Returns an empty string on error.

Examples:

```
echo trim("  some text ")
returns "some text"
echo trim(" \r\t\t\r RESERVE \t\n\x0B\xA0") .. "_TAIL"
returns "RESERVE_TAIL"
echo trim("rm<Xrm<>X>rrm", "rm<>")
returns "Xrm<>X" (characters in the middle are not removed)
echo trim(" vim ", " ", 2)
returns " vim"
```

Can also be used as a `method` :
`GetText()->trim()`

`trunc({expr})`

`trunc()`

Return the largest integral value with magnitude less than or equal to `{expr}` as a `Float` (truncate towards zero). `{expr}` must evaluate to a `Float` or a `Number` . Returns `0.0` if `{expr}` is not a `Float` or a `Number` .
Examples:

```
echo trunc(1.456)
1.0
echo trunc(-5.456)
-5.0
echo trunc(4.0)
4.0
```

Can also be used as a `method` :
`Compute()->trunc()`

`type({expr})`

`type()`

The result is a `Number` representing the type of `{expr}`. Instead of using the number directly, it is better to use the `v:t_` variable that has the value:

```
Number:    0    v:t_number
String:    1    v:t_string
Funcref:   2    v:t_func
List:      3    v:t_list
Dictionary: 4    v:t_dict
Float:     5    v:t_float
Boolean:   6    v:t_bool (v:false and v:true)
None:      7    v:t_none (v:null and v:none)
Job:       8    v:t_job
Channel:   9    v:t_channel
Blob:     10    v:t_blob
Class:    12    v:t_class
Object:   13    v:t_object
Typealias: 14    v:t_typealias
```

For backward compatibility, this method can be used:

```
:if type(myvar) == type(0)
:if type(myvar) == type("")
:if type(myvar) == type(function("tr"))
:if type(myvar) == type([])
:if type(myvar) == type({})
:if type(myvar) == type(0.0)
:if type(myvar) == type(v:false)
:if type(myvar) == type(v:none)
```

To check if the `v:t_` variables exist use this:
`:if exists('v:t_number')`

Can also be used as a `method` :
`mylist->type()`

`typename({expr})`

`typename()`

Return a string representation of the type of `{expr}`.

Example:

```
echo typename([1, 2, 3])
list<number>
```

`undofile({name})`

`undofile()`

Return the name of the undo file that would be used for a file with name `{name}` when writing. This uses the `'undodir'` option, finding directories that exist. It does not check if the undo file exists.

`{name}` is always expanded to the full path, since that is what is used internally.

If `{name}` is empty `undofile()` returns an empty string, since a buffer without a file name will not write an undo file.

Useful in combination with `:wundo` and `:rundo`.

When compiled without the `+persistent_undo` option this always returns an empty string.

Can also be used as a `method` :

```
GetFilename()->undofile()
```

`undotree([buf])`

`undotree()`

Return the current state of the undo tree for the current buffer, or for a specific buffer if `{buf}` is given. The result is a dictionary with the following items:

"seq_last"	The highest undo sequence number used.
"seq_cur"	The sequence number of the current position in the undo tree. This differs from "seq_last" when some changes were undone.
"time_cur"	Time last used for <code>:earlier</code> and related commands. Use <code>strftime()</code> to convert to something readable.
"save_last"	Number of the last file write. Zero when no write yet.
"save_cur"	Number of the current position in the undo tree.
"synced"	Non-zero when the last undo block was synced. This happens when waiting from input from the user. See undo-blocks .
"entries"	A list of dictionaries with information about undo blocks.

The first item in the "entries" list is the oldest undo item.

Each List item is a `Dictionary` with these items:

"seq"	Undo sequence number. Same as what appears in <code>:undolist</code> .
"time"	Timestamp when the change happened. Use <code>strftime()</code> to convert to something readable.
"newhead"	Only appears in the item that is the last one that was added. This marks the last change and where further changes will be added.
"curhead"	Only appears in the item that is the last one that was undone. This marks the current

position in the undo tree, the block that will be used by a redo command. When nothing was undone after the last change this item will not appear anywhere.

"save" Only appears on the last block before a file write. The number is the write count. The first write has number 1, the last one the "save_last" mentioned above.

"alt" Alternate entry. This is again a List of undo blocks. Each item may again have an "alt" item.

`uniq({list} [, {func} [, {dict}]])` uniq() E882

Remove second and succeeding copies of repeated adjacent {list} items in-place. Returns {list}. If you want a list to remain unmodified make a copy first:

```
:let newlist = uniq(copy(mylist))
```

The default compare function uses the string representation of each item. For the use of {func} and {dict} see `sort()`.

Returns zero if {list} is not a List .

Can also be used as a method :

```
mylist->uniq()
```

utf16idx()

`utf16idx({string}, {idx} [, {countcc} [, {charidx}]])`

Same as `charidx()` but returns the UTF-16 code unit index of the byte at {idx} in {string} (after converting it to UTF-16).

When {charidx} is present and TRUE, {idx} is used as the character index in the String {string} instead of as the byte index.

An {idx} in the middle of a UTF-8 sequence is rounded downwards to the beginning of that sequence.

Returns -1 if the arguments are invalid or if there are less than {idx} bytes in {string}. If there are exactly {idx} bytes the length of the string in UTF-16 code units is returned.

See `byteidx()` and `byteidxcomp()` for getting the byte index from the UTF-16 index and `charidx()` for getting the character index from the UTF-16 index.

Refer to `string-offset-encoding` for more information.

Examples:

```
echo utf16idx('a\xff', 3) returns 2
echo utf16idx('a\xff', 7) returns 4
echo utf16idx('a\xff', 1, 0, 1) returns 2
echo utf16idx('a\xff', 2, 0, 1) returns 4
echo utf16idx('a\xc3', 6) returns 2
echo utf16idx('a\xc3', 6, 1) returns 4
echo utf16idx('a\xff', 9) returns -1
```

Can also be used as a method :

GetName()->utf16idx(idx)

values({dict})

values()

Return a `List` with all the values of `{dict}`. The `List` is in arbitrary order. Also see `items()` and `keys()`. Returns zero if `{dict}` is not a `Dict`.

Can also be used as a `method` :
`mydict->values()`

virtcol({expr} [, {list} [, {winid}]])

virtcol()

The result is a `Number`, which is the screen column of the file position given with `{expr}`. That is, the last screen position occupied by the character at that position, when the screen would be of unlimited width. When there is a `<Tab>` at the position, the returned `Number` will be the column at the end of the `<Tab>`. For example, for a `<Tab>` in column 1, with `'ts'` set to 8, it returns 8. `conceal` is ignored. For the byte position use `col()`.

For the use of `{expr}` see `col()`.

When `'virtualedit'` is used `{expr}` can be `[lnum, col, off]`, where "off" is the offset in screen columns from the start of the character. E.g., a position within a `<Tab>` or after the last character. When "off" is omitted zero is used. When Virtual editing is active in the current mode, a position beyond the end of the line can be returned. Also see `'virtualedit'`

The accepted positions are:

- `.` the cursor position
- `$` the end of the cursor line (the result is the number of displayed characters in the cursor line plus one)
- `'x` position of mark x (if the mark is not set, 0 is returned)
- `v` In Visual mode: the start of the Visual area (the cursor is the end). When not in Visual mode returns the cursor position. Differs from `'<` in that it's updated right away.

If `{list}` is present and non-zero then `virtcol()` returns a `List` with the first and last screen position occupied by the character.

With the optional `{winid}` argument the values are obtained for that window instead of the current window.

Note that only marks in the current file can be used.

Examples:

`" With text "foo^Lbar" and cursor on the "^L":`

```
virtcol(".")      " returns 5
virtcol(".", 1)  " returns [4, 5]
virtcol("$")     " returns 9
```

```
" With text "      there", with 't at 'h':
```

```
virtcol("'t")     " returns 6
```

The first column is 1. 0 or [0, 0] is returned for an error.
A more advanced example that echoes the maximum length of all lines:

```
echo max(map(range(1, line('$')), "virtcol([v:val, '$'])"))
```

Can also be used as a `method` :

```
GetPos()->virtcol()
```

```
virtcol2col({winid}, {lnum}, {col})          virtcol2col()
```

The result is a Number, which is the byte index of the character in window `{winid}` at buffer line `{lnum}` and virtual column `{col}`.

If buffer line `{lnum}` is an empty line, 0 is returned.

If `{col}` is greater than the last virtual column in line `{lnum}`, then the byte index of the character at the last virtual column is returned.

For a multi-byte character, the column number of the first byte in the character is returned.

The `{winid}` argument can be the window number or the `window-ID` . If this is zero, then the current window is used.

Returns -1 if the window `{winid}` doesn't exist or the buffer line `{lnum}` or virtual column `{col}` is invalid.

See also `screenpos()` , `virtcol()` and `col()` .

Can also be used as a `method` :

```
GetWinid()->virtcol2col(lnum, col)
```

```
visualmode([{expr}])          visualmode()
```

The result is a String, which describes the last Visual mode used in the current buffer. Initially it returns an empty string, but once Visual mode has been used, it returns "v", "V", or "<CTRL-V>" (a single **CTRL-V** character) for character-wise, line-wise, or block-wise Visual mode respectively.

Example:

```
:exe "normal " .. visualmode()
```

This enters the same Visual mode as before. It is also useful in scripts if you wish to act differently depending on the Visual mode that was used.

If Visual mode is active, use `mode()` to get the Visual mode (e.g., in a `:vmap`).

If `{expr}` is supplied and it evaluates to a non-zero Number or a non-empty String, then the Visual mode will be cleared and the old value is returned. See `non-zero-arg`.

`wildmenumode()` `wildmenumode()`
Returns `TRUE` when the wildmenu is active and `FALSE` otherwise. See `'wildmenu'` and `'wildmode'`.
This can be used in mappings to handle the `'wildcharm'` option gracefully. (Makes only sense with `mapmode-c` mappings).

For example to make `<c-j>` work like `<down>` in wildmode, use:
`:cnoremap <expr> <C-j> wildmenumode() ? "\<Down>\<Tab>" : "\<c-j>"`

(Note, this needs the `'wildcharm'` option set appropriately).

`win_execute({id}, {command} [, {silent}])` `win_execute()`
Like `'execute()'` but in the context of window `{id}`.
The window will temporarily be made the current window, without triggering autocommands or changing directory. When executing `{command}` autocommands will be triggered, this may have unexpected side effects. Use `'noautocmd'` if needed.
Example:

`call win_execute(winid, 'set syntax=python')`
Doing the same with `'setwinvar()'` would not trigger autocommands and not actually show syntax highlighting.

`E994`

Not all commands are allowed in popup windows.
When window `{id}` does not exist then no error is given and an empty string is returned.

Can also be used as a `method`, the base is passed as the second argument:

`GetCommand()->win_execute(winid)`

`win_findbuf({bufnr})` `win_findbuf()`
Returns a `List` with `window-ID`s for windows that contain buffer `{bufnr}`. When there is none the list is empty.

Can also be used as a `method`:

`GetBufnr()->win_findbuf()`

`win_getid([win] [, {tab}])` `win_getid()`
Get the `window-ID` for the specified window.
When `{win}` is missing use the current window.
With `{win}` this is the window number. The top window has number 1.
Without `{tab}` use the current tab, otherwise the tab with number `{tab}`. The first tab has number one.
Return zero if the window cannot be found.

Can also be used as a `method`:

`GetWinnr()->win_getid()`

`win_gettype([{nr}])` `win_gettype()`

Return the type of the window:

<code>"autocmd"</code>	autocommand window. Temporary window used to execute autocommands.
<code>"command"</code>	command-line window <code>cmdwin</code>
<code>(empty)</code>	normal window
<code>"loclist"</code>	<code>location-list-window</code>
<code>"popup"</code>	popup window <code>popup</code>
<code>"preview"</code>	preview window <code>preview-window</code>
<code>"quickfix"</code>	<code>quickfix-window</code>
<code>"unknown"</code>	window <code>{nr}</code> not found

When `{nr}` is omitted return the type of the current window.
 When `{nr}` is given return the type of this window by number or `window-ID` .

Also see the `'buftype'` option. When running a terminal in a popup window then `'buftype'` is "terminal" and `win_gettype()` returns "popup".

Can also be used as a `method` :
`GetWinid()->win_gettype()`

`win_gotoid({expr})` `win_gotoid()`

Go to window with ID `{expr}`. This may also change the current tabpage.
 Return TRUE if successful, FALSE if the window cannot be found.

Can also be used as a `method` :
`GetWinid()->win_gotoid()`

`win_id2tabwin({expr})` `win_id2tabwin()`

Return a list with the tab number and window number of window with ID `{expr}`: `[tabnr, winnr]`.
 Return `[0, 0]` if the window cannot be found.

Can also be used as a `method` :
`GetWinid()->win_id2tabwin()`

`win_id2win({expr})` `win_id2win()`

Return the window number of window with ID `{expr}`.
 Return 0 if the window cannot be found in the current tabpage.

Can also be used as a `method` :
`GetWinid()->win_id2win()`

`win_move_separator({nr}, {offset})` `win_move_separator()`

Move window `{nr}`'s vertical separator (i.e., the right border) by `{offset}` columns, as if being dragged by the mouse. `{nr}` can be a window number or `window-ID` . A positive `{offset}` moves right and a negative `{offset}` moves left. Moving a window's vertical separator will change the width of the window and the width of other windows adjacent to the vertical separator. The magnitude of movement may be smaller than

specified (e.g., as a consequence of maintaining `'winminwidth'`). Returns TRUE if the window can be found and FALSE otherwise.

This will fail for the rightmost window and a full-width window, since it has no separator on the right.

Only works for the current tab page. [E1308](#)

Can also be used as a [method](#) :

`GetWinnr()->win_move_separator(offset)`

`win_move_statusline({nr}, {offset})` [win_move_statusline\(\)](#)

Move window `{nr}`'s status line (i.e., the bottom border) by `{offset}` rows, as if being dragged by the mouse. `{nr}` can be a window number or [window-ID](#) . A positive `{offset}` moves down and a negative `{offset}` moves up. Moving a window's status line will change the height of the window and the height of other windows adjacent to the status line. The magnitude of movement may be smaller than specified (e.g., as a consequence of maintaining `'winminheight'`). Returns TRUE if the window can be found and FALSE otherwise.

Only works for the current tab page.

Can also be used as a [method](#) :

`GetWinnr()->win_move_statusline(offset)`

`win_screenpos({nr})` [win_screenpos\(\)](#)

Return the screen position of window `{nr}` as a list with two numbers: [row, col]. The first window always has position [1, 1], unless there is a tabline, then it is [2, 1].

`{nr}` can be the window number or the [window-ID](#) . Use zero for the current window.

Returns [0, 0] if the window cannot be found.

Can also be used as a [method](#) :

`GetWinid()->win_screenpos()`

`win_splitmove({nr}, {target} [, {options}])` [win_splitmove\(\)](#)

Temporarily switch to window `{target}`, then move window `{nr}` to a new split adjacent to `{target}`.

Unlike commands such as `:split` , no new windows are created (the [window-ID](#) of window `{nr}` is unchanged after the move).

Both `{nr}` and `{target}` can be window numbers or [window-ID](#) s. Both must be in the current tab page.

Returns zero for success, non-zero for failure.

`{options}` is a [Dictionary](#) with the following optional entries:

"vertical" When TRUE, the split is created vertically, like with `:vsplit` .

"rightbelow" When TRUE, the split is made below or to the right (if vertical). When FALSE, it is done above or to the left (if vertical). When not present, the values of `'splitbelow'` and

'splitright' are used.

Can also be used as a `method` :

```
GetWinid()->win_splitmove(target)
```

`winbufnr({nr})` `winbufnr()`
The result is a Number, which is the number of the buffer associated with window `{nr}`. `{nr}` can be the window number or the `window-ID`.

When `{nr}` is zero, the number of the buffer in the current window is returned.

When window `{nr}` doesn't exist, -1 is returned.

Example:

```
:echo "The file in the current window is " . bufname(winbufnr(0))
```

Can also be used as a `method` :

```
FindWindow()->winbufnr()->bufname()
```

`wincol()` `wincol()`
The result is a Number, which is the virtual column of the cursor in the window. This is counting screen cells from the left side of the window. The leftmost column is one.

`windowsversion()` `windowsversion()`
The result is a String. For MS-Windows it indicates the OS version. E.g, Windows 10 is "10.0", Windows 8 is "6.2", Windows XP is "5.1". For non-MS-Windows systems the result is an empty string.

`winheight({nr})` `winheight()`
The result is a Number, which is the height of window `{nr}`. `{nr}` can be the window number or the `window-ID`.
When `{nr}` is zero, the height of the current window is returned. When window `{nr}` doesn't exist, -1 is returned.
An existing window always has a height of zero or more.
This excludes any window toolbar line.

Examples:

```
:echo "The current window has " .. winheight(0) .. " lines."
```

Can also be used as a `method` :

```
GetWinid()->winheight()
```

`winlayout([{{tabnr}}])` `winlayout()`
The result is a nested List containing the layout of windows in a tabpage.

Without `{tabnr}` use the current tabpage, otherwise the tabpage with number `{tabnr}`. If the tabpage `{tabnr}` is not found, returns an empty list.

For a leaf window, it returns:

```
['leaf', {winid}]
```

For horizontally split windows, which form a column, it returns:

```
['col', [{nested list of windows}]]
```

For vertically split windows, which form a row, it returns:

```
['row', [{nested list of windows}]]
```

Example:

```
" Only one window in the tab page
:echo winlayout()
['leaf', 1000]
" Two horizontally split windows
:echo winlayout()
['col', [['leaf', 1000], ['leaf', 1001]]]
" The second tab page, with three horizontally split
" windows, with two vertically split windows in the
" middle window
:echo winlayout(2)
['col', [['leaf', 1002], ['row', [['leaf', 1003],
                                   ['leaf', 1001]]], ['leaf', 1000]]]
```

Can also be used as a `method` :

```
GetTabnr()->winlayout()
```

`winline()` `winline()`
The result is a Number, which is the screen line of the cursor in the window. This is counting screen lines from the top of the window. The first line is one. If the cursor was moved the view on the file will be updated first, this may cause a scroll.

`winnr([arg])` `winnr()`
The result is a Number, which is the number of the current window. The top window has number 1. Returns zero for a popup window.

The optional argument `{arg}` supports the following values:

```
$      the number of the last window (the window
count).
#      the number of the last accessed window (where
CTRL-W_p goes to). If there is no previous
window or it is in another tab page 0 is
returned. May refer to the current window in
some cases (e.g. when evaluating 'statusline'
expressions).
{N}j  the number of the Nth window below the
current window (where CTRL-W_j goes to).
{N}k  the number of the Nth window above the current
window (where CTRL-W_k goes to).
{N}h  the number of the Nth window left of the
current window (where CTRL-W_h goes to).
{N}l  the number of the Nth window right of the
current window (where CTRL-W_l goes to).
```

The number can be used with `CTRL-W_w` and `":wincmd w"`
`:wincmd .`

When `{arg}` is invalid an error is given and zero is returned. Also see `tabpagewinnr()` and `win_getid()` .

Examples:

```
let window_count = winnr('$')
let prev_window = winnr('#')
let wnum = winnr('3k')
```

Can also be used as a `method` :

```
GetWinval()->winnr()
```

`winrestcmd()` Returns a sequence of `:resize` commands that should restore the current window sizes. Only works properly when no windows are opened or closed and the current window and tab page is unchanged.

Example:

```
:let cmd = winrestcmd()
:call MessWithWindowSizes()
:exe cmd
```

`winrestview({dict})` Uses the `Dictionary` returned by `winsaveview()` to restore the view of the current window.

Note: The `{dict}` does not have to contain all values, that are returned by `winsaveview()` . If values are missing, those settings won't be restored. So you can use:

```
:call winrestview({'curswant': 4})
```

This will only set the `curswant` value (the column the cursor wants to move on vertical movements) of the cursor to column 5 (yes, that is 5), while all other settings will remain the same. This is useful, if you set the cursor position manually.

If you have changed the values the result is unpredictable. If the window size changed the result won't be the same.

Can also be used as a `method` :

```
GetView()->winrestview()
```

`winsaveview()` Returns a `Dictionary` that contains information to restore the view of the current window. Use `winrestview()` to restore the view.

This is useful if you have a mapping that jumps around in the buffer and you want to go back to the original view.

This does not save fold information. Use the `'foldenable'` option to temporarily switch off folding, so that folds are not opened when moving around. This may have side effects.

The return value includes:

<code>lnum</code>	cursor line number
<code>col</code>	cursor column (Note: the first column zero, as opposed to what <code>getcurpos()</code> returns)

coladd	cursor column offset for 'virtualedit'
curswant	column for vertical movement (Note: the first column is zero, as opposed to what <code>getcurpos()</code> returns). After <code>\$</code> command it will be a very large number equal to <code>v:maxcol</code> .
topline	first line in the window
topfill	filler lines, only in diff mode
leftcol	first column displayed; only used when 'wrap' is off
skipcol	columns skipped

Note that no option values are saved.

`winwidth({nr})`

`winwidth()`

The result is a Number, which is the width of window `{nr}`. `{nr}` can be the window number or the `window-ID` .

When `{nr}` is zero, the width of the current window is returned. When window `{nr}` doesn't exist, -1 is returned. An existing window always has a width of zero or more.

Examples:

```
:echo "The current window has " .. winwidth(0) .. " columns."
:if winwidth(0) <= 50
: 50 wincmd |
:endif
```

For getting the terminal or screen size, see the **'columns'** option.

Can also be used as a `method` :
`GetWinid()->winwidth()`

`wordcount()`

`wordcount()`

The result is a dictionary of byte/chars/word statistics for the current buffer. This is the same info as provided by `g_CTRL-G`

The return value includes:

bytes	Number of bytes in the buffer
chars	Number of chars in the buffer
words	Number of words in the buffer
cursor_bytes	Number of bytes before cursor position (not in Visual mode)
cursor_chars	Number of chars before cursor position (not in Visual mode)
cursor_words	Number of words before cursor position (not in Visual mode)
visual_bytes	Number of bytes visually selected (only in Visual mode)
visual_chars	Number of chars visually selected (only in Visual mode)
visual_words	Number of words visually selected (only in Visual mode)

writefile()

writefile({object}, {fname} [, {flags}])

When {object} is a `List` write it to file {fname}. Each list item is separated with a NL. Each list item must be a String or Number.

All NL characters are replaced with a NUL character.

Inserting CR characters needs to be done before passing {list} to writefile().

When {object} is a `Blob` write the bytes to file {fname} unmodified, also when binary mode is not specified.

{flags} must be a String. These characters are recognized:

'b' Binary mode is used: There will not be a NL after the last list item. An empty item at the end does cause the last line in the file to end in a NL.

'a' Append mode is used, lines are appended to the file:

```
:call writefile(["foo"], "event.log", "a")
:call writefile(["bar"], "event.log", "a")
```

'D' Delete the file when the current function ends. This works like:

```
:defer delete({fname})
```

Fails when not in a function. Also see `:defer` .

's' fsync() is called after writing the file. This flushes the file to disk, if possible. This takes more time but avoids losing the file if the system crashes.

'S' fsync() is not called, even when `'fsync'` is set.

When {flags} does not contain "S" or "s" then fsync() is called if the `'fsync'` option is set.

An existing file is overwritten, if possible.

When the write fails -1 is returned, otherwise 0. There is an error message if the file can't be created or when writing fails.

Also see `readfile()` .

To copy a file byte for byte:

```
:let fl = readfile("foo", "b")
:call writefile(fl, "foocopy", "b")
```

Can also be used as a `method` :

```
GetText()->writefile("thefile")
```

xor({expr}, {expr})

xor()

Bitwise XOR on the two arguments. The arguments are converted to a number. A List, Dict or Float argument causes an error.

Also see ``and()`` and ``or()``.

Example:

```
:let bits = xor(bits, 0x80)
```

Can also be used as a `method` :

```
:let bits = bits->xor(0x80)
```

3. Feature list

`feature-list`

There are three types of features:

1. Features that are only supported when they have been enabled when Vim was compiled `+feature-list` . Example:

```
:if has("cindent")
```

2. Features that are only supported when certain conditions have been met. Example:

```
:if has("gui_running")
```

`gui_running`

3. Beyond a certain version or at a certain version and including a specific patch. The "patch-7.4.248" feature means that the `Vim version is` 7.5 or later, or it is version 7.4 and patch 248 was included. Example:

```
:if has("patch-7.4.248")
```

Note that it's possible for patch 248 to be omitted even though 249 is included. Only happens when cherry-picking patches.

Note that this form only works for patch 7.4.237 and later, before that you need to check for the patch and the `v:version`. Example (checking version 6.2.148 or later):

```
:if v:version > 602 || (v:version == 602 && has("patch148"))
```

`has-patch`

Hint: To find out if Vim supports backslashes in a file name (MS-Windows), use: ``if exists('+shellslash')``

<code>acl</code>	Compiled with <code>ACL</code> support.
<code>all_builtin_terms</code>	Compiled with all builtin terminals enabled. (always true)
<code>amiga</code>	Amiga version of Vim.
<code>arabic</code>	Compiled with Arabic support <code>Arabic</code> .
<code>arp</code>	Compiled with ARP support (Amiga).
<code>autocmd</code>	Compiled with autocommand support. (always true)
<code>autochdir</code>	Compiled with support for <code>'autochdir'</code>
<code>autoservername</code>	Automatically enable <code>clientserver</code>
<code>balloon_eval</code>	Compiled with <code>balloon-eval</code> support.
<code>balloon_multiline</code>	GUI supports multiline balloons.
<code>beos</code>	BeOS version of Vim.
<code>browse</code>	Compiled with <code>:browse</code> support, and <code>browse()</code> will work.
<code>browsefilter</code>	Compiled with support for <code>browsefilter</code> .
<code>bsd</code>	Compiled on an OS in the BSD family (excluding macOS).
<code>builtin_terms</code>	Compiled with some builtin terminals. (always true)
<code>byte_offset</code>	Compiled with support for 'o' in <code>'statusline'</code>
<code>channel</code>	Compiled with support for <code>channel</code> and <code>job</code>

cindent	Compiled with 'cindent' support. (always true)
clientserver	Compiled with remote invocation support <code>clientserver</code> .
clipboard	Compiled with 'clipboard' support.
clipboard_working	Compiled with 'clipboard' support and it can be used.
cmdline_compl	Compiled with <code>cmdline-completion</code> support.
cmdline_hist	Compiled with <code>cmdline-history</code> support.
cmdline_info	Compiled with 'showcmd' and 'ruler' support.
comments	Compiled with 'comments' support.
compatible	Compiled to be very Vi compatible.
conpty	Platform where <code>ConPTY</code> can be used.
cryptv	Compiled with encryption support <code>encryption</code> .
cscope	Compiled with <code>cscope</code> support.
cursorbind	Compiled with 'cursorbind' (always true)
debug	Compiled with "DEBUG" defined.
dialog_con	Compiled with console dialog support.
dialog_gui	Compiled with GUI dialog support.
diff	Compiled with <code>vimdiff</code> and 'diff' support.
digraphs	Compiled with support for digraphs.
directx	Compiled with support for DirectX and 'renderoptions' .
dnd	Compiled with support for the "~ register <code>quote_~</code> .
drop_file	Compiled with <code>drop_file</code> support.
ebcdic	Compiled on a machine with ebcdic character set.
emacs_tags	Compiled with support for Emacs tags.
eval	Compiled with expression evaluation support. Always true, of course!
ex_extra	<code>+ex_extra</code> (always true)
extra_search	Compiled with support for 'incsearch' and 'hlsearch'
farsi	Support for Farsi was removed <code>farsi</code> .
file_in_path	Compiled with support for <code>gf</code> and <code><cfile></code> (always true)
filterpipe	When 'shelltemp' is off pipes are used for shell read/write/filter commands
find_in_path	Compiled with support for include file searches <code>+find_in_path</code> .
float	Compiled with support for <code>Float</code> .
fname_case	Case in file names matters (for Amiga and MS-Windows this is not present).
folding	Compiled with <code>folding</code> support.
footer	Compiled with GUI footer support. <code>gui-footer</code>
fork	Compiled to use fork()/exec() instead of system().
gettext	Compiled with message translation <code>multi-lang</code>
gui	Compiled with GUI enabled.
gui_athena	Compiled with Athena GUI (always false).
gui_gnome	Compiled with Gnome support (<code>gui_gtk</code> is also defined).
gui_gtk	Compiled with GTK+ GUI (any version).
gui_gtk2	Compiled with GTK+ 2 GUI (<code>gui_gtk</code> is also defined).
gui_gtk3	Compiled with GTK+ 3 GUI (<code>gui_gtk</code> is also defined).
gui_haiku	Compiled with Haiku GUI.
gui_mac	Compiled with Macintosh GUI.
gui_motif	Compiled with Motif GUI.
gui_photon	Compiled with Photon GUI.
gui_running	Vim is running in the GUI, or it will start soon.
gui_win32	Compiled with MS-Windows Win32 GUI.

gui_win32s	idem, and Win32s system being used (Windows 3.1)
haiku	Haiku version of Vim.
hanguk_input	Compiled with Hanguk input support. hanguk
hpux	HP-UX version of Vim.
iconv	Can use iconv() for conversion.
insert_expand	Compiled with support for CTRL-X expansion commands in Insert mode. (always true)
job	Compiled with support for channel and job
ipv6	Compiled with support for IPV6 networking in channel .
jumplist	Compiled with jumplist support. (always true)
keymap	Compiled with ' keymap ' support.
lambda	Compiled with lambda support.
langmap	Compiled with ' langmap ' support.
libcall	Compiled with libcall() support.
linebreak	Compiled with ' linebreak ', ' breakat ', ' showbreak ' and ' breakindent ' support.
linux	Linux version of Vim.
lispindent	Compiled with support for lisp indenting. (always true)
listcmds	Compiled with commands for the buffer list :files and the argument list arglist .
localmap	Compiled with local mappings and abbr. :map-local
lua	Compiled with Lua interface Lua .
mac	Any Macintosh version of Vim cf. osx
macunix	Synonym for osxdarwin
menu	Compiled with support for :menu .
mksession	Compiled with support for :mksession .
modify_fname	Compiled with file name modifiers. filename-modifiers (always true)
mouse	Compiled with support for mouse.
mouse_dec	Compiled with support for Dec terminal mouse.
mouse_gpm	Compiled with support for gpm (Linux console mouse)
mouse_gpm_enabled	GPM mouse is working
mouse_netterm	Compiled with support for netterm mouse.
mouse_pterm	Compiled with support for qnx pterm mouse.
mouse_sysmouse	Compiled with support for sysmouse (*BSD console mouse)
mouse_sgr	Compiled with support for sgr mouse.
mouse_urxvt	Compiled with support for urxvt mouse.
mouse_xterm	Compiled with support for xterm mouse.
mousethape	Compiled with support for ' mousethape '.
multi_byte	Compiled with support for ' encoding ' (always true)
multi_byte_encoding	'encoding' is set to a multibyte encoding.
multi_byte_ime	Compiled with support for IME input method.
multi_lang	Compiled with support for multiple languages.
mzscheme	Compiled with MzScheme interface mzscheme .
nanotime	Compiled with sub-second time stamp checks.
netbeans_enabled	Compiled with support for netbeans and connected.
netbeans_intg	Compiled with support for netbeans .
num64	Compiled with 64-bit Number support. (always true)
ole	Compiled with OLE automation support for Win32.
osx	Compiled for macOS cf. mac
osxdarwin	Compiled for macOS, with mac-darwin-feature
packages	Compiled with packages support.
path_extra	Compiled with up/downwards search in ' path ' and ' tags '

perl	Compiled with Perl interface.
persistent_undo	Compiled with support for persistent undo history.
postscript	Compiled with PostScript file printing.
printer	Compiled with <code>:hardcopy</code> support.
profile	Compiled with <code>:profile</code> support.
prof_nsec	Profile results are in nanoseconds.
python	Python 2.x interface available. <code>has-python</code>
python_compiled	Compiled with Python 2.x interface. <code>has-python</code>
python_dynamic	Python 2.x interface is dynamically loaded. <code>has-python</code>
python3	Python 3.x interface available. <code>has-python</code>
python3_compiled	Compiled with Python 3.x interface. <code>has-python</code>
python3_dynamic	Python 3.x interface is dynamically loaded. <code>has-python</code>
python3_stable	Python 3.x interface is using Python Stable ABI. <code>has-python</code>
pythonx	Python 2.x and/or 3.x interface available. <code>python_x</code>
qnx	QNX version of Vim.
quickfix	Compiled with <code>quickfix</code> support.
reltime	Compiled with <code>reltime()</code> support.
rightleft	Compiled with <code>'rightleft'</code> support.
ruby	Compiled with Ruby interface <code>ruby</code> .
scrollbind	Compiled with <code>'scrollbind'</code> support. (always true)
showcmd	Compiled with <code>'showcmd'</code> support.
signs	Compiled with <code>:sign</code> support.
smartindent	Compiled with <code>'smartindent'</code> support. (always true)
sodium	Compiled with libsodium for better crypt support
sound	Compiled with sound support, e.g. <code>`sound_playevent()</code> `
spell	Compiled with spell checking support <code>spell</code> .
startuptime	Compiled with <code>--startuptime</code> support.
statusline	Compiled with support for <code>'statusline'</code> , <code>'rulerformat'</code> and special formats of <code>'titlestring'</code> and <code>'iconstring'</code> .
sun	SunOS version of Vim.
sun_workshop	Support for Sun <code>workshop</code> has been removed.
syntax	Compiled with syntax highlighting support <code>syntax</code> .
syntax_items	There are active syntax highlighting items for the current buffer.
system	Compiled to use <code>system()</code> instead of <code>fork()/exec()</code> .
tag_binary	Compiled with binary searching in tags files <code>tag-binary-search</code> . (always true)
tag_old_static	Support for old static tags was removed, see <code>tag-old-static</code> .
tcl	Compiled with Tcl interface.
termguicolors	Compiled with true color in terminal support.
terminal	Compiled with <code>terminal</code> support.
terminfo	Compiled with terminfo instead of termcap.
termresponse	Compiled with support for <code>t_RV</code> and <code>v:termresponse</code> .
textobjects	Compiled with support for <code>text-objects</code> .
textprop	Compiled with support for <code>text-properties</code> .
tgetent	Compiled with tgetent support, able to use a termcap or terminfo file.
timers	Compiled with <code>timer_start()</code> support.
title	Compiled with window title support <code>'title'</code> . (always true)
toolbar	Compiled with support for <code>gui-toolbar</code> .
ttyin	input is a terminal (tty)
ttyout	output is a terminal (tty)

unix	Unix version of Vim. +unix
unnamedplus	Compiled with support for "unnamedplus" in 'clipboard'
user_commands	User-defined commands. (always true)
vartabs	Compiled with variable tabstop support 'vartabstop' .
vcon	Win32: Virtual console support is working, can use 'termguicolors' . Also see +vtp .
vertspllit	Compiled with vertically split windows :vsplit . (always true)
vim_starting	True while initial source'ing takes place. startup vim_starting
vim9script	Compiled with Vim9 script support
viminfo	Compiled with viminfo support.
vimscrip-1	Compiled Vim script version 1 support
vimscrip-2	Compiled Vim script version 2 support
vimscrip-3	Compiled Vim script version 3 support
vimscrip-4	Compiled Vim script version 4 support
virtualedit	Compiled with 'virtualedit' option. (always true)
visual	Compiled with Visual mode. (always true)
visualextra	Compiled with extra Visual mode commands. (always true) blockwise-operators .
vms	VMS version of Vim.
vreplace	Compiled with gR and gr commands. (always true)
vtp	Compiled for vcon support +vtp (check vcon to find out if it works in the current console).
wildignore	Compiled with 'wildignore' option.
wildmenu	Compiled with 'wildmenu' option.
win16	old version for MS-Windows 3.1 (always false)
win32	Win32 version of Vim (MS-Windows 95 and later, 32 or 64 bits)
win32unix	Win32 version of Vim, using Unix files (Cygwin)
win64	Win64 version of Vim (MS-Windows 64 bit).
win95	Win32 version for MS-Windows 95/98/ME (always false)
winaltkeys	Compiled with 'winaltkeys' option.
windows	Compiled with support for more than one window. (always true)
writebackup	Compiled with 'writebackup' default on.
xattr	Compiled with extended attributes support xattr (currently only supported on Linux).
xfontset	Compiled with X fontset support xfontset .
xim	Compiled with X input method support xim .
xpm	Compiled with pixmap support.
xpm_w32	Compiled with pixmap support for Win32. (Only for backward compatibility. Use "xpm" instead.)
xsmg	Compiled with X session management support.
xsmg_interact	Compiled with interactive X session management support.
xterm_clipboard	Compiled with support for xterm clipboard.
xterm_save	Compiled with support for saving and restoring the xterm screen.
x11	Compiled with X11 support.

=====
4. Matching a pattern in a String **string-match**

This is common between several functions. A regexp pattern as explained at [pattern](#) is normally used to find a match in the buffer lines. When a pattern is used to find a match in a String, almost everything works in the same way. The difference is that a String is handled like it is one line. When it contains a "\n" character, this is not seen as a line break for the pattern. It can be matched with a "\n" in the pattern, or with ".". Example:

```
:let a = "aaaa\nxxxx"
:echo matchstr(a, "..\n..")
aa
xx
:echo matchstr(a, "a.x")
a
x
```

Don't forget that "^" will only match at the first character of the String and "\$" at the last character of the string. They don't match after or before a "\n".

Inter-process communication

channel

Vim uses channels to communicate with other processes.

A channel uses a socket or pipes.

socket-interface

Jobs can be used to start processes and communicate with them.

The Netbeans interface also uses a channel. netbeans

- | | |
|--------------------------------------|---------------------------|
| 1. Overview | job-channel-overview |
| 2. Channel demo | channel-demo |
| 3. Opening a channel | channel-open |
| 4. Using a JSON or JS channel | channel-use |
| 5. Channel commands | channel-commands |
| 6. Using a RAW or NL channel | channel-raw |
| 7. More channel functions | channel-more |
| 8. Channel functions details | channel-functions-details |
| 9. Starting a job with a channel | job-start |
| 10. Starting a job without a channel | job-start-nochannel |
| 11. Job functions | job-functions-details |
| 12. Job options | job-options |
| 13. Controlling a job | job-control |
| 14. Using a prompt buffer | prompt-buffer |
| 15. Language Server Protocol | language-server-protocol |

E1277

{only when compiled with the |+channel| feature for channel stuff}

You can check this with: ``has('channel')``

{only when compiled with the |+job| feature for job stuff}

You can check this with: ``has('job')``

=====

job-channel-overview

There are four main types of jobs:

1. A daemon, serving several Vim instances.
Vim connects to it with a socket.
2. One job working with one Vim instance, asynchronously.
Uses a socket or pipes.
3. A job performing some work for a short time, asynchronously.
Uses a socket or pipes.
4. Running a filter, synchronously.
Uses pipes.

For when using sockets See [job-start](#) , [job-start-nochannel](#) and [channel-open](#) . For 2 and 3, one or more jobs using pipes, see [job-start](#) . For 4 use the `":{range}!cmd"` command, see [filter](#) .

Over the socket and pipes these protocols are available:

RAW nothing known, Vim cannot tell where a message ends

NL every message ends in a NL (newline) character
JSON JSON encoding `json_encode()`
JS JavaScript style JSON-like encoding `js_encode()`
LSP Language Server Protocol encoding `language-server-protocol`

Common combination are:

- Using a job connected through pipes in NL mode. E.g., to run a style checker and receive errors and warnings.
- Using a daemon, connecting over a socket in JSON mode. E.g. to lookup cross-references in a database.

2. Channel demo

`channel-demo` `demoserver.py`

This requires Python. The demo program can be found in
\$VIMRUNTIME/tools/demoserver.py
Run it in one terminal. We will call this T1.

Run Vim in another terminal. Connect to the demo server with:
`let channel = ch_open('localhost:8765')`

In T1 you should see:

```
=== socket opened ===
```

You can now send a message to the server:

```
echo ch_evalexpr(channel, 'hello!')
```

The message is received in T1 and a response is sent back to Vim.
You can see the raw messages in T1. What Vim sends is:

```
[1,"hello!"]
```

And the response is:

```
[1,"got it"]
```

The number will increase every time you send a message.

The server can send a command to Vim. Type this on T1 (literally, including the quotes):

```
["ex","echo 'hi there'"]
```

And you should see the message in Vim. You can move the cursor a word forward:

```
["normal","w"]
```

To handle asynchronous communication a callback needs to be used:

```
func MyHandler(channel, msg)  
    echo "from the handler: " .. a:msg  
endfunc  
call ch_sendexpr(channel, 'hello!', {'callback': "MyHandler"})
```

Vim will not wait for a response. Now the server can send the response later and MyHandler will be invoked.

Instead of giving a callback with every send call, it can also be specified when opening the channel:

```
call ch_close(channel)  
let channel = ch_open('localhost:8765', {'callback': "MyHandler"})  
call ch_sendexpr(channel, 'hello channel!')
```

When trying out channels it's useful to see what is going on. You can tell Vim to write lines in log file:

```
    call ch_logfile('channellog', 'w')
See  ch_logfile() .
```

3. Opening a channel

channel-open

To open a channel:

```
let channel = ch_open({address} [, {options}])
if ch_status(channel) == "open"
    " use the channel
```

Use `ch_status()` to see if the channel could be opened.

channel-address

`{address}` can be a domain name or an IP address, followed by a port number, or a Unix-domain socket path prefixed by "unix:". E.g.

```
www.example.com:80    " domain + port
127.0.0.1:1234       " IPv4 + port
[2001:db8::1]:8765   " IPv6 + port
unix:/tmp/my-socket  " Unix-domain socket path
```

`{options}` is a dictionary with optional entries:

channel-open-options

"mode" can be:

channel-mode

```
"json" - Use JSON, see below; most convenient way. Default.
"js"   - Use JS (JavaScript) encoding, more efficient than JSON.
"nl"   - Use messages that end in a NL character
"raw"  - Use raw messages
"lsp"  - Use language server protocol encoding
```

channel-callback E921

"callback" A function that is called when a message is received that is not handled otherwise (e.g. a JSON message with ID zero). It gets two arguments: the channel and the received message. Example:

```
func Handle(channel, msg)
    echo 'Received: ' .. a:msg
endfunc
let channel = ch_open("localhost:8765", {"callback": "Handle"})
```

When "mode" is "json" or "js" or "lsp" the "msg" argument is the body of the received message, converted to Vim types.

When "mode" is "nl" the "msg" argument is one message, excluding the NL.

When "mode" is "raw" the "msg" argument is the whole message as a string.

For all callbacks: Use `function()` to bind it to arguments and/or a Dictionary. Or use the form "dict.function" to bind the Dictionary.

Callbacks are only called at a "safe" moment, usually when Vim is waiting for the user to type a character. Vim does not use

multi-threading.

- close_cb**
- "close_cb" A function that is called when the channel gets closed, other than by calling `ch_close()`. It should be defined like this:
- ```
func MyCloseHandler(channel)
 Vim will invoke callbacks that handle data before invoking
 close_cb, thus when this function is called no more data will
 be passed to the callbacks. However, if a callback causes Vim
 to check for messages, the close_cb may be invoked while still
 in the callback. The plugin must handle this somehow, it can
 be useful to know that no more data is coming.
 If it is not known if there is a message to be read, use a
 try/catch block:
 try
 let msg = ch_readraw(a:channel)
 catch
 let msg = 'no message'
 endtry
 try
 let err = ch_readraw(a:channel, #{part: 'err'})
 catch
 let err = 'no error'
 endtry
```
- channel-drop**
- "drop" Specifies when to drop messages:
- "auto" When there is no callback to handle a message. The "close\_cb" is also considered for this.
  - "never" All messages will be kept.
- channel-noblock**
- "noblock" Same effect as `job-noblock`. Only matters for writing.
- waittime**
- "waittime" The time to wait for the connection to be made in milliseconds. A negative number waits forever.
- The default is zero, don't wait, which is useful if a local server is supposed to be running already. On Unix Vim actually uses a 1 msec timeout, that is required on many systems. Use a larger value for a remote server, e.g. 10 msec at least.
- channel-timeout**
- "timeout" The time to wait for a request when blocking, E.g. when using `ch_evalexpr()`. In milliseconds. The default is 2000 (2 seconds).

When "mode" is "json" or "js" the "callback" is optional. When omitted it is only possible to receive a message after sending one.

To change the channel options after opening it use `ch_setoptions()`. The arguments are similar to what is passed to `ch_open()`, but "waittime" cannot be given, since that only applies to opening the channel.

For example, the handler can be added or changed:

```
call ch_setoptions(channel, {'callback': callback})
```

When "callback" is empty (zero or an empty string) the handler is removed.

After a callback has been invoked Vim will update the screen and put the cursor back where it belongs. Thus the callback should not need to do `:redraw``.

The timeout can be changed:

```
call ch_setoptions(channel, {'timeout': msec})
```

channel-close E906

Once done with the channel, disconnect it like this:

```
call ch_close(channel)
```

When a socket is used this will close the socket for both directions. When pipes are used (stdin/stdout/stderr) they are all closed. This might not be what you want! Stopping the job with `job_stop()` might be better. All readahead is discarded, callbacks will no longer be invoked.

**Note** that a channel is closed in three stages:

- The I/O ends, log message: "Closing channel". There can still be queued messages to read or callbacks to invoke.
- The readahead is cleared, log message: "Clearing channel". Some variables may still reference the channel.
- The channel is freed, log message: "Freeing channel".

When the channel can't be opened you will get an error message. There is a difference between MS-Windows and Unix: On Unix when the port doesn't exist `ch_open()` fails quickly. On MS-Windows "waittime" applies.

E898 E901 E902

If there is an error reading or writing a channel it will be closed.

E630 E631

---

#### 4. Using a JSON or JS channel

channel-use

If mode is JSON then a message can be sent synchronously like this:

```
let response = ch_evalexpr(channel, {expr})
```

This awaits a response from the other side.

When mode is JS this works the same, except that the messages use JavaScript encoding. See `js_encode()` for the difference.

To send a message, without handling a response or letting the channel callback handle the response:

```
call ch_sendexpr(channel, {expr})
```

To send a message and letting the response handled by a specific function, asynchronously:

```
call ch_sendexpr(channel, {expr}, {'callback': Handler})
```

Vim will match the response with the request using the message ID. Once the response is received the callback will be invoked. Further responses with the

same ID will be ignored. If your server sends back multiple responses you need to send them with ID zero, they will be passed to the channel callback.

The `{expr}` is converted to JSON and wrapped in an array. An example of the message that the receiver will get when `{expr}` is the string "hello":

```
[12,"hello"]
```

The format of the JSON sent is:

```
[{number},{expr}]
```

In which `{number}` is different every time. It must be used in the response (if any):

```
[{number},{response}]
```

This way Vim knows which sent message matches with which received message and can call the right handler. Also when the messages arrive out of order.

A newline character is terminating the JSON text. This can be used to separate the read text. For example, in Python:

```
splitidx = read_text.find('\n')
message = read_text[:splitidx]
rest = read_text[splitidx + 1:]
```

The sender must always send valid JSON to Vim. Vim can check for the end of the message by parsing the JSON. It will only accept the message if the end was received. A newline after the message is optional.

When the process wants to send a message to Vim without first receiving a message, it must use the number zero:

```
[0,{response}]
```

Then channel handler will then get `{response}` converted to Vim types. If the channel does not have a handler the message is dropped.

It is also possible to use `ch_sendraw()` and `ch_evalraw()` on a JSON or JS channel. The caller is then completely responsible for correct encoding and decoding.

---

## 5. Channel commands

channel-commands

With a JSON channel the process can send commands to Vim that will be handled by Vim internally, it does not require a handler for the channel.

Possible commands are:

E903 E904 E905

```
["redraw", {forced}]
["ex", {Ex command}]
["normal", {Normal mode command}]
["expr", {expression}, {number}]
["expr", {expression}]
["call", {func name}, {argument list}, {number}]
["call", {func name}, {argument list}]
```

With all of these: Be careful what these commands do! You can easily interfere with what the user is doing. To avoid trouble use `mode()` to check that the editor is in the expected state. E.g., to send keys that must be inserted as text, not executed as a command:

```
["ex","if mode() == 'i' | call feedkeys('ClassName') | endif"]
```

Errors in these commands are normally not reported to avoid them messing up the display. If you do want to see them, set the `'verbose'` option to 3 or higher.

### Command "redraw"

The other commands do not explicitly update the screen, so that you can send a sequence of commands without the cursor moving around. A redraw can happen as a side effect of some commands. You must end with the "redraw" command to show any changed text and show the cursor where it belongs.

The argument is normally an empty string:

```
["redraw", ""]
```

To first clear the screen pass "force":

```
["redraw", "force"]
```

### Command "ex"

The "ex" command is executed as any Ex command. There is no response for completion or error. You could use functions in an `autoload` script:

```
["ex","call myscript#MyFunc(arg)"]
```

You can also use `"call feedkeys() "` to insert any key sequence.

When there is an error a message is written to the channel log, if it exists, and `v:errmsg` is set to the error.

### Command "normal"

The "normal" command is executed like with `":normal!"`, commands are not mapped. Example to open the folds under the cursor:

```
["normal" "z0"]
```

### Command "expr" with response

The "expr" command can be used to get the result of an expression. For example, to get the number of lines in the current buffer:

```
["expr","line('$')", -2]
```

It will send back the result of the expression:

```
[-2, "last line"]
```

The format is:

```
[{number}, {result}]
```

Here `{number}` is the same as what was in the request. Use a negative number to avoid confusion with message that Vim sends. Use a different number on every request to be able to match the request with the response.

`{result}` is the result of the evaluation and is JSON encoded. If the evaluation fails or the result can't be encoded in JSON it is the string "ERROR".

### Command "expr" without a response

This command is similar to "expr" above, but does not send back any response. Example:

```
["expr","setline('$', ['one', 'two', 'three'])"]
```

There is no third argument in the request.

### Command "call"

This is similar to "expr", but instead of passing the whole expression as a string this passes the name of a function and a list of arguments. This avoids the conversion of the arguments to a string and escaping and concatenating them. Example:

```
["call", "line", ["$"], -2]
```

Leave out the fourth argument if no response is to be sent:

```
["call", "setline", ["$"], ["one", "two", "three"]]
```

---

## 6. Using a RAW or NL channel

`channel-raw`

If mode is RAW or NL then a message can be sent like this:

```
let response = ch_evalraw(channel, {string})
```

The `{string}` is sent as-is. The response will be what can be read from the channel right away. Since Vim doesn't know how to recognize the end of the message you need to take care of it yourself. The timeout applies for reading the first byte, after that it will not wait for anything more.

If mode is "nl" you can send a message in a similar way. You are expected to put in the NL after each message. Thus you can also send several messages ending in a NL at once. The response will be the text up to and including the first NL. This can also be just the NL for an empty response.

If no NL was read before the channel timeout an empty string is returned.

To send a message, without expecting a response:

```
call ch_sendraw(channel, {string})
```

The process can send back a response, the channel handler will be called with it.

`channel-onetime-callback`

To send a message and letting the response handled by a specific function, asynchronously:

```
call ch_sendraw(channel, {string}, {'callback': 'MyHandler'})
```



This `{string}` can also be JSON, use `json_encode()` to create it and `json_decode()` to handle a received JSON message.

It is not possible to use `ch_evaluate()` or `ch_sendexpr()` on a raw channel.

A String in Vim cannot contain NUL bytes. To send or receive NUL bytes read or write from a buffer. See `in_io-buffer` and `out_io-buffer`.

---

## 7. More channel functions

[channel-more](#)

To obtain the status of a channel: `ch_status(channel)`. The possible results are:

|            |                                                   |
|------------|---------------------------------------------------|
| "fail"     | Failed to open the channel.                       |
| "open"     | The channel can be used.                          |
| "buffered" | The channel was closed but there is data to read. |
| "closed"   | The channel was closed.                           |

To obtain the job associated with a channel: `ch_getjob(channel)`

To read one message from a channel:

```
let output = ch_read(channel)
```

This uses the channel timeout. To read without a timeout, just get any message that is available:

```
let output = ch_read(channel, {'timeout': 0})
```

When no message was available then the result is `v:none` for a JSON or JS mode channels, an empty string for a RAW or NL channel. You can use `ch_canread()` to check if there is something to read.

**Note** that when there is no callback, messages are dropped. To avoid that add a close callback to the channel.

To read all normal output from a RAW channel that is available:

```
let output = ch_readraw(channel)
```

To read all error output from a RAW channel that is available:

```
let output = ch_readraw(channel, {"part": "err"})
```

**Note** that if the channel is in NL mode, `ch_readraw()` will only return one line for each call.

`ch_read()` and `ch_readraw()` use the channel timeout. When there is nothing to read within that time an empty string is returned. To specify a different timeout in msec use the "timeout" option:

```
{"timeout": 123}
```

To read from the error output use the "part" option:

```
{"part": "err"}
```

To read a message with a specific ID, on a JS or JSON channel:

```
{"id": 99}
```

When no ID is specified or the ID is `-1`, the first message is returned. This overrides any callback waiting for this message.

For a RAW channel this returns whatever is available, since Vim does not know where a message ends.

For a NL channel this returns one message.

For a JS or JSON channel this returns one decoded message.  
This includes any sequence number.

---

## 8. Channel functions details

## channel-functions-details

ch\_canread({handle})

ch\_canread()

Return non-zero when there is something to read from {handle}.  
{handle} can be a Channel or a Job that has a Channel.

This is useful to read from a channel at a convenient time,  
e.g. from a timer.

**Note** that messages are dropped when the channel does not have  
a callback. Add a close callback to avoid that.

Can also be used as a **method** :  
GetChannel()->ch\_canread()

ch\_close({handle})

ch\_close()

Close {handle}. See [channel-close](#).  
{handle} can be a Channel or a Job that has a Channel.  
A close callback is not invoked.

Can also be used as a **method** :  
GetChannel()->ch\_close()

ch\_close\_in({handle})

ch\_close\_in()

Close the "in" part of {handle}. See [channel-close-in](#).  
{handle} can be a Channel or a Job that has a Channel.  
A close callback is not invoked.

Can also be used as a **method** :  
GetChannel()->ch\_close\_in()

ch\_evaluate({handle}, {expr} [, {options}])

ch\_evaluate()

Send {expr} over {handle}. The {expr} is encoded  
according to the type of channel. The function cannot be used  
with a raw channel. See [channel-use](#).

{handle} can be a Channel or a Job that has a Channel.  
When using the "lsp" channel mode, {expr} must be a **Dict**.

E917

{options} must be a Dictionary. It must not have a "callback"  
entry. It can have a "timeout" entry to specify the timeout  
for this specific request.

ch\_evaluate() waits for a response and returns the decoded  
expression. When there is an error or timeout it returns an  
empty **String** or, when using the "lsp" channel mode, returns an  
empty **Dict**.

**Note** that while waiting for the response, Vim handles other  
messages. You need to make sure this doesn't cause trouble.

Can also be used as a `method` :  
`GetChannel()->ch_evalexpr(expr)`

`ch_evalraw({handle}, {string} [, {options}])` `ch_evalraw()`  
Send `{string}` over `{handle}`.  
`{handle}` can be a Channel or a Job that has a Channel.

Works like `ch_evalexpr()` , but does not encode the request or decode the response. The caller is responsible for the correct contents. Also does not add a newline for a channel in NL mode, the caller must do that. The NL in the response is removed.

**Note** that Vim does not know when the text received on a raw channel is complete, it may only return the first part and you need to use `ch_readraw()` to fetch the rest.  
See `channel-use` .

Can also be used as a `method` :  
`GetChannel()->ch_evalraw(rawstring)`

`ch_getbufnr({handle}, {what})` `ch_getbufnr()`  
Get the buffer number that `{handle}` is using for String `{what}`.  
`{handle}` can be a Channel or a Job that has a Channel.  
`{what}` can be "err" for stderr, "out" for stdout or empty for socket output.  
Returns -1 when there is no buffer.

Can also be used as a `method` :  
`GetChannel()->ch_getbufnr(what)`

`ch_getjob({channel})` `ch_getjob()`  
Get the Job associated with `{channel}`.  
If there is no job calling `job_status()` on the returned Job will result in "fail".

Can also be used as a `method` :  
`GetChannel()->ch_getjob()`

`ch_info({handle})` `ch_info()`  
Returns a Dictionary with information about `{handle}`. The items are:

|          |                                                               |
|----------|---------------------------------------------------------------|
| "id"     | number of the channel                                         |
| "status" | "open", "buffered" or "closed", like <code>ch_status()</code> |

When opened with `ch_open()`:

|               |                                    |
|---------------|------------------------------------|
| "hostname"    | the hostname of the address        |
| "port"        | the port of the address            |
| "path"        | the path of the Unix-domain socket |
| "sock_status" | "open" or "closed"                 |
| "sock_mode"   | "NL", "RAW", "JSON" or "JS"        |
| "sock_io"     | "socket"                           |

"sock\_timeout" timeout in msec

**Note** that "path" is only present for Unix-domain sockets, for regular ones "hostname" and "port" are present instead.

When opened with `job_start()`:

```
"out_status" "open", "buffered" or "closed"
"out_mode" "NL", "RAW", "JSON" or "JS"
"out_io" "null", "pipe", "file" or "buffer"
"out_timeout" timeout in msec
"err_status" "open", "buffered" or "closed"
"err_mode" "NL", "RAW", "JSON" or "JS"
"err_io" "out", "null", "pipe", "file" or "buffer"
"err_timeout" timeout in msec
"in_status" "open" or "closed"
"in_mode" "NL", "RAW", "JSON", "JS" or "LSP"
"in_io" "null", "pipe", "file" or "buffer"
"in_timeout" timeout in msec
```

Can also be used as a **method** :

```
GetChannel()->ch_info()
```

`ch_log({msg} [, {handle}])` **ch\_log()**

Write String `{msg}` in the channel log file, if it was opened with `ch_logfile()` .

The text "ch\_log():" is prepended to the message to make clear it came from this function call and make it easier to find in the log file.

When `{handle}` is passed the channel number is used for the message.

`{handle}` can be a Channel or a Job that has a Channel. The Channel must be open for the channel number to be used.

Can also be used as a **method** :

```
'did something'->ch_log()
```

`ch_logfile({fname} [, {mode}])` **ch\_logfile()**

Start logging channel activity to `{fname}`.

When `{fname}` is an empty string: stop logging.

When `{mode}` is omitted or contains "a" or is "o" then append to the file.

When `{mode}` contains "w" and not "a" start with an empty file.

When `{mode}` contains "o" then log all terminal output.

Otherwise only some interesting terminal output is logged.

Use `ch_log()` to write log messages. The file is flushed after every message, on Unix you can use "tail -f" to see what is going on in real time.

To enable the log very early, to see what is received from a terminal during startup, use `--log` (this uses mode "ao"):

`vim --log logfile`

This function is not available in the [sandbox](#) .

**NOTE:** the channel communication is stored in the file, be aware that this may contain confidential and privacy sensitive information, e.g. a password you type in a terminal window.

Can also be used as a [method](#) :

`'logfile'->ch_logfile('w')`

`ch_open({address} [, {options}])` [ch\\_open\(\)](#)

Open a channel to `{address}`. See [channel](#) .

Returns a Channel. Use `ch_status()` to check for failure.

`{address}` is a String, see [channel-address](#) for the possible accepted forms.

If `{options}` is given it must be a [Dictionary](#) .

See [channel-open-options](#) .

Can also be used as a [method](#) :

`GetAddress()->ch_open()`

`ch_read({handle} [, {options}])` [ch\\_read\(\)](#)

Read from `{handle}` and return the received message.

`{handle}` can be a Channel or a Job that has a Channel.

For a NL channel this waits for a NL to arrive, except when there is nothing more to read (channel was closed).

See [channel-more](#) .

Can also be used as a [method](#) :

`GetChannel()->ch_read()`

`ch_readblob({handle} [, {options}])` [ch\\_readblob\(\)](#)

Like `ch_read()` but reads binary data and returns a [Blob](#) .

See [channel-more](#) .

Can also be used as a [method](#) :

`GetChannel()->ch_readblob()`

`ch_readraw({handle} [, {options}])` [ch\\_readraw\(\)](#)

Like `ch_read()` but for a JS and JSON channel does not decode the message. For a NL channel it does not block waiting for the NL to arrive, but otherwise works like `ch_read()`.

See [channel-more](#) .

Can also be used as a [method](#) :

`GetChannel()->ch_readraw()`

`ch_sendexpr({handle}, {expr} [, {options}])` `ch_sendexpr()`  
Send `{expr}` over `{handle}`. The `{expr}` is encoded according to the type of channel. The function cannot be used with a raw channel.

See `channel-use` . E912  
`{handle}` can be a Channel or a Job that has a Channel.  
When using the "lsp" channel mode, `{expr}` must be a `Dict` .

If the channel mode is "lsp", then returns a Dict. Otherwise returns an empty String. If the "callback" item is present in `{options}`, then the returned Dict contains the ID of the request message. The ID can be used to send a cancellation request to the LSP server (if needed). Returns an empty Dict on error.

If a response message is not expected for `{expr}`, then don't specify the "callback" item in `{options}`.

Can also be used as a `method` :  
`GetChannel()->ch_sendexpr(expr)`

`ch_sendraw({handle}, {expr} [, {options}])` `ch_sendraw()`  
Send `String` or `Blob` `{expr}` over `{handle}`.  
Works like `ch_sendexpr()` , but does not encode the request or decode the response. The caller is responsible for the correct contents. Also does not add a newline for a channel in NL mode, the caller must do that. The NL in the response is removed.

See `channel-use` .  
Can also be used as a `method` :  
`GetChannel()->ch_sendraw(rawexpr)`

`ch_setopts({handle}, {options})` `ch_setopts()`  
Set options on `{handle}`:  
    "callback"      the channel callback  
    "timeout"      default read timeout in msec  
    "mode"          mode for the whole channel

See `ch_open()` for more explanation.  
`{handle}` can be a Channel or a Job that has a Channel.

**Note** that changing the mode may cause queued messages to be lost.

These options cannot be changed:  
    "waittime"      only applies to `ch_open()`

Can also be used as a `method` :  
`GetChannel()->ch_setopts(options)`

`ch_status({handle} [, {options}])` `ch_status()`

Return the status of `{handle}`:

|            |                                     |
|------------|-------------------------------------|
| "fail"     | failed to open the channel          |
| "open"     | channel can be used                 |
| "buffered" | channel can be read, not written to |
| "closed"   | channel can not be used             |

`{handle}` can be a Channel or a Job that has a Channel.  
"buffered" is used when the channel was closed but there is still data that can be obtained with `ch_read()` .

If `{options}` is given it can contain a "part" entry to specify the part of the channel to return the status for: "out" or "err". For example, to get the error status:

```
ch_status(job, {"part": "err"})
```

Can also be used as a `method` :

```
GetChannel()->ch_status()
```

---

## 9. Starting a job with a channel job-start job

To start a job and open a channel for stdin/stdout/stderr:

```
let job = job_start(command, {options})
```

You can get the channel with:

```
let channel = job_getchannel(job)
```

The channel will use NL mode. If you want another mode it's best to specify this in `{options}`. When changing the mode later some text may have already been received and not parsed correctly.

If the command produces a line of output that you want to deal with, specify a handler for stdout:

```
let job = job_start(command, {"out_cb": "MyHandler"})
```

The function will be called with the channel and a message. You would define it like this:

```
func MyHandler(channel, msg)
```

Without the handler you need to read the output with `ch_read()` or `ch_readraw()` . You can do this in the close callback, see [read-in-close-cb](#) .

**Note** that if the job exits before you read the output, the output may be lost. This depends on the system (on Unix this happens because closing the write end of a pipe causes the read end to get EOF). To avoid this make the job sleep for a short while before it exits.

The handler defined for "out\_cb" will not receive stderr. If you want to handle that separately, add an "err\_cb" handler:

```
let job = job_start(command, {"out_cb": "MyHandler",
 \ "err_cb": "ErrHandler"})
```

If you want to handle both stderr and stdout with one handler use the "callback" option:

```
let job = job_start(command, {"callback": "MyHandler"})
```

Depending on the system, starting a job can put Vim in the background, the started job gets the focus. To avoid that, use the ``foreground()`` function. This might not always work when called early, put in the callback handler or use a timer to call it after the job has started.

You can send a message to the command with `ch_evalraw()`. If the channel is in JSON or JS mode you can use `ch_evalexpr()`.

There are several options you can use, see [job-options](#) .

For example, to start a job and write its output in buffer "dummy":

```
let logjob = job_start("tail -f /tmp/log",
 \ {'out_io': 'buffer', 'out_name': 'dummy'})
sbuf dummy
```

### Job input from a buffer

[in\\_io-buffer](#)

To run a job that reads from a buffer:

```
let job = job_start({command},
 \ {'in_io': 'buffer', 'in_name': 'mybuffer'})
```

[E915](#) [E918](#)

The buffer is found by name, similar to `bufnr()` . The buffer must exist and be loaded when `job_start()` is called.

By default this reads the whole buffer. This can be changed with the "in\_top" and "in\_bot" options.

A special mode is when "in\_top" is set to zero and "in\_bot" is not set: Every time a line is added to the buffer, the last-but-one line will be sent to the job stdin. This allows for editing the last line and sending it when pressing Enter.

[channel-close-in](#)

When not using the special mode the pipe or socket will be closed after the last line has been written. This signals the reading end that the input finished. You can also use `ch_close_in()` to close it sooner.

NUL bytes in the text will be passed to the job (internally Vim stores these as NL bytes).

### Reading job output in the close callback

[read-in-close-cb](#)

If the job can take some time and you don't need intermediate results, you can add a close callback and read the output there:

```
func! CloseHandler(channel)
 while ch_status(a:channel, {'part': 'out'}) == 'buffered'
 echomsg ch_read(a:channel)
 endwhile
endfunc
let job = job_start(command, {'close_cb': 'CloseHandler'})
```

You will want to do something more useful than "echomsg".



---

## 10. Starting a job without a channel

[job-start-nochannel](#)

To start another process without creating a channel:

```
let job = job_start(command,
 \ {"in_io": "null", "out_io": "null", "err_io": "null"})
```

This starts `{command}` in the background, Vim does not wait for it to finish.

When Vim sees that neither stdin, stdout or stderr are connected, no channel will be created. Often you will want to include redirection in the command to avoid it getting stuck.

There are several options you can use, see [job-options](#) .

[job-start-if-needed](#)

To start a job only when connecting to an address does not work, do something like this:

```
let channel = ch_open(address, {"waittime": 0})
if ch_status(channel) == "fail"
 let job = job_start(command)
 let channel = ch_open(address, {"waittime": 1000})
endif
```

**Note** that the `waittime` for `ch_open()` gives the job one second to make the port available.

---

## 11. Job functions

[job-functions-details](#)

`job_getchannel({job})`

[job\\_getchannel\(\)](#)

Get the channel handle that `{job}` is using.

To check if the job has no channel:

```
if string(job_getchannel(job)) == 'channel fail'
```

Can also be used as a [method](#) :

```
GetJob()->job_getchannel()
```

`job_info([{job}])`

[job\\_info\(\)](#)

Returns a Dictionary with information about `{job}`:

|              |                                                 |
|--------------|-------------------------------------------------|
| "status"     | what <a href="#">job_status()</a> returns       |
| "channel"    | what <a href="#">job_getchannel()</a> returns   |
| "cmd"        | List of command arguments used to start the job |
| "process"    | process ID                                      |
| "tty_in"     | terminal input name, empty when none            |
| "tty_out"    | terminal output name, empty when none           |
| "exitval"    | only valid when "status" is "dead"              |
| "exit_cb"    | function to be called on exit                   |
| "stoponexit" | <a href="#">job-stoponexit</a>                  |

Only in Unix:

|           |                                                                                            |
|-----------|--------------------------------------------------------------------------------------------|
| "termsig" | the signal which terminated the process<br>(See <a href="#">job_stop()</a> for the values) |
|-----------|--------------------------------------------------------------------------------------------|

only valid when "status" is "dead"

Only in MS-Windows:

"tty\_type" Type of virtual console in use.  
Values are "winpty" or "conpty".  
See '[termwintype](#)'.

Without any arguments, returns a List with all Job objects.

Can also be used as a [method](#) :

[GetJob\(\)->job\\_info\(\)](#)

[job\\_setopts\({job}, {options}\)](#) [job\\_setopts\(\)](#)

Change options for {job}. Supported are:

"stoponexit" [job-stoponexit](#)  
"exit\_cb" [job-exit\\_cb](#)

Can also be used as a [method](#) :

[GetJob\(\)->job\\_setopts\(options\)](#)

[job\\_start\({command} \[, {options}\]\)](#) [job\\_start\(\)](#)

Start a job and return a Job object. Unlike [system\(\)](#) and [:!cmd](#) this does not wait for the job to finish.

To start a job in a terminal window see [term\\_start\(\)](#) .

If the job fails to start then [job\\_status\(\)](#) on the returned Job object results in "fail" and none of the callbacks will be invoked.

{command} can be a String. This works best on MS-Windows. On Unix it is split up in white-separated parts to be passed to [execvp\(\)](#). Arguments in double quotes can contain white space.

{command} can be a List, where the first item is the executable and further items are the arguments. All items are converted to String. This works best on Unix.

On MS-Windows, [job\\_start\(\)](#) makes a GUI application hidden. If you want to show it, use [:!start](#) instead.

The command is executed directly, not through a shell, the '[shell](#)' option is not used. To use the shell:

```
let job = job_start(["/bin/sh", "-c", "echo hello"])
```

Or:

```
let job = job_start('/bin/sh -c "echo hello"')
```

**Note** that this will start two processes, the shell and the command it executes. If you don't want this use the "exec" shell command.

On Unix \$PATH is used to search for the executable only when the command does not contain a slash.

The job will use the same terminal as Vim. If it reads from stdin the job and Vim will be fighting over input, that doesn't work. Redirect stdin and stdout to avoid problems:

```
let job = job_start(['sh', '-c', "myserver </dev/null >/dev/null"])
```

The returned Job object can be used to get the status with `job_status()` and stop the job with `job_stop()` .

**Note** that the job object will be deleted if there are no references to it. This closes the stdin and stderr, which may cause the job to fail with an error. To avoid this keep a reference to the job. Thus instead of:

```
call job_start('my-command')
```

use:

```
let myjob = job_start('my-command')
```

and unlet "myjob" once the job is not needed or is past the point where it would fail (e.g. when it prints a message on startup). Keep in mind that variables local to a function will cease to exist if the function returns. Use a script-local variable if needed:

```
let s:myjob = job_start('my-command')
```

`{options}` must be a Dictionary. It can contain many optional items, see `job-options` .

Can also be used as a `method` :

```
BuildCommand()->job_start()
```

`job_status({job})` `job_status()` E916

Returns a String with the status of `{job}`:

```
"run" job is running
"fail" job failed to start
"dead" job died or was stopped after running
```

On Unix a non-existing command results in "dead" instead of "fail", because a fork happens before the failure can be detected.

If in Vim9 script a variable is declared with type "job" but never assigned to, passing that variable to `job_status()` returns "fail".

If an exit callback was set with the "exit\_cb" option and the job is now detected to be "dead" the callback will be invoked.

For more information see `job_info()` .

Can also be used as a `method` :

```
GetJob()->job_status()
```

`job_stop({job} [, {how}])` `job_stop()`

Stop the `{job}`. This can also be used to signal the job.

When `{how}` is omitted or is "term" the job will be terminated. For Unix SIGTERM is sent. On MS-Windows the job will be terminated forcibly (there is no "gentle" way). This goes to the process group, thus children may also be affected.

Effect for Unix:

|        |                                 |
|--------|---------------------------------|
| "term" | SIGTERM (default)               |
| "hup"  | SIGHUP                          |
| "quit" | SIGQUIT                         |
| "int"  | SIGINT                          |
| "kill" | SIGKILL (strongest way to stop) |
| number | signal with that number         |

Effect for MS-Windows:

|        |                                      |
|--------|--------------------------------------|
| "term" | terminate process forcibly (default) |
| "hup"  | CTRL_BREAK                           |
| "quit" | CTRL_BREAK                           |
| "int"  | CTRL_C                               |
| "kill" | terminate process forcibly           |
| Others | CTRL_BREAK                           |

On Unix the signal is sent to the process group. This means that when the job is "sh -c command" it affects both the shell and the command.

The result is a Number: 1 if the operation could be executed, 0 if "how" is not supported on the system.

**Note** that even when the operation was executed, whether the job was actually stopped needs to be checked with `job_status()` .

If the status of the job is "dead", the signal will not be sent. This is to avoid to stop the wrong job (esp. on Unix, where process numbers are recycled).

When using "kill" Vim will assume the job will die and close the channel.

Can also be used as a `method` :  
`GetJob()->job_stop()`

---

## 12. Job options job-options

The `{options}` argument in `job_start()` is a dictionary. All entries are optional. Some options can be used after the job has started, using `job_setoptions(job, {options})`. Many options can be used with the channel related to the job, using `ch_setoptions(channel, {options})`. See `job_setoptions()` and `ch_setoptions()` .

`in_mode`   `out_mode`   `err_mode`

|                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>"in_mode"</p> <p>"out_mode"</p> <p>"err_mode"</p> | <p>mode specifically for stdin, only when using pipes</p> <p>mode specifically for stdout, only when using pipes</p> <p>mode specifically for stderr, only when using pipes</p> <p>See <a href="#">channel-mode</a> for the values.</p> <p><b>Note:</b> when setting "mode" the part specific mode is overwritten. Therefore set "mode" first and the part specific mode later.</p> <p><b>Note:</b> when writing to a file or buffer and when reading from a buffer NL mode is used by default.</p>                 |
| <p>"noblock": 1</p>                                  | <p style="text-align: center;"><a href="#">job-noblock</a></p> <p>When writing use a non-blocking write call. This avoids getting stuck if Vim should handle other messages in between, e.g. when a job sends back data to Vim. It implies that when <code>ch_sendraw()</code> returns not all data may have been written yet. This option was added in patch 8.1.0350, test with:</p> <pre> if has("patch-8.1.350")   let options['noblock'] = 1 endif </pre>                                                      |
| <p>"callback": handler</p>                           | <p style="text-align: center;"><a href="#">job-callback</a></p> <p>Callback for something to read on any part of the channel.</p>                                                                                                                                                                                                                                                                                                                                                                                   |
| <p>"out_cb": handler</p>                             | <p style="text-align: center;"><a href="#">job-out_cb</a>   <a href="#">out_cb</a></p> <p>Callback for when there is something to read on stdout. Only for when the channel uses pipes. When "out_cb" wasn't set the channel callback is used. The two arguments are the channel and the message.</p>                                                                                                                                                                                                               |
| <p>"err_cb": handler</p>                             | <p style="text-align: center;"><a href="#">job-err_cb</a>   <a href="#">err_cb</a></p> <p>Callback for when there is something to read on stderr. Only for when the channel uses pipes. When "err_cb" wasn't set the channel callback is used. The two arguments are the channel and the message.</p>                                                                                                                                                                                                               |
| <p>"close_cb": handler</p>                           | <p style="text-align: center;"><a href="#">job-close_cb</a></p> <p>Callback for when the channel is closed. Same as "close_cb" on <a href="#">ch_open()</a> , see <a href="#">close_cb</a> .</p>                                                                                                                                                                                                                                                                                                                    |
| <p>"drop": when</p>                                  | <p style="text-align: center;"><a href="#">job-drop</a></p> <p>Specifies when to drop messages. Same as "drop" on <a href="#">ch_open()</a> , see <a href="#">channel-drop</a> . For "auto" the <code>exit_cb</code> is not considered.</p>                                                                                                                                                                                                                                                                         |
| <p>"exit_cb": handler</p>                            | <p style="text-align: center;"><a href="#">job-exit_cb</a></p> <p>Callback for when the job ends. The arguments are the job and the exit status. Vim checks up to 10 times per second for jobs that ended. The check can also be triggered by calling <a href="#">job_status()</a> , which may then invoke the <code>exit_cb</code> handler.</p> <p><b>Note</b> that data can be buffered, callbacks may still be called after the process ends.</p> <p style="text-align: center;"><a href="#">job-timeout</a></p> |

"timeout": time The time to wait for a request when blocking, E.g. when using `ch_evalexpr()`. In milliseconds. The default is 2000 (2 seconds).

"out\_timeout": time Timeout for stdout. Only when using pipes. out\_timeout err\_timeout

"err\_timeout": time Timeout for stderr. Only when using pipes. out\_timeout err\_timeout  
**Note:** when setting "timeout" the part specific mode is overwritten. Therefore set "timeout" first and the part specific mode later.

"stoponexit": {signal} Send {signal} to the job when Vim exits. See job-stoponexit  
job\_stop() for possible values.

"stoponexit": "" Do not stop the job when Vim exits. The default is "term".

"term": "open" Start a terminal in a new window and connect the job stdin/stdout/stderr to it. Similar to using job-term  
`:terminal``.  
**NOTE:** Not implemented yet!

"channel": {channel} Use an existing channel instead of creating a new one. The parts of the channel that get used for the new job will be disconnected from what they were used before. If the channel was still used by another job this may cause I/O errors. Existing callbacks and other settings remain.

"pty": 1 Use a pty (pseudo-tty) instead of a pipe when possible. This is most useful in combination with a terminal window, see terminal .  
{only on Unix and Unix-like systems}

"in\_io": "null" disconnect stdin (read from /dev/null) job-in\_io in\_top in\_bot in\_name in\_buf

"in\_io": "pipe" stdin is connected to the channel (default)

"in\_io": "file" stdin reads from a file

"in\_io": "buffer" stdin reads from a buffer

"in\_top": number when using "buffer": first line to send (default: 1)

"in\_bot": number when using "buffer": last line to send (default: last)

"in\_name": "/path/file" the name of the file or buffer to read from

"in\_buf": number the number of the buffer to read from

"out\_io": "null" disconnect stdout (goes to /dev/null) job-out\_io out\_name out\_buf

"out\_io": "pipe" stdout is connected to the channel (default)

"out\_io": "file" stdout writes to a file

"out\_io": "buffer" stdout appends to a buffer (see below)

"out\_name": "/path/file" the name of the file or buffer to write to

"out\_buf": number the number of the buffer to write to

"out\_modifiable": 0 when writing to a buffer, **'modifiable'** will be off (see below)

"out\_msg": 0 when writing to a new buffer, the first line will be

set to "Reading from channel output..."

|                          |                                                                                                  |                       |                      |
|--------------------------|--------------------------------------------------------------------------------------------------|-----------------------|----------------------|
|                          | <code>job-err_io</code>                                                                          | <code>err_name</code> | <code>err_buf</code> |
| "err_io": "out"          | stderr messages to go to stdout                                                                  |                       |                      |
| "err_io": "null"         | disconnect stderr (goes to /dev/null)                                                            |                       |                      |
| "err_io": "pipe"         | stderr is connected to the channel (default)                                                     |                       |                      |
| "err_io": "file"         | stderr writes to a file                                                                          |                       |                      |
| "err_io": "buffer"       | stderr appends to a buffer (see below)                                                           |                       |                      |
| "err_name": "/path/file" | the name of the file or buffer to write to                                                       |                       |                      |
| "err_buf": number        | the number of the buffer to write to                                                             |                       |                      |
| "err_modifiable": 0      | when writing to a buffer, <b>'modifiable'</b> will be off (see below)                            |                       |                      |
| "err_msg": 0             | when writing to a new buffer, the first line will be set to "Reading from channel error..."      |                       |                      |
| "block_write": number    | only for testing: pretend every other write to stdin will block                                  |                       |                      |
| "env": dict              | environment variables for the new process                                                        |                       |                      |
| "cwd": "/path/to/dir"    | current working directory for the new process; if the directory does not exist an error is given |                       |                      |

## Writing to a buffer

### `out_io-buffer`

When the `out_io` or `err_io` mode is "buffer" and there is a callback, the text is appended to the buffer before invoking the callback.

When a buffer is used both for input and output, the output lines are put above the last line, since the last line is what is written to the channel input. Otherwise lines are appended below the last line.

When using JS or JSON mode with "buffer", only messages with zero or negative ID will be added to the buffer, after decoding + encoding. Messages with a positive number will be handled by a callback, commands are handled as usual.

The name of the buffer from "out\_name" or "err\_name" is compared the full name of existing buffers, also after expanding the name for the current directory. E.g., when a buffer was created with ":edit somename" and the buffer name is "somename" it will use that buffer.

If there is no matching buffer a new buffer is created. Use an empty name to always create a new buffer. `ch_getbufnr()` can then be used to get the buffer number.

For a new buffer **'buftype'** is set to "nofile" and **'bufhidden'** to "hide". If you prefer other settings, create the buffer first and pass the buffer number.

### `out_modifiable` `err_modifiable`

The "out\_modifiable" and "err\_modifiable" options can be used to set the **'modifiable'** option off, or write to a buffer that has **'modifiable'** off. That means that lines will be appended to the buffer, but the user can't easily change the buffer.

### `out_msg` `err_msg`

The "out\_msg" option can be used to specify whether a new buffer will have the

first line set to "Reading from channel output...". The default is to add the message. "err\_msg" does the same for channel error.

When an existing buffer is to be written where **'modifiable'** is off and the "out\_modifiable" or "err\_modifiable" options is not zero, an error is given and the buffer will not be written to.

When the buffer written to is displayed in a window and the cursor is in the first column of the last line, the cursor will be moved to the newly added line and the window is scrolled up to show the cursor if needed.

Undo is synced for every added line. NUL bytes are accepted (internally Vim stores these as NL bytes).

### Writing to a file

E920

The file is created with permissions 600 (read-write for the user, not accessible for others). Use `setfperm()` to change this.

If the file already exists it is truncated.

---

## 13. Controlling a job

job-control

To get the status of a job:  
`echo job_status(job)`

To make a job stop running:  
`job_stop(job)`

This is the normal way to end a job. On Unix it sends a SIGTERM to the job. It is possible to use other ways to stop the job, or even send arbitrary signals. E.g. to force a job to stop, "kill it":

`job_stop(job, "kill")`

For more options see `job_stop()` .

---

## 14. Using a prompt buffer

prompt-buffer

If you want to type input for the job in a Vim window you have a few options:

- Use a normal buffer and handle all possible commands yourself. This will be complicated, since there are so many possible commands.
- Use a terminal window. This works well if what you type goes directly to the job and the job output is directly displayed in the window. See `terminal-window` .
- Use a window with a prompt buffer. This works well when entering a line for the job in Vim while displaying (possibly filtered) output from the job.

A prompt buffer is created by setting **'buftype'** to "prompt". You would normally only do that in a newly created buffer.

The user can edit and enter one line of text at the very last line of the



buffer. When pressing Enter in the prompt line the callback set with `prompt_setcallback()` is invoked. It would normally send the line to a job. Another callback would receive the output from the job and display it in the buffer, below the prompt (and above the next prompt).

Only the text in the last line, after the prompt, is editable. The rest of the buffer is not modifiable with Normal mode commands. It can be modified by calling functions, such as `append()`. Using other commands may mess up the buffer.

After setting `'buftype'` to "prompt" Vim does not automatically start Insert mode, use `:`startinsert`` if you want to enter Insert mode, so that the user can start typing a line.

The text of the prompt can be set with the `prompt_setprompt()` function. If no prompt is set with `prompt_setprompt()`, "% " is used. You can get the effective prompt text for a buffer, with `prompt_getprompt()`.

The user can go to Normal mode and navigate through the buffer. This can be useful to see older output or copy text.

The **CTRL-W** key can be used to start a window command, such as **CTRL-W** w to switch to the next window. This also works in Insert mode (use Shift-CTRL-W to delete a word). When leaving the window Insert mode will be stopped. When coming back to the prompt window Insert mode will be restored.

Any command that starts Insert mode, such as "a", "i", "A" and "I", will move the cursor to the last line. "A" will move to the end of the line, "I" to the start of the line.

Here is an example for Unix. It starts a shell in the background and prompts for the next shell command. Output from the shell is displayed above the prompt.

```
" Create a channel log so we can see what happens.
call ch_logfile('logfile', 'w')

" Function handling a line of text that has been typed.
func TextEntered(text)
 " Send the text to a shell with Enter appended.
 call ch_sendraw(g:shell_job, a:text .. "\n")
endfunc

" Function handling output from the shell: Add it above the prompt.
func GotOutput(channel, msg)
 call append(line("$") - 1, "- " .. a:msg)
endfunc

" Function handling the shell exits: close the window.
func JobExit(job, status)
 quit!
endfunc

" Start a shell in the background.
```

```

let shell_job = job_start(["/bin/sh"], #{
 \ out_cb: function('GotOutput'),
 \ err_cb: function('GotOutput'),
 \ exit_cb: function('JobExit'),
 \ })

new
set buftype=prompt
let buf = bufnr('')
call prompt_setcallback(buf, function("TextEntered"))
eval prompt_setprompt(buf, "shell command: ")

" start accepting shell commands
startinsert

```

The same in `Vim9` script:

```

vim9script

Create a channel log so we can see what happens.
ch_logfile('logfile', 'w')

var shell_job: job

Function handling a line of text that has been typed.
def TextEntered(text: string)
 # Send the text to a shell with Enter appended.
 ch_sendraw(shell_job, text .. "\n")
enddef

Function handling output from the shell: Add it above the prompt.
def GotOutput(channel: channel, msg: string)
 append(line("$") - 1, "- " .. msg)
enddef

Function handling the shell exits: close the window.
def JobExit(job: job, status: number)
 quit!
enddef

Start a shell in the background.
shell_job = job_start(["/bin/sh"], {
 out_cb: GotOutput,
 err_cb: GotOutput,
 exit_cb: JobExit,
})

new
set buftype=prompt
var buf = bufnr('')
prompt_setcallback(buf, TextEntered)
prompt_setprompt(buf, "shell command: ")

start accepting shell commands

```

## 15. Language Server Protocol

language-server-protocol

The language server protocol specification is available at:

<https://microsoft.github.io/language-server-protocol/specification>

Each LSP protocol message starts with a simple HTTP header followed by the payload encoded in JSON-RPC format. This is described in:

<https://www.jsonrpc.org/specification>

To encode and send a LSP request/notification message in a Vim `Dict` into a LSP JSON-RPC message and to receive and decode a LSP JSON-RPC response/notification message into a Vim `Dict`, connect to the LSP server with the `channel-mode` set to "lsp".

For messages received on a channel with `channel-mode` set to "lsp", Vim will process the HTTP header and decode the JSON-RPC payload into a Vim `Dict` type and call the `channel-callback` function or the specified `channel-onetime-callback` function. When sending messages on a channel using the `ch_evaluate()` or `ch_sendexpr()` functions, Vim will add the HTTP header and encode the Vim expression into JSON. Refer to `json_encode()` and `json_decode()` for more information about how Vim encodes and decodes the builtin types into JSON.

To open a channel using the `'lsp'` mode, set the `'mode'` item in the `ch_open()` `{options}` argument to `'lsp'`. Example:

```
let ch = ch_open(..., #{mode: 'lsp'})
```

To open a channel using the `'lsp'` mode with a job, set the `'in_mode'` and `'out_mode'` items in the `job_start()` `{options}` argument to `'lsp'`. Example:

```
let cmd = ['clangd', '--background-index', '--clang-tidy']
let opts = {}
let opts.in_mode = 'lsp'
let opts.out_mode = 'lsp'
let opts.err_mode = 'nl'
let opts.out_cb = function('LspOutCallback')
let opts.err_cb = function('LspErrCallback')
let opts.exit_cb = function('LspExitCallback')
let job = job_start(cmd, opts)
```

**Note** that if a job outputs LSP messages on stdout and non-LSP messages on stderr, then the channel-callback function should handle both the message formats appropriately or you should use a separate callback function for "out\_cb" and "err\_cb" to handle them as shown above.

To synchronously send a JSON-RPC request to the server, use the `ch_evaluate()` function. This function will wait and return the decoded response message from the server. You can use either the `channel-timeout` or

the `'timeout'` field in the `{options}` argument to control the response wait time. If the request times out, then an empty `Dict` is returned. Example:

```
let req = {}
let req.method = 'textDocument/definition'
let req.params = {}
let req.params.textDocument = #{uri: 'a.c'}
let req.params.position = #{line: 10, character: 3}
let defs = ch_evalexpr(ch, req, #{timeout: 100})
if defs->empty()
 ... <handle failure>
endif
```

**Note** that in the request message the `'id'` field should not be specified. If it is specified, then Vim will overwrite the value with an internally generated identifier. Vim currently supports only a number type for the `'id'` field. The callback function will be invoked for both a successful and a failed RPC request.

To send a JSON-RPC request to the server and asynchronously process the response, use the `ch_sendexpr()` function and supply a callback function. If the `"id"` field is present in the request message, then Vim will overwrite it with an internally generated number. This function returns a `Dict` with the identifier used for the message. This can be used to send cancellation request to the LSP server (if needed). Example:

```
let req = {}
let req.method = 'textDocument/hover'
let req.id = 200
let req.params = {}
let req.params.textDocument = #{uri: 'a.c'}
let req.params.position = #{line: 10, character: 3}
let resp = ch_sendexpr(ch, req, #{callback: 'HoverFunc'})
```

To cancel an outstanding asynchronous LSP request sent to the server using the `ch_sendexpr()` function, send a cancellation message to the server using the `ch_sendexpr()` function with the ID returned by the `ch_sendexpr()` function for the request. Example:

```
" send a completion request
let req = {}
let req.method = 'textDocument/completion'
let req.params = {}
let req.params.textDocument = #{uri: 'a.c'}
let req.params.position = #{line: 10, character: 3}
let reqstatus = ch_sendexpr(ch, req, #{callback: 'LspComplete'})
" send a cancellation notification
let notif = {}
let notif.method = '$/cancelRequest'
let notif.id = reqstatus.id
call ch_sendexpr(ch, notif)
```

To send a JSON-RPC notification message to the server, use the `ch_sendexpr()` function. As the server will not send a response message to the notification,

don't specify the "callback" item. Example:

```
call ch_sendexpr(ch, #{method: 'initialized'})
```

To respond to a JSON-RPC request message from the server, use the `ch_sendexpr()` function. In the response message, copy the `'id'` field value from the server request message. Example:

```
let resp = {}
let resp.id = req.id
let resp.result = 1
call ch_sendexpr(ch, resp)
```

The JSON-RPC notification messages from the server are delivered through the `channel-callback` function.

Depending on the use case, you can use the `ch_evalexpr()`, `ch_sendexpr()` and `ch_sendraw()` functions on the same channel.

A LSP request message has the following format (expressed as a Vim Dict). The "params" field is optional:

```
{
 "jsonrpc": "2.0",
 "id": <number>,
 "method": <string>,
 "params": <list|dict>
}
```

A LSP response message has the following format (expressed as a Vim Dict). The "result" and "error" fields are optional:

```
{
 "jsonrpc": "2.0",
 "id": <number>,
 "result": <vim type>
 "error": <dict>
}
```

A LSP notification message has the following format (expressed as a Vim Dict). The "params" field is optional:

```
{
 "jsonrpc": "2.0",
 "method": <string>,
 "params": <list|dict>
}
```



## Folding

Folding folding folds

You can find an introduction on folding in chapter 28 of the user manual.  
[usr\\_28.txt](#)

1. Fold methods [fold-methods](#)
2. Fold commands [fold-commands](#)
3. Fold options [fold-options](#)
4. Behavior of folds [fold-behavior](#)

{not available when compiled without the |+folding| feature}

=====

1. Fold methods

[fold-methods](#)

The folding method can be set with the **'foldmethod'** option.

When setting **'foldmethod'** to a value other than "manual", all folds are deleted and new ones created. Switching to the "manual" method doesn't remove the existing folds. This can be used to first define the folds automatically and then change them manually.

There are six methods to select folds:

|        |                                       |
|--------|---------------------------------------|
| manual | manually define folds                 |
| indent | more indent means a higher fold level |
| expr   | specify an expression to define folds |
| syntax | folds defined by syntax highlighting  |
| diff   | folds for unchanged text              |
| marker | folds defined by markers in the text  |

**MANUAL**[fold-manual](#)

Use commands to manually define the fold regions. This can also be used by a script that parses text to find folds.

The level of a fold is only defined by its nesting. To increase the fold level of a fold for a range of lines, define a fold inside it that has the same lines.

The manual folds are lost when you abandon the file. To save the folds use the **:mkview** command. The view can be restored later with **:loadview** .

**INDENT**[fold-indent](#)

The folds are automatically defined by the indent of the lines.

The `foldlevel` is computed from the indent of the line, divided by the `'shiftwidth'` (rounded down). A sequence of lines with the same or higher fold level form a fold, with the lines with a higher level forming a nested fold.

The nesting of folds is limited with `'foldnestmax'`.

Some lines are ignored and get the fold level of the line above or below it, whichever is lower. These are empty or white lines and lines starting with a character in `'foldignore'`. White space is skipped before checking for characters in `'foldignore'`. For C use `"#"` to ignore preprocessor lines.

When you want to ignore lines in another way, use the `"expr"` method. The `indent()` function can be used in `'foldexpr'` to get the indent of a line.

## EXPR

## fold-expr

The folds are automatically defined by their `foldlevel`, like with the `"indent"` method. The value of the `'foldexpr'` option is evaluated to get the `foldlevel` of a line. Examples:

This will create a fold for all consecutive lines that start with a tab:

```
:set foldexpr=getline(v:lnum)[0]=="\t"
```

This will make a fold out of paragraphs separated by blank lines:

```
:set foldexpr=getline(v:lnum) =~ '\s*$' && getline(v:lnum+1) =~ '\S?' <1':1
```

This does the same:

```
:set foldexpr=getline(v:lnum-1) =~ '\s*$' && getline(v:lnum) =~ '\S?' >1':1
```

**Note** that backslashes must be used to escape characters that `:"set"` handles differently (space, backslash, double quote, etc., see [option-backslash](#)).

The most efficient is to call a compiled function without arguments:

```
:set foldexpr=MyFoldLevel()
```

The function must use `v:lnum`. See [expr-option-function](#).

These are the conditions with which the expression is evaluated:

- The current buffer and window are set for the line.
- The variable `"v:lnum"` is set to the line number.
- The result is used for the fold level in this way:

| value          | meaning                                                                                                       |
|----------------|---------------------------------------------------------------------------------------------------------------|
| 0              | the line is not in a fold                                                                                     |
| 1, 2, ..       | the line is in a fold with this level                                                                         |
| -1             | the fold level is undefined, use the fold level of a line before or after this line, whichever is the lowest. |
| "="            | use fold level from the previous line                                                                         |
| "a1", "a2", .. | add one, two, .. to the fold level of the previous line, use the result for the current line                  |
| "s1", "s2", .. | subtract one, two, .. from the fold level of the previous line, use the result for the next line              |
| "<1", "<2", .. | a fold with this level ends at this line                                                                      |
| ">1", ">2", .. | a fold with this level starts at this line                                                                    |

It is not required to mark the start (end) of a fold with `">1"` ("`<1"`), a fold will also start (end) when the fold level is higher (lower) than the fold



level of the previous line.

There must be no side effects from the expression. The text in the buffer, cursor position, the search patterns, options etc. must not be changed. You can change and restore them if you are careful.

If there is some error in the expression, or the resulting value isn't recognized, there is no error message and the fold level will be zero. For debugging the **'debug'** option can be set to "msg", the error messages will be visible then.

**Note:** Since the expression has to be evaluated for every line, this fold method can be very slow!

Try to avoid the "=", "a" and "s" return values, since Vim often has to search backwards for a line for which the fold level is defined. This can be slow.

If the **'foldexpr'** expression starts with s: or <SID> , then it is replaced with the script ID ( **local-function** ). Examples:

```
set foldexpr=s:MyFoldExpr()
set foldexpr=<SID>SomeFoldExpr()
```

An example of using "a1" and "s1": For a multi-line C comment, a line containing "/\*" would return "a1" to start a fold, and a line containing "\*/" would return "s1" to end the fold after that line:

```
if match(thisline, '/*') >= 0
 return 'a1'
elseif match(thisline, '*/') >= 0
 return 's1'
else
 return '='
endif
```

However, this won't work for single line comments, strings, etc.

**foldlevel()** can be useful to compute a fold level relative to a previous fold level. But **note** that **foldlevel()** may return -1 if the level is not known yet. And it returns the level at the start of the line, while a fold might end in that line.

It may happen that folds are not updated properly. You can use **zx** or **zX** to force updating folds.

## SYNTAX

## fold-syntax

A fold is defined by syntax items that have the "fold" argument. **:syn-fold**

The fold level is defined by nesting folds. The nesting of folds is limited with **'foldnestmax'**.

Be careful to specify proper syntax syncing. If this is not done right, folds may differ from the displayed highlighting. This is especially relevant when using patterns that match more than one line. In case of doubt, try using brute-force syncing:

```
:syn sync fromstart
```

## DIFF

## fold-diff

The folds are automatically defined for text that is not part of a change or close to a change.

This method only works properly when the **'diff'** option is set for the current window and changes are being displayed. Otherwise the whole buffer will be one big fold.

The **'diffopt'** option can be used to specify the context. That is, the number of lines between the fold and a change that are not included in the fold. For example, to use a context of 8 lines:

```
:set diffopt=filler,context:8
```

The default context is six lines.

When **'scrollbind'** is also set, Vim will attempt to keep the same folds open in other diff windows, so that the same text is visible.

## MARKER

## fold-marker

Markers in the text tell where folds start and end. This allows you to precisely specify the folds. This will allow deleting and putting a fold, without the risk of including the wrong lines. The **'foldtext'** option is normally set such that the text before the marker shows up in the folded line. This makes it possible to give a name to the fold.

Markers can have a level included, or can use matching pairs. Including a level is easier, you don't have to add end markers and avoid problems with non-matching marker pairs. Example:

```
/* global variables {{{1 */
int varA, varB;
```

```
/* functions {{{1 */
/* funcA() {{{2 */
void funcA() {}
```

```
/* funcB() {{{2 */
void funcB() {}
```

```
{{{ }}}
```

A fold starts at a "**{{{**" marker. The following number specifies the fold level. What happens depends on the difference between the current fold level and the level given by the marker:

1. If a marker with the same fold level is encountered, the previous fold ends and another fold with the same level starts.
2. If a marker with a higher fold level is found, a nested fold is started.
3. If a marker with a lower fold level is found, all folds up to and including this level end and a fold with the specified level starts.

The number indicates the fold level. A zero cannot be used (a marker with level zero is ignored). You can use "**}}}"** with a digit to indicate the level

of the fold that ends. The fold level of the following line will be one less than the indicated level. **Note** that Vim doesn't look back to the level of the matching marker (that would take too much time). Example:

```
{{{1
fold level here is 1
{{{3
fold level here is 3
}}}3
fold level here is 2
```

You can also use matching pairs of "{{{" and "}}}" markers to define folds. Each "{{{" increases the fold level by one, each "}}}" decreases the fold level by one. Be careful to keep the markers matching! Example:

```
{{{
fold level here is 1
{{{
fold level here is 2
}}}
fold level here is 1
```

You can mix using markers with a number and without a number. A useful way of doing this is to use numbered markers for large folds, and unnumbered markers locally in a function. For example use level one folds for the sections of your file like "structure definitions", "local variables" and "functions". Use level 2 markers for each definition and function, Use unnumbered markers inside functions. When you make changes in a function to split up folds, you don't have to renumber the markers.

The markers can be set with the **'foldmarker'** option. It is recommended to keep this at the default value of "{{{,}}}", so that files can be exchanged between Vim users. Only change it when it is required for the file (e.g., it contains markers from another folding editor, or the default markers cause trouble for the language of the file).

#### fold-create-marker

"zf" can be used to create a fold defined by markers. Vim will insert the markers for you. Vim will append the start and end marker, as specified with **'foldmarker'**. The markers are appended to the end of the line.

**'commentstring'** is used if it isn't empty.

This does not work properly when:

- The line already contains a marker with a level number. Vim then doesn't know what to do.
- Folds nearby use a level number in their marker which gets in the way.
- The line is inside a comment, **'commentstring'** isn't empty and nested comments don't work. For example with C: adding /\* {{{ \*/ inside a comment will truncate the existing comment. Either put the marker before or after the comment, or add the marker manually.

Generally it's not a good idea to let Vim create markers when you already have markers with a level number.

#### fold-delete-marker

"zd" can be used to delete a fold defined by markers. Vim will delete the

markers for you. Vim will search for the start and end markers, as specified with **'foldmarker'**, at the start and end of the fold. When the text around the marker matches with **'commentstring'**, that text is deleted as well.

This does not work properly when:

- A line contains more than one marker and one of them specifies a level. Only the first one is removed, without checking if this will have the desired effect of deleting the fold.
- The marker contains a level number and is used to start or end several folds at the same time.

---

## 2. Fold commands

fold-commands E490

All folding commands start with "z". Hint: the "z" looks like a folded piece of paper, if you look at it from the side.

### CREATING AND DELETING FOLDS

zf E350

zf{motion} or  
{Visual}zf Operator to create a fold.  
This only works when **'foldmethod'** is "manual" or "marker".  
The new fold will be closed for the "manual" method.  
**'foldenable'** will be set.  
Also see [fold-create-marker](#) .

zF

zF Create a fold for [count] lines. Works like "zf".

:{range}fo[ld]

:fold :fo

Create a fold for the lines in {range}. Works like "zf".

zd E351

zd Delete one fold at the cursor. When the cursor is on a folded line, that fold is deleted. Nested folds are moved one level up. In Visual mode one level of all folds (partially) in the selected area are deleted.  
Careful: This easily deletes more folds than you expect and there is no undo for manual folding.  
This only works when **'foldmethod'** is "manual" or "marker".  
Also see [fold-delete-marker](#) .

zD

zD Delete folds recursively at the cursor. In Visual mode all folds (partially) in the selected area and all nested folds in them are deleted.  
This only works when **'foldmethod'** is "manual" or "marker".  
Also see [fold-delete-marker](#) .

zE E352

zE Eliminate all folds in the window.  
This only works when **'foldmethod'** is "manual" or "marker".  
Also see [fold-delete-marker](#) .

## OPENING AND CLOSING FOLDS

A fold smaller than **'foldminlines'** will always be displayed like it was open. Therefore the commands below may work differently on small folds.

- zo** **zo**  
Open one fold under the cursor. When a count is given, that many folds deep will be opened. In Visual mode one level of folds is opened for all lines in the selected area.
- zO** **zO**  
Open all folds under the cursor recursively. Folds that don't contain the cursor line are unchanged. In Visual mode it opens all folds that are in the selected area, also those that are only partly selected.
- zc** **zc**  
Close one fold under the cursor. When a count is given, that many folds deep are closed. In Visual mode one level of folds is closed for all lines in the selected area.  
**'foldenable'** will be set.
- zC** **zC**  
Close all folds under the cursor recursively. Folds that don't contain the cursor line are unchanged. In Visual mode it closes all folds that are in the selected area, also those that are only partly selected.  
**'foldenable'** will be set.
- za** **za**  
Summary: Toggle the fold under the cursor.  
When on a closed fold: open it. When folds are nested, you may have to use "za" several times. When a count is given, that many closed folds are opened.  
When on an open fold: close it and set **'foldenable'**. This will only close one level, since using "za" again will open the fold. When a count is given that many folds will be closed (that's not the same as repeating "za" that many times).
- zA** **zA**  
When on a closed fold: open it recursively.  
When on an open fold: close it recursively and set **'foldenable'**.
- zv** **zv**  
View cursor line: Open just enough folds to make the line in which the cursor is located not folded.
- zx** **zx**  
Update folds: Undo manually opened and closed folds: re-apply **'foldlevel'**, then do "zv": View cursor line.  
Also forces recomputing folds. This is useful when using

'foldexpr' and the buffer is changed in a way that results in folds not to be updated properly.

- zX zX  
Undo manually opened and closed folds: re-apply 'foldlevel'. Also forces recomputing folds, like zx .
- zm zm  
Fold more: Subtract v:count1 from 'foldlevel'. If 'foldlevel' was already zero nothing happens. 'foldenable' will be set.
- zM zM  
Close all folds: set 'foldlevel' to 0. 'foldenable' will be set.
- zr zr  
Reduce folding: Add v:count1 to 'foldlevel'.
- zR zR  
Open all folds. This sets 'foldlevel' to highest fold level.
- :foldo :foldopen  
:{range}foldo[pen][!]  
Open folds in {range}. When [!] is added all folds are opened. Useful to see all the text in {range}. Without [!] one level of folds is opened.
- :foldc :foldclose  
:{range}foldc[lose][!]  
Close folds in {range}. When [!] is added all folds are closed. Useful to hide all the text in {range}. Without [!] one level of folds is closed.
- zn zn  
Fold none: reset 'foldenable'. All folds will be open.
- zN zN  
Fold normal: set 'foldenable'. All folds will be as they were before.
- zi zi  
Invert 'foldenable'.

## MOVING OVER FOLDS

- [z [z  
Move to the start of the current open fold. If already at the start, move to the start of the fold that contains it. If there is no containing fold, the command fails. When a count is used, repeats the command [count] times.
- ]z ]z  
Move to the end of the current open fold. If already at the

end, move to the end of the fold that contains it. If there is no containing fold, the command fails.  
When a count is used, repeats the command [count] times.

**zj** zj  
Move downwards to the start of the next fold. A closed fold is counted as one fold.  
When a count is used, repeats the command [count] times.  
This command can be used after an operator .

**zk** zk  
Move upwards to the end of the previous fold. A closed fold is counted as one fold.  
When a count is used, repeats the command [count] times.  
This command can be used after an operator .

## EXECUTING COMMANDS ON FOLDS

**:[range]foldd[open] {cmd}** :foldd :folddo :folddoopen  
Execute {cmd} on all lines that are not in a closed fold.  
When [range] is given, only these lines are used.  
Each time {cmd} is executed the cursor is positioned on the line it is executed for.  
This works like the ":global" command: First all lines that are not in a closed fold are marked. Then the {cmd} is executed for all marked lines. Thus when {cmd} changes the folds, this has no influence on where it is executed (except when lines are deleted, of course).  
Example:  
`:folddoopen s/end/loop_end/ge`  
**Note** the use of the "e" flag to avoid getting an error message where "end" doesn't match.

**:[range]folddoc[losed] {cmd}** :folddoc :folddoclosed  
Execute {cmd} on all lines that are in a closed fold.  
Otherwise like ":folddoopen".

---

### 3. Fold options

fold-options

#### COLORS

fold-colors

The colors of a closed fold are set with the Folded group `hl-Folded` . The colors of the fold column are set with the FoldColumn group `hl-FoldColumn` .  
Example to set the colors:

```
:highlight Folded guibg=grey guifg=blue
:highlight FoldColumn guibg=darkgrey guifg=white
```

#### FOLDLEVEL

fold-foldlevel

**'foldlevel'** is a number option: The higher the more folded regions are open.

When **'foldlevel'** is 0, all folds are closed.  
 When **'foldlevel'** is positive, some folds are closed.  
 When **'foldlevel'** is very high, all folds are open.  
**'foldlevel'** is applied when it is changed. After that manually folds can be opened and closed.  
 When increased, folds above the new level are opened. No manually opened folds will be closed.  
 When decreased, folds above the new level are closed. No manually closed folds will be opened.

## FOLDTEXT

## fold-foldtext

**'foldtext'** is a string option that specifies an expression. This expression is evaluated to obtain the text displayed for a closed fold. Example:

```
:set foldtext=v:folddashes.substitute(getline(v:foldstart),'/*\\\|*\|\\\|{\{\d\\='
```

This shows the first line of the fold, with **"/\***, **"\*/"** and **"{{"** removed.  
**Note** the use of backslashes to avoid some characters to be interpreted by the **":set"** command. It is much simpler to define a function and call it:

```
:set foldtext=MyFoldText()
:function MyFoldText()
: let line = getline(v:foldstart)
: let sub = substitute(line, '/*\\\|*\|\\\|{\{\d\\=', '', 'g')
: return v:folddashes .. sub
:endfunction
```

The advantage of using a function call without arguments is that it is faster, see [expr-option-function](#).

Evaluating **'foldtext'** is done in the [sandbox](#). The current window is set to the window that displays the line. The context is set to the script where the option was last set.

Errors are ignored. For debugging set the **'debug'** option to **"throw"**.

The default value is [foldtext\(\)](#). This returns a reasonable text for most types of folding. If you don't like it, you can specify your own **'foldtext'** expression. It can use these special Vim variables:

|              |                                                           |
|--------------|-----------------------------------------------------------|
| v:foldstart  | line number of first line in the fold                     |
| v:foldend    | line number of last line in the fold                      |
| v:folddashes | a string that contains dashes to represent the foldlevel. |
| v:foldlevel  | the foldlevel of the fold                                 |

In the result a TAB is replaced with a space and unprintable characters are made into printable characters.

The resulting line is truncated to fit in the window, it never wraps. When there is room after the text, it is filled with the character specified by **'fillchars'**.



If the **'foldtext'** expression starts with `s:` or `<SID>` , then it is replaced with the script ID ( `local-function` ). Examples:

```
set foldtext=s:MyFoldText()
set foldtext=<SID>SomeFoldText()
```

**Note** that backslashes need to be used for characters that the `":set"` command handles differently: Space, backslash and double-quote. `option-backslash`

## FOLDCOLUMN

## fold-foldcolumn

**'foldcolumn'** is a number, which sets the width for a column on the side of the window to indicate folds. When it is zero, there is no foldcolumn. A normal value is 4 or 5. The minimal useful value is 2, although 1 still provides some information. The maximum is 12.

An open fold is indicated with a column that has a '-' at the top and '|' characters below it. This column stops where the open fold stops. When folds nest, the nested fold is one character right of the fold it's contained in.

A closed fold is indicated with a '+'.

These characters can be changed with the **'fillchars'** option.

Where the fold column is too narrow to display all nested folds, digits are shown to indicate the nesting level.

The mouse can also be used to open and close folds by clicking in the fold column:

- Click on a '+' to open the closed fold at this row.
- Click on any other non-blank character to close the open fold at this row.

## OTHER OPTIONS

|                       |                |                                                                   |
|-----------------------|----------------|-------------------------------------------------------------------|
| <b>'foldenable'</b>   | <b>'fen'</b> : | Open all folds while not set.                                     |
| <b>'foldexpr'</b>     | <b>'fde'</b> : | Expression used for "expr" folding.                               |
| <b>'foldignore'</b>   | <b>'fdi'</b> : | Characters used for "indent" folding.                             |
| <b>'foldmarker'</b>   | <b>'fmr'</b> : | Defined markers used for "marker" folding.                        |
| <b>'foldmethod'</b>   | <b>'fdm'</b> : | Name of the current folding method.                               |
| <b>'foldminlines'</b> | <b>'fml'</b> : | Minimum number of screen lines for a fold to be displayed closed. |
| <b>'foldnestmax'</b>  | <b>'fdn'</b> : | Maximum nesting for "indent" and "syntax" folding.                |
| <b>'foldopen'</b>     | <b>'fdo'</b> : | Which kinds of commands open closed folds.                        |
| <b>'foldclose'</b>    | <b>'fcl'</b> : | When the folds not under the cursor are closed.                   |

---

## 4. Behavior of folds fold-behavior

When moving the cursor upwards or downwards and when scrolling, the cursor will move to the first line of a sequence of folded lines. When the cursor is already on a folded line, it moves to the next unfolded line or the next closed fold.

While the cursor is on folded lines, the cursor is always displayed in the first column. The ruler does show the actual cursor position, but since the line is folded, it cannot be displayed there.

Many movement commands handle a sequence of folded lines like an empty line. For example, the "w" command stops once in the first column.

When starting a search in a closed fold it will not find a match in the current fold. It's like a forward search always starts from the end of the closed fold, while a backwards search starts from the start of the closed fold.

When in Insert mode, the cursor line is never folded. That allows you to see what you type!

When using an operator, a closed fold is included as a whole. Thus "dl" deletes the whole closed fold under the cursor.

For Ex commands that work on buffer lines the range is adjusted to always start at the first line of a closed fold and end at the last line of a closed fold. Thus this command:

```
:s/foo/bar/g
```

when used with the cursor on a closed fold, will replace "foo" with "bar" in all lines of the fold.

This does not happen for `:folddoopen` and `:folddoclosed` .

When editing a buffer that has been edited before, the last used folding settings are used again. For manual folding the defined folds are restored. For all folding methods the manually opened and closed folds are restored. If this buffer has been edited in this window, the values from back then are used. Otherwise the values from the window where the buffer was edited last are used.

=====

Testing Vim and Vim script [testing-support](#)

Expression evaluation is explained in [eval.txt](#) . This file goes into details about writing tests in Vim script. This can be used for testing Vim itself and for testing plugins.

- 1. Testing Vim [testing](#)
- 2. Test functions [test-functions-details](#)
- 3. Assert functions [assert-functions-details](#)

=====

1. Testing Vim [testing](#)

Vim can be tested after building it, usually with "make test". The tests are located in the directory "src/testdir".

There are two types of tests added over time:

- test20.in oldest, only for tiny builds
- test\_something.vim new style tests

[new-style-testing](#)

New tests should be added as new style tests. The test scripts are named test\_<feature>.vim (replace <feature> with the feature under test). These use functions such as [assert\\_equal\(\)](#) to keep the test commands and the expected result in one place.

[old-style-testing](#)

These tests are used only for testing Vim without the [+eval](#) feature.

Find more information in the file src/testdir/README.txt.

=====

2. Test functions [test-functions-details](#)

test\_alloc\_fail({id}, {countdown}, {repeat}) [test\\_alloc\\_fail\(\)](#)

This is for testing: If the memory allocation with {id} is called, then decrement {countdown}, and when it reaches zero let memory allocation fail {repeat} times. When {repeat} is smaller than one it fails one time.

Can also be used as a [method](#) :

[GetAllocId\(\)->test\\_alloc\\_fail\(\)](#)

test\_autochdir() [test\\_autochdir\(\)](#)

Set a flag to enable the effect of ['autochdir'](#) before Vim startup has finished.

`test_feedinput({string})` `test_feedinput()`  
Characters in `{string}` are queued for processing as if they were typed by the user. This uses a low level input buffer. This function works only when with `+unix` or GUI is running.

Can also be used as a `method` :  
`GetText()->test_feedinput()`

`test_garbagecollect_now()` `test_garbagecollect_now()`  
Like `garbagecollect()`, but executed right away. This must only be called directly to avoid any structure to exist internally, and `v:testing` must have been set before calling any function. `E1142`  
This will not work when called from a `:def` function, because variables on the stack will be freed.

`test_garbagecollect_soon()` `test_garbagecollect_soon()`  
Set the flag to call the garbagecollector as if in the main loop. Only to be used in tests.

`test_getvalue({name})` `test_getvalue()`  
Get the value of an internal variable. These values for `{name}` are supported:  
    `need_fileinfo`

Can also be used as a `method` :  
`GetName()->test_getvalue()`

`test_gui_event({event}, {args})` `test_gui_event()`  
Generate a GUI `{event}` with arguments `{args}` for testing Vim functionality. This function works only when the GUI is running.

`{event}` is a String and the supported values are:  
    `"dropfiles"` drop one or more files in a window.  
    `"findrepl"` search and replace text.  
    `"mouse"` mouse button click event.  
    `"scrollbar"` move or drag the scrollbar.  
    `"key"` send a low-level keyboard event.  
    `"tabline"` select a tab page by mouse click.  
    `"tabmenu"` select a tabline menu entry.

`{args}` is a Dict and contains the arguments for the event.

`"dropfiles"`:  
    Drop one or more files in a specified window. The supported items in `{args}` are:  
    `files:` List of file names  
    `row:` window row number  
    `col:` window column number

modifiers: key modifiers. The supported values are:  
0x4 Shift  
0x8 Alt  
0x10 Ctrl

The files are added to the `argument-list` and the first file in `{files}` is edited in the window. See `drag-n-drop` for more information. This event works only when the `drop_file` feature is present.

"findrepl":

{only available when the GUI has a find/replace dialog}

Perform a search and replace of text. The supported items in `{args}` are:

find\_text: string to find.

repl\_text: replacement string.

flags: flags controlling the find/replace. Supported values are:

- 1 search next string (find dialog)
- 2 search next string (replace dialog)
- 3 replace string once
- 4 replace all matches
- 8 match whole words only
- 16 match case

forward: set to 1 for forward search.

"mouse":

Inject either a mouse button click, or a mouse move, event. The supported items in `{args}` are:

button: mouse button. The supported values are:

- 0 left mouse button
- 1 middle mouse button
- 2 right mouse button
- 3 mouse button release
- 4 scroll wheel down
- 5 scroll wheel up
- 6 scroll wheel left
- 7 scroll wheel right

row: mouse click row number. The first row of the Vim window is 1 and the last row is `'lines'`.

col: mouse click column number. The maximum value of `{col}` is `'columns'`.

multiclick: set to 1 to inject a multiclick mouse event.

modifiers: key modifiers. The supported values are:

- 4 shift is pressed
- 8 alt is pressed
- 16 ctrl is pressed

move: Optional; if used and TRUE then a mouse move event can be generated.

Only `{args}` row: and col: are used and required; they are interpreted as pixels or screen cells, depending on "cell".

Only results in an event when `'mousemoveevent'` is set or a popup uses mouse move events.

cell: Optional: when present and TRUE then "move"

uses screen cells instead of pixel positions

"scrollbar":

Set or drag the left, right or horizontal scrollbar. Only works when the scrollbar actually exists. The supported items in {args} are:

which: Selects the scrollbar. The supported values are:

left Left scrollbar of the current window  
right Right scrollbar of the current window  
hor Horizontal scrollbar

value: Amount to scroll. For the vertical scrollbars the value can be between 0 to the line-count of the buffer minus one. For the horizontal scrollbar the value can be between 1 and the maximum line length, assuming 'wrap' is not set.

dragging: 1 to drag the scrollbar and 0 to click in the scrollbar.

"key":

Send a low-level keyboard event (e.g. key-up or down). Currently only supported on MS-Windows.

The supported items in {args} are:

event: The supported string values are:  
keyup generate a keyup event  
keydown generate a keydown event

keycode: Keycode to use for a keyup or a keydown event.

E1291

"tabline":

Inject a mouse click event on the tabline to select a tabpage. The supported items in {args} are:

tabnr: tab page number

"tabmenu":

Inject an event to select a tabline menu entry. The supported items in {args} are:

tabnr: tab page number

item: tab page menu item number. 1 for the first menu item, 2 for the second item and so on.

After injecting the GUI events you probably should call `feedkeys()` to have them processed, e.g.:

```
call feedkeys("y", 'Lx!')
```

Returns TRUE if the event is successfully added, FALSE if there is a failure.

Can also be used as a `method` :

```
GetEvent()->test_gui_event({args})
```

`test_ignore_error({expr})`

`test_ignore_error()`

Ignore any error containing {expr}. A normal message is given

instead.

This is only meant to be used in tests, where catching the error with try/catch cannot be used (because it skips over following code).

`{expr}` is used literally, not as a pattern.

When the `{expr}` is the string "RESET" then the list of ignored errors is made empty.

Can also be used as a `method` :

```
GetErrorText()->test_ignore_error()
```

```
test_mswin_event({event}, {args}) test_mswin_event()
Generate a low-level MS-Windows {event} with arguments {args}
for testing Vim functionality. It works for MS-Windows GUI
and for the console.
```

`{event}` is a String and the supported values are:

```
"mouse" mouse event.
"key" keyboard event.
"set_keycode_trans_strategy"
 Change the key translation method.
```

"mouse":

Inject either a mouse button click, or a mouse move, event.  
The supported items in `{args}` are:

```
button: mouse button. The supported values are:
 0 right mouse button
 1 middle mouse button
 2 left mouse button
 3 mouse button release
 4 scroll wheel down
 5 scroll wheel up
 6 scroll wheel left
 7 scroll wheel right
```

```
row: mouse click row number. The first row of the
 Vim window is 1 and the last row is 'lines'.
```

```
col: mouse click column number. The maximum value
 of {col} is 'columns'.
```

**Note:** row and col are always interpreted as screen cells for the console application.

But, they may be interpreted as pixels for the GUI, depending on "cell".

```
multiclick: set to 1 to inject a double-click mouse event.
```

```
modifiers: key modifiers. The supported values are:
```

```
 4 shift is pressed
 8 alt is pressed
 16 ctrl is pressed
```

```
move: Optional; if used and TRUE then a mouse move
 event can be generated.
```

Only `{args}` row: and col: are used and required.

Only results in an event when 'mousemoveevent' is set or a popup uses mouse move events.

cell: Optional for the GUI: when present and TRUE then "move" uses screen cells instead of pixel positions. Not used by the console.

"key":

Send a low-level keyboard event (e.g. keyup or keydown).

The supported items in `{args}` are:

event: The supported string values are:  
    keyup generate a keyup event  
    keydown generate a keydown event

keycode: Keycode to use for a keyup or a keydown event.

modifiers: Optional; key modifiers.  
The supported values are:  
    2 shift is pressed  
    4 ctrl is pressed  
    8 alt is pressed

**Note:** These values are different from the mouse modifiers.

execute: Optional. Similar to `feedkeys()` mode x. When this is included and set to true (non-zero) then Vim will process any buffered unprocessed key events. All other `{args}` items are optional when this is set and true.

"set\_keycode\_trans\_strategy":

[w32-experimental-keycode-trans-strategy](#)

Switch the keycode translation method. The supported methods are:

experimental: The method used after Patch v8.2.4807 using ToUnicode() Win API call.

classic: The method used pre Patch v8.2.4807 using the TranslateMessage() Win API call.

Returns TRUE if the event is successfully added or executed, FALSE if there is a failure.

Can also be used as a [method](#) :

`GetEvent()->test_mswin_event({args})`

`test_null_blob()` [test\\_null\\_blob\(\)](#)  
Return a [Blob](#) that is null. Only useful for testing.

`test_null_channel()` [test\\_null\\_channel\(\)](#)  
Return a [Channel](#) that is null. Only useful for testing.  
[{only available when compiled with the +channel feature}](#)

`test_null_dict()` [test\\_null\\_dict\(\)](#)  
Return a [Dict](#) that is null. Only useful for testing.

`test_null_function()` [test\\_null\\_function\(\)](#)



Return a `Funcref` that is null. Only useful for testing.

`test_null_job()` `test_null_job()`  
Return a `Job` that is null. Only useful for testing.  
{only available when compiled with the `+job` feature}

`test_null_list()` `test_null_list()`  
Return a `List` that is null. Only useful for testing.

`test_null_partial()` `test_null_partial()`  
Return a `Partial` that is null. Only useful for testing.

`test_null_string()` `test_null_string()`  
Return a `String` that is null. Only useful for testing.

`test_option_not_set({name})` `test_option_not_set()`  
Reset the flag that indicates option `{name}` was set. Thus it looks like it still has the default value. Use like this:  
`set ambiwidth=double`  
`call test_option_not_set('ambiwidth')`  
Now the `'ambiwidth'` option behaves like it was never changed, even though the value is "double".  
Only to be used for testing!

Can also be used as a `method` :  
`GetOptionName()->test_option_not_set()`

`test_override({name}, {val})` `test_override()`  
Overrides certain parts of Vim's internal processing to be able to run tests. Only to be used for testing Vim!  
The override is enabled when `{val}` is non-zero and removed when `{val}` is zero.  
Current supported values for `{name}` are:

| <code>{name}</code>         | effect when <code>{val}</code> is non-zero                                                                                 |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>alloc_lines</code>    | make a copy of every buffer line into allocated memory, so that memory access errors can be found by <code>valgrind</code> |
| <code>autoload</code>       | <code>`import autoload`</code> will load the script right away, not postponed until an item is used                        |
| <code>char_avail</code>     | disable the <code>char_avail()</code> function                                                                             |
| <code>nfa_fail</code>       | makes the NFA regexp engine fail to force a fallback to the old engine                                                     |
| <code>no_query_mouse</code> | do not query the mouse position for "dec" terminals                                                                        |
| <code>no_wait_return</code> | set the "no_wait_return" flag. Not restored with "ALL".                                                                    |
| <code>redraw</code>         | disable the <code>redrawing()</code> function                                                                              |

|             |                                                                                          |
|-------------|------------------------------------------------------------------------------------------|
| redraw_flag | ignore the RedrawingDisabled flag                                                        |
| starting    | reset the "starting" variable, see below                                                 |
| term_props  | reset all terminal properties when the version string is detected                        |
| ui_delay    | time in msec to use in ui_delay(); overrules a wait time of up to 3 seconds for messages |
| unreachable | no error for code after <code>:throw</code> and <code>:return</code>                     |
| uptime      | overrules sysinfo.uptime                                                                 |
| vterm_title | setting the window title by a job running in a terminal window                           |
| ALL         | clear all overrides, except alloc_lines ({val} is not used)                              |

"starting" is to be used when a test should behave like startup was done. Since the tests are run by sourcing a script the "starting" variable is non-zero. This is usually a good thing (tests run faster), but sometimes this changes behavior in a way that the test doesn't work properly.

When using:

```
call test_override('starting', 1)
```

The value of "starting" is saved. It is restored by:

```
call test_override('starting', 0)
```

To make sure the flag is reset later using `:defer` can be useful:

```
call test_override('unreachable', 1)
defer call test_override('unreachable', 0)
```

Can also be used as a `method` :

```
GetOverrideVal()-> test_override('starting')
```

test\_refcount({expr}) test\_refcount()

Return the reference count of {expr}. When {expr} is of a type that does not have a reference count, returns -1. Only to be used for testing.

Can also be used as a `method` :

```
GetVarname()->test_refcount()
```

test\_setmouse({row}, {col}) test\_setmouse()

Set the mouse position to be used for the next mouse action. {row} and {col} are one based.

For example:

```
call test_setmouse(4, 20)
call feedkeys("\<LeftMouse>", "xt")
```

test\_settime({expr}) test\_settime()

Set the time Vim uses internally. Currently only used for timestamps in the history, as they are used in viminfo, and for undo.

Using a value of 1 makes Vim not sleep after a warning or

error message.

`{expr}` must evaluate to a number. When the value is zero the normal behavior is restored.

Can also be used as a `method` :

`GetTime()->test_settime()`

`test_srand_seed([seed])` [test\\_srand\\_seed\(\)](#)

When `[seed]` is given this sets the seed value used by ``srand()``. When omitted the test seed is removed.

`test_unknown()` [test\\_unknown\(\)](#)

Return a value with unknown type. Only useful for testing.

`test_void()` [test\\_void\(\)](#)

Return a value with void type. Only useful for testing.

---

### 3. Assert functions

[assert-functions-details](#)

`assert_beeeps({cmd})` [assert\\_beeeps\(\)](#)

Run `{cmd}` and add an error message to `v:errors` if it does NOT produce a beep or visual bell. Also see `assert_fails()`, `assert_nobeep()` and `assert-return`.

Can also be used as a `method` :

`GetCmd()->assert_beeeps()`

`assert_equal({expected}, {actual} [, {msg}])` [assert\\_equal\(\)](#)

When `{expected}` and `{actual}` are not equal an error message is added to `v:errors` and 1 is returned. Otherwise zero is returned. `assert-return`

The error is in the form "Expected `{expected}` but got `{actual}`". When `{msg}` is present it is prefixed to that.

There is no automatic conversion, the String "4" is different from the Number 4. And the number 4 is different from the Float 4.0. The value of `'ignorecase'` is not used here, case always matters.

Example:

```
assert_equal('foo', 'bar')
```

Will result in a string to be added to `v:errors` :

```
test.vim line 12: Expected 'foo' but got 'bar'
```

Can also be used as a `method`, the base is passed as the second argument:

```
mylist->assert_equal([1, 2, 3])
```

`assert_equalfile({fname-one}, {fname-two} [, {msg}])` `assert_equalfile()`  
When the files `{fname-one}` and `{fname-two}` do not contain exactly the same text an error message is added to `v:errors` . Also see `assert-return` .  
When `{fname-one}` or `{fname-two}` does not exist the error will mention that.  
Mainly useful with `terminal-diff` .

Can also be used as a `method` :  
`GetLog()->assert_equalfile('expected.log')`

`assert_exception({error} [, {msg}])` `assert_exception()`  
When `v:exception` does not contain the string `{error}` an error message is added to `v:errors` . Also see `assert-return` .  
This can be used to assert that a command throws an exception. Using the error number, followed by a colon, avoids problems with translations:

```
try
 commandthatfails
 call assert_false(1, 'command should have failed')
catch
 call assert_exception('E492:')
endtry
```

`assert_fails({cmd} [, {error} [, {msg} [, {lnum} [, {context}]]]])` `assert_fails()`  
Run `{cmd}` and add an error message to `v:errors` if it does NOT produce an error or when `{error}` is not found in the error message. Also see `assert-return` .

`E856`

When `{error}` is a string it must be found literally in the first reported error. Most often this will be the error code, including the colon, e.g. "E123:".

```
assert_fails('bad cmd', 'E987:')
```

When `{error}` is a `List` with one or two strings, these are used as patterns. The first pattern is matched against the first reported error:

```
assert_fails('cmd', ['E987:.*expected bool'])
```

The second pattern, if present, is matched against the last reported error.

If there is only one error then both patterns must match. This can be used to check that there is only one error.

To only match the last error use an empty string for the first error:

```
assert_fails('cmd', ['', 'E987:'])
```

If `{msg}` is empty then it is not used. Do this to get the default message when passing the `{lnum}` argument.

`E1115`

When `{lnum}` is present and not negative, and the `{error}` argument is present and matches, then this is compared with the line number at which the error was reported. That can be

the line number in a function or in a script.

E1116

When `{context}` is present it is used as a pattern and matched against the context (script name or function name) where `{lnum}` is located in.

**Note** that beeping is not considered an error, and some failing commands only beep. Use `assert_beeps()` for those.

Can also be used as a **method** :

```
GetCmd()->assert_fails('E99:')
```

`assert_false({actual} [, {msg}])` `assert_false()`

When `{actual}` is not false an error message is added to `v:errors`, like with `assert_equal()`.

The error is in the form "Expected False but got `{actual}`".

When `{msg}` is present it is prepended to that.

Also see `assert-return`.

A value is false when it is zero. When `{actual}` is not a number the assert fails.

Can also be used as a **method** :

```
GetResult()->assert_false()
```

`assert_inrange({lower}, {upper}, {actual} [, {msg}])` `assert_inrange()`

This asserts number and `Float` values. When `{actual}` is lower than `{lower}` or higher than `{upper}` an error message is added to `v:errors`. Also see `assert-return`.

The error is in the form "Expected range `{lower} - {upper}`, but got `{actual}`". When `{msg}` is present it is prefixed to that.

`assert_match({pattern}, {actual} [, {msg}])` `assert_match()`

When `{pattern}` does not match `{actual}` an error message is added to `v:errors`. Also see `assert-return`.

The error is in the form "Pattern `{pattern}` does not match `{actual}`". When `{msg}` is present it is prefixed to that.

`{pattern}` is used as with `=~` : The matching is always done like `'magic'` was set and `'coptions'` is empty, no matter what the actual value of `'magic'` or `'coptions'` is.

`{actual}` is used as a string, automatic conversion applies. Use `"^"` and `"$"` to match with the start and end of the text. Use both to match the whole text.

Example:

```
assert_match('^f.*o$', 'foobar')
```

Will result in a string to be added to `v:errors` :

```
test.vim line 12: Pattern '^f.*o$' does not match 'foobar'
```

Can also be used as a **method** :

`getFile()->assert_match('foo.*')`

`assert_nobeep({cmd})` `assert_nobeep()`  
Run `{cmd}` and add an error message to `v:errors` if it produces a beep or visual bell.  
Also see `assert_beeps()` .

Can also be used as a `method` :  
`GetCmd()->assert_nobeep()`

`assert_notequal({expected}, {actual} [, {msg}])` `assert_notequal()`  
The opposite of `assert_equal()`: add an error message to `v:errors` when `{expected}` and `{actual}` are equal.  
Also see `assert-return` .

Can also be used as a `method` :  
`mylist->assert_notequal([1, 2, 3])`

`assert_notmatch({pattern}, {actual} [, {msg}])` `assert_notmatch()`  
The opposite of `assert_match()`: add an error message to `v:errors` when `{pattern}` matches `{actual}`.  
Also see `assert-return` .

Can also be used as a `method` :  
`getFile()->assert_notmatch('bar.*')`

`assert_report({msg})` `assert_report()`  
Report a test failure directly, using String `{msg}`.  
Always returns one.

Can also be used as a `method` :  
`GetMessage()->assert_report()`

`assert_true({actual} [, {msg}])` `assert_true()`  
When `{actual}` is not true an error message is added to `v:errors` , like with `assert_equal()` .  
Also see `assert-return` .  
A value is TRUE when it is a non-zero number. When `{actual}` is not a number the assert fails.  
When `{msg}` is given it precedes the default message.

Can also be used as a `method` :  
`GetResult()->assert_true()`

Printing

printing

- 1. Introduction print-intro
- 2. Print options print-options
- 3. PostScript Printing postscript-printing
- 4. PostScript Printing Encoding postscript-print-encoding
- 5. PostScript CJK Printing postscript-cjk-printing
- 6. PostScript Printing Troubleshooting postscript-print-trouble
- 7. PostScript Utilities postscript-print-util
- 8. Formfeed Characters printing-formfeed

{only available when compiled with the |+printer| feature}

=====

1. Introduction print-intro

On MS-Windows Vim can print your text on any installed printer. On other systems a PostScript file is produced. This can be directly sent to a PostScript printer. For other printers a program like ghostscript needs to be used.

**Note:** If you have problems printing with `:hardcopy`, an alternative is to use `:TOhtml` and print the resulting html file from a browser.

`:ha` `:hardcopy` E237 E238 E324

`:[range]ha[rdcopy][!]` [*arguments*]

Send [*range*] lines (default whole file) to the printer.

On MS-Windows a dialog is displayed to allow selection of printer, paper size etc. To skip the dialog, use the `[!]`. In this case the printer defined by `'printdevice'` is used, or, if `'printdevice'` is empty, the system default printer.

For systems other than MS-Windows, PostScript is written in a temp file and `'printexpr'` is used to actually print it. Then [*arguments*] can be used by `'printexpr'` through `v:cmdarg`. Otherwise [*arguments*] is ignored. `'printoptions'` can be used to specify paper size, duplex, etc.

**Note:** If you want PDF, there are tools such as "ps2pdf" that can convert the PostScript to PDF.

`:[range]ha[rdcopy][!]` >{filename}

As above, but write the resulting PostScript in file {filename}.

Things like "%" are expanded `cmdline-special`

Careful: An existing file is silently overwritten.  
{only available when compiled with the `+postscript`  
feature}  
On MS-Windows use the "print to file" feature of the  
printer driver.

Progress is displayed during printing as a page number and a percentage. To abort printing use the interrupt key (CTRL-C or, on MS-systems, **CTRL-Break**).

Printer output is controlled by the `'printfont'` and `'printoptions'` options. `'printhead'` specifies the format of a page header.

The printed file is always limited to the selected margins, irrespective of the current window's `'wrap'` or `'linebreak'` settings. The "wrap" item in `'printoptions'` can be used to switch wrapping off.

The current highlighting colors are used in the printout, with the following considerations:

- 1) The normal background is always rendered as white (i.e. blank paper).
- 2) White text or the default foreground is rendered as black, so that it shows up!
- 3) If `'background'` is "dark", then the colours are darkened to compensate for the fact that otherwise they would be too bright to show up clearly on white paper.

---

## 2. Print options print-options

Here are the details for the options that change the way printing is done. For generic info about setting options see `options.txt`.

`'printdevice'` `'pdev'` string (default empty) pdev-option  
global

This defines the name of the printer to be used when the `:hardcopy` command is issued with a bang (!) to skip the printer selection dialog. On Win32, it should be the printer name exactly as it appears in the standard printer dialog.

If the option is empty, then vim will use the system default printer for `":hardcopy!"`

`'printencoding'` `'penc'` String (default empty, except for: penc-option E620  
MS-Windows: cp1252,  
Macintosh: mac-roman,  
VMS: dec-mcs,  
HPUX: hp-roman8,  
EBCDIC: ebcdic-uk)  
global

Sets the character encoding used when printing. This option tells Vim which print character encoding file from the "print" directory in `'runtimepath'` to use.

This option will accept any value from `encoding-names`. Any recognized names are converted to Vim standard names - see `'encoding'` for more details. Names



not recognized by Vim will just be converted to lower case and underscores replaced with '-' signs.

If **'printencoding'** is empty or Vim cannot find the file then it will use **'encoding'** (if it is set an 8-bit encoding) to find the print character encoding file. If Vim is unable to find a character encoding file then it will use the "latin1" print character encoding file.

When **'encoding'** is set to a multibyte encoding, Vim will try to convert characters to the printing encoding for printing (if **'printencoding'** is empty then the conversion will be to latin1). Conversion to a printing encoding other than latin1 will require Vim to be compiled with the **+iconv** feature. If no conversion is possible then printing will fail. Any characters that cannot be converted will be replaced with upside down question marks.

Four print character encoding files are provided to support default Mac, VMS, HPUNIX, and EBCDIC character encodings and are used by default on these platforms. Code page 1252 print character encoding is used by default on MS-Windows platform.

**'printexpr' 'pexpr'** String (default: see below) pexpr-option  
global

Expression that is evaluated to print the PostScript produced with **:hardcopy** .

The file name to be printed is in **v:fname\_in** .

The arguments to the **:hardcopy** command are in **v:cmdarg** .

The expression must take care of deleting the file after printing it.

When there is an error, the expression must return a non-zero number.

If there is no error, return zero or an empty string.

The default for non MS-Windows or VMS systems is to simply use "lpr" to print the file:

```
system('lpr' .. (&printdevice == '' ? '' : '-P' .. &printdevice)
.. ' ' .. v:fname_in) .. delete(v:fname_in) + v:shell_error
```

On MS-Windows machines the default is to copy the file to the currently specified printdevice:

```
system('copy' .. ' ' .. v:fname_in .. (&printdevice == ''
? ' LPT1:' : (' \"' .. &printdevice .. '\"')))
.. delete(v:fname_in)
```

On VMS machines the default is to send the file to either the default or currently specified printdevice:

```
system('print' .. (&printdevice == '' ? '' : '/queue=' ..
&printdevice) .. ' ' .. v:fname_in) .. delete(v:fname_in)
```

If you change this option, using a function is an easy way to avoid having to escape all the spaces. Example:

```
:set printexpr=PrintFile()
:function PrintFile()
```

```

: call system("ghostview " .. v:fname_in)
: call delete(v:fname_in)
: return v:shell_error
:endifunc

```

It is more efficient if the option is set to just a function call, see [expr-option-function](#) .

Be aware that some print programs return control before they have read the file. If you delete the file too soon it will not be printed. These programs usually offer an option to have them remove the file when printing is done.

[E365](#)

If evaluating the expression fails or it results in a non-zero number, you get an error message. In that case Vim will delete the file. In the default value for non-MS-Windows a trick is used: Adding "v:shell\_error" will result in a non-zero number when the system() call fails.

If the expression starts with s: or <SID> , then it is replaced with the script ID ( [local-function](#) ). Example:

```

set printexpr=s:MyPrintFile()
set printexpr=<SID>SomePrintFile()

```

Otherwise, the expression is evaluated in the context of the script where the option was set, thus script-local items are available.

This option cannot be set from a [modeline](#) or in the [sandbox](#) , for security reasons.

[pfn-option](#) [E613](#)

```

'printfont' 'pfn' string (default "courier")
 global

```

This is the name of the font that will be used for the [:hardcopy](#) command's output. It has the same format as the '[guifont](#)' option, except that only one font may be named, and the special "guifont=\*" syntax is not available.

In the Win32 GUI version this specifies a font name with its extra attributes, as with the '[guifont](#)' option.

For other systems, only ":h11" is recognized, where "11" is the point size of the font. When omitted, the point size is 10.

[pheader-option](#)

```

'printhead' 'pheader' string (default "%<%f%h%m%=Page %N")
 global

```

This defines the format of the header produced in [:hardcopy](#) output. The option is defined in the same way as the '[statusline](#)' option. If Vim has not been compiled with the [+statusline](#) feature, this option has no effect and a simple default header is used, which shows the page number. The same simple header is used when this option is empty.

[pmbcs-option](#)

```

'printmbcharset' 'pmbcs' string (default "")
 global

```

Sets the CJK character set to be used when generating CJK output from [:hardcopy](#) . The following predefined values are currently recognised by Vim:

|                          | Value        | Description                                    |
|--------------------------|--------------|------------------------------------------------|
| Chinese<br>(Simplified)  | GB_2312-80   |                                                |
|                          | GBT_12345-90 |                                                |
|                          | MAC          | Apple Mac Simplified Chinese                   |
|                          | GBT-90_MAC   | GB/T 12345-90 Apple Mac Simplified Chinese     |
|                          | GBK          | GBK (GB 13000.1-93)                            |
| Chinese<br>(Traditional) | ISO10646     | ISO 10646-1:1993                               |
|                          | CNS_1993     | CNS 11643-1993, Planes 1 & 2                   |
|                          | BIG5         |                                                |
|                          | ETEN         | Big5 with ETen extensions                      |
| Japanese                 | ISO10646     | ISO 10646-1:1993                               |
|                          | JIS_C_1978   |                                                |
|                          | JIS_X_1983   |                                                |
|                          | JIS_X_1990   |                                                |
|                          | MSWINDOWS    | Win3.1/95J (JIS X 1997 + NEC + IBM extensions) |
|                          | KANJITALK6   | Apple Mac KanjiTalk V6.x                       |
| Korean                   | KANJITALK7   | Apple Mac KanjiTalk V7.x                       |
|                          | KS_X_1992    |                                                |
|                          | MAC          | Apple Macintosh Korean                         |
|                          | MSWINDOWS    | KS X 1992 with MS extensions                   |
|                          | ISO10646     | ISO 10646-1:1993                               |

Only certain combinations of the above values and **'printencoding'** are possible. The following tables show the valid combinations:

|                          |              | eu-cn | gbk   | ucs-2 | utf-8 |
|--------------------------|--------------|-------|-------|-------|-------|
| Chinese<br>(Simplified)  | GB_2312-80   | x     |       |       |       |
|                          | GBT_12345-90 | x     |       |       |       |
|                          | MAC          | x     |       |       |       |
|                          | GBT-90_MAC   | x     |       |       |       |
|                          | GBK          |       | x     |       |       |
|                          | ISO10646     |       |       | x     | x     |
| Chinese<br>(Traditional) | CNS_1993     | x     |       |       |       |
|                          | BIG5         |       | x     |       |       |
|                          | ETEN         |       | x     |       |       |
|                          | ISO10646     |       |       | x     | x     |
| Japanese                 | JIS_C_1978   | x     | x     |       |       |
|                          | JIS_X_1983   | x     | x     |       |       |
|                          | JIS_X_1990   | x     |       | x     | x     |
|                          | MSWINDOWS    |       | x     |       |       |
|                          | KANJITALK6   |       | x     |       |       |
|                          | KANJITALK7   |       | x     |       |       |
|                          |              |       |       |       |       |
|                          |              | eu-kr | cp949 | ucs-2 | utf-8 |

|        |           |   |   |   |   |
|--------|-----------|---|---|---|---|
| Korean | KS_X_1992 | x |   |   |   |
|        | MAC       | x |   |   |   |
|        | MSWINDOWS |   | x |   |   |
|        | ISO10646  |   |   | x | x |

To set up the correct encoding and character set for printing some Japanese text you would do the following;

```
:set printencoding=euc-jp
:set printmbcharset=JIS_X_1983
```

If **'printmbcharset'** is not one of the above values then it is assumed to specify a custom multibyte character set and no check will be made that it is compatible with the value for **'printencoding'**. Vim will look for a file defining the character set in the "print" directory in **'runtimepath'**.

pmbfn-option

**'printmbfont'** **'pmbfn'** string (default "")  
global

This is a comma-separated list of fields for font names to be used when generating CJK output from **:hardcopy**. Each font name has to be preceded with a letter indicating the style the font is to be used for as follows:

```
r:{font-name} font to use for normal characters
b:{font-name} font to use for bold characters
i:{font-name} font to use for italic characters
o:{font-name} font to use for bold-italic characters
```

A field with the r: prefix must be specified when doing CJK printing. The other fontname specifiers are optional. If a specifier is missing then another font will be used as follows:

```
if b: is missing, then use r:
if i: is missing, then use r:
if o: is missing, then use b:
```

Some CJK fonts do not contain characters for codes in the ASCII code range. Also, some characters in the CJK ASCII code ranges differ in a few code points from traditional ASCII characters. There are two additional fields to control printing of characters in the ASCII code range.

```
c:yes Use Courier font for characters in the ASCII
c:no (default) code range.

a:yes Use ASCII character set for codes in the ASCII
a:no (default) code range.
```

The following is an example of specifying two multibyte fonts, one for normal and italic printing and one for bold and bold-italic printing, and using Courier to print codes in the ASCII code range but using the national character set:

```
:set printmbfont=r:WadaMin-Regular,b:WadaMin-Bold,c:yes
```

popt-option

**'printoptions'** **'popt'** string (default "")

## global

This is a comma-separated list of items that control the format of the output of `:hardcopy` :

|                                    |                                                                                                                                                                                                                     |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>left:{spec}</code>           | left margin (default: 10pc)                                                                                                                                                                                         |
| <code>right:{spec}</code>          | right margin (default: 5pc)                                                                                                                                                                                         |
| <code>top:{spec}</code>            | top margin (default: 5pc)                                                                                                                                                                                           |
| <code>bottom:{spec}</code>         | bottom margin (default: 5pc)                                                                                                                                                                                        |
|                                    | <code>{spec}</code> is a number followed by "in" for inches, "pt" for points (1 point is 1/72 of an inch), "mm" for millimeters or "pc" for a percentage of the media size.                                         |
|                                    | Weird example:<br><code>left:2in,top:30pt,right:16mm,bottom:3pc</code>                                                                                                                                              |
|                                    | If the unit is not recognized there is no error and the default value is used.                                                                                                                                      |
| <code>header:{nr}</code>           | Number of lines to reserve for the header.<br>Only the first line is actually filled, thus when <code>{nr}</code> is 2 there is one empty line. The header is formatted according to ' <code>printhheader</code> '. |
| <code>header:0</code>              | Do not print a header.                                                                                                                                                                                              |
| <code>header:2</code> (default)    | Use two lines for the header                                                                                                                                                                                        |
| <code>syntax:n</code>              | Do not use syntax highlighting. This is faster and thus useful when printing large files.                                                                                                                           |
| <code>syntax:y</code>              | Do syntax highlighting.                                                                                                                                                                                             |
| <code>syntax:a</code> (default)    | Use syntax highlighting if the printer appears to be able to print color or grey.                                                                                                                                   |
| <code>number:y</code>              | Include line numbers in the printed output.                                                                                                                                                                         |
| <code>number:n</code> (default)    | No line numbers.                                                                                                                                                                                                    |
| <code>wrap:y</code> (default)      | Wrap long lines.                                                                                                                                                                                                    |
| <code>wrap:n</code>                | Truncate long lines.                                                                                                                                                                                                |
| <code>duplex:off</code>            | Print on one side.                                                                                                                                                                                                  |
| <code>duplex:long</code> (default) | Print on both sides (when possible), bind on long side.                                                                                                                                                             |
| <code>duplex:short</code>          | Print on both sides (when possible), bind on short side.                                                                                                                                                            |
| <code>collate:y</code> (default)   | Collating: 1 2 3, 1 2 3, 1 2 3                                                                                                                                                                                      |
| <code>collate:n</code>             | No collating: 1 1 1, 2 2 2, 3 3 3                                                                                                                                                                                   |
| <code>jobsplit:n</code> (default)  | Do all copies in one print job                                                                                                                                                                                      |
| <code>jobsplit:y</code>            | Do each copy as a separate print job. Useful when doing N-up postprocessing.                                                                                                                                        |
| <code>portrait:y</code> (default)  | Orientation is portrait.                                                                                                                                                                                            |
| <code>portrait:n</code>            | Orientation is landscape.                                                                                                                                                                                           |
| <code>paper:A4</code> (default)    | Paper size: A4                                                                                                                                                                                                      |
| <code>paper:{name}</code>          | Paper size from this table:                                                                                                                                                                                         |

a4 letter

| {name}    | size in cm    | size in inch  |
|-----------|---------------|---------------|
| 10x14     | 25.4 x 35.57  | 10 x 14       |
| A3        | 29.7 x 42     | 11.69 x 16.54 |
| A4        | 21 x 29.7     | 8.27 x 11.69  |
| A5        | 14.8 x 21     | 5.83 x 8.27   |
| B4        | 25 x 35.3     | 10.12 x 14.33 |
| B5        | 17.6 x 25     | 7.17 x 10.12  |
| executive | 18.42 x 26.67 | 7.25 x 10.5   |
| folio     | 21 x 33       | 8.27 x 13     |
| ledger    | 43.13 x 27.96 | 17 x 11       |
| legal     | 21.59 x 35.57 | 8.5 x 14      |
| letter    | 21.59 x 27.96 | 8.5 x 11      |
| quarto    | 21.59 x 27.5  | 8.5 x 10.83   |
| statement | 13.97 x 21.59 | 5.5 x 8.5     |
| tabloid   | 27.96 x 43.13 | 11 x 17       |

- formfeed:n (default) Treat form feed characters (0x0c) as a normal print character.
- formfeed:y When a form feed character is encountered, continue printing of the current line at the beginning of the first line on a new page.

The item indicated with (default) is used when the item is not present. The values are not always used, especially when using a dialog to select the printer and options.

Example:

```
:set printoptions=paper:letter,duplex:off
```

### 3. PostScript Printing

postscript-printing  
E455 E456 E457 E624

Provided you have enough disk space there should be no problems generating a PostScript file. You need to have the runtime files correctly installed (if you can find the help files, they probably are).

There are currently a number of limitations with PostScript printing:

- **'printfont'** - The font name is ignored (the Courier family is always used - it should be available on all PostScript printers) but the font size is used.
- **'printoptions'** - The duplex setting is used when generating PostScript output, but it is up to the printer to take notice of the setting. If the printer does not support duplex printing then it should be silently ignored. Some printers, however, don't print at all.
- 8-bit support - While a number of 8-bit print character encodings are supported it is possible that some characters will not print. Whether a character will print depends on the font in the printer knowing the character. Missing characters will be replaced with an upside down question mark, or a space if that character is also not known by the font. It may be possible to get all the characters in an encoding to print by installing a new version of the Courier font family.

- Multi-byte support - Currently Vim will try to convert multibyte characters to the 8-bit encoding specified by **'printencoding'** (or latin1 if it is empty). Any characters that are not successfully converted are shown as unknown characters. Printing will fail if Vim cannot convert the multibyte to the 8-bit encoding.

---

#### 4. Custom 8-bit Print Character Encodings postscript-print-encoding E618 E619

To use your own print character encoding when printing 8-bit character data you need to define your own PostScript font encoding vector. Details on how to define a font encoding vector is beyond the scope of this help file, but you can find details in the PostScript Language Reference Manual, 3rd Edition, published by Addison-Wesley and available in PDF form at <http://www.adobe.com/>. The following describes what you need to do for Vim to locate and use your print character encoding.

- Decide on a unique name for your encoding vector, one that does not clash with any of the recognized or standard encoding names that Vim uses (see [encoding-names](#) for a list), and that no one else is likely to use.
- Copy \$VIMRUNTIME/print/latin1.ps to the print subdirectory in your **'runtimepath'** and rename it with your unique name.
- Edit your renamed copy of latin1.ps, replacing all occurrences of latin1 with your unique name (don't forget the line starting %%Title:), and modify the array of glyph names to define your new encoding vector. The array must have exactly 256 entries or you will not be able to print!
- Within Vim, set **'printencoding'** to your unique encoding name and then print your file. Vim will now use your custom print character encoding.

Vim will report an error with the resource file if you change the order or content of the first 3 lines, other than the name of the encoding on the line starting %%Title: or the version number on the line starting %%Version:.

[Technical explanation for those that know PostScript - Vim looks for a file with the same name as the encoding it will use when printing. The file defines a new PostScript Encoding resource called /VIM-name, where name is the print character encoding Vim will use.]

---

#### 5. PostScript CJK Printing postscript-cjk-printing E673 E674 E675

Vim supports printing of Chinese, Japanese, and Korean files. Setting up Vim to correctly print CJK files requires setting up a few more options.

Each of these countries has many standard character sets and encodings which require that both be specified when printing. In addition, CJK fonts normally do not have the concept of italic glyphs and use different weight or stroke style to achieve emphasis when printing. This in turn requires a different approach to specifying fonts to use when printing.

The encoding and character set are specified with the **'printencoding'** and **'printmbcharset'** options. If **'printencoding'** is not specified then **'encoding'** is used as normal. If **'printencoding'** is specified then characters will be

translated to this encoding for printing. You should ensure that the encoding is compatible with the character set needed for the file contents or some characters may not appear when printed.

The fonts to use for CJK printing are specified with `'printmbfont'`. This option allows you to specify different fonts to use when printing characters which are syntax highlighted with the font styles normal, italic, bold and bold-italic.

No CJK fonts are supplied with Vim. There are some free Korean, Japanese, and Traditional Chinese fonts available at:

<http://examples.oreilly.com/cjkvinfo/adobe/samples/>

You can find descriptions of the various fonts in the read me file at

[http://examples.oreilly.de/english\\_examples/cjkvinfo/adobe/00README](http://examples.oreilly.de/english_examples/cjkvinfo/adobe/00README)

Please read your printer documentation on how to install new fonts.

CJK fonts can be large containing several thousand glyphs, and it is not uncommon to find that they only contain a subset of a national standard. It is not unusual to find the fonts to not include characters for codes in the ASCII code range. If you find half-width Roman characters are not appearing in your printout then you should configure Vim to use the Courier font the half-width ASCII characters with `'printmbfont'`. If your font does not include other characters then you will need to find another font that does.

Another issue with ASCII characters, is that the various national character sets specify a couple of different glyphs in the ASCII code range. If you print ASCII text using the national character set you may see some unexpected characters. If you want true ASCII code printing then you need to configure Vim to output ASCII characters for the ASCII code range with `'printmbfont'`.

It is possible to define your own multibyte character set although this should not be attempted lightly. A discussion on the process is beyond the scope of these help files. You can find details on CMap (character map) files in the document 'Adobe CMap and CIDFont Files Specification, Version 1.0', available from <http://www.adobe.com> as a PDF file.

---

## 6. PostScript Printing Troubleshooting

`postscript-print-trouble`

E621

Usually the only sign of a problem when printing with PostScript is that your printout does not appear. If you are lucky you may get a printed page that tells you the PostScript operator that generated the error that prevented the print job completing.

There are a number of possible causes as to why the printing may have failed:

- Wrong version of the prolog resource file. The prolog resource file contains some PostScript that Vim needs to be able to print. Each version of Vim needs one particular version. Make sure you have correctly installed the runtime files, and don't have any old versions of a file called prolog



in the print directory in your `'runtimepath'` directory.

- Paper size. Some PostScript printers will abort printing a file if they do not support the requested paper size. By default Vim uses A4 paper. Find out what size paper your printer normally uses and set the appropriate paper size with `'printoptions'`. If you cannot find the name of the paper used, measure a sheet and compare it with the table of supported paper sizes listed for `'printoptions'`, using the paper that is closest in both width AND height. **Note:** The dimensions of actual paper may vary slightly from the ones listed. If there is no paper listed close enough, then you may want to try `psresize` from PSUtils, discussed below.
- Two-sided printing (duplex). Normally a PostScript printer that does not support two-sided printing will ignore any request to do it. However, some printers may abort the job altogether. Try printing with duplex turned off. **Note:** Duplex prints can be achieved manually using PS utils - see below.
- Collated printing. As with Duplex printing, most PostScript printers that do not support collating printouts will ignore a request to do so. Some may not. Try printing with collation turned off.
- Syntax highlighting. Some print management code may prevent the generated PostScript file from being printed on a black and white printer when syntax highlighting is turned on, even if solid black is the only color used. Try printing with syntax highlighting turned off.

A safe `printoptions` setting to try is:

```
:set printoptions=paper:A4,duplex:off,collate:n,syntax:n
```

Replace "A4" with the paper size that best matches your printer paper.

---

## 7. PostScript Utilities postscript-print-util

### 7.1 Ghostscript

Ghostscript is a PostScript and PDF interpreter that can be used to display and print on non-PostScript printers PostScript and PDF files. It can also generate PDF files from PostScript.

Ghostscript will run on a wide variety of platforms.

There are three available versions:

- AFPL Ghostscript (formerly Aladdin Ghostscript) which is free for non-commercial use. It can be obtained from:

<http://www.cs.wisc.edu/~ghost/>

- GNU Ghostscript which is available under the GNU General Public License. It can be obtained from:

<ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/gnu/>

- A commercial version for inclusion in commercial products.

Additional information on Ghostscript can also be found at:

<http://www.ghostscript.com/>

Support for a number of non PostScript printers is provided in the distribution as standard, but if you cannot find support for your printer check the Ghostscript site for other printers not included by default.

## 7.2 Ghostscript Previewers.

The interface to Ghostscript is very primitive so a number of graphical front ends have been created. These allow easier PostScript file selection, previewing at different zoom levels, and printing. Check supplied documentation for full details.

X11

- Ghostview. Obtainable from:

<http://www.cs.wisc.edu/~ghost/gv/>

- gv. Derived from Ghostview. Obtainable from:

<http://wwwthep.physik.uni-mainz.de/~plass/gv/>

Copies (possibly not the most recent) can be found at:

<http://www.cs.wisc.edu/~ghost/gv/>

OpenVMS

- Is apparently supported in the main code now (untested). See:

<http://wwwthep.physik.uni-mainz.de/~plass/gv/>

MS-Windows

- GSview. Obtainable from:

<http://www.cs.wisc.edu/~ghost/gsview/>

Linux

- GSview. Linux version of the popular MS-Windows previewer. Obtainable from:

<http://www.cs.wisc.edu/~ghost/gsview/>

- BMV. Different from Ghostview and gv in that it doesn't use X but svgalib. Obtainable from:

<ftp://sunsite.unc.edu/pub/Linux/apps/graphics/viewers/svgabmv-1.2.tgz>

### 7.3 PSUtils

PSUtils is a collection of utility programs for manipulating PostScript documents. Binary distributions are available for many platforms, as well as the full source. PSUtils can be found at:

<http://knackered.org/angus/psutils>

The utilities of interest include:

- psnup. Convert PS files for N-up printing.
- psselect. Select page range and order of printing.
- psresize. Change the page size.
- psbook. Reorder and lay out pages ready for making a book.

The output of one program can be used as the input to the next, allowing for complex print document creation.

#### N-UP PRINTING

The psnup utility takes an existing PostScript file generated from Vim and convert it to an n-up version. The simplest way to create a 2-up printout is to first create a PostScript file with:

```
:hardcopy > test.ps
```

Then on your command line execute:

```
psnup -n 2 test.ps final.ps
```

**Note:** You may get warnings from some Ghostscript previewers for files produced by psnup - these may safely be ignored.

Finally print the file final.ps to your PostScript printer with your platform's print command. (You will need to delete the two PostScript files afterwards yourself.) **'printexpr'** could be modified to perform this extra step before printing.

#### ALTERNATE DUPLEX PRINTING

It is possible to achieve a poor man's version of duplex printing using the PS utility psselect. This utility has options -e and -o for printing just the even or odd pages of a PS file respectively.

First generate a PS file with the **'hardcopy'** command, then generate new files with all the odd and even numbered pages with:

```
psselect -o test.ps odd.ps
```

```
psselect -e test.ps even.ps
```

Next print odd.ps with your platform's normal print command. Then take the print output, turn it over and place it back in the paper feeder. Now print even.ps with your platform's print command. All the even pages should now appear on the back of the odd pages.

There are a couple of points to bear in mind:

1. Position of the first page. If the first page is on top of the printout when printing the odd pages then you need to reverse the order that the odd pages are printed. This can be done with the `-r` option to `psselect`. This will ensure page 2 is printed on the back of page 1.  
**Note:** it is better to reverse the odd numbered pages rather than the even numbered in case there are an odd number of pages in the original PS file.
2. Paper flipping. When turning over the paper with the odd pages printed on them you may have to either flip them horizontally (along the long edge) or vertically (along the short edge), as well as possibly rotating them 180 degrees. All this depends on the printer - it will be more obvious for desktop ink jets than for small office laser printers where the paper path is hidden from view.

---

## 8. Formfeed Characters

printing-formfeed

By default Vim does not do any special processing of formfeed control characters. Setting the `'printoptions'` formfeed item will make Vim recognize formfeed characters and continue printing the current line at the beginning of the first line on a new page. The use of formfeed characters provides rudimentary print control but there are certain things to be aware of.

Vim will always start printing a line (including a line number if enabled) containing a formfeed character, even if it is the first character on the line. This means if a line starting with a formfeed character is the first line of a page then Vim will print a blank page.

Since the line number is printed at the start of printing the line containing the formfeed character, the remainder of the line printed on the new page will not have a line number printed for it (in the same way as the wrapped lines of a long line when wrap in `'printoptions'` is enabled).

If the formfeed character is the last character on a line, then printing will continue on the second line of the new page, not the first. This is due to Vim processing the end of the line after the formfeed character and moving down a line to continue printing.

Due to the points made above it is recommended that when formfeed character processing is enabled, printing of line numbers is disabled, and that form feed characters are not the last character on a line. Even then you may need to adjust the number of lines before a formfeed character to prevent accidental blank pages.





## Vim client-server communication

## client-server

- 1. Common functionality `clientserver`
- 2. X11 specific items `x11-clientserver`
- 3. MS-Windows specific items `w32-clientserver`

## 1. Common functionality

## clientserver

When compiled with the `+clientserver` option, Vim can act as a command server. It accepts messages from a client and executes them. At the same time, Vim can function as a client and send commands to a Vim server.

The following command line arguments are available:

| argument                                              | meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>--remote [+{cmd}] {file} ...</code>             | <code>--remote</code><br>Open the file list in a remote Vim. When there is no Vim server, execute locally. There is one optional init command: <code>+{cmd}</code> . This must be an Ex command that can be followed by <code> </code> . The rest of the command line is taken as the file list. Thus any non-file arguments must come before this. You cannot edit stdin this way <code>--</code> . The remote Vim is raised. If you don't want this use<br><code>vim --remote-send "&lt;C-\&gt;&lt;C-N&gt;:n filename&lt;CR&gt;"</code> |
| <code>--remote-silent [+{cmd}] {file} ...</code>      | <code>--remote-silent</code><br>As above, but don't complain if there is no server and the file is edited locally.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>--remote-wait [+{cmd}] {file} ...</code>        | <code>--remote-wait</code><br>As <code>--remote</code> , but wait for files to complete (unload) in remote Vim.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>--remote-wait-silent [+{cmd}] {file} ...</code> | <code>--remote-wait-silent</code><br>As <code>--remote-wait</code> , but don't complain if there is no server.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>--remote-tab</code>                             | <code>--remote-tab</code><br>Like <code>--remote</code> but open each file in a new tabpage.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>--remote-tab-silent</code>                      | <code>--remote-tab-silent</code><br>Like <code>--remote-silent</code> but open each file in a new tabpage.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>--remote-tab-wait</code>                        | <code>--remote-tab-wait</code><br>Like <code>--remote-wait</code> but open each file in a new                                                                                                                                                                                                                                                                                                                                                                                                                                             |

tabpage.

|                                       |                                                                                                                                                                                                                       |                                       |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
|                                       |                                                                                                                                                                                                                       | <code>--remote-tab-wait-silent</code> |
| <code>--remote-tab-wait-silent</code> | Like <code>--remote-wait-silent</code> but open each file in a new tabpage.                                                                                                                                           |                                       |
|                                       |                                                                                                                                                                                                                       | <code>--servername</code>             |
| <code>--servername {name}</code>      | Become the server <code>{name}</code> . When used together with one of the <code>--remote</code> commands: connect to server <code>{name}</code> instead of the default (see below). The name used will be uppercase. |                                       |
|                                       |                                                                                                                                                                                                                       | <code>--remote-send</code>            |
| <code>--remote-send {keys}</code>     | Send <code>{keys}</code> to server and exit. The <code>{keys}</code> are not mapped. Special key names are recognized, e.g., " <code>&lt;CR&gt;</code> " results in a CR character.                                   |                                       |
|                                       |                                                                                                                                                                                                                       | <code>--remote-expr</code>            |
| <code>--remote-expr {expr}</code>     | Evaluate <code>{expr}</code> in server and print the result on stdout.                                                                                                                                                |                                       |
|                                       |                                                                                                                                                                                                                       | <code>--serverlist</code>             |
| <code>--serverlist</code>             | Output a list of server names.                                                                                                                                                                                        |                                       |

## Examples

Edit "file.txt" in an already running GVIM server:

```
gvim --remote file.txt
```

Edit "file.txt" in an already running server called FOOBAR:

```
gvim --servername FOOBAR --remote file.txt
```

Edit "file.txt" in server "FILES" if it exists, become server "FILES" otherwise:

```
gvim --servername FILES --remote-silent file.txt
```

This doesn't work, all arguments after `--remote` will be used as file names:

```
gvim --remote --servername FOOBAR file.txt
```

Edit file "+foo" in a remote server (note the use of "./" to avoid the special meaning of the leading plus):

```
vim --remote ./+foo
```

Tell the remote server "BLA" to write all files and exit:

```
vim --servername BLA --remote-send '<C-\><C-N>:wqa<CR>'
```

## SERVER NAME

`client-server-name`

By default Vim will try to register the name under which it was invoked (`gvim`, `gvim ...`). This can be overridden with the `--servername` argument. If the specified name is not available, a postfix is applied until a free name is encountered, i.e. "gvim1" for the second invocation of `gvim` on a particular X-server. The resulting name is available in the `servername` builtin variable `v:servername`. The case of the server name is ignored, thus "gvim" and "GVIM" are considered equal.



When Vim is invoked with `--remote`, `--remote-wait` or `--remote-send` it will try to locate the server name determined by the invocation name and `--servername` argument as described above. If an exact match is not available, the first server with the number postfix will be used. If a name with the number postfix is specified with the `--servername` argument, it must match exactly.

If no server can be located and `--remote` or `--remote-wait` was used, Vim will start up according to the rest of the command line and do the editing by itself. This way it is not necessary to know whether `gvim` is already started when sending command to it.

The `--serverlist` argument will cause Vim to print a list of registered command servers on the standard output (stdout) and exit.

The `{server}` argument is used by several functions. When this is an empty string then on Unix the default server name is used, which is "GVIM". On MS-Windows an empty string does not work.

Win32 **Note:** Making the Vim server go to the foreground doesn't always work, because MS-Windows doesn't allow it. The client will move the server to the foreground when using the `--remote` or `--remote-wait` argument and the server name starts with "g".

## REMOTE EDITING

The `--remote` argument will cause a `:drop` command to be constructed from the rest of the command line and sent as described above.

The `--remote-wait` argument does the same thing and additionally sets up to wait for each of the files to have been edited. This uses the `BufUnload` event, thus as soon as a file has been unloaded, Vim assumes you are done editing it.

**Note** that the `--remote` and `--remote-wait` arguments will consume the rest of the command line. I.e. all remaining arguments will be regarded as filenames. You can not put options there!

## FUNCTIONS

There are a number of Vim functions for scripting the command server. See the description in `builtin.txt` or use **CTRL-]** on the function name to jump to the full explanation.

### synopsis

```
remote_startserver(name)
remote_expr(server, string, idvar)
remote_send(server, string, idvar)
serverlist()
remote_peek(serverid, retvar)
remote_read(serverid)
server2client(serverid, string)
remote_foreground(server)
```

### explanation

```
run a server
send expression
send key sequence
get a list of available servers
check for reply string
read reply string
send reply string
bring server to the front
```

See also the explanation of `CTRL-\_CTRL-N` . Very useful as a leading key sequence.

The `{serverid}` for `server2client()` can be obtained with `expand("<client>")`

---

## 2. X11 specific items

`x11-clientserver`

`E247` `E248` `E251` `E258` `E277`

The communication between client and server goes through the X server. The display of the Vim server must be specified. The usual protection of the X server is used, you must be able to open a window on the X server for the communication to work. It is possible to communicate between different systems.

By default, a GUI Vim will register a name on the X-server by which it can be addressed for subsequent execution of injected strings. Vim can also act as a client and send strings to other instances of Vim on the same X11 display.

When an X11 GUI Vim (`gvim`) is started, it will try to register a send-server name on the 'VimRegistry' property on the root window.

A non GUI Vim with access to the X11 display ( `xterm-clipboard` enabled), can also act as a command server if a server name is explicitly given with the `--servername` argument, or when Vim was built with the `+autoservername` feature.

An empty `--servername` argument will cause the command server to be disabled.

To send commands to a Vim server from another application, read the source file `src/if_xcmdsrv.c`, it contains some hints about the protocol used.

---

## 3. Win32 specific items

`w32-clientserver`

Every Win32 Vim can work as a server, also in the console. You do not need a version compiled with OLE. Windows messages are used, this works on any version of MS-Windows. But only communication within one system is possible.

Since MS-Windows messages are used, any other application should be able to communicate with a Vim server. An alternative is using the OLE functionality `ole-interface` .

When using `gvim`, the `--remote-wait` only works properly this way:

```
start /w gvim --remote-wait file.txt
```

## Terminal information

[terminal-info](#)

Vim uses information about the terminal you are using to fill the screen and recognize what keys you hit. If this information is not correct, the screen may be messed up or keys may not be recognized. The actions which have to be performed on the screen are accomplished by outputting a string of characters. Special keys produce a string of characters. These strings are stored in the terminal options, see [terminal-options](#) .

**NOTE:** Most of this is not used when running the [GUI](#) .

- |                            |                                    |
|----------------------------|------------------------------------|
| 1. Startup                 | <a href="#">startup-terminal</a>   |
| 2. Terminal options        | <a href="#">terminal-options</a>   |
| 3. Window size             | <a href="#">window-size</a>        |
| 4. Slow and fast terminals | <a href="#">slow-fast-terminal</a> |
| 5. Using the mouse         | <a href="#">mouse-using</a>        |

## =====

## 1. Startup

[startup-terminal](#)

When Vim is started a default terminal type is assumed. For the Amiga this is a standard CLI window, for MS-Windows the pc terminal, for Unix an ansi terminal. A few other terminal types are always available, see below [builtin-terms](#) .

You can give the terminal name with the '-T' Vim argument. If it is not given Vim will try to get the name from the TERM environment variable.

[termcap](#) [terminfo](#) [E557](#) [E558](#) [E559](#)

On Unix the terminfo database or termcap file is used. This is referred to as "termcap" in all the documentation. At compile time, when running configure, the choice whether to use terminfo or termcap is done automatically. When running Vim the output of ":version" will show [+terminfo](#) if terminfo is used. Also see [xterm-screens](#) .

On non-Unix systems a termcap is only available if Vim was compiled with TERMCAP defined.

[builtin-terms](#) [builtin\\_terms](#)

A number of builtin terminals are available. Since patch 9.0.0280 there is no difference between Vim versions. You can see a list of available builtin terminals in the error message you get for `:set term=xxx` (when not running the GUI). Also see [++builtin\\_terms](#) .

If the termcap code is included Vim will try to get the strings for the terminal you are using from the termcap file and the builtin termcaps. Both are always used, if an entry for the terminal you are using is present. Which one is used first depends on the ['ttybuiltin'](#) option:

|                               |                     |                     |
|-------------------------------|---------------------|---------------------|
| <code>'ttybuiltin'</code> on  | 1: builtin termcap  | 2: external termcap |
| <code>'ttybuiltin'</code> off | 1: external termcap | 2: builtin termcap  |

If an option is missing in one of them, it will be obtained from the other one. If an option is present in both, the one first encountered is used.

Which external termcap file is used varies from system to system and may depend on the environment variables "TERMCAP" and "TERMPATH". See "man tgetent".

Settings depending on terminal

[term-dependent-settings](#)

If you want to set options or mappings, depending on the terminal name, you can do this best in your .vimrc. Example:

```
if &term == "xterm"
 ... xterm maps and settings ...
elseif &term =~ "vt10."
 ... vt100, vt102 maps and settings ...
endif
```

[raw-terminal-mode](#)

For normal editing the terminal will be put into "raw" mode. The strings defined with `'t_ti'`, `'t_TI'` and `'t_ks'` will be sent to the terminal. Normally this puts the terminal in a state where the termcap codes are valid and activates the cursor and function keys.

When Vim exits the terminal will be put back into the mode it was before Vim started. The strings defined with `'t_te'`, `'t_TE'` and `'t_ke'` will be sent to the terminal. On the Amiga, with commands that execute an external command (e.g., "!!"), the terminal will be put into Normal mode for a moment. This means that you can stop the output to the screen by hitting a printing key. Output resumes when you hit `<BS>`.

**Note:** When `'t_ti'` is not empty, Vim assumes that it causes switching to the alternate screen. This may slightly change what happens when executing a shell command or exiting Vim. To avoid this use `'t_TI'` and `'t_TE'` (but make sure to add to them, not overwrite).

Vim will try to detect what keyboard protocol the terminal is using with the `'t_RK'` termcap entry. This is sent after `'t_TI'`, but only when there is no work to do (no typeahead and no pending commands). That is to avoid the response to end up in a shell command or arrive after Vim exits.

[xterm-bracketed-paste](#)

When the `'t_BE'` option is set then `'t_BE'` will be sent to the terminal when entering "raw" mode and `'t_BD'` when leaving "raw" mode. The terminal is then expected to put `'t_PS'` before pasted text and `'t_PE'` after pasted text. This way Vim can separate text that is pasted from characters that are typed. The pasted text is handled like when the middle mouse button is used, it is inserted literally and not interpreted as commands.

Please **note:** while bracketed paste is trying to prevent nasty side-effects from pasting (like the `CTRL-C` or `<ESC>` key), it's not a guaranteed security

measure because different terminals may implement this mode slightly differently. You should still be careful with what you paste into Vim.

When the cursor is in the first column, the pasted text will be inserted before it. Otherwise the pasted text is appended after the cursor position. This means one cannot paste after the first column. Unfortunately Vim does not have a way to tell where the mouse pointer was.

**Note** that in some situations Vim will not recognize the bracketed paste and you will get the raw text. In other situations Vim will only get the first pasted character and drop the rest, e.g. when using the "r" command. If you have a problem with this, disable bracketed paste by putting this in your .vimrc:

```
set t_BE=
```

If this is done while Vim is running the '**t\_BD**' will be sent to the terminal to disable bracketed paste.

If **t\_PS** or **t\_PE** is not set, then **t\_BE** will not be used. This is to make sure that bracketed paste is not enabled when the escape codes surrounding pasted text cannot be recognized.

**Note:** bracketed paste mode will be disabled, when the '**esckey**s' option is not set (also when the '**compatible**' option is set).

If your terminal supports bracketed paste, but the options are not set automatically, you can try using something like this:

```
if &term =~ "screen"
 let &t_BE = "\e[?2004h"
 let &t_BD = "\e[?2004l"
 exec "set t_PS=\e[200~"
 exec "set t_PE=\e[201~"
endif
```

The terminfo entries "BE", "BD", "PS" and "PE" were added in ncurses version 6.4, early 2023, for some terminals. If you have this version then you may not have to manually configure your terminal.

#### tmux-integration

If you experience issues when running Vim inside tmux, here are a few hints. You can comment-out parts if something doesn't work (it may depend on the terminal that tmux is running in):

```
if !has('gui_running') && &term =~ '^%(screen|tmux\)'
 " Better mouse support, see :help 'ttymouse'
 set ttymouse=sgr

 " Enable true colors, see :help xterm-true-color
 let &termguicolors = v:true
 let &t_8f = "\<Esc>[38;2;%lu;%lu;%lum"
 let &t_8b = "\<Esc>[48;2;%lu;%lu;%lum"

 " Enable bracketed paste mode, see :help xterm-bracketed-paste
 let &t_BE = "\<Esc>[?2004h"
```

```

let &t_BD = "\<Esc>[?2004]"
let &t_PS = "\<Esc>[200~"
let &t_PE = "\<Esc>[201~"

" Enable focus event tracking, see :help xterm-focus-event
let &t_fe = "\<Esc>[?1004h"
let &t_fd = "\<Esc>[?1004l"
execute "set <FocusGained>=\<Esc>[I"
execute "set <FocusLost>=\<Esc>[O"

" Enable modified arrow keys, see :help arrow_modifiers
execute "silent! set <xUp>=\<Esc>[O;A"
execute "silent! set <xDown>=\<Esc>[O;B"
execute "silent! set <xRight>=\<Esc>[O;C"
execute "silent! set <xLeft>=\<Esc>[O;D"
endif

```

### cs7-problem

**Note:** If the terminal settings are changed after running Vim, you might have an illegal combination of settings. This has been reported on Solaris 2.5 with "stty cs8 parenb", which is restored as "stty cs7 parenb". Use "stty cs8 -parenb -istrip" instead, this is restored correctly.

Some termcap entries are wrong in the sense that after sending '**t\_ks**' the cursor keys send codes different from the codes defined in the termcap. To avoid this you can set '**t\_ks**' (and '**t\_ke**') to empty strings. This must be done during initialization (see [initialization](#)), otherwise it's too late.

Some termcap entries assume that the highest bit is always reset. For example: The cursor-up entry for the Amiga could be ":ku=\E[A:". But the Amiga really sends "\233A". This works fine if the highest bit is reset, e.g., when using an Amiga over a serial line. If the cursor keys don't work, try the entry ":ku=\233A:".

Some termcap entries have the entry ":ku=\E[A:". But the Amiga really sends "\233A". On output "\E[" and "\233" are often equivalent, on input they aren't. You will have to change the termcap entry, or change the key code with the :set command to fix this.

Many cursor key codes start with an **<Esc>**. Vim must find out if this is a single hit of the **<Esc>** key or the start of a cursor key sequence. It waits for a next character to arrive. If it does not arrive within one second a single **<Esc>** is assumed. On very slow systems this may fail, causing cursor keys not to work sometimes. If you discover this problem reset the '**timeout**' option. Vim will wait for the next character to arrive after an **<Esc>**. If you want to enter a single **<Esc>** you must type it twice. Resetting the '**esckey**' option avoids this problem in Insert mode, but you lose the possibility to use cursor and function keys in Insert mode.

On the Amiga the recognition of window resizing is activated only when the terminal name is "amiga" or "builtin\_amiga".

Some terminals have confusing codes for the cursor keys. The televideo 925 is such a terminal. It sends a **CTRL-H** for cursor-left. This would make it

impossible to distinguish a backspace and cursor-left. To avoid this problem **CTRL-H** is never recognized as cursor-left.

#### vt100-cursor-keys    xterm-cursor-keys

Other terminals (e.g., vt100 and xterm) have cursor keys that send `<Esc>OA`, `<Esc>OB`, etc. Unfortunately these are valid commands in insert mode: Stop insert, Open a new line above the new one, start inserting 'A', 'B', etc. Instead of performing these commands Vim will erroneously recognize this typed key sequence as a cursor key movement. To avoid this and make Vim do what you want in either case you could use these settings:

```
:set notimeout " don't timeout on mappings
:set ttimeout " do timeout on terminal key codes
:set timeoutlen=100 " timeout after 100 msec
```

This requires the key-codes to be sent within 100 msec in order to recognize them as a cursor key. When you type you normally are not that fast, so they are recognized as individual typed commands, even though Vim receives the same sequence of bytes.

#### vt100-function-keys    xterm-function-keys

An xterm can send function keys F1 to F4 in two modes: vt100 compatible or not. Because Vim may not know what the xterm is sending, both types of keys are recognized. The same happens for the `<Home>` and `<End>` keys.

|                           |                   | normal                       |                            | vt100                      |                                  |
|---------------------------|-------------------|------------------------------|----------------------------|----------------------------|----------------------------------|
| <code>&lt;F1&gt;</code>   | <code>t_k1</code> | <code>&lt;Esc&gt;[11~</code> | <code>&lt;xF1&gt;</code>   | <code>&lt;Esc&gt;OP</code> | <code>&lt;xF1&gt;-xterm</code>   |
| <code>&lt;F2&gt;</code>   | <code>t_k2</code> | <code>&lt;Esc&gt;[12~</code> | <code>&lt;xF2&gt;</code>   | <code>&lt;Esc&gt;OQ</code> | <code>&lt;xF2&gt;-xterm</code>   |
| <code>&lt;F3&gt;</code>   | <code>t_k3</code> | <code>&lt;Esc&gt;[13~</code> | <code>&lt;xF3&gt;</code>   | <code>&lt;Esc&gt;OR</code> | <code>&lt;xF3&gt;-xterm</code>   |
| <code>&lt;F4&gt;</code>   | <code>t_k4</code> | <code>&lt;Esc&gt;[14~</code> | <code>&lt;xF4&gt;</code>   | <code>&lt;Esc&gt;OS</code> | <code>&lt;xF4&gt;-xterm</code>   |
| <code>&lt;Home&gt;</code> | <code>t_kh</code> | <code>&lt;Esc&gt;[7~</code>  | <code>&lt;xHome&gt;</code> | <code>&lt;Esc&gt;OH</code> | <code>&lt;xHome&gt;-xterm</code> |
| <code>&lt;End&gt;</code>  | <code>t_@7</code> | <code>&lt;Esc&gt;[4~</code>  | <code>&lt;xEnd&gt;</code>  | <code>&lt;Esc&gt;OF</code> | <code>&lt;xEnd&gt;-xterm</code>  |

When Vim starts, `<xF1>` is mapped to `<F1>`, `<xF2>` to `<F2>` etc. This means that by default both codes do the same thing. If you make a mapping for `<xF2>`, because your terminal does have two keys, the default mapping is overwritten, thus you can use the `<F2>` and `<xF2>` keys for something different.

#### xterm-shifted-keys

Newer versions of xterm support shifted function keys and special keys. Vim recognizes most of them. Use `":set termcap"` to check which are supported and what the codes are. Mostly these are not in a termcap, they are only supported by the builtin\_xterm termcap.

#### xterm-modifier-keys

Newer versions of xterm support Alt and Ctrl for most function keys. To avoid having to add all combinations of Alt, Ctrl and Shift for every key a special sequence is recognized at the end of a termcap entry: `";*X"`. The "X" can be any character, often `'~'` is used. The `";*"` stands for an optional modifier argument. `";2"` is Shift, `";3"` is Alt, `";5"` is Ctrl and `";9"` is Meta (when it's different from Alt). They can be combined. Examples:

```
:set <F8>=^[[19;*~
:set <Home>=^[[1;*H
```

Another speciality about these codes is that they are not overwritten by another code. That is to avoid that the codes obtained from xterm directly `t_RV` overwrite them.

Another special value is a termcap entry ending in "@;\*X". This is for cursor keys, which either use "CSI X" or "CSI 1 ; modifier X". Thus the "@" stands for either "1" if a modifier follows, or nothing.

#### arrow\_modifiers

Several terminal emulators (alacritty, gnome, konsole, etc.) send special codes for keys with modifiers, but these do not have an entry in the termcap/terminfo database. You can make them work by adding a few lines in your vimrc. For example, to make the Control modifier work with arrow keys for the gnome terminal:

```
if &term =~ 'gnome'
 execute "set <xUp>=\<Esc>[@;*A"
 execute "set <xDown>=\<Esc>[@;*B"
 execute "set <xRight>=\<Esc>[@;*C"
 execute "set <xLeft>=\<Esc>[@;*D"
endif
```

#### xterm-scroll-region

The default termcap entry for xterm on Sun and other platforms does not contain the entry for scroll regions. Add ":cs=\E[%i%d;%dr:" to the xterm entry in /etc/termcap and everything should work.

#### xterm-end-home-keys

On some systems (at least on FreeBSD with XFree86 3.1.2) the codes that the <End> and <Home> keys send contain a <Null> character. To make these keys send the proper key code, add these lines to your ~/.Xdefaults file:

```
*VT100.Translations: #override \n\
 <Key>Home: string("\0x1b") string("[7~") \n\
 <Key>End: string("\0x1b") string("[8~")
```

#### xterm-8bit xterm-8-bit

Xterm can be run in a mode where it uses 8-bit escape sequences. The CSI code is used instead of <Esc>[. The advantage is that an <Esc> can quickly be recognized in Insert mode, because it can't be confused with the start of a special key.

For the builtin termcap entries, Vim checks if the 'term' option contains "8bit" anywhere. It then uses 8-bit characters for the termcap entries, the mouse and a few other things. You would normally set \$TERM in your shell to "xterm-8bit" and Vim picks this up and adjusts to the 8-bit setting automatically.

When Vim receives a response to the t\_RV (request version) sequence and it starts with CSI, it assumes that the terminal is in 8-bit mode and will convert all key sequences to their 8-bit variants.

#### xterm-terminfo-entries

For some time the terminfo entries were insufficient to describe all the features that Vim can use. The builtin xterm termcap entries did have these, with the result that several terminals that were similar enough to xterm took advantage of these by prefixing "xterm-" to the terminal name in \$TERM.

This leads to problems, because quite often these terminals are not 100% compatible with xterm. At the start of 2023 several entries have been added to the terminfo database to make it possible to use these features without using the "xterm" workaround. These are the relevant entries (so far):



| name | xterm value                               | description                                    |
|------|-------------------------------------------|------------------------------------------------|
| RV   | "\033[>c"                                 | Request version <code>t_RV</code>              |
| BE   | "\033[?2004h"                             | enable bracketed paste mode <code>t_BE</code>  |
| BD   | "\033[?2004l"                             | disable bracketed paste mode <code>t_BD</code> |
| PS   | "\033[200~"                               | pasted text start <code>t_PS</code>            |
| PE   | "\033[201~"                               | pasted text end <code>t_PE</code>              |
| XM   | "\033[?1006;1004;1000%?%p1%{1}%=%th%el%;" | mouse enable / disable <code>t_XM</code>       |
| FE   | "\033[?1004h"                             | enable focus event tracking <code>t_fe</code>  |
| FD   | "\033[?1004l"                             | disable focus event tracking <code>t_fd</code> |

The "XM" entry includes "1006" to enable SGR style mouse reporting. This supports columns above 223. It also includes "1004" which enables focus reporting.

**Note:** As of 2023, the "1004" is currently not used by Vim itself, instead it is recommended to set focus reporting independently of mouse tracking by the `t_fe` and `t_fd` entries, as ncurses also starts to use with the latest versions (and will then also end up in terminfo/termcap).

### xterm-kitty    kitty-terminal

The Kitty terminal is a special case. Mainly because it works differently from most other terminals, but also because, instead of trying to fit in and make it behave like other terminals by default, it dictates how applications need to work when using Kitty. This makes it very difficult for Vim to work in a Kitty terminal. Some exceptions have been hard coded, but it is not at all nice to have to make exceptions for one specific terminal.

One of the problems is that the value for \$TERM is set to "xterm-kitty". For Vim this is an indication that the terminal is xterm-compatible and the builtin xterm termcap entries should be used. Many other terminals depend on this. However, Kitty is not fully xterm compatible. The author suggested to ignore the "xterm-" prefix and use the terminfo entry anyway, so that is what happens now, the builtin xterm termcap entries are not used. However, the `t_RV` is set, otherwise other things would not work, such as automatically setting `'tymouse'` to "sgr" (at least until `t_XM` is being used for this).

It is not clear why kitty sets \$TERM to "xterm-kitty", the terminal isn't really xterm compatible. "kitty" would be more appropriate, but a terminfo entry with that name is not widespread.

**Note** that using the kitty keyboard protocol is a separate feature, see [kitty-keyboard-protocol](#) .

---

## 2. Terminal options terminal-options    termcap-options    E436

The terminal options can be set just like normal options. But they are not shown with the `":set all"` command. Instead use `":set termcap"`.

It is always possible to change individual strings by setting the appropriate option. For example:

```
:set t_ce=^V^[K (CTRL-V, <Esc>, [, K)
```

The options are listed below. The associated termcap code is always equal to the last two characters of the option name. Only one termcap code is required: Cursor motion, 't\_cm'.

The options 't\_da', 't\_db', 't\_ms', 't\_xs', 't\_xn' represent flags in the termcap. When the termcap flag is present, the option will be set to "y". But any non-empty string means that the flag is set. An empty string means that the flag is not set. 't\_CS' works like this too, but it isn't a termcap flag.

## OUTPUT CODES

terminal-output-codes

option meaning

|      |                                                            |           |        |
|------|------------------------------------------------------------|-----------|--------|
| t_AB | set background color (ANSI)                                | t_AB      | 't_AB' |
| t_AF | set foreground color (ANSI)                                | t_AF      | 't_AF' |
| t_AL | add number of blank lines                                  | t_AL      | 't_AL' |
| t_al | add new blank line                                         | t_al      | 't_al' |
| t_bc | backspace character                                        | t_bc      | 't_bc' |
| t_cd | clear to end of screen                                     | t_cd      | 't_cd' |
| t_ce | clear to end of line                                       | t_ce      | 't_ce' |
| t_cl | clear screen                                               | t_cl      | 't_cl' |
| t_cm | cursor motion (required!)                                  | E437 t_cm | 't_cm' |
| t_Co | number of colors                                           | t_Co      | 't_Co' |
| t_CS | if non-empty, cursor relative to scroll region             | t_CS      | 't_CS' |
| t_cs | define scrolling region                                    | t_cs      | 't_cs' |
| t_CV | define vertical scrolling region                           | t_CV      | 't_CV' |
| t_da | if non-empty, lines from above scroll down                 | t_da      | 't_da' |
| t_db | if non-empty, lines from below scroll up                   | t_db      | 't_db' |
| t_DL | delete number of lines                                     | t_DL      | 't_DL' |
| t_dl | delete line                                                | t_dl      | 't_dl' |
| t_fs | set window title end (from status line)                    | t_fs      | 't_fs' |
| t_ke | exit "keypad transmit" mode                                | t_ke      | 't_ke' |
| t_ks | start "keypad transmit" mode                               | t_ks      | 't_ks' |
| t_le | move cursor one char left                                  | t_le      | 't_le' |
| t_mb | blinking mode                                              | t_mb      | 't_mb' |
| t_md | bold mode                                                  | t_md      | 't_md' |
| t_me | Normal mode (undoes t_mr, t_mb, t_md and color)            | t_me      | 't_me' |
| t_mr | reverse (invert) mode                                      | t_mr      | 't_mr' |
| t_ms |                                                            | t_ms      | 't_ms' |
| t_ms | if non-empty, cursor can be moved in standout/inverse mode |           |        |
| t_nd | non destructive space character                            | t_nd      | 't_nd' |
| t_op | reset to original color pair                               | t_op      | 't_op' |
| t_RI | cursor number of chars right                               | t_RI      | 't_RI' |
| t_Sb | set background color                                       | t_Sb      | 't_Sb' |
| t_Sf | set foreground color                                       | t_Sf      | 't_Sf' |
| t_se | standout end                                               | t_se      | 't_se' |
| t_so | standout mode                                              | t_so      | 't_so' |
| t_sr | scroll reverse (backward)                                  | t_sr      | 't_sr' |
| t_te | end of "termcap" mode                                      | t_te      | 't_te' |
| t_ti | put terminal into "termcap" mode                           | t_ti      | 't_ti' |
| t_ts | set window title start (to status line)                    | t_ts      | 't_ts' |
| t_ue | underline end                                              | t_ue      | 't_ue' |

|      |                                                                                    |      |        |
|------|------------------------------------------------------------------------------------|------|--------|
| t_us | underline mode                                                                     | t_us | 't_us' |
| t_ut | clearing uses the current background color                                         | t_ut | 't_ut' |
| t_vb | visual bell                                                                        | t_vb | 't_vb' |
| t_ve | cursor visible                                                                     | t_ve | 't_ve' |
| t_vi | cursor invisible                                                                   | t_vi | 't_vi' |
| t_vs | cursor very visible (blink)                                                        | t_vs | 't_vs' |
|      |                                                                                    | t_xs | 't_xs' |
| t_xs | if non-empty, standout not erased by overwriting (hpterm)                          |      |        |
|      |                                                                                    | t_xn | 't_xn' |
| t_xn | if non-empty, writing a character at the last screen cell does not cause scrolling |      |        |
| t_ZH | italics mode                                                                       | t_ZH | 't_ZH' |
| t_ZR | italics end                                                                        | t_ZR | 't_ZR' |

Added by Vim (there are no standard codes for these):

|      |                                                                                                                                                                                                                           |      |        |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|--------|
| t_AU | set underline color (ANSI)                                                                                                                                                                                                | t_AU | 't_AU' |
| t_Ce | undercurl and underline end                                                                                                                                                                                               | t_Ce | 't_Ce' |
| t-Cs | undercurl (curly underline) mode                                                                                                                                                                                          | t-Cs | 't-Cs' |
| t_CF | set alternate font (using index 0 - 10)                                                                                                                                                                                   | t_CF | 't_CF' |
| t_Us | double underline mode                                                                                                                                                                                                     | t_Us | 't_Us' |
| t_ds | dotted underline mode                                                                                                                                                                                                     | t_ds | 't_ds' |
| t_Ds | dashed underline mode                                                                                                                                                                                                     | t_Ds | 't_Ds' |
| t_Te | strikethrough end                                                                                                                                                                                                         | t_Te | 't_Te' |
| t_Ts | strikethrough mode                                                                                                                                                                                                        | t_Ts | 't_Ts' |
| t_IS | set icon text start                                                                                                                                                                                                       | t_IS | 't_IS' |
| t_IE | set icon text end                                                                                                                                                                                                         | t_IE | 't_IE' |
| t_WP | set window position (Y, X) in pixels                                                                                                                                                                                      | t_WP | 't_WP' |
| t_GP | get window position (Y, X) in pixels                                                                                                                                                                                      | t_GP | 't_GP' |
| t_WS | set window size (height, width in cells)                                                                                                                                                                                  | t_WS | 't_WS' |
| t_VS | cursor normally visible (no blink)                                                                                                                                                                                        | t_VS | 't_VS' |
| t_SI | start insert mode (bar cursor shape)                                                                                                                                                                                      | t_SI | 't_SI' |
| t_SR | start replace mode (underline cursor shape)                                                                                                                                                                               | t_SR | 't_SR' |
| t_EI | end insert or replace mode (block cursor shape)                                                                                                                                                                           | t_EI | 't_EI' |
|      | <a href="#">termcap-cursor-shape</a>                                                                                                                                                                                      |      |        |
| t_RV | request terminal version string (works for xterm and other terminal emulators) The response is stored in <a href="#">v:termresponse</a> <a href="#">xterm-8bit</a> <a href="#">'ttymouse'</a> <a href="#">xterm-codes</a> | t_RV | 't_RV' |
| t_XM | enable/disable mouse reporting, see <a href="#">mouse-reporting</a> below                                                                                                                                                 | t_XM | 't_XM' |
| t_RK | request terminal keyboard protocol state; sent after <a href="#">t_TI</a>                                                                                                                                                 | t_RK | 't_RK' |
| t_u7 | request cursor position (for xterm) see <a href="#">'ambiwidth'</a>                                                                                                                                                       | t_u7 | 't_u7' |
|      | The response is stored in <a href="#">v:termu7resp</a>                                                                                                                                                                    |      |        |
| t_RF | request terminal foreground color                                                                                                                                                                                         | t_RF | 't_RF' |
|      | The response is stored in <a href="#">v:termrfgresp</a>                                                                                                                                                                   |      |        |
| t_RB | request terminal background color                                                                                                                                                                                         | t_RB | 't_RB' |
|      | The response is stored in <a href="#">v:termrbgresp</a>                                                                                                                                                                   |      |        |
| t_8f | set foreground color (R, G, B)                                                                                                                                                                                            | t_8f | 't_8f' |
|      | <a href="#">xterm-true-color</a>                                                                                                                                                                                          |      |        |
| t_8b | set background color (R, G, B)                                                                                                                                                                                            | t_8b | 't_8b' |
|      | <a href="#">xterm-true-color</a>                                                                                                                                                                                          |      |        |
| t_8u | set underline color (R, G, B)                                                                                                                                                                                             | t_8u | 't_8u' |

|      |                                                                                            |      |        |
|------|--------------------------------------------------------------------------------------------|------|--------|
| t_BE | enable bracketed paste mode<br><code>xterm-bracketed-paste</code>                          | t_BE | 't_BE' |
| t_BD | disable bracketed paste mode<br><code>xterm-bracketed-paste</code>                         | t_BD | 't_BD' |
| t_SC | set cursor color start                                                                     | t_SC | 't_SC' |
| t_EC | set cursor color end                                                                       | t_EC | 't_EC' |
| t_SH | set cursor shape                                                                           | t_SH | 't_SH' |
| t_RC | request terminal cursor blinking<br>The response is stored in <code>v:termblinkresp</code> | t_RC | 't_RC' |
| t_RS | request terminal cursor style<br>The response is stored in <code>v:termstyleresp</code>    | t_RS | 't_RS' |
| t_ST | save window title to stack                                                                 | t_ST | 't_ST' |
| t_RT | restore window title from stack                                                            | t_RT | 't_RT' |
| t_Si | save icon text to stack                                                                    | t_Si | 't_Si' |
| t_Ri | restore icon text from stack                                                               | t_Ri | 't_Ri' |
| t_TE | end of "raw" mode                                                                          | t_TE | 't_TE' |
| t_TI | put terminal into "raw" mode                                                               | t_TI | 't_TI' |
| t_fe | enable focus-event tracking<br><code>xterm-focus-event</code>                              | t_fe | 't_fe' |
| t_fd | disable focus-event tracking<br><code>xterm-focus-event</code>                             | t_fd | 't_fd' |

Some codes have a start, middle and end part. The start and end are defined by the termcap option, the middle part is text.

```
set title text: t_ts {title text} t_fs
set icon text: t_IS {icon text} t_IE
set cursor color: t_SC {color name} t_EC
```

t\_SH must take one argument:

```
0, 1 or none blinking block cursor
2 block cursor
3 blinking underline cursor
4 underline cursor
5 blinking vertical bar cursor
6 vertical bar cursor
```

t\_RS is sent only if the response to t\_RV has been received. It is not used on Mac OS when Terminal.app could be recognized from the termresponse.

### mouse-reporting

Many terminals can report mouse clicks and some can report mouse movement and dragging. Vim needs to know what codes are being used for this.

The "XM" terminfo/termcap entry is used for this. Vim also has the **'ttymouse'** option to specify the mouse protocol being used. See the option for the possible values.

If Vim can read the "XM" terminfo/termcap entry then it will be used for enabling and disabling the mouse reporting. If it is missing, then the value from **'ttymouse'** is used to decide how to do this.

If the "XM" entry exists and the first number is "1006" then **'ttymouse'** will be set to "sgr", unless it was already set earlier.

## KEY CODES

Note: Use the <> form if possible

terminal-key-codes

| option | name      | meaning                |                   |
|--------|-----------|------------------------|-------------------|
| t_ku   | <Up>      | arrow up               | t_ku 't_ku'       |
| t_kd   | <Down>    | arrow down             | t_kd 't_kd'       |
| t_kr   | <Right>   | arrow right            | t_kr 't_kr'       |
| t_kl   | <Left>    | arrow left             | t_kl 't_kl'       |
|        | <xUp>     | alternate arrow up     | <xUp>             |
|        | <xDown>   | alternate arrow down   | <xDown>           |
|        | <xRight>  | alternate arrow right  | <xRight>          |
|        | <xLeft>   | alternate arrow left   | <xLeft>           |
|        | <S-Up>    | shift arrow up         |                   |
|        | <S-Down>  | shift arrow down       |                   |
| t_%i   | <S-Right> | shift arrow right      | t_%i 't_%i'       |
| t_#4   | <S-Left>  | shift arrow left       | t_#4 't_#4'       |
| t_k1   | <F1>      | function key 1         | t_k1 't_k1'       |
|        | <xF1>     | alternate F1           | <xF1>             |
| t_k2   | <F2>      | function key 2         | <F2> t_k2 't_k2'  |
|        | <xF2>     | alternate F2           | <xF2>             |
| t_k3   | <F3>      | function key 3         | <F3> t_k3 't_k3'  |
|        | <xF3>     | alternate F3           | <xF3>             |
| t_k4   | <F4>      | function key 4         | <F4> t_k4 't_k4'  |
|        | <xF4>     | alternate F4           | <xF4>             |
| t_k5   | <F5>      | function key 5         | <F5> t_k5 't_k5'  |
| t_k6   | <F6>      | function key 6         | <F6> t_k6 't_k6'  |
| t_k7   | <F7>      | function key 7         | <F7> t_k7 't_k7'  |
| t_k8   | <F8>      | function key 8         | <F8> t_k8 't_k8'  |
| t_k9   | <F9>      | function key 9         | <F9> t_k9 't_k9'  |
| t_k;   | <F10>     | function key 10        | <F10> t_k; 't_k;' |
| t_F1   | <F11>     | function key 11        | <F11> t_F1 't_F1' |
| t_F2   | <F12>     | function key 12        | <F12> t_F2 't_F2' |
| t_F3   | <F13>     | function key 13        | <F13> t_F3 't_F3' |
| t_F4   | <F14>     | function key 14        | <F14> t_F4 't_F4' |
| t_F5   | <F15>     | function key 15        | <F15> t_F5 't_F5' |
| t_F6   | <F16>     | function key 16        | <F16> t_F6 't_F6' |
| t_F7   | <F17>     | function key 17        | <F17> t_F7 't_F7' |
| t_F8   | <F18>     | function key 18        | <F18> t_F8 't_F8' |
| t_F9   | <F19>     | function key 19        | <F19> t_F9 't_F9' |
|        | <S-F1>    | shifted function key 1 |                   |
|        | <S-xF1>   | alternate <S-F1>       | <S-xF1>           |
|        | <S-F2>    | shifted function key 2 | <S-F2>            |
|        | <S-xF2>   | alternate <S-F2>       | <S-xF2>           |
|        | <S-F3>    | shifted function key 3 | <S-F3>            |
|        | <S-xF3>   | alternate <S-F3>       | <S-xF3>           |
|        | <S-F4>    | shifted function key 4 | <S-F4>            |
|        | <S-xF4>   | alternate <S-F4>       | <S-xF4>           |
|        | <S-F5>    | shifted function key 5 | <S-F5>            |
|        | <S-F6>    | shifted function key 6 | <S-F6>            |
|        | <S-F7>    | shifted function key 7 | <S-F7>            |
|        | <S-F8>    | shifted function key 8 | <S-F8>            |
|        | <S-F9>    | shifted function key 9 | <S-F9>            |

|      |               |                                                   |                   |
|------|---------------|---------------------------------------------------|-------------------|
|      | <S-F10>       | shifted function key 10                           | <S-F10>           |
|      | <S-F11>       | shifted function key 11                           | <S-F11>           |
|      | <S-F12>       | shifted function key 12                           | <S-F12>           |
| t_%1 | <Help>        | help key                                          | t_%1 't_%1'       |
| t_&8 | <Undo>        | undo key                                          | t_&8 't_&8'       |
| t_kI | <Insert>      | insert key                                        | t_kI 't_kI'       |
|      | <kInsert>     | keypad insert key                                 |                   |
| t_kD | <Del>         | delete key                                        | t_kD 't_kD'       |
| t_kb | <BS>          | backspace key                                     | t_kb 't_kb'       |
| t_kB | <S-Tab>       | back-tab (shift-tab) <S-Tab>                      | t_kB 't_kB'       |
| t_kh | <Home>        | home key                                          | t_kh 't_kh'       |
| t_#2 | <S-Home>      | shifted home key <S-Home>                         | t_#2 't_#2'       |
|      | <xHome>       | alternate home key                                | <xHome>           |
| t_@7 | <End>         | end key                                           | t_@7 't_@7'       |
| t_*7 | <S-End>       | shifted end key <S-End>                           | t_star7 't_star7' |
|      | <xEnd>        | alternate end key                                 | <xEnd>            |
| t_kP | <PageUp>      | page-up key                                       | t_kP 't_kP'       |
| t_kN | <PageDown>    | page-down key                                     | t_kN 't_kN'       |
| t_K1 | <kHome>       | keypad home key                                   | t_K1 't_K1'       |
| t_K4 | <kEnd>        | keypad end key                                    | t_K4 't_K4'       |
| t_K3 | <kPageUp>     | keypad page-up key                                | t_K3 't_K3'       |
| t_K5 | <kPageDown>   | keypad page-down key                              | t_K5 't_K5'       |
| t_K6 | <kPlus>       | keypad plus key <kPlus>                           | t_K6 't_K6'       |
| t_K7 | <kMinus>      | keypad minus key <kMinus>                         | t_K7 't_K7'       |
| t_K8 | <kDivide>     | keypad divide <kDivide>                           | t_K8 't_K8'       |
| t_K9 | <kMultiply>   | keypad multiply <kMultiply>                       | t_K9 't_K9'       |
| t_KA | <kEnter>      | keypad enter key <kEnter>                         | t_KA 't_KA'       |
| t_KB | <kPoint>      | keypad decimal point <kPoint>                     | t_KB 't_KB'       |
| t_KC | <k0>          | keypad 0 <k0>                                     | t_KC 't_KC'       |
| t_KD | <k1>          | keypad 1 <k1>                                     | t_KD 't_KD'       |
| t_KE | <k2>          | keypad 2 <k2>                                     | t_KE 't_KE'       |
| t_KF | <k3>          | keypad 3 <k3>                                     | t_KF 't_KF'       |
| t_KG | <k4>          | keypad 4 <k4>                                     | t_KG 't_KG'       |
| t_KH | <k5>          | keypad 5 <k5>                                     | t_KH 't_KH'       |
| t_KI | <k6>          | keypad 6 <k6>                                     | t_KI 't_KI'       |
| t_KJ | <k7>          | keypad 7 <k7>                                     | t_KJ 't_KJ'       |
| t_KK | <k8>          | keypad 8 <k8>                                     | t_KK 't_KK'       |
| t_KL | <k9>          | keypad 9 <k9>                                     | t_KL 't_KL'       |
|      | <Mouse>       | leader of mouse code                              | <Mouse>           |
| t_PS | <PasteStart>  | start of bracketed paste<br>xterm-bracketed-paste | t_PS 't_PS'       |
| t_PE | <PasteEnd>    | end of bracketed paste<br>xterm-bracketed-paste   | t_PE 't_PE'       |
|      | <FocusGained> | Vim window got focus (internal only)              |                   |
|      | <FocusLost>   | Vim window lost focus (internal only)             |                   |

**Note** about t\_so and t\_mr: When the termcap entry "so" is not present the entry for "mr" is used. And vice versa. The same is done for "se" and "me". If your terminal supports both inversion and standout mode, you can see two different modes. If your terminal supports only one of the modes, both will look the same.

keypad-comma

The keypad keys, when they are not mapped, behave like the equivalent normal key. There is one exception: if you have a comma on the keypad instead of a decimal point, Vim will use a dot anyway. Use these mappings to fix that:

```
:noremap <kPoint> ,
:noremap! <kPoint> ,
```

#### xterm-codes

There is a special trick to obtain the key codes which currently only works for xterm. When `t_RV` is defined and a response is received which indicates an xterm with patchlevel 141 or higher, Vim uses special escape sequences to request the key codes directly from the xterm. The responses are used to adjust the various `t_` codes. This avoids the problem that the xterm can produce different codes, depending on the mode it is in (8-bit, VT102, VT220, etc.). The result is that codes like `<xF1>` are no longer needed.

One of the codes that can change is `'t_Co'`, the number of colors. This will trigger a redraw. If this is a problem, reset the `'xtermcodes'` option as early as possible:

```
set noxtermcodes
```

**Note:** Requesting the key codes is only done on startup. If the xterm options are changed after Vim has started, the escape sequences may not be recognized anymore.

#### xterm-true-color

Vim supports using true colors in the terminal (taken from `highlight-guifg` and `highlight-guibg`), given that the terminal supports this. To make this work the `'termguicolors'` option needs to be set. See <https://github.com/termstandard/colors> for a list of terminals that support true colors.

For telling the terminal what RGB color to use the `t_8f` and `t_8b` termcap entries are used. These are set by default to values that work for most terminals. If that does not work for your terminal you can set them manually. The default values are set like this:

```
let &t_8f = "\<Esc>[38;2;%lu;%lu;%lum"
let &t_8b = "\<Esc>[48;2;%lu;%lu;%lum"
```

Some terminals accept the same sequences, but with all semicolons replaced by colons (this is actually more compatible, but less widely supported):

```
let &t_8f = "\<Esc>[38:2;%lu;%lu;%lum"
let &t_8b = "\<Esc>[48:2;%lu;%lu;%lum"
```

These options contain printf strings, with `printf()` (actually, its C equivalent hence ``l`` modifier) invoked with the `t_` option value and three unsigned long integers that may have any value between 0 and 255 (inclusive) representing red, green and blue colors respectively.

#### xterm-resize

Window resizing with xterm only works if the `allowWindowOps` resource is enabled. On some systems and versions of xterm it's disabled by default because someone thought it would be a security issue. It's not clear if this is actually the case.

To overrule the default, put this line in your `~/.Xdefaults` or

~/ .Xresources:

```
XTerm*allowWindowOps: true
```

And run "xrdb -merge .Xresources" to make it effective. You can check the value with the context menu (right mouse button while CTRL key is pressed), there should be a tick at allow-window-ops.

#### xterm-focus-event

Some terminals including xterm support the focus event tracking feature. If this feature is enabled by the '**t\_fe**' sequence, special key sequences are sent from the terminal to Vim every time the terminal gains or loses focus. Vim fires focus events ( FocusGained||FocusLost ) by handling them accordingly. Focus event tracking is disabled by a '**t\_fd**' sequence when exiting "raw" mode. If you would like to disable this feature, add the following to your .vimrc:

```
`set t_fd=`
`set t_fe=`
```

If your terminal does support this but Vim does not recognize the terminal, you may have to set the options yourself:

```
let &t_fe = "\<Esc>[?1004h"
let &t_fd = "\<Esc>[?1004l"
execute "set <FocusGained>=\<Esc>[I"
execute "set <FocusLost>=\<Esc>[O"
```

If this causes garbage to show when Vim starts up then it doesn't work.

#### termcap-colors

**Note** about colors: The '**t\_Co**' option tells Vim the number of colors available. When it is non-zero, the '**t\_AB**' and '**t\_AF**' options are used to set the color. If one of these is not available, '**t\_Sb**' and '**t\_Sf**' are used. '**t\_me**' is used to reset to the default colors. Also see '**termguicolors**'. When the GUI is running '**t\_Co**' is set to 16777216.

#### termcap-cursor-shape termcap-cursor-color

When Vim enters Insert mode the '**t\_SI**' escape sequence is sent. When Vim enters Replace mode the '**t\_SR**' escape sequence is sent if it is set, otherwise '**t\_SI**' is sent. When leaving Insert mode or Replace mode '**t\_EI**' is used.

**Note:** When '**t\_EI**' is not set then '**t\_SI**' and '**t\_SR**' will not be sent. And when '**t\_SI**' or '**t\_SR**' is not set then '**t\_EI**' is sent only once.

This can be used to change the shape or color of the cursor in Insert or Replace mode. These are not standard termcap/terminfo entries, you need to set them yourself.

Example for an xterm, this changes the color of the cursor:

```
if &term =~ "xterm"
 let &t_SI = "\<Esc>]12;purple\x7"
 let &t_SR = "\<Esc>]12;red\x7"
 let &t_EI = "\<Esc>]12;blue\x7"
endif
```

**NOTE:** When Vim exits the shape for Normal mode will remain. The shape from before Vim started will not be restored.

For Windows Terminal you can use something like this:

```
" Note: This should be set after `set termguicolors` or `set t_Co=256`.
if &term =~ 'xterm' || &term == 'win32'
```



```

" Use DECSCUSR escape sequences
let &t_SI = "\e[5 q" " blink bar
let &t_SR = "\e[3 q" " blink underline
let &t_EI = "\e[1 q" " blink block
let &t_ti .= "\e[1 q" " blink block
let &t_te .= "\e[0 q" " default (depends on terminal, normally blink
 " block)

endif

```

{not available when compiled without the |+cursorshape| feature}

#### termcap-title

The **'t\_ts'** and **'t\_fs'** options are used to set the window title if the terminal allows title setting via sending strings. They are sent before and after the title string, respectively. Similar **'t\_IS'** and **'t\_IE'** are used to set the icon text. These are Vim-internal extensions of the Unix termcap, so they cannot be obtained from an external termcap. However, the builtin termcap contains suitable entries for xterm and iris-ansi, so you don't need to set them here.

#### hpterm

If inversion or other highlighting does not work correctly, try setting the **'t\_xs'** option to a non-empty string. This makes the **'t\_ce'** code be used to remove highlighting from a line. This is required for "hpterm". Setting the **'weirdinvert'** option has the same effect as making **'t\_xs'** non-empty, and vice versa.

#### scroll-region

Some termcaps do not include an entry for "cs" (scroll region), although the terminal does support it. For example: xterm on a Sun. You can use the builtin\_xterm or define t\_cs yourself. For example:

```
:set t_cs=^V^[[%i%d;%dr
```

Where ^V is **CTRL-V** and ^[ is <Esc>.

The vertical scroll region t\_CV is not a standard termcap code. Vim uses it internally in the GUI. But it can also be defined for a terminal, if you can find one that supports it. The two arguments are the left and right column of the region which to restrict the scrolling to. Just like t\_cs defines the top and bottom lines. Defining t\_CV will make scrolling in vertically split windows a lot faster. Don't set t\_CV when t\_da or t\_db is set (text isn't cleared when scrolling).

Unfortunately it is not possible to deduce from the termcap how cursor positioning should be done when using a scrolling region: Relative to the beginning of the screen or relative to the beginning of the scrolling region. Most terminals use the first method. The **'t\_CS'** option should be set to any string when cursor positioning is relative to the start of the scrolling region. It should be set to an empty string otherwise.

**Note** for xterm users: The shifted cursor keys normally don't work. You can make them work with the xmodmap command and some mappings in Vim.

Give these commands in the xterm:

```
xmodmap -e "keysym Up = Up F13"
xmodmap -e "keysym Down = Down F16"
```

```
xmodmap -e "keysym Left = Left F18"
xmodmap -e "keysym Right = Right F19"
```

And use these mappings in Vim:

```
:map <t_F3> <S-Up>
:map! <t_F3> <S-Up>
:map <t_F6> <S-Down>
:map! <t_F6> <S-Down>
:map <t_F8> <S-Left>
:map! <t_F8> <S-Left>
:map <t_F9> <S-Right>
:map! <t_F9> <S-Right>
```

Instead of, say, `<S-Up>` you can use any other command that you want to use the shift-cursor-up key for. (Note: To help people that have a Sun keyboard with left side keys F14 is not used because it is confused with the undo key; F15 is not used, because it does a window-to-front; F17 is not used, because it closes the window. On other systems you can probably use them.)

---

### 3. Window size

window-size

[This is about the size of the whole window Vim is using, not a window that is created with the `:split` command.]

If you are running Vim on an Amiga and the terminal name is "amiga" or "builtin\_amiga", the amiga-specific window resizing will be enabled. On Unix systems three methods are tried to get the window size:

- an ioctl call (TIOCGSIZE or TIOCGWINSZ, depends on your system)
- the environment variables "LINES" and "COLUMNS"
- from the termcap entries "li" and "co"

If everything fails a default size of 24 lines and 80 columns is assumed. If a window-resize signal is received the size will be set again. If the window size is wrong you can use the `'lines'` and `'columns'` options to set the correct values.

One command can be used to set the screen size:

```
:mod[e] [mode] :mod :mode E359
```

Without argument this only detects the screen size and redraws the screen. `[mode]` was used on MS-DOS, but it doesn't work anymore. In Vim9 this command is not supported.

---

### 4. Slow and fast terminals

slow-fast-terminal  
slow-terminal

If you have a fast terminal you may like to set the `'ruler'` option. The cursor position is shown in the status line. If you are using horizontal scrolling (`'wrap'` option off) consider setting `'sidescroll'` to a small number.

If you have a slow terminal you may want to reset the `'showcmd'` option. The command characters will not be shown in the status line. If the terminal scrolls very slowly, set the `'scrolljump'` to 5 or so. If the cursor is moved off the screen (e.g., with "j") Vim will scroll 5 lines at a time. Another possibility is to reduce the number of lines that Vim uses with the command `"z{height}<CR>"`.

If the characters from the terminal are arriving with more than 1 second between them you might want to set the `'timeout'` and/or `'ttimeout'` option. See the "Options" chapter [options](#) .

If your terminal does not support a scrolling region, but it does support insert/delete line commands, scrolling with multiple windows may make the lines jump up and down. This would happen if the `'ttyfast'` option has been reset. Check that with:

```
verbose set ttyfast?
```

If your terminal scrolls very slowly, but redrawing is not slow, set the `'ttyscroll'` option to a small number, e.g., 3. This will make Vim redraw the screen instead of scrolling, when there are more than 3 lines to be scrolled.

If you are using a color terminal that is slow, use this command:

```
hi NonText cterm=NONE ctermfg=NONE
```

This avoids that spaces are sent when they have different attributes. On most terminals you can't see this anyway.

If you are using Vim over a slow serial line, you might want to try running Vim inside the "screen" program. Screen will optimize the terminal I/O quite a bit.

If you are testing termcap options, but you cannot see what is happening, you might want to set the `'writedelay'` option. When non-zero, one character is sent to the terminal at a time. This makes the screen updating a lot slower, making it possible to see what is happening.

---

## 5. Using the mouse mouse-using

This section is about using the mouse on a terminal or a terminal window. How to use the mouse in a GUI window is explained in [gui-mouse](#) . For scrolling with a mouse wheel see [scroll-mouse-wheel](#) .

Don't forget to enable the mouse with this command:

```
:set mouse=a
```

Otherwise Vim won't recognize the mouse in all modes (See `'mouse'`).

Currently the mouse is supported for Unix in an xterm window, in a \*BSD console with [sysmouse](#) , in a Linux console (with GPM [gpm-mouse](#) ), and in a Windows console.

Mouse clicks can be used to position the cursor, select an area and paste.

These characters in the `'mouse'` option tell in which situations the mouse will be used by Vim:

|   |                                        |
|---|----------------------------------------|
| n | Normal mode                            |
| v | Visual mode                            |
| i | Insert mode                            |
| c | Command-line mode                      |
| h | all previous modes when in a help file |
| a | all previous modes                     |
| r | for <code>hit-enter</code> prompt      |

The default for `'mouse'` is empty, the mouse is not used. Normally you would do:

```
:set mouse=a
```

to start using the mouse (this is equivalent to setting `'mouse'` to "nvich"). If you only want to use the mouse in a few modes or also want to use it for the two questions you will have to concatenate the letters for those modes. For example:

```
:set mouse=nv
```

Will make the mouse work in Normal mode and Visual mode.

```
:set mouse=h
```

Will make the mouse work in help files only (so you can use "g<LeftMouse>" to jump to tags).

Whether the selection that is started with the mouse is in Visual mode or Select mode depends on whether "mouse" is included in the `'selectmode'` option.

#### terminal-mouse

In an xterm, with the currently active mode included in the `'mouse'` option, normal mouse clicks are used by Vim, mouse clicks with the shift or ctrl key pressed go to the xterm. With the currently active mode not included in `'mouse'` all mouse clicks go to the xterm.

For terminals where it is not possible to have the mouse events be used by the terminal itself by using a modifier, a workaround is to not use mouse events for Vim in command-line mode:

```
:set mouse=nvi
```

Then to select text with the terminal, use ":" to go to command-line mode, select and copy the text to the system, then press Esc.

Another way is to temporarily use ":sh" to run a shell, copy the text, then exit the shell. `'mouse'` can remain set to "a" then.

#### xterm-clipboard

In the Motif GUI version, when running in a terminal and there is access to the X-server (DISPLAY is set), the copy and paste will behave like in the GUI. If not, the middle mouse button will insert the unnamed register. In that case, here is how you copy and paste a piece of text:

Copy/paste with the mouse and Visual mode (`'mouse'` option must be set, see above):

1. Press left mouse button on first letter of text, move mouse pointer to last letter of the text and release the button. This will start Visual mode and highlight the selected area.
2. Press "y" to yank the Visual text in the unnamed register.
3. Click the left mouse button at the insert position.
4. Click the middle mouse button.

Shortcut: If the insert position is on the screen at the same time as the Visual text, you can do 2, 3 and 4 all in one: Click the middle mouse button at the insert position.

**Note:** When the `-X` command line argument is used, Vim will not connect to the X server and copy/paste to the X clipboard (selection) will not work. Use the shift key with the mouse buttons to let the xterm do the selection.

[xterm-command-server](#)

When the X-server clipboard is available, the command server described in [x11-clientserver](#) can be enabled with the `--servername` command line argument.

[xterm-copy-paste](#)

**NOTE:** In some (older) xterms, it's not possible to move the cursor past column 95 or 223. This is an xterm problem, not Vim's. Get a newer xterm [color-xterm](#) . Also see ['ttymouse'](#) .

Copy/paste in xterm with (current mode NOT included in ['mouse'](#)):

1. Press left mouse button on first letter of text, move mouse pointer to last letter of the text and release the button.
2. Use normal Vim commands to put the cursor at the insert position.
3. Press "a" to start Insert mode.
4. Click the middle mouse button.
5. Press ESC to end Insert mode.

(The same can be done with anything in ['mouse'](#) if you keep the shift key pressed while using the mouse.)

**Note:** if you lose the 8th bit when pasting (special characters are translated into other characters), you may have to do `"stty cs8 -istrip -parenb"` in your shell before starting Vim.

Thus in an xterm the shift and ctrl keys cannot be used with the mouse. Mouse commands requiring the CTRL modifier can be simulated by typing the "g" key before using the mouse:

```
"g<LeftMouse>" is "<C-LeftMouse>" (jump to tag under mouse click)
" " ("CTRL-T")
" " ("CTRL-T")
" " ("CTRL-T")
```

[mouse-mode-table](#)   [mouse-overview](#)

A short overview of what the mouse buttons do, when ['mousemodel'](#) is "extend":

Normal Mode:

| event          | position<br>cursor | selection               | change<br>window | action                     |
|----------------|--------------------|-------------------------|------------------|----------------------------|
| <LeftMouse>    | yes                | end                     | yes              |                            |
| <C-LeftMouse>  | yes                | end                     | yes              | "CTRL-]" (2)               |
| <S-LeftMouse>  | yes                | no change               | yes              | "*" (2)     <S-LeftMouse>  |
| <LeftDrag>     | yes                | start or extend (1)     | no               | <LeftDrag>                 |
| <LeftRelease>  | yes                | start or extend (1)     | no               |                            |
| <MiddleMouse>  | yes                | if not active           | no               | put                        |
| <MiddleMouse>  | yes                | if active               | no               | yank and put               |
| <RightMouse>   | yes                | start or extend         | yes              |                            |
| <A-RightMouse> | yes                | start or extend blockw. | yes              | <A-RightMouse>             |
| <S-RightMouse> | yes                | no change               | yes              | "#" (2)     <S-RightMouse> |
| <C-RightMouse> | no                 | no change               | no               | "CTRL-T"                   |

|                |     |        |    |                |
|----------------|-----|--------|----|----------------|
| <RightDrag>    | yes | extend | no | <RightDrag>    |
| <RightRelease> | yes | extend | no | <RightRelease> |

Insert or Replace Mode:

| event          | position<br>cursor | selection               | change<br>window | action          |
|----------------|--------------------|-------------------------|------------------|-----------------|
| <LeftMouse>    | yes                | (cannot be active)      | yes              |                 |
| <C-LeftMouse>  | yes                | (cannot be active)      | yes              | "CTRL-O^]" (2)  |
| <S-LeftMouse>  | yes                | (cannot be active)      | yes              | "CTRL-O*" (2)   |
| <LeftDrag>     | yes                | start or extend (1)     | no               | like CTRL-O (1) |
| <LeftRelease>  | yes                | start or extend (1)     | no               | like CTRL-O (1) |
| <MiddleMouse>  | no                 | (cannot be active)      | no               | put register    |
| <RightMouse>   | yes                | start or extend         | yes              | like CTRL-O     |
| <A-RightMouse> | yes                | start or extend blockw. | yes              |                 |
| <S-RightMouse> | yes                | (cannot be active)      | yes              | "CTRL-O#" (2)   |
| <C-RightMouse> | no                 | (cannot be active)      | no               | "CTRL-O CTRL-T" |

In a help window:

| event         | position<br>cursor | selection          | change<br>window | action                  |
|---------------|--------------------|--------------------|------------------|-------------------------|
| <2-LeftMouse> | yes                | (cannot be active) | no               | "^]" (jump to help tag) |

When **'mousemodel'** is "popup", these are different:

Normal Mode:

| event         | position<br>cursor | selection               | change<br>window | action        |
|---------------|--------------------|-------------------------|------------------|---------------|
| <S-LeftMouse> | yes                | start or extend (1)     | no               |               |
| <A-LeftMouse> | yes                | start or extend blockw. | no               | <A-LeftMouse> |
| <RightMouse>  | no                 | popup menu              | no               |               |

Insert or Replace Mode:

| event         | position<br>cursor | selection               | change<br>window | action          |
|---------------|--------------------|-------------------------|------------------|-----------------|
| <S-LeftMouse> | yes                | start or extend (1)     | no               | like CTRL-O (1) |
| <A-LeftMouse> | yes                | start or extend blockw. | no               |                 |
| <RightMouse>  | no                 | popup menu              | no               |                 |

(1) only if mouse pointer moved since press

(2) only if click is in same buffer

Clicking the left mouse button causes the cursor to be positioned. If the click is in another window that window is made the active window. When editing the command-line the cursor can only be positioned on the command-line. When in Insert mode Vim remains in Insert mode. If **'scrolloff'** is set, and the cursor is positioned within **'scrolloff'** lines from the window border, the text is scrolled.

A selection can be started by pressing the left mouse button on the first character, moving the mouse to the last character, then releasing the mouse button. You will not always see the selection until you release the button, only in some versions (GUI, Win32) will the dragging be shown immediately.

**Note** that you can make the text scroll by moving the mouse at least one character in the first/last line in the window when **'scrolloff'** is non-zero.

In Normal, Visual and Select mode clicking the right mouse button causes the Visual area to be extended. When `'mousemodel'` is "popup", the left button has to be used while keeping the shift key pressed. When clicking in a window which is editing another buffer, the Visual or Select mode is stopped.

In Normal, Visual and Select mode clicking the right mouse button with the alt key pressed causes the Visual area to become blockwise. When `'mousemodel'` is "popup" the left button has to be used with the alt key. **Note** that this won't work on systems where the window manager consumes the mouse events when the alt key is pressed (it may move the window).

#### double-click

Double, triple and quadruple clicks are supported when the GUI is active, for Win32, and for an xterm (if the `gettimeofday()` function is available). For selecting text, extra clicks extend the selection:

| click     | select            |               |
|-----------|-------------------|---------------|
| double    | word or % match   | <2-LeftMouse> |
| triple    | line              | <3-LeftMouse> |
| quadruple | rectangular block | <4-LeftMouse> |

Exception: In a Help window a double click jumps to help for the word that is clicked on.

A double click on a word selects that word. `'iskeyword'` is used to specify which characters are included in a word. A double click on a character that has a match selects until that match (like using `"v%"`). If the match is an `#if/#else/#endif` block, the selection becomes linewise.

For MS-Windows and xterm the time for double clicking can be set with the `'mousetime'` option. For the other systems this time is defined outside of Vim. An example, for using a double click to jump to the tag under the cursor:

```
:map <2-LeftMouse> :exe "tag " .. expand("<cword>")<CR>
```

Dragging the mouse with a double click (button-down, button-up, button-down and then drag) will result in whole words to be selected. This continues until the button is released, at which point the selection is per character again.

For scrolling with the mouse see [scroll-mouse-wheel](#) .

#### gpm-mouse

The GPM mouse is only supported when the `+mouse_gpm` feature was enabled at compile time. The GPM mouse driver (Linux console) does not support quadruple clicks.

In Insert mode, when a selection is started, Vim goes into Normal mode temporarily. When Visual or Select mode ends, it returns to Insert mode. This is like using **CTRL-O** in Insert mode. Select mode is used when the `'selectmode'` option contains "mouse".

#### sysmouse

The `sysmouse` is only supported when the `+mouse_sysmouse` feature was enabled at compile time. The `sysmouse` driver (\*BSD console) does not support keyboard modifiers.

#### drag-status-line

When working with several windows, the size of the windows can be changed by

dragging the status line with the mouse. Point the mouse at a status line, press the left button, move the mouse to the new position of the status line, release the button. Just clicking the mouse in a status line makes that window the current window, without moving the cursor. If by selecting a window it will change position or size, the dragging of the status line will look confusing, but it will work (just try it).

|                                    |                            | <code>&lt;MiddleRelease&gt;</code> | <code>&lt;MiddleDrag&gt;</code> |
|------------------------------------|----------------------------|------------------------------------|---------------------------------|
| <code>code</code>                  | <code>mouse button</code>  | <code>normal action</code>         |                                 |
| <code>&lt;LeftMouse&gt;</code>     | left pressed               | set cursor position                |                                 |
| <code>&lt;LeftDrag&gt;</code>      | left moved while pressed   | extend selection                   |                                 |
| <code>&lt;LeftRelease&gt;</code>   | left released              | set selection end                  |                                 |
| <code>&lt;MiddleMouse&gt;</code>   | middle pressed             | paste text at cursor position      |                                 |
| <code>&lt;MiddleDrag&gt;</code>    | middle moved while pressed | -                                  |                                 |
| <code>&lt;MiddleRelease&gt;</code> | middle released            | -                                  |                                 |
| <code>&lt;RightMouse&gt;</code>    | right pressed              | extend selection                   |                                 |
| <code>&lt;RightDrag&gt;</code>     | right moved while pressed  | extend selection                   |                                 |
| <code>&lt;RightRelease&gt;</code>  | right released             | set selection end                  |                                 |
| <code>&lt;X1Mouse&gt;</code>       | X1 button pressed          | -                                  | <code>X1Mouse</code>            |
| <code>&lt;X1Drag&gt;</code>        | X1 moved while pressed     | -                                  | <code>X1Drag</code>             |
| <code>&lt;X1Release&gt;</code>     | X1 button release          | -                                  | <code>X1Release</code>          |
| <code>&lt;X2Mouse&gt;</code>       | X2 button pressed          | -                                  | <code>X2Mouse</code>            |
| <code>&lt;X2Drag&gt;</code>        | X2 moved while pressed     | -                                  | <code>X2Drag</code>             |
| <code>&lt;X2Release&gt;</code>     | X2 button release          | -                                  | <code>X2Release</code>          |

The X1 and X2 buttons refer to the extra buttons found on some mice. The 'Microsoft Explorer' mouse has these buttons available to the right thumb. Currently X1 and X2 only work on Win32 and X11 environments.

Examples:

```
:noremap <MiddleMouse> <LeftMouse><MiddleMouse>
```

Paste at the position of the middle mouse button click (otherwise the paste would be done at the cursor position).

```
:noremap <LeftRelease> <LeftRelease>y
```

Immediately yank the selection, when using Visual mode.

Note the use of ":noremap" instead of "map" to avoid a recursive mapping.

```
:map <X1Mouse> <C-O>
```

```
:map <X2Mouse> <C-I>
```

Map the X1 and X2 buttons to go forwards and backwards in the jump list, see `CTRL-O` and `CTRL-I`.

#### mouse-swap-buttons

To swap the meaning of the left and right mouse buttons:

```
:noremap <LeftMouse> <RightMouse>
:noremap <LeftDrag> <RightDrag>
:noremap <LeftRelease> <RightRelease>
:noremap <RightMouse> <LeftMouse>
:noremap <RightDrag> <LeftDrag>
:noremap <RightRelease> <LeftRelease>
:noremap g<LeftMouse> <C-RightMouse>
```



```
:noremap g<RightMouse> <C-LeftMouse>
:noremap! <LeftMouse> <RightMouse>
:noremap! <LeftDrag> <RightDrag>
:noremap! <LeftRelease> <RightRelease>
:noremap! <RightMouse> <LeftMouse>
:noremap! <RightDrag> <LeftDrag>
:noremap! <RightRelease> <LeftRelease>
```



Terminal window support [terminal](#) [terminal-window](#)

The terminal feature is optional, use this to check if your Vim has it:

```
echo has('terminal')
```

If the result is "1" you have it.

|                                    |                                           |
|------------------------------------|-------------------------------------------|
| 1. Basic use                       | <a href="#">terminal-use</a>              |
| Typing                             | <a href="#">terminal-typing</a>           |
| Size and color                     | <a href="#">terminal-size-color</a>       |
| Command syntax                     | <a href="#">:terminal</a>                 |
| Resizing                           | <a href="#">terminal-resizing</a>         |
| Terminal Modes                     | <a href="#">Terminal-mode</a>             |
| Cursor style                       | <a href="#">terminal-cursor-style</a>     |
| Session                            | <a href="#">terminal-session</a>          |
| Special keys                       | <a href="#">terminal-special-keys</a>     |
| Unix                               | <a href="#">terminal-unix</a>             |
| MS-Windows                         | <a href="#">terminal-ms-windows</a>       |
| 2. Terminal functions              | <a href="#">terminal-function-details</a> |
| 3. Terminal communication          | <a href="#">terminal-communication</a>    |
| Vim to job: term_sendkeys()        | <a href="#">terminal-to-job</a>           |
| Job to Vim: JSON API               | <a href="#">terminal-api</a>              |
| Using the client-server feature    | <a href="#">terminal-client-server</a>    |
| 4. Remote testing                  | <a href="#">terminal-testing</a>          |
| 5. Diffing screen dumps            | <a href="#">terminal-diff</a>             |
| Writing a screen dump test for Vim | <a href="#">terminal-dumptest</a>         |
| Creating a screen dump             | <a href="#">terminal-screendump</a>       |
| Comparing screen dumps             | <a href="#">terminal-diffscreeendump</a>  |
| 6. Debugging                       | <a href="#">terminal-debug</a>            |
| Starting                           | <a href="#">termdebug-starting</a>        |
| Example session                    | <a href="#">termdebug-example</a>         |
| Stepping through code              | <a href="#">termdebug-stepping</a>        |
| Inspecting variables               | <a href="#">termdebug-variables</a>       |
| Navigating stack frames            | <a href="#">termdebug-frames</a>          |
| Other commands                     | <a href="#">termdebug-commands</a>        |
| Events                             | <a href="#">termdebug-events</a>          |
| Prompt mode                        | <a href="#">termdebug-prompt</a>          |
| Mappings                           | <a href="#">termdebug-mappings</a>        |
| Communication                      | <a href="#">termdebug-communication</a>   |
| Customizing                        | <a href="#">termdebug-customizing</a>     |

{only available when compiled with the |+terminal| feature}

The terminal feature requires the +job and +channel features.

=====

|              |                              |
|--------------|------------------------------|
| 1. Basic use | <a href="#">terminal-use</a> |
|--------------|------------------------------|

This feature is for running a terminal emulator in a Vim window. A job can be started connected to the terminal emulator. For example, to run a shell:

```
:term bash
```

Or to run build command:

```
:term make myprogram
```

The job runs asynchronously from Vim, the window will be updated to show output from the job, also while editing in another window.

## Typing

[terminal-typing](#)

When the keyboard focus is in the terminal window, typed keys will be sent to the job. This uses a pty when possible. You can click outside of the terminal window to move keyboard focus elsewhere.

**CTRL-W** can be used to navigate between windows and other **CTRL-W** commands, e.g.:

```
CTRL-W CTRL-W move focus to the next window
```

```
CTRL-W : enter an Ex command
```

See [CTRL-W](#) for more commands.

Special in the terminal window:

[t\\_CTRL-W\\_CTRL-W](#) [t\\_CTRL-W\\_:](#)

```
CTRL-W . send a CTRL-W to the job in the terminal
```

```
CTRL-W CTRL-\ send a CTRL-\ to the job in the terminal
```

```
CTRL-W N go to Terminal-Normal mode, see Terminal-mode
```

```
CTRL-\ CTRL-N go to Terminal-Normal mode, see Terminal-mode
```

```
CTRL-W " {reg} paste register {reg} t_CTRL-W_quote
```

Also works with the = register to insert the result of evaluating an expression.

```
CTRL-W CTRL-C ends the job, see below t_CTRL-W_CTRL-C
```

```
CTRL-W gt go to next tabpage, same as `gt` t_CTRL-W_gt
```

```
CTRL-W gT go to previous tabpage, same as `gT` t_CTRL-W_gT
```

See option '[termwinkey](#)' for specifying another key instead of **CTRL-W** that will work like **CTRL-W**. However, typing '[termwinkey](#)' twice sends '[termwinkey](#)' to the job. For example:

```
'termwinkey' CTRL-W move focus to the next window
```

```
'termwinkey' : enter an Ex command
```

```
'termwinkey' 'termwinkey' send 'termwinkey' to the job in the terminal
```

```
'termwinkey' . send 'termwinkey' to the job in the terminal
```

```
'termwinkey' CTRL-\ send a CTRL-\ to the job in the terminal
```

```
'termwinkey' N go to terminal Normal mode, see below
```

```
'termwinkey' CTRL-N same as CTRL-W N t_CTRL-W_N
```

```
'termwinkey' CTRL-C same as CTRL-W CTRL-C t_CTRL-W_CTRL-C
```

[t\\_CTRL-\\\_CTRL-N](#)

The special key combination **CTRL-\ CTRL-N** can be used to switch to Normal mode, just like this works in any other mode.

[t\\_CTRL-W\\_CTRL-C](#)

**CTRL-W CTRL-C** can be typed to forcefully end the job. On MS-Windows a **CTRL-Break** will also kill the job.

If you type **CTRL-C** the effect depends on what the pty has been configured to

do. For simple commands this causes a SIGINT to be sent to the job, which would end it. Other commands may ignore the SIGINT or handle the **CTRL-C** themselves (like Vim does).

To change the keys you type use terminal mode mappings, see `:tmap`. These are defined like any mapping, but apply only when typing keys that are sent to the job running in the terminal. For example, to make F1 switch to Terminal-Normal mode:

```
tnoremap <F1> <C-W>N
```

You can use Esc, but you need to make sure it won't cause other keys to break (cursor keys start with an Esc, so they may break), this probably only works in the GUI:

```
tnoremap <Esc> <C-W>N
set notimeout ttimeout timeoutlen=100
```

You can also create menus similar to terminal mode mappings, but you have to use `:tmenu` instead of `:tmenu`.

### options-in-terminal

After opening the terminal window and setting `'buftype'` to "terminal" the `TerminalWinOpen` autocommand event is triggered. This makes it possible to set options specifically for the terminal window and buffer. Example:

```
au TerminalWinOpen * setlocal bufhidden=hide
```

This only works properly if the terminal is not hidden.

For both hidden and non-hidden terminals this works, both for buffer-local and window-local options:

```
au TerminalWinOpen,BufWinEnter * if &buftype == 'terminal'
 \ | setlocal bufhidden=hide colorcolumn=123
 \ | endif
```

**Note** that for a hidden terminal the options are not set until the terminal is no longer hidden.

There is also the `TerminalOpen` event. Keep in mind this may be triggered for a hidden terminal, then the current window and buffer are not that of the new terminal.

You need to use `<abuf>`, which is set to the terminal buffer. Example:

```
au TerminalOpen * call setbufvar(expand('<abuf>')->str2nr(),
 \ '&termwscroll', 1000)
```

For a window-local option, you need to delay setting the option until the terminal window has been created (this only works for a hidden terminal):

```
au TerminalOpen * exe printf(
 \ 'au BufWinEnter <buffer=%d> ++once setlocal colorcolumn=%d',
 \ expand('<abuf>')->str2nr(), 123)
```

For a non-hidden terminal use `TerminalWinOpen`.

Mouse events (click and drag) are passed to the terminal. Mouse move events are only passed when Vim itself is receiving them. For a terminal that is when `'balloonevalterm'` is enabled.

### Size and color

### terminal-size-color

See option `'termwsize'` for controlling the size of the terminal window.

(TODO: scrolling when the terminal is larger than the window)

The job running in the terminal can change the colors. The default foreground and background colors are taken from Vim, the Normal highlight group.

For a color terminal the **'background'** option is used to decide whether the terminal window will start with a white or black background.

To use a different color the Terminal highlight group can be used, for example:

```
hi Terminal ctermbg=lightgrey ctermfg=blue guibg=lightgrey guifg=blue
```

Instead of Terminal another group can be specified with the "term\_highlight" option for ``term_start()``.

In GUI mode or with **'termguicolors'**, the 16 ANSI colors used by default in new terminal windows may be configured using the variable ``g:terminal_ansi_colors``, which should be a list of 16 color names or hexadecimal color codes, similar to those accepted by `highlight-guifg`. When not using GUI colors, the terminal window always uses the 16 ANSI colors of the underlying terminal.

When using ``term_start()`` the colors can be set with the "ansi\_colors" option. The `term_setansicolors()` function can be used to change the colors, and `term_getansicolors()` to get the currently used colors.

## Command syntax

```
:[range]ter[minal] [options] [command] :ter :terminal
Open a new terminal window.
```

If `[command]` is provided run it as a job and connect the input and output to the terminal.

If `[command]` is not given the **'shell'** option is used.

if `[command]` is NONE no job is started, the pty of the terminal can be used by a command like gdb.

If `[command]` is missing the default behavior is to close the terminal when the shell exits. This can be changed with the `++noclose` argument.

If `[command]` is present the default behavior is to keep the terminal open in Terminal-Normal mode. This can be changed with the `++close` argument.

No Vim command can follow, any `|` is included in `[command]`. Use ``:execute`` if you must have a Vim command following in the same line.

A new buffer will be created, using `[command]` or **'shell'** as the name, prefixed with a `"!"`. If a buffer by this name already exists a number is added in parentheses. E.g. if "gdb" exists the second terminal buffer will use `"!gdb (1)"`.

If `[range]` is given the specified lines are used as input for the job. It will not be possible to type keys in the terminal window. For MS-Windows see the `++eof` argument below.

`term++close` `term++open`

Supported `[options]` are:

`++close` The terminal window will close automatically when the job terminates.  
`terminal-close`

`++noclose` The terminal window will NOT close automatically when the job terminates.

`++open` When the job terminates and no window shows it, a window will be opened.  
**Note** that this can be interruptive.

The last of `++close`, `++noclose` and `++open` matters and rules out earlier arguments.

`++curwin` Open the terminal in the current window, do not split the current window. Fails if the current buffer cannot be `abandon` ed.

`++hidden` Open the terminal in a hidden buffer, no window will be used.

`++norestore` Do not include this terminal window in a session file.

`++shell` Instead of executing `{command}` directly, use a shell, like with ``:!command`` **E279**

`{only works on Unix and MS-Windows}`

`++kill={how}` When trying to close the terminal window kill the job with `{how}`. See `term_setkill()` for the values.

`++rows={height}` Use `{height}` for the terminal window height. If the terminal uses the full Vim height (no window above or below the terminal window) the command line height will be reduced as needed.

`++cols={width}` Use `{width}` for the terminal window width. If the terminal uses the full Vim width (no window left or right of the terminal window) this value is ignored.

`++eof={text}` When using `[range]`: text to send after the last line was written. Cannot contain white space. A CR is appended. For MS-Windows the default is to send **CTRL-D**.

E.g. for a shell use `"++eof=exit"` and for Python `"++eof=exit()"`. Special codes can be used like with ``:map``, e.g. `"<C-Z"` for **CTRL-Z**.

`++type={pty}` (MS-Windows only): Use `{pty}` as the

virtual console. See **'termwintype'** for the values.

`++api={expr}` Permit the function name starting with `{expr}` to be called as `terminal-api` function. If `{expr}` is empty then no function can be called.

If you want to use more options use the `term_start()` function.

If you want to split the window vertically, use:  
`:vertical terminal`

Or short:  
`:vert ter`

When the buffer associated with the terminal is forcibly unloaded or wiped out the job is killed, similar to calling ``job_stop(job, "kill")``.

Closing the window normally results in `E947`. When a kill method was set with `"++kill={how}"` or `term_setkill()` then closing the window will use that way to kill or interrupt the job. For example:

```
:term ++kill=term tail -f /tmp/log
```

So long as the job is running the window behaves like it contains a modified buffer. Trying to close the window with ``CTRL-W :quit`` fails. When using ``CTRL-W :quit!`` the job is ended. The text in the window is lost, the buffer is deleted. With ``CTRL-W :bunload!`` the buffer remains but will be empty.

Trying to close the window with ``CTRL-W :close`` also fails. Using ``CTRL-W :close!`` will close the window and make the buffer hidden.

You can use ``CTRL-W :hide`` to close the terminal window and make the buffer hidden, the job keeps running. The ``:buffer`` command can be used to turn the current window into a terminal window. If there are unsaved changes this fails, use `!` to force, as usual.

#### **terminal-close**

When the terminal window is closed, e.g. when the shell exits and `"++close"` argument was used, and this is the last normal Vim window, then Vim will exit. This is like using `:quit` in a normal window. Help and preview windows are not counted.

To have a background job run without a window, and open the window when it's done, use options like this:

```
:term ++hidden ++open make
```

**Note** that the window will open at an unexpected moment, this will interrupt what you are doing.

#### **E947 E948**

So long as the job is running, the buffer is considered modified and Vim cannot be quit easily, see `abandon`.

When the job has finished and no changes were made to the buffer: closing the window will wipe out the buffer.

Before changes can be made to a terminal buffer, the **'modifiable'** option must



be set. This is only possible when the job has finished. At the first change the buffer will become a normal buffer and the highlighting is removed. You may want to change the buffer name with `:file` to be able to write, since the buffer name will still be set to the command.

## Resizing

[terminal-resizing](#)

The size of the terminal can be in one of three modes:

1. The `'termwinsize'` option is empty: The terminal size follows the window size. The minimal size is 2 screen lines with 10 cells.
2. The `'termwinsize'` option is "rows\*cols", where "rows" is the minimal number of screen rows and "cols" is the minimal number of cells.
3. The `'termwinsize'` option is "rowsXcols" (where the x is upper or lower case). The terminal size is fixed to the specified number of screen lines and cells. If the window is bigger there will be unused empty space.

If the window is smaller than the terminal size, only part of the terminal can be seen (the lower-left part).

The `term_getsize()` function can be used to get the current size of the terminal. `term_setsize()` can be used only when in the first or second mode, not when `'termwinsize'` is "rowsXcols".

## Terminal-Job and Terminal-Normal mode

[Terminal-mode](#) [Terminal-Job](#)

When the job is running the contents of the terminal is under control of the job. That includes the cursor position. Typed keys are sent to the job. The terminal contents can change at any time. This is called Terminal-Job mode.

Use `CTRL-W N` (or `'termwinkey' N`) to switch to Terminal-Normal mode. Now the contents of the terminal window is under control of Vim, the job output is suspended. `CTRL-\ CTRL-N` does the same.

Terminal-Job mode is where `:tmap` mappings are applied. Keys sent by `term_sendkeys()` are not subject to tmap, but keys from `feedkeys()` are.

It is not possible to enter Insert mode from Terminal-Job mode.

[Terminal-Normal](#) [E946](#)

In Terminal-Normal mode you can move the cursor around with the usual Vim commands, Visually mark text, yank text, etc. But you cannot change the contents of the buffer. The commands that would start insert mode, such as `'i'` and `'a'`, return to Terminal-Job mode. The window will be updated to show the contents of the terminal. `:startinsert` is ineffective.

In Terminal-Normal mode the statusline and window title show "(Terminal)". If the job ends while in Terminal-Normal mode this changes to "(Terminal-finished)".

When the job outputs lines in the terminal, such that the contents scrolls off the top, those lines are remembered and can be seen in Terminal-Normal mode. The number of lines is limited by the `'termwincscroll'` option. When going over this limit, the first 10% of the scrolled lines are deleted and are lost.

## Cursor style

`terminal-cursor-style`

By default the cursor in the terminal window uses a not blinking block. The normal xterm escape sequences can be used to change the blinking state and the shape. Once focus leaves the terminal window Vim will restore the original cursor.

An exception is when xterm is started with the `"-bc"` argument, or another way that causes the cursor to blink. This actually means that the blinking flag is inverted. Since Vim cannot detect this, the terminal window cursor blinking will also be inverted.

## Session

`terminal-session`

A terminal window will be restored when using a session file, if possible and wanted.

If `"terminal"` was removed from `'sessionoptions'` then no terminal windows will be restored.

If the job in the terminal was finished the window will not be restored.

If the terminal can be restored, the command that was used to open it will be used again. To change this use the `term_setrestore()` function. This can also be used to not restore a specific terminal by setting the command to `"NONE"`.

## Special keys

`terminal-special-keys`

Since the terminal emulator simulates an xterm, only escape sequences that both Vim and xterm recognize will be available in the terminal window. If you want to pass on other escape sequences to the job running in the terminal you need to set up forwarding. Example:

```
tmap <expr> <Esc>]b SendToTerm("\<Esc>]b")
func SendToTerm(what)
 call term_sendkeys(' ', a:what)
 return ''
endfunc
```

## Unix

`terminal-unix`

On Unix a pty is used to make it possible to run all kinds of commands. You can even run Vim in the terminal! That's used for debugging, see below.

Environment variables are used to pass information to the running job:

|                |                                                                                                                                       |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| TERM           | the name of the terminal, from the <b>'term'</b> option or \$TERM in the GUI; falls back to "xterm" if it does not start with "xterm" |
| ROWS           | number of rows in the terminal initially                                                                                              |
| LINES          | same as ROWS                                                                                                                          |
| COLUMNS        | number of columns in the terminal initially                                                                                           |
| COLORS         | number of colors, <b>'t_Co'</b> (256*256*256 in the GUI)                                                                              |
| VIM_SERVERNAME | v:servername                                                                                                                          |
| VIM_TERMINAL   | v:version                                                                                                                             |

## MS-Windows

[terminal-ms-windows](#)

On MS-Windows winpty is used to make it possible to run all kind of commands. Obviously, they must be commands that run in a terminal, not open their own window.

You need the following two files from winpty:

```
winpty.dll
winpty-agent.exe
```

You can download them from the following page:

<https://github.com/rprichard/winpty>

Just put the files somewhere in your PATH. You can set the **'winptydll'** option to point to the right file, if needed. If you have both the 32-bit and 64-bit version, rename to winpty32.dll and winpty64.dll to match the way Vim was build.

[ConPTY](#) [E982](#)

On more recent versions of MS-Windows 10 (beginning with the "October 2018 Update"), winpty is no longer required. On those versions, **:terminal** will use Windows' built-in support for hosting terminal applications, "ConPTY". When ConPTY is in use, there may be rendering artifacts regarding ambiguous-width characters. If you encounter any such issues, install "winpty". Until the ConPTY problems have been fixed "winpty" will be preferred.

Environment variables are used to pass information to the running job:

|                |              |
|----------------|--------------|
| VIM_SERVERNAME | v:servername |
|----------------|--------------|

---

## 2. Terminal functions

[terminal-function-details](#)

[term\\_dumpdiff\(\)](#)

```
term_dumpdiff({filename}, {filename} [, {options}])
```

Open a new window displaying the difference between the two files. The files must have been created with

```
term_dumpwrite() .
```

Returns the buffer number or zero when the diff fails.

Also see [terminal-diff](#) .

**NOTE:** this does not work with double-width characters yet.

The top part of the buffer contains the contents of the first file, the bottom part of the buffer contains the contents of the second file. The middle part shows the differences. The parts are separated by a line of equals.

If the `{options}` argument is present, it must be a Dict with these possible members:

|             |                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "term_name" | name to use for the buffer name, instead of the first file name.                                                                                                                                      |
| "term_rows" | vertical size to use for the terminal, instead of using <code>'termwinsize'</code> , but respecting the minimal size                                                                                  |
| "term_cols" | horizontal size to use for the terminal, instead of using <code>'termwinsize'</code> , but respecting the minimal size                                                                                |
| "vertical"  | split the window vertically                                                                                                                                                                           |
| "curwin"    | use the current window, do not split the window; fails if the current buffer cannot be abandoned                                                                                                      |
| "bufnr"     | do not create a new buffer, use the existing buffer "bufnr". This buffer must have been previously created with <code>term_dumpdiff()</code> or <code>term_dumpload()</code> and visible in a window. |
| "norestore" | do not add the terminal window to a session file                                                                                                                                                      |

Each character in the middle part indicates a difference. If there are multiple differences only the first in this list is used:

|   |                                              |
|---|----------------------------------------------|
| X | different character                          |
| w | different width                              |
| f | different foreground color                   |
| b | different background color                   |
| a | different attribute                          |
| + | missing position in first file               |
| - | missing position in second file              |
| > | cursor position in first file, not in second |
| < | cursor position in second file, not in first |

Using the "s" key the top and bottom parts are swapped. This makes it easy to spot a difference.

Can also be used as a `method` :

`GetFilename()->term_dumpdiff(otherfile)`

`term_dumpload()`

`term_dumpload({filename} [, {options}])`

Open a new window displaying the contents of `{filename}`. The file must have been created with `term_dumpwrite()`. Returns the buffer number or zero when it fails. Also see `terminal-diff`.

For `{options}` see `term_dumpdiff()`.

Can also be used as a `method` :  
`GetFilename()->term_dumpload()`

`term_dumpwrite({buf}, {filename} [, {options}])` `term_dumpwrite()`  
Dump the contents of the terminal screen of `{buf}` in the file `{filename}`. This uses a format that can be used with `term_dumpload()` and `term_dumpdiff()` .  
If the job in the terminal already finished an error is given: `E958`  
If `{filename}` already exists an error is given: `E953`  
Also see `terminal-diff` .

`{options}` is a dictionary with these optional entries:  
"rows" maximum number of rows to dump  
"columns" maximum number of columns to dump

Can also be used as a `method` , the base is used for the file name:

`GetFilename()->term_dumpwrite(bufnr)`

`term_getaltscreen({buf})` `term_getaltscreen()`  
Returns 1 if the terminal of `{buf}` is using the alternate screen.  
`{buf}` is used as with `term_getsize()` .

Can also be used as a `method` :  
`GetBufnr()->term_getaltscreen()`

`term_getansicolors({buf})` `term_getansicolors()`  
Get the ANSI color palette in use by terminal `{buf}`.  
Returns a List of length 16 where each element is a String representing a color in hexadecimal "#rrggbb" format.  
Also see `term_setansicolors()` and `g:terminal_ansi_colors` .  
If neither was used returns the default colors.

`{buf}` is used as with `term_getsize()` . If the buffer does not exist or is not a terminal window, an empty list is returned.

Can also be used as a `method` :  
`GetBufnr()->term_getansicolors()`

{only available when compiled with GUI enabled and/or the `+termguicolors` feature}

`term_getattr({attr}, {what})` `term_getattr()`  
Given `{attr}`, a value returned by `term_scrape()` in the "attr" item, return whether `{what}` is on. `{what}` can be one of:  
bold  
italic  
underline  
strike

reverse

Can also be used as a [method](#) :  
[GetAttr\(\)->term\\_getattr\(\)](#)

`term_getcursor({buf})` [term\\_getcursor\(\)](#)  
Get the cursor position of terminal `{buf}`. Returns a list with two numbers and a dictionary: [row, col, dict].

"row" and "col" are one based, the first screen cell is row 1, column 1. This is the cursor position of the terminal itself, not of the Vim window.

"dict" can have these members:

|           |                                                                 |
|-----------|-----------------------------------------------------------------|
| "visible" | one when the cursor is visible, zero when it is hidden.         |
| "blink"   | one when the cursor is blinking, zero when it is not blinking.  |
| "shape"   | 1 for a block cursor, 2 for underline and 3 for a vertical bar. |
| "color"   | color of the cursor, e.g. "green"                               |

`{buf}` must be the buffer number of a terminal window. If the buffer does not exist or is not a terminal window, an empty list is returned.

Can also be used as a [method](#) :  
[GetBufnr\(\)->term\\_getcursor\(\)](#)

`term_getjob({buf})` [term\\_getjob\(\)](#)  
Get the Job associated with terminal window `{buf}`.  
`{buf}` is used as with [term\\_getsize\(\)](#) .  
Returns `v:null` when there is no job.

Can also be used as a [method](#) :  
[GetBufnr\(\)->term\\_getjob\(\)](#)

`term_getline({buf}, {row})` [term\\_getline\(\)](#)  
Get a line of text from the terminal window of `{buf}`.  
`{buf}` is used as with [term\\_getsize\(\)](#) .

The first line has `{row}` one. When `{row}` is "." the cursor line is used. When `{row}` is invalid an empty string is returned.

To get attributes of each character use [term\\_scrape\(\)](#) .

Can also be used as a [method](#) :  
[GetBufnr\(\)->term\\_getline\(row\)](#)

`term_getscrolled({buf})` [term\\_getscrolled\(\)](#)

Return the number of lines that scrolled to above the top of terminal `{buf}`. This is the offset between the row number used for `term_getline()` and `getline()` , so that:

```
term_getline(buf, N)
is equal to:
getline(N + term_getscrolled(buf))
(if that line exists).
```

`{buf}` is used as with `term_getsize()` .

Can also be used as a `method` :  
`GetBufnr()->term_getscrolled()`

`term_getsize({buf})` `term_getsize()`  
Get the size of terminal `{buf}`. Returns a list with two numbers: [rows, cols]. This is the size of the terminal, not the window containing the terminal.

`{buf}` must be the buffer number of a terminal window. Use an empty string for the current buffer. If the buffer does not exist or is not a terminal window, an empty list is returned.

Can also be used as a `method` :  
`GetBufnr()->term_getsize()`

`term_getstatus({buf})` `term_getstatus()`  
Get the status of terminal `{buf}`. This returns a String with a comma-separated list of these items:

|          |                         |
|----------|-------------------------|
| running  | job is running          |
| finished | job has finished        |
| normal   | in Terminal-Normal mode |

One of "running" or "finished" is always present.

`{buf}` must be the buffer number of a terminal window. If the buffer does not exist or is not a terminal window, an empty string is returned.

Can also be used as a `method` :  
`GetBufnr()->term_getstatus()`

`term_gettitle({buf})` `term_gettitle()`  
Get the title of terminal `{buf}`. This is the title that the job in the terminal has set.

`{buf}` must be the buffer number of a terminal window. If the buffer does not exist or is not a terminal window, an empty string is returned.

Can also be used as a `method` :  
`GetBufnr()->term_gettitle()`

`term_gettty({buf} [, {input}])` `term_gettty()`  
Get the name of the controlling terminal associated with terminal window `{buf}`. `{buf}` is used as with `term_getsize()` .

When `{input}` is omitted or 0, return the name for writing (stdout). When `{input}` is 1 return the name for reading (stdin). On UNIX, both return same name.

Can also be used as a `method` :  
`GetBufnr()->term_gettty()`

`term_list()` `term_list()`  
Return a list with the buffer numbers of all buffers for terminal windows.

`term_scrape({buf}, {row})` `term_scrape()`  
Get the contents of `{row}` of terminal screen of `{buf}`. For `{buf}` see `term_getsize()` .

The first line has `{row}` one. When `{row}` is "." the cursor line is used. When `{row}` is invalid an empty string is returned.

Return a List containing a Dict for each screen cell:

|         |                                                                                     |
|---------|-------------------------------------------------------------------------------------|
| "chars" | character(s) at the cell                                                            |
| "fg"    | foreground color as #rrggbb                                                         |
| "bg"    | background color as #rrggbb                                                         |
| "attr"  | attributes of the cell, use <code>term_getattr()</code> to get the individual flags |
| "width" | cell width: 1 or 2                                                                  |

For a double-width cell there is one item, thus the list can be shorter than the width of the terminal.

Can also be used as a `method` :  
`GetBufnr()->term_scrape(row)`

`term_sendkeys({buf}, {keys})` `term_sendkeys()`  
Send keystrokes `{keys}` to terminal `{buf}`. `{buf}` is used as with `term_getsize()` .

`{keys}` are translated as key sequences. For example, "`\<c-x`" means the character **CTRL-X**.

Can also be used as a `method` :  
`GetBufnr()->term_sendkeys(keys)`

`term_setansicolors({buf}, {colors})` `term_setansicolors()`  
Set the ANSI color palette used by terminal `{buf}`. `{colors}` must be a List of 16 valid color names or hexadecimal



color codes, like those accepted by `highlight-guifg` .  
Also see `term_getansicolors()` and `g:terminal_ansi_colors` .

The colors normally are:

|    |              |
|----|--------------|
| 0  | black        |
| 1  | dark red     |
| 2  | dark green   |
| 3  | brown        |
| 4  | dark blue    |
| 5  | dark magenta |
| 6  | dark cyan    |
| 7  | light grey   |
| 8  | dark grey    |
| 9  | red          |
| 10 | green        |
| 11 | yellow       |
| 12 | blue         |
| 13 | magenta      |
| 14 | cyan         |
| 15 | white        |

These colors are used in the GUI and in the terminal when `'termguicolors'` is set. When not using GUI colors (GUI mode or `'termguicolors'`), the terminal window always uses the 16 ANSI colors of the underlying terminal.

Can also be used as a `method` :

```
GetBufnr()->term_setansicolors(colors)
```

{only available with GUI enabled and/or the `+termguicolors` feature}

```
term_setapi({buf}, {expr}) term_setapi()
Set the function name prefix to be used for the terminal-api
function in terminal {buf}. For example:
:call term_setapi(buf, "Myapi_")
:call term_setapi(buf, "")
```

The default is "Tapi\_". When `{expr}` is an empty string then no `terminal-api` function can be used for `{buf}`.

When used as a method the base is used for `{buf}`:

```
GetBufnr()->term_setapi({expr})
```

```
term_setkill({buf}, {how}) term_setkill()
When exiting Vim or trying to close the terminal window in
another way, {how} defines whether the job in the terminal can
be stopped.
When {how} is empty (the default), the job will not be
stopped, trying to exit will result in E947 .
Otherwise, {how} specifies what signal to send to the job.
See job_stop() for the values.
```

After sending the signal Vim will wait for up to a second to check that the job actually stopped.

Can also be used as a `method` :  
`GetBufnr()->term_setkill(how)`

`term_setrestore({buf}, {command})` `term_setrestore()`  
Set the command to write in a session file to restore the job in this terminal. The line written in the session file is:  
`terminal ++curwin ++cols=%d ++rows=%d {command}`  
Make sure to escape the command properly.

Use an empty `{command}` to run `'shell'`.  
Use "NONE" to not restore this window.

Can also be used as a `method` :  
`GetBufnr()->term_setrestore(command)`

`term_setsize({buf}, {rows}, {cols})` `term_setsize()` E955  
Set the size of terminal `{buf}`. The size of the window containing the terminal will also be adjusted, if possible. If `{rows}` or `{cols}` is zero or negative, that dimension is not changed.

`{buf}` must be the buffer number of a terminal window. Use an empty string for the current buffer. If the buffer does not exist or is not a terminal window, an error is given.

Can also be used as a `method` :  
`GetBufnr()->term_setsize(rows, cols)`

`term_start({cmd} [, {options}])` `term_start()`  
Open a terminal window and run `{cmd}` in it.  
`{cmd}` can be a string or a List, like with `job_start()` . The string "NONE" can be used to open a terminal window without starting a job, the pty of the terminal can be used by a command like gdb.

Returns the buffer number of the terminal window. If `{cmd}` cannot be executed the window does open and shows an error message.

If opening the window fails zero is returned.

`{options}` are similar to what is used for `job_start()` , see `job-options` . However, not all options can be used. These are supported:

- all timeout options
- "stoponexit", "cwd", "env"
- "callback", "out\_cb", "err\_cb", "exit\_cb", "close\_cb"

"in\_io", "in\_top", "in\_bot", "in\_name", "in\_buf"  
 "out\_io", "out\_name", "out\_buf", "out\_modifiable", "out\_msg"  
 "err\_io", "err\_name", "err\_buf", "err\_modifiable", "err\_msg"  
 However, at least one of stdin, stdout or stderr must be connected to the terminal. When I/O is connected to the terminal then the callback function for that part is not used.

There are extra options:

|                  |                                                                                                                                                                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "term_name"      | name to use for the buffer name, instead of the command name.                                                                                                                                                                           |
| "term_rows"      | vertical size to use for the terminal, instead of using <code>'termwinsize'</code> ; valid range is from zero to 1000                                                                                                                   |
| "term_cols"      | horizontal size to use for the terminal, instead of using <code>'termwinsize'</code>                                                                                                                                                    |
| "vertical"       | split the window vertically; <b>note</b> that other window position can be defined with command modifiers, such as <code>:belowright</code> .                                                                                           |
| "curwin"         | use the current window, do not split the window; fails if the current buffer cannot be <code>abandoned</code>                                                                                                                           |
| "hidden"         | do not open a window                                                                                                                                                                                                                    |
| "norestore"      | do not add the terminal window to a session file                                                                                                                                                                                        |
| "term_kill"      | what to do when trying to close the terminal window, see <code>term_setkill()</code>                                                                                                                                                    |
| "term_finish"    | What to do when the job is finished:<br>"close": close any windows<br>"open": open window if needed<br><b>Note</b> that "open" can be interruptive. See <code>term++close</code> and <code>term++open</code> .                          |
| "term_opencmd"   | command to use for opening the window when "open" is used for "term_finish"; must have "%d" where the buffer number goes, e.g. <code>"!0split buffer %d"</code> ; when not specified <code>"botright sbuf %d"</code> is used            |
| "term_highlight" | highlight group to use instead of "Terminal"                                                                                                                                                                                            |
| "eof_chars"      | Text to send after all buffer lines were written to the terminal. When not set <b>CTRL-D</b> is used on MS-Windows. For Python use <b>CTRL-Z</b> or <code>"exit()"</code> . For a shell use <code>"exit"</code> . A CR is always added. |
| "ansi_colors"    | A list of 16 color names or hex codes defining the ANSI palette used in GUI color modes. See <code>g:terminal_ansi_colors</code> .                                                                                                      |
| "tty_type"       | (MS-Windows only): Specify which pty to use. See <code>'termwintype'</code> for the values.                                                                                                                                             |
| "term_api"       | function name prefix for the <code>terminal-api</code> function. See <code>term_setapi()</code> .                                                                                                                                       |

Can also be used as a `method` :  
`GetCommand()->term_start()`

`term_wait({buf} [, {time}])` `term_wait()`  
Wait for pending updates of `{buf}` to be handled.  
`{buf}` is used as with `term_getsize()` .  
`{time}` is how long to wait for updates to arrive in msec. If not set then 10 msec will be used.

Can also be used as a `method` :  
`GetBufnr()->term_wait()`

---

### 3. Terminal communication

`terminal-communication`

There are several ways to communicate with the job running in a terminal:

- Use `term_sendkeys()` to send text and escape sequences from Vim to the job.
- Use the JSON API to send encoded commands from the job to Vim.
- Use the `client-server` mechanism. This works on machines with an X server and on MS-Windows.

#### Vim to job: `term_sendkeys()`

`terminal-to-job`

This allows for remote controlling the job running in the terminal. It is a one-way mechanism. The job can update the display to signal back to Vim. For example, if a shell is running in a terminal, you can do:

```
call term_sendkeys(buf, "ls *.java\<CR>")
```

This requires for the job to be in the right state where it will do the right thing when receiving the keys. For the above example, the shell must be waiting for a command to be typed.

For a job that was written for the purpose, you can use the JSON API escape sequence in the other direction. E.g.:

```
call term_sendkeys(buf, "\<Esc>]51;[\"response\"]\x07")
```

#### Job to Vim: JSON API

`terminal-api`

The job can send JSON to Vim, using a special escape sequence. The JSON encodes a command that Vim understands. Example of such a message:

```
<Esc>]51;[\"drop\", \"README.md\"]<07>
```

The body is always a list, making it easy to find the end: `]<07>`.

The `<Esc>]51;msg<07>` sequence is reserved by xterm for "Emacs shell", which is similar to what we are doing here.

Currently supported commands:

```
call {funcname} {argument}
```

Call a user defined function with `{argument}`.  
The function is called with two arguments: the buffer number of the terminal and `{argument}`, the decoded JSON argument.

By default, the function name must start with "Tapi\_" to avoid accidentally calling a function not meant to be used for the terminal API. This can be changed with `term_setapi()`. The user function should sanity check the argument. The function can use `term_sendkeys()` to send back a reply. Example in JSON:

```
["call", "Tapi_Impression", ["play", 14]]
Calls a function defined like this:
function Tapi_Impression(bufnum, arglist)
 if len(a:arglist) == 2
 echomsg "impression " .. a:arglist[0]
 echomsg "count " .. a:arglist[1]
 endif
endfunc
```

Output from `:echo` may be erased by a redraw, use `:echomsg` to be able to see it with `:messages`.

`drop {filename} [options]`

Let Vim open a file, like the `:drop` command. If `{filename}` is already open in a window, switch to that window. Otherwise open a new window to edit `{filename}`.

**Note** that both the job and Vim may change the current directory, thus it's best to use the full path.

`[options]` is only used when opening a new window. If present, it must be a Dict. Similarly to `++opt`, these entries are recognized:

|              |                                                               |
|--------------|---------------------------------------------------------------|
| "ff"         | file format: "dos", "mac" or "unix"                           |
| "fileformat" | idem                                                          |
| "enc"        | overrides <b>'fileencoding'</b>                               |
| "encoding"   | idem                                                          |
| "bin"        | sets <b>'binary'</b>                                          |
| "binary"     | idem                                                          |
| "nobin"      | resets <b>'binary'</b>                                        |
| "nobinary"   | idem                                                          |
| "bad"        | specifies behavior for bad characters, see <code>++bad</code> |

Example in JSON:

```
["drop", "path/file.txt", {"ff": "dos"}]
```

A trick to have Vim send this escape sequence:

```
exe "set t_ts=\<Esc>]51; t_fs=\x07"
let &titlestring = '["call","Tapi_TryThis",["hello",123]]'
redraw
set t_ts& t_fs&
```

Rationale: Why not allow for any command or expression? Because that might create a security problem.

`terminal-autoshellidir`

This can be used to pass the current directory from a shell to Vim.

Put this in your `.vimrc`:

```
def g:Tapi_lcd(_, path: string)
```

```

 if isdirectory(path)
 execute 'silent lcd ' .. fnameescape(path)
 endif
 enddef
enddef

```

And, in a bash init file:

```

if [[-n "$VIM_TERMINAL"]]; then
 PROMPT_COMMAND='_vim_sync_PWD'
 function _vim_sync_PWD() {
 printf '\033]51;["call", "Tapi_lcd", "%q"]\007' "$PWD"
 }
fi

```

Or, for zsh:

```

if [[-n "$VIM_TERMINAL"]]; then
 autoload -Uz add-zsh-hook
 add-zsh-hook -Uz chpwd _vim_sync_PWD
 function _vim_sync_PWD() {
 printf '\033]51;["call", "Tapi_lcd", "%q"]\007' "$PWD"
 }
fi

```

Or, for fish:

```

if test -n "$VIM_TERMINAL"
 function _vim_sync_PWD --on-variable=PWD
 printf '\033]51;["call", "Tapi_lcd", "%s"]\007' "$PWD"
 end
end

```

### Using the client-server feature

#### terminal-client-server

This only works when v:servername is not empty. If needed you can set it, before opening the terminal, with:

```
call remote_startserver('vim-server')
```

\$VIM\_SERVERNAME is set in the terminal to pass on the server name.

In the job you can then do something like:

```
vim --servername $VIM_SERVERNAME --remote +123 some_file.c
```

This will open the file "some\_file.c" and put the cursor on line 123.

### ===== 4. Remote testing terminal-testing

Most Vim tests execute a script inside Vim. For some tests this does not work, running the test interferes with the code being tested. To avoid this Vim is executed in a terminal window. The test sends keystrokes to it and inspects the resulting screen state.

### Functions

|                              |                                                     |
|------------------------------|-----------------------------------------------------|
| <code>term_sendkeys()</code> | send keystrokes to a terminal (not subject to tmap) |
| <code>term_wait()</code>     | wait for screen to be updated                       |

`term_scrape()` inspect terminal screen

---

## 5. Diffing screen dumps

`terminal-diff`

In some cases it can be bothersome to test that Vim displays the right characters on the screen. E.g. with syntax highlighting. To make this simpler it is possible to take a screen dump of a terminal and compare it to an expected screen dump.

Vim uses the window size, text, color and other attributes as displayed. The Vim screen size, font and other properties do not matter. Therefore this mechanism is portable across systems. A conventional screenshot would reflect all differences, including font size and family.

### Writing a screen dump test for Vim

`terminal-dumptest`

For an example see the `Test_syntax_c()` function in `src/testdir/test_syntax.vim`. The main parts are:

- Write a file you want to test with. This is useful for testing syntax highlighting. You can also start Vim with an empty buffer.
- Run Vim in a terminal with a specific size. The default is 20 lines of 75 characters. This makes sure the dump is always this size. The function `RunVimInTerminal()` takes care of this. Pass it the arguments for the Vim command.
- Send any commands to Vim using `term_sendkeys()`. For example:  
`call term_sendkeys(buf, ":echo &lines &columns\<CR>")`
- Check that the screen is now in the expected state, using `VerifyScreenDump()`. This expects the reference screen dump to be in the `src/testdir/dumps/` directory. Pass the name without `".dump"`. It is recommended to use the name of the test function and a sequence number, so that we know what test is using the file.
- Repeat sending commands and checking the state.
- Finally stop Vim by calling `StopVimInTerminal()`.

The first time you do this you won't have a screen dump yet. Create an empty file for now, e.g.:

```
touch src/testdir/dumps/Test_function_name_01.dump
```

The test will then fail, giving you the command to compare the reference dump and the failed dump, e.g.:

```
call term_dumpdiff("failed/Test_func.dump", "dumps/Test_func.dump")
```

Use this command in Vim, with the current directory set to `src/testdir`. Once you are satisfied with the test, move the failed dump in place of the reference:

```
:!mv failed/Test_func.dump dumps/Test_func.dump
```

### Creating a screen dump

`terminal-screendump`

To create the screen dump, run Vim (or any other program) in a terminal and

make it show the desired state. Then use the `term_dumpwrite()` function to create a screen dump file. For example:

```
:call term_dumpwrite(77, "mysyntax.dump")
```

Here "77" is the buffer number of the terminal. Use `:ls!` to see it.

You can view the screen dump with `term_dumpload()` :

```
:call term_dumpload("mysyntax.dump")
```

To verify that Vim still shows exactly the same screen, run Vim again with exactly the same way to show the desired state. Then create a screen dump again, using a different file name:

```
:call term_dumpwrite(88, "test.dump")
```

To assert that the files are exactly the same use `assert_equalfile()` :

```
call assert_equalfile("mysyntax.dump", "test.dump")
```

If there are differences then `v:errors` will contain the error message.

### Comparing screen dumps

`terminal-diffscreen`

`assert_equalfile()` does not make it easy to see what is different.

To spot the problem use `term_dumpdiff()` :

```
call term_dumpdiff("mysyntax.dump", "test.dump")
```

This will open a window consisting of three parts:

1. The contents of the first dump
2. The difference between the first and second dump
3. The contents of the second dump

You can usually see what differs in the second part. Use the `'ruler'` to relate it to the position in the first or second dump. Letters indicate the kind of difference:

|   |                                                    |
|---|----------------------------------------------------|
| X | different character                                |
| > | cursor in first but not in second                  |
| < | cursor in second but not in first                  |
| w | character width differs (single vs double width)   |
| f | foreground color differs                           |
| b | background color differs                           |
| a | attribute differs (bold, underline, reverse, etc.) |
| ? | character missing in both                          |
| + | character missing in first                         |
| - | character missing in second                        |

Alternatively, press "s" to swap the first and second dump. Do this several times so that you can spot the difference in the context of the text.

### 6. Debugging `terminal-debug` `terminal-debugger`

The Terminal debugging plugin can be used to debug a program with `gdb` and view the source code in a Vim window. Since this is completely contained inside Vim this also works remotely over an `ssh` connection.



When the `+terminal` feature is missing, the plugin will use the "prompt" buffer type, if possible. The running program will then use a newly opened terminal window. See `termdebug-prompt` below for details.

## Starting

`termdebug-starting`

Load the plugin with this command:  
`packadd termdebug`

`:Termdebug`

To start debugging use `:Termdebug`` or `:TermdebugCommand`` followed by the command name, for example:

`:Termdebug vim`

This opens two windows:

**gdb window** A terminal window in which "gdb vim" is executed. Here you can directly interact with gdb. The buffer name is "!gdb".

**program window** A terminal window for the executed program. When "run" is used in gdb the program I/O will happen in this window, so that it does not interfere with controlling gdb. The buffer name is "debugged program".

The current window is used to show the source code. When gdb pauses the source file location will be displayed, if possible. A sign is used to highlight the current position, using highlight group `debugPC`.

If the buffer in the current window is modified, another window will be opened to display the current gdb position. You can use `:Winbar`` to add a window toolbar there.

Focus the terminal of the executed program to interact with it. This works the same as any command running in a terminal window.

When the debugger ends, typically by typing "quit" in the gdb window, the two opened windows are closed.

Only one debugger can be active at a time.

`:TermdebugCommand`

If you want to give specific commands to the command being debugged, you can use the `:TermdebugCommand`` command followed by the command name and additional parameters.

`:TermdebugCommand vim --clean -c ':set nu'`

Both the `:Termdebug`` and `:TermdebugCommand`` support an optional "!" bang argument to start the command right away, without pausing at the gdb window (and cursor will be in the debugged window). For example:

`:TermdebugCommand! vim --clean`

To attach gdb to an already running executable or use a core file, pass extra arguments. E.g.:

`:Termdebug vim core`

```
:Termdebug vim 98343
```

If no argument is given, you'll end up in a gdb window, in which you need to specify which command to run using e.g. the gdb ``file`` command.

### Example session

termdebug-example

Start in the Vim "src" directory and build Vim:

```
% make
```

Make sure that debug symbols are present, usually that means that `$CFLAGS` includes `"-g"`.

Start Vim:

```
% ./vim
```

Load the termdebug plugin and start debugging Vim:

```
:packadd termdebug
:Termdebug vim
```

You should now have three windows:

- source - where you started, has a window toolbar with buttons
- gdb - you can type gdb commands here
- program - the executed program will use this window

You can use **CTRL-W CTRL-W** or the mouse to move focus between windows.

Put focus on the gdb window and type:

```
break ex_help
run
```

Vim will start running in the program window. Put focus there and type:

```
:help gui
```

Gdb will run into the `ex_help` breakpoint. The source window now shows the `ex_cmds.c` file. A red "1" marker will appear in the signcolumn where the breakpoint was set. The line where the debugger stopped is highlighted. You can now step through the program. Let's use the mouse: click on the "Next" button in the window toolbar. You will see the highlighting move as the debugger executes a line of source code.

Click "Next" a few times until the for loop is highlighted. Put the cursor on the end of `"eap->arg"`, then click "Eval" in the toolbar. You will see this displayed:

```
"eap->arg": 0x555555e68855 "gui"
```

This way you can inspect the value of local variables. You can also focus the gdb window and use a "print" command, e.g.:

```
print *eap
```

If mouse pointer movements are working, Vim will also show a balloon when the mouse rests on text that can be evaluated by gdb.

Now go back to the source window and put the cursor on the first line after the for loop, then type:

```
:Break
```

You will see a ">>" marker appear, this indicates the new breakpoint. Now click "Cont" in the toolbar and the code until the breakpoint will be executed.

You can type more advanced commands in the gdb window. For example, type:

```
watch curbuf
```

Now click "Cont" in the toolbar (or type "cont" in the gdb window). Execution will now continue until the value of "curbuf" changes, which is in do\_ecmd().

To remove this watchpoint again type in the gdb window:

```
delete 3
```

You can see the stack by typing in the gdb window:

```
where
```

Move through the stack frames, e.g. with:

```
frame 3
```

The source window will show the code, at the point where the call was made to a deeper level.

## Stepping through code

[termdebug-stepping](#)

Put focus on the gdb window to type commands there. Some common ones are:

- **CTRL-C** interrupt the program
- next execute the current line and stop at the next line
- step execute the current line and stop at the next statement, entering functions
- until execute until past the current cursor line or past a specified position or the current stack frame returns
- finish execute until leaving the current function
- where show the stack
- frame N go to the Nth stack frame
- continue continue execution

[:Run](#) [:Arguments](#)

In the window showing the source code these commands can be used to control gdb:

```
`:Run` [args] run the program with [args] or the previous arguments
```

```
`:Arguments` {args} set arguments for the next `:Run`
```

```
:Break set a breakpoint at the cursor position
```

```
:Break {position}
```

```
set a breakpoint at the specified position
```

```
:Tbreak set a temporary breakpoint at the cursor position
```

```
:Tbreak {position}
```

```
set a temporary breakpoint at the specified position
```

```
:Clear delete the breakpoint at the cursor position
```

```
:Step execute the gdb "step" command
```

```
:Over execute the gdb "next" command (`:Next` is a Vim command)
```

```
:Until execute the gdb "until" command
```

```
:Finish execute the gdb "finish" command
```

```
:Continue execute the gdb "continue" command
```

```
:Stop interrupt the program
```

If **'mouse'** is set the plugin adds a window toolbar with these entries:

```
Step `:Step`
```

```
Next `:Over`
```

```
Finish `:Finish`
```

```

Cont `:Continue`
Stop `:Stop`
Eval `:Evaluate`

```

This way you can use the mouse to perform the most common commands. You need to have the **'mouse'** option set to enable mouse clicks. See [termdebug\\_winbar](#) for configuring this toolbar.

[:Winbar](#)

You can add the window toolbar in other windows you open with:  
[:Winbar](#)

If gdb stops at a source line and there is no window currently showing the source code, a new window will be created for the source code. This also happens if the buffer in the source code window has been modified and can't be abandoned.

Gdb gives each breakpoint a number. In Vim the number shows up in the sign column, with a red background. You can use these gdb commands:

```

- info break list breakpoints
- delete N delete breakpoint N

```

You can also use the ``:Clear`` command if the cursor is in the line with the breakpoint, or use the "Clear breakpoint" right-click menu entry.

### Inspecting variables

[termdebug-variables](#) [:Evaluate](#)

```

`:Evaluate` evaluate the expression under the cursor
`K` same (see termdebug_map_K to disable)
`:Evaluate` {expr} evaluate {expr}
`:<,>Evaluate` evaluate the Visually selected text

```

This is similar to using "print" in the gdb window. You can usually shorten ``:Evaluate`` to ``:Ev``.

### Navigating stack frames

[termdebug-frames](#) [:Frame](#) [:Up](#) [:Down](#)

```

`:Frame` [frame] select frame [frame], which is a frame number,
 address, or function name (default: current frame)
`:Up` [count] go up [count] frames (default: 1; the frame that
 called the current)
`+` same (see termdebug_map_plus to disable)
`:Down` [count] go down [count] frames (default: 1; the frame called
 by the current)
`-` same (see termdebug_map_minus to disable)

```

### Other commands

[termdebug-commands](#)

```

:Gdb jump to the gdb window
:Program jump to the window with the running program
:Source jump to the window with the source code, create it if there
 isn't one
:Asm jump to the window with the disassembly, create it if there
 isn't one

```

`:Var` jump to the window with the local and argument variables, create it if there isn't one. This window updates whenever the program is stopped

## Events

[termdebug-events](#)

Four autocommands can be used:

```
au User TermdebugStartPre echomsg 'debugging starting'
au User TermdebugStartPost echomsg 'debugging started'
au User TermdebugStopPre echomsg 'debugging stopping'
au User TermdebugStopPost echomsg 'debugging stopped'
```

### TermdebugStartPre

TermdebugStartPre Before starting debugging.  
Not triggered if the debugger is already running or the debugger command cannot be executed.

### TermdebugStartPost

TermdebugStartPost After debugging has initialized.  
If a "!" bang is passed to `:Termdebug`` or `:TermdebugCommand`` the event is triggered before running the provided command in gdb.

### TermdebugStopPre

TermdebugStopPre Before debugging ends, when gdb is terminated, most likely after issuing a "quit" command in the gdb window.

### TermdebugStopPost

TermdebugStopPost After debugging has ended, gdb-related windows are closed, debug buffers wiped out and the state before the debugging was restored.

## Customizing

[termdebug-customizing](#) `g:termdebug_config`

In the past several global variables were used for configuration. These are deprecated and using the `g:termdebug_config` dictionary is preferred. When `g:termdebug_config` exists the other global variables will NOT be used.

The recommended way is to start with an empty dictionary:

```
let g:termdebug_config = {}
```

Then you can add entries to the dictionary as mentioned below. The deprecated global variable names are mentioned for completeness. If you are switching over to using `g:termdebug_config` you can find the old variable name and take over the value, then delete the deprecated variable.

## Prompt mode

[termdebug-prompt](#)

When the `+terminal` feature is not supported and on MS-Windows, gdb will run in a buffer with `'buftype'` set to "prompt". This works slightly differently:

- The gdb window will be in Insert mode while typing commands. Go to Normal mode with `<Esc>`, then you can move around in the buffer, copy/paste, etc. Go back to editing the gdb command with any command that starts Insert mode, such as ``a`` or ``i``.

- The program being debugged will run in a separate window. On MS-Windows this is a new console window. On Unix, if the `+terminal` feature is available a Terminal window will be opened to run the debugged program in.

#### `termdebug_use_prompt`

Prompt mode can be used even when the `+terminal` feature is present with:

```
let g:termdebug_config['use_prompt'] = 1
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebug_use_prompt = 1
```

### Mappings

#### `termdebug_map_K` `termdebug-mappings`

The K key is normally mapped to `:Evaluate` unless a buffer local (`:map-local`) mapping to K already exists. If you do not want this use:

```
let g:termdebug_config['map_K'] = 0
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebug_map_K = 0
```

#### `termdebug_map_minus`

The - key is normally mapped to `:Down` unless a buffer local mapping to the - key already exists. If you do not want this use:

```
let g:termdebug_config['map_minus'] = 0
```

#### `termdebug_map_plus`

The + key is normally mapped to `:Up` unless a buffer local mapping to the + key already exists. If you do not want this use:

```
let g:termdebug_config['map_plus'] = 0
```

#### `termdebug_disasm_window`

If you want the Asm window shown by default, set the "disasm\_window" flag to 1. The "disasm\_window\_height" entry can be used to set the window height:

```
let g:termdebug_config['disasm_window'] = 1
let g:termdebug_config['disasm_window_height'] = 15
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebug_disasm_window = 15
```

Any value greater than 1 will set the Asm window height to that value.

If the current window has enough horizontal space, it will be vertically split and the Asm window will be shown side by side with the source code window (and the height option won't be used).

#### `termdebug_variables_window`

If you want the Var window shown by default, set the "variables\_window" flag to 1. The "variables\_window\_height" entry can be used to set the window height:

```
let g:termdebug_config['variables_window'] = 1
let g:termdebug_config['variables_window_height'] = 15
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebug_variables_window = 15
```

Any value greater than 1 will set the Var window height to that value.

If the current window has enough horizontal space, it will be vertically split and the Var window will be shown side by side with the source code window (and the height options won't be used).

### Communication

## termdebug-communication

There is another, hidden, buffer, which is used for Vim to communicate with gdb. The buffer name is "gdb communication". Do not delete this buffer, it will break the debugger.

Gdb has some weird behavior, the plugin does its best to work around that. For example, after typing "continue" in the gdb window a **CTRL-C** can be used to interrupt the running program. But after using the MI command "-exec-continue" pressing **CTRL-C** does not interrupt. Therefore you will see "continue" being used for the ``:Continue`` command, instead of using the communication channel.

## GDB command

### g:termdebugger

To change the name of the gdb command, set "debugger" entry in `g:termdebug_config` or the `"g:termdebugger"` variable before invoking ``:Termdebug``:

```
let g:termdebug_config['command'] = "mygdb"
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebugger = "mygdb"
```

If the command needs an argument use a List:

```
let g:termdebug_config['command'] = ['rr', 'replay', '--']
```

If there is no `g:termdebug_config` you can use:

```
let g:termdebugger = ['rr', 'replay', '--']
```

Several arguments will be added to make gdb work well for the debugger.

If you want to modify them, add a function to filter the argument list:

```
let g:termdebug_config['command_filter'] = MyDebugFilter
```

If you do not want the arguments to be added, but you do need to set the "pty", use a function to add the necessary arguments:

```
let g:termdebug_config['command_add_args'] = MyAddArguments
```

The function will be called with the list of arguments so far, and a second argument that is the name of the pty.

### gdb-version

Only debuggers fully compatible with gdb will work. Vim uses the GDB/MI interface. The "new-ui" command requires gdb version 7.12 or later. If you get this error:

```
Undefined command: "new-ui". Try "help".
```

Then your gdb is too old.

## Colors

### hl-debugPC hl-debugBreakpoint

The color of the signs can be adjusted with these highlight groups:

- debugPC the current position
- debugBreakpoint a breakpoint

The defaults are, when `'background'` is "light":

```
hi debugPC term=reverse ctermbg=lightblue guibg=lightblue
hi debugBreakpoint term=reverse ctermbg=red guibg=red
```

When **'background'** is "dark":

```
hi debugPC term=reverse ctermbg=darkblue guibg=darkblue
hi debugBreakpoint term=reverse ctermbg=red guibg=red
```

## Shortcuts

[termdebug\\_shortcuts](#)

You can define your own shortcuts (mappings) to control gdb, that can work in any window, using the TermDebugSendCommand() function. Example:

```
map ,w :call TermDebugSendCommand('where')<CR>
```

The argument is the gdb command.

## Popup menu

[termdebug\\_popup](#)

By default the Termdebug plugin sets **'mousemodel'** to "popup\_setpos" and adds these entries to the popup menu:

```
Set breakpoint `:Break`
Clear breakpoint `:Clear`
Evaluate `:Evaluate`
```

If you don't want this then disable it with:

```
let g:termdebug_config['popup'] = 0
```

If there is no g:termdebug\_config you can use:

```
let g:termdebug_popup = 0
```

## Change default signs

[termdebug\\_signs](#)

Termdebug uses the hex number of the breakpoint ID in the signcolumn to represent breakpoints. if it is greater than "0xFF", then it will be displayed as "F+", due to we really only have two screen cells for the sign.

If you want to customize the breakpoint signs:

```
let g:termdebug_config['sign'] = '>>'
```

If there is no g:terminal\_config yet you can use:

```
let g:termdebug_config = {'sign': '>>'}
```

After this, breakpoints will be displayed as `>>` in the signcolumn.

## Window toolbar

[termdebug\\_winbar](#)

By default the Termdebug plugin creates a window toolbar if the mouse is enabled (see [:Winbar](#) ). If you don't want this then disable it with:

```
let g:termdebug_config['winbar'] = 0
```

## Vim window width

[termdebug\\_wide](#)

To change the width of the Vim window when debugging starts and use a vertical split:

```
let g:termdebug_config['wide'] = 163
```

If there is no g:termdebug\_config you can use:

```
let g:termdebug_wide = 163
```



This will set `'columns'` to 163 when ``:Termdebug`` is used. The value is restored when quitting the debugger.

If the wide value is set and `'columns'` is already a greater value, then a vertical split will be used without modifying `'columns'`.

Set the wide value to 1 to use a vertical split without ever changing `'columns'`. This is useful when the terminal can't be resized by Vim.



Displaying text in a floating window. [popup](#) [popup-window](#) [popupwin](#)

|                                              |                                        |
|----------------------------------------------|----------------------------------------|
| 1. Introduction                              | <a href="#">popup-intro</a>            |
| Window position and size                     | <a href="#">popup-position</a>         |
| Closing the popup window                     | <a href="#">popup-close</a>            |
| Popup buffer and window                      | <a href="#">popup-buffer</a>           |
| Terminal in popup window                     | <a href="#">popup-terminal</a>         |
| 2. Functions                                 | <a href="#">popup-functions</a>        |
| Details                                      | <a href="#">popup-function-details</a> |
| 3. Usage                                     | <a href="#">popup-usage</a>            |
| <a href="#">popup_create()</a> arguments     | <a href="#">popup_create-arguments</a> |
| Popup text properties                        | <a href="#">popup-props</a>            |
| Position popup with <a href="#">textprop</a> | <a href="#">popup-textprop-pos</a>     |
| Popup filter                                 | <a href="#">popup-filter</a>           |
| Popup callback                               | <a href="#">popup-callback</a>         |
| Popup scrollbar                              | <a href="#">popup-scrollbar</a>        |
| Popup mask                                   | <a href="#">popup-mask</a>             |
| 4. Examples                                  | <a href="#">popup-examples</a>         |

{not available if the |+popupwin| feature was disabled at compile time}

=====

1. Introduction [popup-intro](#)

We are talking about popup windows here, text that goes on top of the regular windows and is under control of a plugin. You cannot edit the text in the popup window like with regular windows.

- A popup window can be used for such things as:
- briefly show a message without overwriting the command line
  - prompt the user with a dialog
  - display contextual information while typing
  - give extra information for auto-completion

The text in the popup window can be colored with [text-properties](#) . It is also possible to use syntax highlighting.

The default color used is "Pmenu". If you prefer something else use the "highlight" argument or the **'wincolor'** option, e.g.:

```
hi MyPopupColor ctermbg=lightblue guibg=lightblue
call setwinvar(winid, '&wincolor', 'MyPopupColor')
```

**'hlsearch'** highlighting is not displayed in a popup window.

A popup window has a window-ID like other windows, but behaves differently. The size can be up to the whole Vim window and it overlaps other windows.

Popup windows can also overlap each other. The "zindex" property specifies what goes on top of what.

E366

The popup window contains a buffer, and that buffer is always associated with the popup window. The window cannot be in Normal, Visual or Insert mode, it does not get keyboard focus. You can use functions like ``setbufline()`` to change the text in the buffer. There are more differences from how this window and buffer behave compared to regular windows and buffers, see [popup-buffer](#) .

If this is not what you are looking for, check out other popup functionality:

- popup menu, see [popup-menu](#)
- balloon, see [balloon-eval](#)

## WINDOW POSITION AND SIZE

[popup-position](#)

The height of the window is normally equal to the number of, possibly wrapping, lines in the buffer. It can be limited with the "maxheight" property. You can use empty lines to increase the height or the "minheight" property.

The width of the window is normally equal to the longest visible line in the buffer. It can be limited with the "maxwidth" property. You can use spaces to increase the width or use the "minwidth" property.

By default the `'wrap'` option is set, so that no text disappears. Otherwise, if there is not enough space then the window is shifted left in order to display more text. When right-aligned the window is shifted right to display more text. The shifting can be disabled with the "fixed" property.

Vim tries to show the popup in the location you specify. In some cases, e.g. when the popup would go outside of the Vim window, it will show it somewhere nearby. E.g. if you use ``popup_atcursor()`` the popup normally shows just above the current cursor position, but if the cursor is close to the top of the Vim window it will be placed below the cursor position.

When the screen scrolls up for output of an Ex command, popups move too, so that they will not cover the output.

The current cursor position is displayed even when it is under a popup window. That way you can still see where it is, even though you cannot see the text that it is in.

## CLOSING THE POPUP WINDOW

[popup-close](#)

Normally the plugin that created the popup window is also in charge of closing it. If somehow a popup hangs around, you can close all of them with:

```
call popup_clear(1)
```

Some popups, such as notifications, close after a specified time. This can be set with the "time" property on ``popup_create()``.

Otherwise, a popup can be closed by clicking on the X in the top-right corner or by clicking anywhere inside the popup. This must be enabled with the

"close" property. It is set by default for notifications.

## POPUP BUFFER AND WINDOW

popup-buffer

If a popup function is called to create a popup from text, a new buffer is created to hold the text and text properties of the popup window. The buffer is always associated with the popup window and manipulation is restricted:

- the buffer has no name
- **'buftype'** is "popup"
- **'swapfile'** is off
- **'bufhidden'** is "hide"
- **'buflisted'** is off
- **'undolevels'** is -1: no undo at all
- all other buffer-local and window-local options are set to their Vim default value.

It is possible to change the specifically mentioned options, but anything might break then, so better leave them alone.

The window does have a cursor position, but the cursor is not displayed. In fact, the cursor in the underlying window is displayed, as if it peeks through the popup, so you can see where it is.

To execute a command in the context of the popup window and buffer use ``win_execute()``. Example:  
`call win_execute(winid, 'syntax enable')`

Options can be set on the window with ``setwinvar()``, e.g.:  
`call setwinvar(winid, '&wrap', 0)`

And options can be set on the buffer with ``setbufvar()``, e.g.:  
`call setbufvar(winbufnr(winid), '&filetype', 'java')`

You can also use ``win_execute()`` with a `":setlocal"` command.

## TERMINAL IN POPUP WINDOW

popup-terminal

A special case is running a terminal in a popup window. Many rules are then different:

- The popup window always has focus, it is not possible to switch to another window.
- When the job ends, the popup window shows the buffer in Terminal-Normal mode. Use ``:q`` to close it or use "term\_finish" value "close".
- The popup window can be closed with ``popup_close()``, the terminal buffer then becomes hidden.
- It is not possible to open a second popup window with a terminal. **E861**
- The default Pmenu color is only used for the border and padding. To change the color of the terminal itself set the Terminal highlight group before creating the terminal. Setting **'wincolor'** later can work but requires the program in the terminal to redraw everything.
- The default minimal size is 5 lines of 20 characters; Use the "minwidth" and "minheight" parameters to set a different value.
- The terminal size will grow if the program running in the terminal writes text. Set "maxheight" and "maxwidth" to restrict the size.

To run a terminal in a popup window, first create the terminal hidden. Then pass the buffer number to `popup_create()`. Example:

```
hi link Terminal Search
let buf = term_start(['picker', 'Something'], #{hidden: 1, term_finish: 'close'})
let winid = popup_create(buf, #{minwidth: 50, minheight: 20})
```

---

## 2. Functions

popup-functions

Creating a popup window:

|                                   |                                                                                   |
|-----------------------------------|-----------------------------------------------------------------------------------|
| <code>popup_create()</code>       | centered in the screen                                                            |
| <code>popup_atcursor()</code>     | just above the cursor position, closes when the cursor moves away                 |
| <code>popup_beval()</code>        | at the position indicated by v:beval_ variables, closes when the mouse moves away |
| <code>popup_notification()</code> | show a notification for three seconds                                             |
| <code>popup_dialog()</code>       | centered with padding and border                                                  |
| <code>popup_menu()</code>         | prompt for selecting an item from a list                                          |

Manipulating a popup window:

|                                 |                                         |
|---------------------------------|-----------------------------------------|
| <code>popup_hide()</code>       | hide a popup temporarily                |
| <code>popup_show()</code>       | show a previously hidden popup          |
| <code>popup_move()</code>       | change the position and size of a popup |
| <code>popup_setoptions()</code> | override options of a popup             |
| <code>popup_settext()</code>    | replace the popup buffer contents       |

Closing popup windows:

|                            |                  |
|----------------------------|------------------|
| <code>popup_close()</code> | close one popup  |
| <code>popup_clear()</code> | close all popups |

Filter functions:

|                                   |                                    |
|-----------------------------------|------------------------------------|
| <code>popup_filter_menu()</code>  | select from a list of items        |
| <code>popup_filter_yesno()</code> | blocks until 'y' or 'n' is pressed |

Other:

|                                 |                                         |
|---------------------------------|-----------------------------------------|
| <code>popup_getoptions()</code> | get current options for a popup         |
| <code>popup_getpos()</code>     | get actual position and size of a popup |
| <code>popup_locate()</code>     | find popup window at a screen position  |
| <code>popup_list()</code>       | get list of all popups                  |

## DETAILS

popup-function-details

`popup_atcursor({what}, {options})`

popup\_atcursor()

Show the `{what}` above the cursor, and close it when the cursor moves. This works like:

```
call popup_create({what}, #{
 \ pos: 'botleft',
 \ line: 'cursor-1',
 \ col: 'cursor',
 \ moved: 'WORD',
 \ })
```

Use `{options}` to change the properties.

If "pos" is passed as "topleft" then the default for "line" becomes "cursor+1".

Can also be used as a `method` :  
`GetText()->popup_atcursor({})`

`popup_beval({what}, {options})` `popup_beval()`  
Show the `{what}` above the position from `'ballooneval'` and close it when the mouse moves. This works like:  
`let pos = screenpos(v:beval_winnr, v:beval_lnum, v:beval_col)`  
`call popup_create({what}, #{`  
    `\ pos: 'botleft',`  
    `\ line: pos.row - 1,`  
    `\ col: pos.col,`  
    `\ mousemoved: 'WORD',`  
    `\ })`

Use `{options}` to change the properties.  
See `popup_beval_example` for an example.

Can also be used as a `method` :  
`GetText()->popup_beval({})`

`popup_clear([force])` `popup_clear()`  
Emergency solution to a misbehaving plugin: close all popup windows for the current tab and global popups. Close callbacks are not invoked. When `{force}` is not present this will fail if the current window is a popup. When `{force}` is present and `TRUE` the popup is also closed when it is the current window. If a terminal is running in a popup it is killed.

`popup_close({id} [, {result}])` `popup_close()`  
Close popup `{id}`. The window and the associated buffer will be deleted.  
  
If the popup has a callback it will be called just before the popup window is deleted. If the optional `{result}` is present it will be passed as the second argument of the callback. Otherwise zero is passed to the callback.

Can also be used as a `method` :  
`GetPopup()->popup_close()`

`popup_create({what}, {options})` `popup_create()`  
Open a popup window showing `{what}`, which is either: `E450`  
- a buffer number  
- a string  
- a list of strings  
- a list of text lines with text properties

When `{what}` is not a buffer number, a buffer is created with `'buftype'` set to "popup". That buffer will be wiped out once the popup closes.

if `{what}` is a buffer number and loading the buffer runs into an existing swap file, it is silently opened read-only, as if a `SwapExists` autocommand had set `v:swapchoice` to 'o'. This is because we assume the buffer is only used for viewing.

`{options}` is a dictionary with many possible entries. See `popup_create-arguments` for details.

Returns a window-ID, which can be used with other popup functions. Use `'winbufnr()'` to get the number of the buffer in the window:

```
let winid = popup_create('hello', {})
let bufnr = winbufnr(winid)
call setbufline(bufnr, 2, 'second line')
```

In case of failure zero is returned.

Can also be used as a `method` :  
`GetText()->popup_create({})`

`popup_dialog({what}, {options})` `popup_dialog()`

Just like `popup_create()` but with these default options:

```
call popup_create({what}, #{
 \ pos: 'center',
 \ zindex: 200,
 \ drag: 1,
 \ border: [],
 \ padding: [],
 \ mapping: 0,
 \ })
```

Use `{options}` to change the properties. E.g. add a `'filter'` option with value 'popup\_filter\_yesno'. Example:

```
call popup_create('do you want to quit (Yes/no)?', #{
 \ filter: 'popup_filter_yesno',
 \ callback: 'QuitCallback',
 \ })
```

By default the dialog can be dragged, so that text below it can be read if needed.

Can also be used as a `method` :  
`GetText()->popup_dialog({})`

`popup_filter_menu({id}, {key})` `popup_filter_menu()`

Filter that can be used for a popup. These keys can be used:

|   |                   |                          |
|---|-------------------|--------------------------|
| j | <Down> <C-N>      | select item below        |
| k | <Up> <C-P>        | select item above        |
|   | <Space> <Enter>   | accept current selection |
| x | Esc <b>CTRL-C</b> | cancel the menu          |



Other keys are ignored.  
Always returns `v:true` .

A match is set on that line to highlight it, see [popup\\_menu\(\)](#) .

When the current selection is accepted the "callback" of the popup menu is invoked with the index of the selected line as the second argument. The first entry has index one. Cancelling the menu invokes the callback with -1.

To add shortcut keys, see the example here: [popup\\_menu-shortcut-example](#)

`popup_filter_yesno({id}, {key})` [popup\\_filter\\_yesno\(\)](#)  
Filter that can be used for a popup. It handles only the keys 'y', 'Y' and 'n' or 'N'. Invokes the "callback" of the popup menu with the 1 for 'y' or 'Y' and zero for 'n' or 'N' as the second argument. Pressing Esc and 'x' works like pressing 'n'. **CTRL-C** invokes the callback with -1. Other keys are ignored.  
See the example here: [popup\\_dialog-example](#)

`popup_findecho()` [popup\\_findecho\(\)](#)  
Get the `window-ID` for the popup that shows messages for the `:echowindow`` command. Return zero if there is none.  
Mainly useful to hide the popup.

`popup_findinfo()` [popup\\_findinfo\(\)](#)  
Get the `window-ID` for the popup info window, as it used by the popup menu. See [complete-popup](#) . The info popup is hidden when not used, it can be deleted with [popup\\_clear\(\)](#) and [popup\\_close\(\)](#) . Use [popup\\_show\(\)](#) to reposition it to the item in the popup menu.  
Returns zero if there is none.

`popup_findpreview()` [popup\\_findpreview\(\)](#)  
Get the `window-ID` for the popup preview window.  
Return zero if there is none.

`popup_getoptions({id})` [popup\\_getoptions\(\)](#)  
Return the `{options}` for popup `{id}` in a Dict.  
A zero value means the option was not set. For "zindex" the default value is returned, not zero.

The "moved" entry is a list with line number, minimum and maximum column, `[0, 0, 0]` when not set.

The "mousemoved" entry is a list with screen row, minimum and maximum screen column, `[0, 0, 0]` when not set.

"firstline" is the property set on the popup, unlike the "firstline" obtained with `popup_getpos()` which is the actual buffer line at the top of the popup window.

"border" and "padding" are not included when all values are zero. When all values are one then an empty list is included.

"borderhighlight" is not included when all values are empty. "scrollbarhighlight" and "thumbhighlight" are only included when set.

"tabpage" will be -1 for a global popup, zero for a popup on the current tabpage and a positive number for a popup on another tabpage.

"textprop", "textpropid" and "textpropwin" are only present when "textprop" was set.

If popup window `{id}` is not found an empty Dict is returned.

Can also be used as a `method` :

`GetPopup()->popup_getoptions()`

`popup_getpos({id})`

`popup_getpos()`

Return the position and size of popup `{id}`. Returns a Dict with these entries:

|                          |                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------|
| <code>col</code>         | screen column of the popup, one-based                                                        |
| <code>line</code>        | screen line of the popup, one-based                                                          |
| <code>width</code>       | width of the whole popup in screen cells                                                     |
| <code>height</code>      | height of the whole popup in screen cells                                                    |
| <code>core_col</code>    | screen column of the text box                                                                |
| <code>core_line</code>   | screen line of the text box                                                                  |
| <code>core_width</code>  | width of the text box in screen cells                                                        |
| <code>core_height</code> | height of the text box in screen cells                                                       |
| <code>firstline</code>   | line of the buffer at top (1 unless scrolled)<br>(not the value of the "firstline" property) |
| <code>lastline</code>    | line of the buffer at the bottom (updated when<br>the popup is redrawn)                      |
| <code>scrollbar</code>   | non-zero if a scrollbar is displayed                                                         |
| <code>visible</code>     | one if the popup is displayed, zero if hidden                                                |

**Note** that these are the actual screen positions. They differ from the values in ``popup_getoptions()`` for the sizing and positioning mechanism applied.

The "core\_" values exclude the padding and border.

If popup window `{id}` is not found an empty Dict is returned.

Can also be used as a `method` :

`GetPopup()->popup_getpos()`

popup\_hide({id}) popup\_hide()  
If {id} is a displayed popup, hide it now. If the popup has a filter it will not be invoked for so long as the popup is hidden.  
If window {id} does not exist nothing happens. If window {id} exists but is not a popup window an error is given. E993  
If popup window {id} contains a terminal an error is given.  
  
Can also be used as a method :  
    GetPopup()->popup\_hide()

popup\_list() popup\_list()  
Return a List with the window-ID of all existing popups.

popup\_locate({row}, {col}) popup\_locate()  
Return the window-ID of the popup at screen position {row} and {col}. If there are multiple popups the one with the highest zindex is returned. If there are no popups at this position then zero is returned.

popup\_menu({what}, {options}) popup\_menu()  
Show the {what} near the cursor, handle selecting one of the items with cursorkeys, and close it an item is selected with Space or Enter. {what} should have multiple lines to make this useful. This works like:  
    call popup\_create({what}, #{  
        \ pos: 'center',  
        \ zindex: 200,  
        \ drag: 1,  
        \ wrap: 0,  
        \ border: [],  
        \ cursorline: 1,  
        \ padding: [0,1,0,1],  
        \ filter: 'popup\_filter\_menu',  
        \ mapping: 0,  
        \ })

The current line is highlighted with a match using "PopupSelected", or "PmenuSel" if that is not defined.

Use {options} to change the properties. Should at least set "callback" to a function that handles the selected item.

Example:

```
func ColorSelected(id, result)
 use a:result
endfunc
call popup_menu(['red', 'green', 'blue'], #{
 \ callback: 'ColorSelected',
 \ })
```

Can also be used as a method :  
    GetChoices()->popup\_menu({})

`popup_move({id}, {options})` `popup_move()`  
 Move popup `{id}` to the position specified with `{options}`.  
`{options}` may contain the items from `popup_create()` that specify the popup position:
 

- line
- col
- pos
- maxheight
- minheight
- maxwidth
- minwidth
- fixed

 For `{id}` see ``popup_hide()``.  
 For other options see `popup_setoptions()` .

Can also be used as a `method` :  
`GetPopup()->popup_move(options)`

`popup_notification({what}, {options})` `popup_notification()`  
 Show the `{what}` for 3 seconds at the top of the Vim window.  
 This works like:
 

```

call popup_create({what}, #{
 \ line: 1,
 \ col: 10,
 \ minwidth: 20,
 \ time: 3000,
 \ tabpage: -1,
 \ zindex: 300,
 \ drag: 1,
 \ highlight: 'WarningMsg',
 \ border: [],
 \ close: 'click',
 \ padding: [0,1,0,1],
 \ })

```

The `PopupNotification` highlight group is used instead of `WarningMsg` if it is defined.

Without the `+timers` feature the popup will not disappear automatically, the user has to click in it.

The position will be adjusted to avoid overlap with other notifications.

Use `{options}` to change the properties.

Can also be used as a `method` :  
`GetText()->popup_notification({})`

`popup_setoptions({id}, {options})` `popup_setoptions()`  
 Override options in popup `{id}` with entries in `{options}`.  
 These options can be set:

border  
borderchars  
borderhighlight  
callback  
close  
cursorline  
drag  
filter  
firstline  
flip  
highlight  
mapping  
mask  
moved  
padding  
resize  
scrollbar  
scrollbarhighlight  
thumbhighlight  
time  
title  
wrap  
zindex

The options from `popup_move()` can also be used. Generally, setting an option to zero or an empty string resets it to the default value, but there are exceptions. For "hidden" use `popup_hide()` and `popup_show()`. "tabpage" cannot be changed.

Can also be used as a `method` :  
`GetPopup()->popup_setopt(options)`

`popup_settext({id}, {text})` `popup_settext()`  
Set the text of the buffer in popup win `{id}`. `{text}` is the same as supplied to `popup_create()`, except that a buffer number is not allowed. Does not change the window size or position, other than caused by the different text.

Can also be used as a `method` :  
`GetPopup()->popup_settext('hello')`

`popup_show({id})` `popup_show()`  
If `{id}` is a hidden popup, show it now. For `{id}` see `popup_hide()`. If `{id}` is the info popup it will be positioned next to the current popup menu item.

---

### 3. Usage `popup-usage`

## POPUP\_CREATE() ARGUMENTS

## popup\_create-arguments

The first argument of `popup_create()` (and the second argument to `popup_settext()`) specifies the text to be displayed, and optionally text properties. It is in one of four forms: [E1284](#)

- a buffer number
- a string
- a list of strings
- a list of dictionaries, where each dictionary has these entries:

|       |                                                                                                                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| text  | String with the text to display.                                                                                                                                                                                                     |
| props | A list of text properties. Optional.<br>Each entry is a dictionary, like the third argument of <code>prop_add()</code> , but specifying the column in the dictionary with a "col" entry, see below:<br><a href="#">popup-props</a> . |

If you want to create a new buffer yourself use `bufadd()` and pass the buffer number to `popup_create()`.

The second argument of `popup_create()` is a dictionary with options:

|             |                                                                                                                                                                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| line        | Screen line where to position the popup. Can use a number or "cursor", "cursor+1" or "cursor-1" to use the line of the cursor and add or subtract a number of lines. If omitted or zero the popup is vertically centered. The first line is 1.<br>When using "textprop" the number is relative to the text property and can be negative.          |
| col         | Screen column where to position the popup. Can use a number or "cursor" to use the column of the cursor, "cursor+9" or "cursor-9" to add or subtract a number of columns. If omitted or zero the popup is horizontally centered. The first column is 1.<br>When using "textprop" the number is relative to the text property and can be negative. |
| pos         | "topleft", "topright", "botleft" or "botright": defines what corner of the popup "line" and "col" are used for. When not set "topleft" is used.<br>Alternatively "center" can be used to position the popup in the center of the Vim window, in which case "line" and "col" are ignored.                                                          |
| posinvert   | When FALSE the value of "pos" is always used. When TRUE (the default) and the popup does not fit vertically and there is more space on the other side then the popup is placed on the other side of the position indicated by "line".                                                                                                             |
| textprop    | When present the popup is positioned next to a text property with this name and will move when the text property moves. Use an empty string to remove. See <a href="#">popup-textprop-pos</a> .                                                                                                                                                   |
| textpropwin | What window to search for the text property. When omitted or invalid the current window is used. Used when "textprop" is present.                                                                                                                                                                                                                 |
| textpropid  | Used to identify the text property when "textprop" is present. Use zero to reset.                                                                                                                                                                                                                                                                 |

|           |                                                                                                                                                                                                                                                                                                                                               |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fixed     | When FALSE (the default), and: <ul style="list-style-type: none"> <li>- "pos" is "botleft" or "topleft", and</li> <li>- "wrap" is off, and</li> <li>- the popup would be truncated at the right edge of the screen, then</li> </ul> the popup is moved to the left so as to fit the contents on the screen. Set to TRUE to disable this.      |
| flip      | When TRUE (the default) and the position is relative to the cursor, flip to below or above the cursor to avoid overlap with the <code>popupmenu-completion</code> or another popup with a higher "zindex". When there is no space above/below the cursor then show the popup to the side of the popup or popup menu.<br>{not implemented yet} |
| maxheight | Maximum height of the contents, excluding border and padding.                                                                                                                                                                                                                                                                                 |
| minheight | Minimum height of the contents, excluding border and padding.                                                                                                                                                                                                                                                                                 |
| maxwidth  | Maximum width of the contents, excluding border, padding and scrollbar.                                                                                                                                                                                                                                                                       |
| minwidth  | Minimum width of the contents, excluding border, padding and scrollbar.                                                                                                                                                                                                                                                                       |
| firstline | First buffer line to display. When larger than one it looks like the text scrolled up. When out of range the last buffer line will at the top of the window. Set to zero to leave the position as set by commands. Also see "scrollbar".                                                                                                      |
| hidden    | When TRUE the popup exists but is not displayed; use <code>`popup_show()`</code> to unhide it.                                                                                                                                                                                                                                                |
| tabpage   | When -1: display the popup on all tab pages.<br>When 0 (the default): display the popup on the current tab page.<br>Otherwise the number of the tab page the popup is displayed on; when invalid the popup is not created and an error is given. E997                                                                                         |
| title     | Text to be displayed above the first item in the popup, on top of any border. If there is no top border one line of padding is added to put the title on. You might want to add one or more spaces at the start and end as padding.                                                                                                           |
| wrap      | TRUE to make the lines wrap (default TRUE).                                                                                                                                                                                                                                                                                                   |
| drag      | TRUE to allow the popup to be dragged with the mouse by grabbing at the border. Has no effect if the popup does not have a border. As soon as dragging starts and "pos" is "center" it is changed to "topleft".                                                                                                                               |
| dragall   | TRUE to allow the popup to be dragged from every position. Makes it very difficult to select text in the popup.                                                                                                                                                                                                                               |
| resize    | TRUE to allow the popup to be resized with the mouse by grabbing at the bottom right corner. Has no effect if the popup does not have a border.                                                                                                                                                                                               |
| close     | When "button" an X is displayed in the top-right, on top of any border, padding or text. When clicked on                                                                                                                                                                                                                                      |

the X the popup will close. Any callback is invoked with the value -2.

When "click" any mouse click in the popup will close it.

When "none" (the default) mouse clicks do not close the popup window.

**highlight** Highlight group name to use for the text, stored in the **'wincolor'** option.

**padding** List with numbers, defining the padding above/right/below/left of the popup (similar to CSS). An empty list uses a padding of 1 all around. The padding goes around the text, inside any border. Padding uses the **'wincolor'** highlight.  
Example: [1, 2, 1, 3] has 1 line of padding above, 2 columns on the right, 1 line below and 3 columns on the left.

**border** List with numbers, defining the border thickness above/right/below/left of the popup (similar to CSS). Only values of zero and non-zero are currently recognized. An empty list uses a border all around.

**borderhighlight** List of highlight group names to use for the border. When one entry it is used for all borders, otherwise the highlight for the top/right/bottom/left border.  
Example: ['TopColor', 'RightColor', 'BottomColor', 'LeftColor']

**borderchars** List with characters, defining the character to use for the top/right/bottom/left border. Optionally followed by the character to use for the topleft/topright/botright/botleft corner.  
Example: ['- ', '|', '- ', '|', '┌', '┐', '└', '┘']  
When the list has one character it is used for all. When the list has two characters the first is used for the border lines, the second for the corners. By default a double line is used all around when **'encoding'** is "utf-8" and **'ambiwidth'** is "single", otherwise ASCII characters are used.

**scrollbar** 1 or true: show a scrollbar when the text doesn't fit.  
zero: do not show a scrollbar. Default is non-zero.  
Also see [popup-scrollbar](#) .

**scrollbarhighlight** Highlight group name for the scrollbar. The background color is what matters. When not given then PmenuSbar is used.

**thumbhighlight** Highlight group name for the scrollbar thumb. The background color is what matters. When not given then PmenuThumb is used.

**zindex** Priority for the popup, default 50. Minimum value is 1, maximum value is 32000.

**mask** A list of lists with coordinates, defining parts of the popup that are transparent. See [popup-mask](#) .

**time** Time in milliseconds after which the popup will close. When omitted [popup\\_close\(\)](#) must be used.

**moved** Specifies to close the popup if the cursor moved:  
- "any": if the cursor moved at all  
- "word": if the cursor moved outside `<word>`



- "WORD": if the cursor moved outside `<cWORD>`
- "expr": if the cursor moved outside `<cexpr>`
- [{start}, {end}]: if the cursor moved before column {start} or after {end}
- [{lnum}, {start}, {end}]: if the cursor moved away from line {lnum}, before column {start} or after {end}
- [0, 0, 0] do not close the popup when the cursor moves

The popup also closes if the cursor moves to another line or to another window.

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mousemoved | Like "moved" but referring to the mouse pointer position                                                                                                                                                                                                                                                                                                                                                                        |
| cursorline | TRUE: Highlight the cursor line. Also scrolls the text to show this line (only works properly when 'wrap' is off).<br>zero: Do not highlight the cursor line. Default is zero, except for <code>popup_menu()</code> .                                                                                                                                                                                                           |
| filter     | A callback that can filter typed characters, see <code>popup-filter</code> .                                                                                                                                                                                                                                                                                                                                                    |
| mapping    | Allow for key mapping. When FALSE and the popup is visible and has a filter callback key mapping is disabled. Default value is TRUE.                                                                                                                                                                                                                                                                                            |
| filtermode | In which modes the filter is used (same flags as with <code>hasmapto()</code> plus "a"): <ul style="list-style-type: none"> <li>n Normal mode</li> <li>v Visual and Select mode</li> <li>x Visual mode</li> <li>s Select mode</li> <li>o Operator-pending mode</li> <li>i Insert mode</li> <li>l Language-Argument ("r", "f", "t", etc.)</li> <li>c Command-line mode</li> <li>a all modes</li> </ul> The default value is "a". |
| callback   | A callback that is called when the popup closes, e.g. when using <code>popup_filter_menu()</code> , see <code>popup-callback</code> .                                                                                                                                                                                                                                                                                           |

Depending on the "zindex" the popup goes under or above other popups. The completion menu (`popup-menu`) has zindex 100. For messages that occur for a short time the suggestion is to use zindex 1000.

By default text wraps, which causes a line in `{lines}` to occupy more than one screen line. When "wrap" is FALSE then the text outside of the popup or outside of the Vim window will not be displayed, thus truncated.

## POPUP TEXT PROPERTIES

`popup-props`

These are similar to the third argument of `prop_add()` except:

- "lnum" is always the current line in the list
- "bufnr" is always the buffer of the popup
- "col" is in the Dict instead of a separate argument

So we get:

|          |                                                                                                                                       |
|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| col      | starting column, counted in bytes, use one for the first column.                                                                      |
| length   | length of text in bytes; can be zero                                                                                                  |
| end_lnum | line number for the end of the text                                                                                                   |
| end_col  | column just after the text; not used when "length" is present; when {col} and "end_col" are equal, this is a zero-width text property |
| id       | user defined ID for the property; when omitted zero is used                                                                           |
| type     | name of the text property type, as added with <code>prop_type_add()</code>                                                            |

## POSITION POPUP WITH TEXTPROP

popup-textprop-pos

Positioning a popup next to a text property causes the popup to move when text is inserted or deleted. The popup functions like a tooltip.

These steps are needed to make this work:

- Define a text property type, it defines the name.  
`call prop_type_add('popupMarker', {})`

- Place a text property at the desired text:  
`let lnum = {line of the text}`  
`let col = {start column of the text}`  
`let len = {length of the text}`  
`let propId = {arbitrary but unique number}`  
`call prop_add(lnum, col, #{`  
`\ length: len,`  
`\ type: 'popupMarker',`  
`\ id: propId,`  
`\ })`

- Create a popup:  
`let winid = popup_create('the text', #{`  
`\ pos: 'botleft',`  
`\ textprop: 'popupMarker',`  
`\ textpropid: propId,`  
`\ border: [],`  
`\ padding: [0,1,0,1],`  
`\ close: 'click',`  
`\ })`

By default the popup is positioned at the corner of the text, opposite of the "pos" specified for the popup. Thus when the popup uses "botleft", the bottom-left corner of the popup is positioned next to the top-right corner of the text property:

```

+-----+
| the text |
+-----+
just some PROPERTY as an example

```

Here the text property is on "PROPERTY". Move the popup to the left by

passing a negative "col" value to `popup_create()`. With "col: -5" you get:

```
+-----+
| the text |
+-----+
```

just some PROPERTY as an example

If the text property moves out of view then the popup will be hidden.  
If the window for which the popup was defined is closed, the popup is closed.

If the popup cannot fit in the desired position, it may show at a nearby position.

Some hints:

- To avoid collision with other plugins the text property type name has to be unique. You can also use the "bufnr" item to make it local to a buffer.
- You can leave out the text property ID if there is only ever one text property visible.
- The popup may be in the way of what the user is doing, making it close with a click, as in the example above, helps for that.
- If the text property is removed the popup is closed. Use something like this:

```
call prop_remove({type: 'popupMarker', id: propId})
```

## POPUP FILTER

## popup-filter

A callback that gets any typed keys while a popup is displayed. The filter is not invoked when the popup is hidden.

The filter can return TRUE to indicate the key has been handled and is to be discarded, or FALSE to let Vim handle the key as usual in the current state. In case it returns FALSE and there is another popup window visible, that filter is also called. The filter of the popup window with the highest zindex is called first.

The filter function is called with two arguments: the ID of the popup and the key as a string, e.g.:

```
func MyFilter(winid, key)
 if a:key == "\<F2>"
 " do something
 return 1
 endif
 if a:key == 'x'
 call popup_close(a:winid)
 return 1
 endif
 return 0
endfunc
```

## popup-filter-mode

The "filtermode" property can be used to specify in what mode the filter is invoked. The default is "a": all modes. When using "nvi" Command-line mode is not included, so that any command typed on the command line is not filtered. However, to get to Command-line mode the filter must not consume

:". Just like it must not consume "v" to allow for entering Visual mode.

#### popup-mapping

Normally the key is what results after any mapping, since the keys pass on as normal input if the filter does not use it. If the filter consumes all the keys, set the "mapping" property to zero so that mappings do not get in the way. This is default for `popup_menu()` and `popup_dialog()` .

Some recommended key actions:

|             |                                                  |
|-------------|--------------------------------------------------|
| x           | close the popup (see <a href="#">note</a> below) |
| cursor keys | select another entry                             |
| Tab         | accept current suggestion                        |

When **CTRL-C** is pressed the popup is closed, the filter will not be invoked.

A mouse click arrives as `<LeftMouse>`. The coordinates can be obtained with `getmousepos()` .

Vim provides standard filters `popup_filter_menu()` and `popup_filter_yesno()` .

Keys coming from a `:normal` command do not pass through the filter. This can be used to move the cursor in a popup where the "cursorline" option is set:

```
call win_execute(winid, 'normal! 10Gzz')
```

Keys coming from `feedkeys()` are passed through the filter.

**Note** that "x" is the normal way to close a popup. You may want to use Esc, but since many keys start with an Esc character, there may be a delay before Vim recognizes the Esc key. If you do use Esc, it is recommended to set the `'timeoutlen'` option to 100 and set `'timeout'` and/or `'ttimeout'`.

#### popup-filter-errors

If the filter function can't be called, e.g. because the name is wrong, then the popup is closed. If the filter causes an error then it is assumed to return zero. If this happens three times in a row the popup is closed. If the popup gives errors fewer than 10% of the calls then it won't be closed.

## POPUP CALLBACK

#### popup-callback

A callback that is invoked when the popup closes.

The callback is invoked with two arguments: the ID of the popup window and the result, which could be an index in the popup lines, or whatever was passed as the second argument of `popup_close()`.

If the popup is force-closed, e.g. because the cursor moved or **CTRL-C** was pressed, the number -1 is passed to the callback.

Example:

```
func SelectedColor(id, result)
 echo 'choice made: ' .. a:result
endfunc
```

## POPUP SCROLLBAR

popup-scrollbar

If the text does not fit in the popup a scrollbar is displayed on the right of the window. This can be disabled by setting the "scrollbar" option to zero. When the scrollbar is displayed mouse scroll events, while the mouse pointer is on the popup, will cause the text to scroll up or down as you would expect. A click in the upper half of the scrollbar will scroll the text down one line. A click in the lower half will scroll the text up one line. However, this is limited so that the popup does not get smaller.

## POPUP MASK

popup-mask

To minimize the text that the popup covers, parts of it can be made transparent. This is defined by a "mask" which is a list of lists, where each list has four numbers:

|         |                                                                                                                           |
|---------|---------------------------------------------------------------------------------------------------------------------------|
| col     | start column, positive for counting from the left, 1 for leftmost, negative for counting from the right, -1 for rightmost |
| endcol  | last column, like "col"                                                                                                   |
| line    | start line, positive for counting from the top, 1 for top, negative for counting from the bottom, -1 for bottom           |
| endline | end line, like "line"                                                                                                     |

For example, to make the last 10 columns of the last line transparent:

```
[[-10, -1, -1, -1]]
```

To make the four corners transparent:

```
[[1, 1, 1, 1], [-1, -1, 1, 1], [1, 1, -1, -1], [-1, -1, -1, -1]]
```

---

## 4. Examples

popup-examples

These examples use `Vim9` script.

popup\_dialog-example

Prompt the user to press y/Y or n/N:

```
popup_dialog('Continue? y/n', {
 filter: 'popup_filter_yesno',
 callback: (id, result) => {
 if result == 1
 echomsg "'y' or 'Y' was pressed"
 else
 echomsg "'y' or 'Y' was NOT pressed"
 endif
 },
 padding: [2, 4, 2, 4],
})
```

popup\_menu-shortcut-example

Extend `popup_filter_menu()` with shortcut keys:

```

popup_menu(['Save', 'Cancel', 'Discard'], {
 callback: (_, result) => {
 echo 'dialog result is' result
 },
 filter: (id, key) => {
 # Handle shortcuts
 if key == 'S' || key == 's'
 popup_close(id, 1)
 elseif key == 'C' || key == 'c'
 popup_close(id, 2)
 elseif key == 'D' || key == 'd'
 popup_close(id, 3)
 else
 # No shortcut, pass to generic filter
 return popup_filter_menu(id, key)
 endif
 return true
 },
})

```

popup\_beval\_example

Example for using a popup window for **'ballooneval'**:

```

set ballooneval balloonevalterm
set balloonexpr=BalloonExpr()
var winid: number
var last_text: string

def BalloonExpr(): string
 # here you would use "v:beval_text" to lookup something interesting
 var text = v:beval_text
 if winid > 0 && popup_getpos(winid) != null_dict
 # previous popup window still shows
 if text == last_text
 # still the same text, keep the existing popup
 return null_string
 endif
 popup_close(winid)
 endif

 winid = popup_beval(text, {})
 last_text = text
 return null_string
enddef

```

If the text has to be obtained asynchronously return an empty string from the expression function and call popup\_beval() once the text is available. In this example simulated with a timer callback:

```

set ballooneval balloonevalterm
set balloonexpr=BalloonExpr()
var winid: number
var last_text: string

```

```

def BalloonExpr(): string
 var text = v:beval_text
 if winid > 0 && popup_getpos(winid) != null_dict
 # previous popup window still shows
 if text == last_text
 # still the same text, keep the existing popup
 return null_string
 endif
 popup_close(winid)
 endif

 # Simulate an asynchronous lookup that takes half a second for the
 # text to display.
 last_text = text
 timer_start(500, 'ShowPopup')
 return null_string
enddef

def ShowPopup(timerid: number)
 winid = popup_beval('Result: ' .. last_text, {})
enddef

```





Vim9 script commands and expressions.

Vim9 vim9

Most expression help is in [eval.txt](#) . This file is about the new syntax and features in Vim9 script.

- |                                 |                                  |
|---------------------------------|----------------------------------|
| 1. What is Vim9 script?         | <a href="#">Vim9-script</a>      |
| 2. Differences                  | <a href="#">vim9-differences</a> |
| 3. New style functions          | <a href="#">fast-functions</a>   |
| 4. Types                        | <a href="#">vim9-types</a>       |
| 5. Namespace, Import and Export | <a href="#">vim9script</a>       |
| 6. Classes and interfaces       | <a href="#">vim9-classes</a>     |
| 9. Rationale                    | <a href="#">vim9-rationale</a>   |

---

## 1. What is Vim9 script?

Vim9-script

Vim script has been growing over time, while preserving backwards compatibility. That means bad choices from the past often can't be changed and compatibility with Vi restricts possible solutions. Execution is quite slow, each line is parsed every time it is executed.

The main goal of Vim9 script is to drastically improve performance. This is accomplished by compiling commands into instructions that can be efficiently executed. An increase in execution speed of 10 to 100 times can be expected.

A secondary goal is to avoid Vim-specific constructs and get closer to commonly used programming languages, such as JavaScript, TypeScript and Java.

The performance improvements can only be achieved by not being 100% backwards compatible. For example, making function arguments available in the "a:" dictionary adds quite a lot of overhead. In a Vim9 function this dictionary is not available. Other differences are more subtle, such as how errors are handled.

The Vim9 script syntax and semantics are used in:

- a function defined with the `:def` command
- a script file where the first command is `vim9script`
- an autocommand defined in the context of the above
- a command prefixed with the `vim9cmd` command modifier

When using `:function` in a Vim9 script file the legacy syntax is used, with the highest `scriptversion` . However, this can be confusing and is therefore discouraged.

Vim9 script and legacy Vim script can be mixed. There is no requirement to rewrite old scripts, they keep working as before. You may want to use a few ``:def`` functions for code that needs to be fast.

```
:vim9[cmd] {cmd} :vim9 :vim9cmd E1164
Evaluate and execute {cmd} using Vim9 script syntax and semantics. Useful when typing a command and in a legacy script or function.
```

```
:leg[acy] {cmd} :leg :legacy E1189 E1234
Evaluate and execute {cmd} using legacy script syntax and semantics. Only useful in a Vim9 script or a :def function.
Note that {cmd} cannot use local variables, since it is parsed with legacy expression syntax.
```

---

## 2. Differences from legacy Vim script

[vim9-differences](#)

### Overview

[E1146](#)

Brief summary of the differences you will most often encounter when using Vim9 script and ``:def`` functions; details are below:

- Comments start with #, not "

```
echo "hello" # comment
```
- Using a backslash for line continuation is hardly ever needed:

```
echo "hello "
 .. yourName
 .. ", how are you?"
```
- White space is required in many places to improve readability.
- Assign values without ``:let`` [E1126](#) , declare variables with ``:var``:

```
var count = 0
count += 3
```
- Constants can be declared with ``:final`` and ``:const``:

```
final matches = [] # add to the list later
const names = ['Betty', 'Peter'] # cannot be changed
```
- ``:final`` cannot be used as an abbreviation of ``:finally``.
- Variables and functions are script-local by default.
- Functions are declared with argument types and return type:

```
def CallMe(count: number, message: string): bool
```
- Call functions without ``:call``:

```
writefile(['done'], 'file.txt')
```
- You cannot use old Ex commands:

```
`:Print`
`:append`
`:change`
`:d` directly followed by 'd' or 'p'.
`:insert`
`:k`
`:mode`
`:open`
`:s` with only flags
`:t`
`:xit`
```

- Some commands, especially those used for flow control, cannot be shortened. E.g., ``:throw`` cannot be written as ``:th``. [vim9-no-shorten](#)
- You cannot use curly-braces names.
- A range before a command must be prefixed with a colon:
 

```
:%s/this/that
```
- Executing a register with "@r" does not work, you can prepend a colon or use ``:exe``:
 

```
:exe @a
```
- Unless mentioned specifically, the highest `scriptversion` is used.
- When defining an expression mapping, the expression will be evaluated in the context of the script where it was defined.
- When indexing a string the index is counted in characters, not bytes:
 

```
vim9-string-index
```
- Some possibly unexpected differences: [vim9-gotchas](#) .

### Comments starting with #

In legacy Vim script comments start with double quote. In Vim9 script comments start with #.

```
declarations
var count = 0 # number of occurrences
```

The reason is that a double quote can also be the start of a string. In many places, especially halfway through an expression with a line break, it's hard to tell what the meaning is, since both a string and a comment can be followed by arbitrary text. To avoid confusion only # comments are recognized. This is the same as in shell scripts and Python programs.

In Vi # is a command to list text with numbers. In Vim9 script you can use ``:number`` for that.

```
:101 number
```

To improve readability there must be a space between a command and the # that starts a comment:

```
var name = value # comment
var name = value# error!
```

**E1170**

Do not start a comment with #{, it looks like the legacy dictionary literal and produces an error where this might be confusing. #{} or #{{{ are OK, these can be used to start a fold.

When starting to read a script file Vim doesn't know it is `Vim9` script until the ``vim9script`` command is found. Until that point you would need to use legacy comments:

```
" legacy comment
vim9script
Vim9 comment
```

That looks ugly, better put ``vim9script`` in the very first line:

```
vim9script
Vim9 comment
```

In legacy Vim script # is also used for the alternate file name. In Vim9

script you need to use %% instead. Instead of ## use %%% (stands for all arguments).

## Vim9 functions

E1099

A function defined with `:def` is compiled. Execution is many times faster, often 10 to 100 times.

Many errors are already found when compiling, before the function is executed. The syntax is strict, to enforce code that is easy to read and understand.

Compilation is done when any of these is encountered:

- the first time the function is called
- when the `:defcompile` command is encountered in the script after the function was defined
- `:disassemble` is used for the function.
- a function that is compiled calls the function or uses it as a function reference (so that the argument and return types can be checked)

E1091 E1191

If compilation fails it is not tried again on the next call, instead this error is given: "E1091: Function is not compiled: {name}".

Compilation will fail when encountering a user command that has not been created yet. In this case you can call `execute()` to invoke it at runtime.

```
def MyFunc()
 execute('DefinedLater')
enddef
```

`:def` has no options like `:function` does: "range", "abort", "dict" or "closure". A `:def` function always aborts on an error (unless `:silent!` was used for the command or the error was caught a `:try` block), does not get a range passed, cannot be a "dict" function, and can always be a closure.

vim9-no-dict-function

You can use a Vim9 Class ( `Vim9-class` ) instead of a "dict function".

You can also pass the dictionary explicitly:

```
def DictFunc(self: dict<any>, arg: string)
 echo self[arg]
enddef
var ad = {item: 'value', func: DictFunc}
ad.func(ad, 'item')
```

You can call a legacy dict function though:

```
func Legacy() dict
 echo self.value
endfunc
def CallLegacy()
 var d = {func: Legacy, value: 'text'}
 d.func()
enddef
```

E1096 E1174 E1175

The argument types and return type need to be specified. The "any" type can be used, type checking will then be done at runtime, like with legacy functions.

E1106

Arguments are accessed by name, without "a:", just like any other language. There is no "a:" dictionary or "a:000" list.

vim9-variable-arguments E1055 E1160 E1180

Variable arguments are defined as the last argument, with a name and have a list type, similar to TypeScript. For example, a list of numbers:

```
def MyFunc(...itemlist: list<number>)
 for item in itemlist
 ...
```

When a function argument is optional (it has a default value) passing `v:none` as the argument results in using the default value. This is useful when you want to specify a value for an argument that comes after an argument that should use its default value. Example:

```
def MyFunc(one = 'one', last = 'last')
 ...
enddef
MyFunc(v:none, 'LAST') # first argument uses default value 'one'
```

vim9-ignored-argument E1181

The argument "\_" (an underscore) can be used to ignore the argument. This is most useful in callbacks where you don't need it, but do need to give an argument to match the call. E.g. when using map() two arguments are passed, the key and the value, to ignore the key:

```
map(numberList, (_, v) => v * 2)
```

There is no error for using the "\_" argument multiple times. No type needs to be given.

## Functions and variables are script-local by default

vim9-scopes

When using `:function` or `:def` to specify a new function at the script level in a Vim9 script, the function is local to the script. Like prefixing "s:" in legacy script. To define a global function or variable the "g:" prefix must be used. For functions in a script that is to be imported and in an autoload script "export" needs to be used for those to be used elsewhere.

```
def ThisFunction() # script-local
def g:ThatFunction() # global
export def Function() # for import and import autoload
 E1058 E1075
```

When using `:function` or `:def` to specify a nested function inside a `:def` function and no namespace was given, this nested function is local to the code block it is defined in. It cannot be used in `function()` with a string argument, pass the function reference itself:

```
def Outer()
 def Inner()
 echo 'inner'
 enddef
 var Fok = function(Inner) # OK
 var Fbad = function('Inner') # does not work
```

Detail: this is because "Inner" will actually become a function reference to a function with a generated name.

It is not possible to define a script-local function in a function. You can

define a local function and assign it to a script-local Funcref (it must have been declared at the script level). It is possible to define a global function by using the "g:" prefix.

When referring to a function and no "s:" or "g:" prefix is used, Vim will search for the function:

- in the function scope, in block scopes
- in the script scope

Imported functions are found with the prefix from the `:import` command.

Since a script-local function reference can be used without "s:" the name must start with an upper case letter even when using the "s:" prefix. In legacy script "s:funcref" could be used, because it could not be referred to with "funcref". In Vim9 script it can, therefore "s:Funcref" must be used to avoid that the name interferes with builtin functions.

[vim9-s-namespace](#) [E1268](#)

The use of the "s:" prefix is not supported at the Vim9 script level. All functions and variables without a prefix are script-local.

In `:def` functions the use of "s:" depends on the script: Script-local variables and functions in a legacy script do use "s:", while in a Vim9 script they do not use "s:". This matches what you see in the rest of the file.

In legacy functions the use of "s:" for script items is required, as before. No matter if the script is Vim9 or legacy.

In all cases the function must be defined before used. That is when it is called, when `:defcompile` causes it to be compiled, or when code that calls it is being compiled (to figure out the return type).

The result is that functions and variables without a namespace can usually be found in the script, either defined there or imported. Global functions and variables could be defined anywhere (good luck finding out where! You can often see where it was last set using `:verbose`).

[E1102](#)

Global functions can still be defined and deleted at nearly any time. In Vim9 script script-local functions are defined once when the script is sourced and cannot be deleted or replaced by itself (it can be by reloading the script).

When compiling a function and a function call is encountered for a function that is not (yet) defined, the `FuncUndefined` autocommand is not triggered. You can use an autoloading function if needed, or call a legacy function and have `FuncUndefined` triggered there.

### Reloading a Vim9 script clears functions and variables by default

[vim9-reload](#) [E1149](#) [E1150](#)

When loading a legacy Vim script a second time nothing is removed, the commands will replace existing variables and functions, create new ones, and leave removed things hanging around.

When loading a Vim9 script a second time all existing script-local functions

and variables are deleted, thus you start with a clean slate. This is useful if you are developing a plugin and want to try a new version. If you renamed something you don't have to worry about the old name still hanging around.

If you do want to keep items, use:

```
vim9script noclear
```

You want to use this in scripts that use a `finish` command to bail out at some point when loaded again. E.g. when a buffer local option is set to a function, the function does not need to be defined more than once:

```
vim9script noclear
setlocal completefunc=SomeFunc
if exists('*SomeFunc')
 finish
endif
def SomeFunc()
....
```

Variable declarations with `:var`, `:final` and `:const`

```
vim9-declaration :var E1079
E1017 E1020 E1054 E1087 E1124
```

Local variables need to be declared with `:var`. Local constants need to be declared with `:final` or `:const`. We refer to both as "variables" in this section.

Variables can be local to a script, function or code block:

```
vim9script
var script_var = 123
def SomeFunc()
 var func_var = script_var
 if cond
 var block_var = func_var
 ...
```

The variables are only visible in the block where they are defined and nested blocks. Once the block ends the variable is no longer accessible:

```
if cond
 var inner = 5
else
 var inner = 0
endif
echo inner # Error!
```

The declaration must be done earlier:

```
var inner: number
if cond
 inner = 5
else
 inner = 0
endif
echo inner
```

Although this is shorter and faster for simple values:

```

var inner = 0
if cond
 inner = 5
endif
echo inner

```

E1025 E1128

To intentionally hide a variable from code that follows, a block can be used:

```

{
 var temp = 'temp'
 ...
}
echo temp # Error!

```

This is especially useful in a user command:

```

command -range Rename {
 var save = @a
 @a = 'some expression'
 echo 'do something with ' .. @a
 @a = save
}

```

And with autocommands:

```

au BufWritePre *.go {
 var save = winsaveview()
 silent! exe ':%! some formatting command'
 winrestview(save)
}

```

Although using a `:def` function probably works better.

E1022 E1103 E1130 E1131 E1133  
E1134

Declaring a variable with a type but without an initializer will initialize to false (for bool), empty (for string, list, dict, etc.) or zero (for number, any, etc.). This matters especially when using the "any" type, the value will default to the number zero. For example, when declaring a list, items can be added:

```

var myList: list<number>
myList->add(7)

```

Initializing a variable to a null value, e.g. ``null_list``, differs from not initializing the variable. This throws an error:

```

var myList = null_list
myList->add(7) # E1130: Cannot add to null list

```

E1016 E1052 E1066

In Vim9 script ``:let`` cannot be used. An existing variable is assigned to without any command. The same for global, window, tab, buffer and Vim variables, because they are not really declared. Those can also be deleted with ``:unlet``.

E1065

You cannot use ``:va`` to declare a variable, it must be written with the full name ``:var``. Just to make sure it is easy to read.



E1178

`:lockvar` does not work on local variables. Use `:const` and `:final` instead.

The `exists()` and `exists\_compiled()` functions do not work on local variables or arguments.

E1006 E1041 E1167 E1168 E1213

Variables, functions and function arguments cannot shadow previously defined or imported variables and functions in the same script file. Variables may shadow Ex commands, rename the variable if needed.

Global variables must be prefixed with "g:", also at the script level.

```
vim9script
var script_local = 'text'
g:global = 'value'
var Funcref = g:ThatFunction
```

Global functions must be prefixed with "g:":

```
vim9script
def g:GlobalFunc(): string
 return 'text'
enddef
echo g:GlobalFunc()
```

The "g:" prefix is not needed for auto-load functions.

vim9-function-defined-later

Although global functions can be called without the "g:" prefix, they must exist when compiled. By adding the "g:" prefix the function can be defined later. Example:

```
def CallPluginFunc()
 if exists('g:loaded_plugin')
 g:PluginFunc()
 endif
enddef
```

If you do it like this, you get an error at compile time that "PluginFunc" does not exist, even when "g:loaded\_plugin" does not exist:

```
def CallPluginFunc()
 if exists('g:loaded_plugin')
 PluginFunc() # Error - function not found
 endif
enddef
```

You can use `exists_compiled()` to avoid the error, but then the function would not be called, even when "g:loaded\_plugin" is defined later:

```
def CallPluginFunc()
 if exists_compiled('g:loaded_plugin')
 PluginFunc() # Function may never be called
 endif
enddef
```

Since `&opt = value` is now assigning a value to option "opt", ":%" cannot be used to repeat a `:substitute` command.

vim9-unpack-ignore

For an unpack assignment the underscore can be used to ignore a list item, similar to how a function argument can be ignored:

```
[a, _, c] = theList
```

To ignore any remaining items:

```
[a, b; _] = longList
```

E1163 E1080

Declaring more than one variable at a time, using the unpack notation, is possible. Each variable can have a type or infer it from the value:

```
var [v1: number, v2] = GetValues()
```

Use this only when there is a list with values, declaring one variable per line is much easier to read and change later.

## Constants

vim9-const vim9-final

How constants work varies between languages. Some consider a variable that can't be assigned another value a constant. JavaScript is an example. Others also make the value immutable, thus when a constant uses a list, the list cannot be changed. In Vim9 we can use both.

E1021 E1307

``:const`` is used for making both the variable and the value a constant. Use this for composite structures that you want to make sure will not be modified. Example:

```
const myList = [1, 2]
myList = [3, 4] # Error!
myList[0] = 9 # Error!
myList->add(3) # Error!
```

:final E1125

``:final`` is used for making only the variable a constant, the value can be changed. This is well known from Java. Example:

```
final myList = [1, 2]
myList = [3, 4] # Error!
myList[0] = 9 # OK
myList->add(3) # OK
```

It is common to write constants as ALL\_CAPS, but you don't have to.

The constant only applies to the value itself, not what it refers to.

```
final females = ["Mary"]
const NAMES = ["John", "Peter"], females]
NAMES[0] = ["Jack"] # Error!
NAMES[0][0] = "Jack" # Error!
NAMES[1] = ["Emma"] # Error!
NAMES[1][0] = "Emma" # OK, now females[0] == "Emma"
```

## Omitting `:call` and `:eval`

E1190

Functions can be called without ``:call``:

```
writefile(lines, 'file')
```

Using ``:call`` is still possible, but this is discouraged.

A method call without ``eval`` is possible, so long as the start is an identifier or can't be an Ex command. For a function either "(" or "->" must

be following, without a line break. Examples:

```
myList->add(123)
g:myList->add(123)
[1, 2, 3]->Process()
{a: 1, b: 2}->Process()
"foobar"->Process()
("foobar")->Process()
'foobar'->Process()
('foobar')->Process()
```

In the rare case there is ambiguity between a function name and an Ex command, prepend ":" to make clear you want to use the Ex command. For example, there is both the `:substitute` command and the `substitute()` function. When the line starts with `substitute()` this will use the function. Prepend a colon to use the command instead:

```
:substitute(pattern replacement (
```

If the expression starts with "!" this is interpreted as a shell command, not negation of a condition. Thus this is a shell command:

```
!shellCommand->something
```

Put the expression in parentheses to use the "!" for negation:

```
(!expression)->Method()
```

**Note** that while variables need to be defined before they can be used, functions can be called before being defined. This is required to allow for cyclic dependencies between functions. It is slightly less efficient, since the function has to be looked up by name. And a typo in the function name will only be found when the function is called.

### Omitting function()

A user defined function can be used as a function reference in an expression without `function()`. The argument types and return type will then be checked. The function must already have been defined.

```
var Funcref = MyFunction
```

When using `function()` the resulting type is "func", a function with any number of arguments and any return type (including void). The function can be defined later if the argument is in quotes.

### Lambda using => instead of ->

[vim9-lambda](#)

In legacy script there can be confusion between using "->" for a method call and for a lambda. Also, when a "{" is found the parser needs to figure out if it is the start of a lambda or a dictionary, which is now more complicated because of the use of argument types.

To avoid these problems Vim9 script uses a different syntax for a lambda, which is similar to JavaScript:

```
var Lambda = (arg) => expression
var Lambda = (arg): type => expression
```

E1157

No line break is allowed in the arguments of a lambda up to and including the "=>" (so that Vim can tell the difference between an expression in parentheses and lambda arguments). This is OK:

```
filter(list, (k, v) =>
 v > 0)
```

This does not work:

```
filter(list, (k, v)
 => v > 0)
```

This also does not work:

```
filter(list, (k,
 v) => v > 0)
```

But you can use a backslash to concatenate the lines before parsing:

```
filter(list, (k,
 \ v)
 \ => v > 0)
```

vim9-lambda-arguments E1172

In legacy script a lambda could be called with any number of extra arguments, there was no way to warn for not using them. In Vim9 script the number of arguments must match. If you do want to accept any arguments, or any further arguments, use "...\_", which makes the function accept

vim9-variable-arguments . Example:

```
var Callback = (..._) => 'anything'
echo Callback(1, 2, 3) # displays "anything"
```

inline-function E1171

Additionally, a lambda can contain statements in {}:

```
var Lambda = (arg) => {
 g:was_called = 'yes'
 return expression
}
```

This can be useful for a timer, for example:

```
var count = 0
var timer = timer_start(500, (_) => {
 count += 1
 echom 'Handler called ' .. count
}, {repeat: 3})
```

The ending "}" must be at the start of a line. It can be followed by other characters, e.g.:

```
var d = mapnew(dict, (k, v): string => {
 return 'value'
})
```

No command can follow the "{", only a comment can be used there.

command-block E1026

The block can also be used for defining a user command. Inside the block Vim9 syntax will be used.

If the statements include a dictionary, its closing bracket must not be written at the start of a line. Otherwise, it would be parsed as the end of the block. This does not work:

```
command NewCommand {
 g:mydict = {
```

```

 'key': 'value',
 } # ERROR: will be recognized as the end of the block
}

```

Put the '}' after the last item to avoid this:

```

command NewCommand {
 g:mydict = {
 'key': 'value' }
}

```

Rationale: The "}" cannot be after a command because it would require parsing the commands to find it. For consistency with that no command can follow the "{". Unfortunately this means using "() => { command }" does not work, line breaks are always required.

#### vim9-curly

To avoid the "{" of a dictionary literal to be recognized as a statement block wrap it in parentheses:

```

var Lambda = (arg) => ({key: 42})

```

Also when confused with the start of a command block:

```

({
 key: value
})->method()

```

#### Automatic line continuation

#### vim9-line-continuation E1097

In many cases it is obvious that an expression continues on the next line. In those cases there is no need to prefix the line with a backslash (see [line-continuation](#)). For example, when a list spans multiple lines:

```

var mylist = [
 'one',
 'two',
]

```

And when a dict spans multiple lines:

```

var mydict = {
 one: 1,
 two: 2,
}

```

With a function call:

```

var result = Func(
 arg1,
 arg2
)

```

For binary operators in expressions not in [], {} or () a line break is possible just before or after the operator. For example:

```

var text = lead
 .. middle
 .. end
var total = start +
 end -
 correction
var result = positive

```

```
? PosFunc(arg)
: NegFunc(arg)
```

For a method call using "->" and a member using a dot, a line break is allowed before it:

```
var result = GetBuilder()
 ->BuilderSetWidth(333)
 ->BuilderSetHeight(777)
 ->BuilderBuild()
var result = MyDict
 .member
```

For commands that have an argument that is a list of commands, the | character at the start of the line indicates line continuation:

```
autocmd BufNewFile *.match if condition
 | echo 'match'
 | endif
```

Note that this means that in heredoc the first line cannot start with a bar:

```
var lines =<< trim END
 | this doesn't work
END
```

Either use an empty line at the start or do not use heredoc. Or temporarily add the "C" flag to 'coptions':

```
set cpo+=C
var lines =<< trim END
 | this works
END
set cpo-=C
```

If the heredoc is inside a function 'coptions' must be set before :def and restored after the :enddef.

In places where line continuation with a backslash is still needed, such as splitting up a long Ex command, comments can start with '#\ ':

```
syn region Text
 \ start='foo'
 #\ comment
 \ end='bar'
```

Like with legacy script '"\ ' is used. This is also needed when line continuation is used without a backslash and a line starts with a bar:

```
au CursorHold * echom 'BEFORE bar'
 #\ some comment
 | echom 'AFTER bar'
```

E1050

To make it possible for the operator at the start of the line to be recognized, it is required to put a colon before a range. This example will add "start" and "print":

```
var result = start
+ print
```

Like this:

```
var result = start + print
```

This will assign "start" and print a line:

```
var result = start
:+ print
```

After the range an Ex command must follow. Without the colon you can call a function without `:call`, but after a range you do need it:

```
MyFunc()
:% call MyFunc()
```

Note that the colon is not required for the `+cmd` argument:

```
edit +6 fname
```

It is also possible to split a function header over multiple lines, in between arguments:

```
def MyFunc(
 text: string,
 separator = '-'
): string
```

Since a continuation line cannot be easily recognized the parsing of commands has been made stricter. E.g., because of the error in the first line, the second line is seen as a separate command:

```
popup_create(some invalid expression, {
 exit_cb: Func})
```

Now "exit\_cb: Func})" is actually a valid command: save any changes to the file "\_cb: Func})" and exit. To avoid this kind of mistake in Vim9 script there must be white space between most command names and the argument.

E1144

However, the argument of a command that is a command won't be recognized. For example, after "windo echo expr" a line break inside "expr" will not be seen.

#### Notes:

- "endif" cannot be used at the start of a continuation line, it ends the current function.
- No line break is allowed in the LHS of an assignment. Specifically when unpacking a list `:let-unpack`. This is OK:

```
[var1, var2] =
 Func()
```

This does not work:

```
[var1,
 var2] =
 Func()
```

- No line break is allowed in between arguments of an `:echo`, `:execute` and similar commands. This is OK:

```
echo [1,
 2] [3,
 4]
```

This does not work:

```
echo [1, 2]
 [3, 4]
```

- In some cases it is difficult for Vim to parse a command, especially when commands are used as an argument to another command, such as `:windo`. In those cases the line continuation with a backslash has to be used.

## White space

Vim9 script enforces proper use of white space. This is no longer allowed:

```
E1004 E1068 E1069 E1074 E1127 E1202
var name=234 # Error!
var name= 234 # Error!
var name =234 # Error!
```

There must be white space before and after the "=":

```
var name = 234 # OK
```

White space must also be put before the # that starts a comment after a command:

```
var name = 234# Error!
var name = 234 # OK
```

White space is required around most operators.

White space is required in a sublist (list slice) around the ":", except at the start and end:

```
otherlist = mylist[v : count] # v:count has a different meaning
otherlist = mylist[:] # make a copy of the List
otherlist = mylist[v :]
otherlist = mylist[: v]
```

White space is not allowed:

- Between a function name and the "(":

```
Func (arg) # Error!
Func
 \ (arg) # Error!
Func
 (arg) # Error!
Func(arg) # OK
Func(
 arg) # OK
Func(
 arg # OK
)
```

E1205

White space is not allowed in a `:set` command between the option name and a following "&", "!", "<", "=", "+=", "-=" or "^=".

## No curly braces expansion

`curly-braces-names` cannot be used.

## Command modifiers are not ignored

Using a command modifier for a command that does not use it gives an error.

Also, using a command modifier without a following command is now an error.



## Dictionary literals

vim9-literal-dict E1014

Traditionally Vim has supported dictionary literals with a `{}` syntax:

```
let dict = {'key': value}
```

Later it became clear that using a simple text key is very common, thus literal dictionaries were introduced in a backwards compatible way:

```
let dict = #{key: value}
```

However, this `#{}` syntax is unlike any existing language. As it turns out that using a literal key is much more common than using an expression, and considering that JavaScript uses this syntax, using the `{}` form for dictionary literals is considered a much more useful syntax. In Vim9 script the `{}` form uses literal keys:

```
var dict = {key: value}
```

This works for alphanumeric characters, underscore and dash. If you want to use another character, use a single or double quoted string:

```
var dict = {'key with space': value}
var dict = {"key\twith\ttabs": value}
var dict = {'': value} # empty key
E1139
```

In case the key needs to be an expression, square brackets can be used, just like in JavaScript:

```
var dict = {"key" .. nr}: value}
```

The key type can be string, number, bool or float. Other types result in an error. Without using `[]` the value is used as a string, keeping leading zeros. An expression given with `[]` is evaluated and then converted to a string. Leading zeros will then be dropped:

```
var dict = {000123: 'without', [000456]: 'with'}
echo dict
{'456': 'with', '000123': 'without'}
```

A float only works inside `[]` because the dot is not accepted otherwise:

```
var dict = {[00.013]: 'float'}
echo dict
{'0.013': 'float'}
```

No `:xit`, `:t`, `:k`, `:append`, `:change` or `:insert`

E1100

These commands are too easily confused with local variable names.

Instead of ``:x`` or ``:xit`` you can use ``:exit``.

Instead of ``:t`` you can use ``:copy``.

Instead of ``:k`` you can use ``:mark``.

## Comparators

The `'ignorecase'` option is not used for comparators that use strings.

Thus `"=~"` works like `"=~#"`.

`"is"` and `"isnot"` (`expr-is` and `expr-isnot`) when used on strings now return false. In legacy script they just compare the strings, in Vim9 script they

check identity, and strings are copied when used, thus two strings are never the same (this might change someday if strings are not copied but reference counted).

### Abort after error

In legacy script, when an error is encountered, Vim continues to execute following lines. This can lead to a long sequence of errors and need to type **CTRL-C** to stop it. In Vim9 script execution of commands stops at the first error. Example:

```
vim9script
var x = does-not-exist
echo 'not executed'
```

### For loop

E1254

The loop variable must not be declared yet:

```
var i = 1
for i in [1, 2, 3] # Error!
```

It is possible to use a global variable though:

```
g:i = 1
for g:i in [1, 2, 3]
 echo g:i
endfor
```

Legacy Vim script has some tricks to make a for loop over a list handle deleting items at the current or previous item. In Vim9 script it just uses the index, if items are deleted then items in the list will be skipped.

Example legacy script:

```
let l = [1, 2, 3, 4]
for i in l
 echo i
 call remove(l, index(l, i))
endfor
```

Would echo:

```
1
2
3
4
```

In compiled Vim9 script you get:

```
1
3
```

Generally, you should not change the list that is iterated over. Make a copy first if needed.

When looping over a list of lists, the nested lists can be changed. The loop variable is "final", it cannot be changed but what its value can be changed.

E1306

The depth of loops, :for and :while loops added together, cannot exceed 10.

### Conditions and expressions

## vim9-boolean

Conditions and expressions are mostly working like they do in other languages. Some values are different from legacy Vim script:

| value  | legacy Vim script | Vim9 script |
|--------|-------------------|-------------|
| 0      | falsy             | falsy       |
| 1      | truthy            | truthy      |
| 99     | truthy            | Error!      |
| "0"    | falsy             | Error!      |
| "99"   | truthy            | Error!      |
| "text" | falsy             | Error!      |

For the "??" operator and when using "!" then there is no error, every value is either falsy or truthy. This is mostly like JavaScript, except that an empty list and dict is falsy:

| type       | truthy when                                      |
|------------|--------------------------------------------------|
| bool       | true, v:true or 1                                |
| number     | non-zero                                         |
| float      | non-zero                                         |
| string     | non-empty                                        |
| blob       | non-empty                                        |
| list       | non-empty (different from JavaScript)            |
| dictionary | non-empty (different from JavaScript)            |
| func       | when there is a function name                    |
| special    | true or v:true                                   |
| job        | when not NULL                                    |
| channel    | when not NULL                                    |
| class      | when not NULL                                    |
| object     | when not NULL (TODO: when isTrue() returns true) |

The boolean operators "||" and "&&" expect the values to be boolean, zero or one:

```
1 || false == true
0 || 1 == true
0 || false == false
1 && true == true
0 && 1 == false
8 || 0 Error!
'yes' && 0 Error!
[] || 99 Error!
```

When using "!" for inverting, there is no error for using any type and the result is a boolean. "!!" can be used to turn any value into boolean:

```
!'yes' == false
!![] == false
!![1, 2, 3] == true
```

When using "`.."` for string concatenation arguments of simple types are always converted to string:

```
'hello ' .. 123 == 'hello 123'
'hello ' .. v:true == 'hello true'
```

Simple types are Number, Float, Special and Bool. For other types `string()` should be used.

```
false true null null_blob null_channel
null_class null_dict null_function null_job
null_list null_object null_partial null_string
E1034
```

In Vim9 script one can use the following predefined values:

```
true
false
null
null_blob
null_channel
null_class
null_dict
null_function
null_job
null_list
null_object
null_partial
null_string
```

``true`` is the same as ``v:true``, ``false`` the same as ``v:false``, ``null`` the same as ``v:null``.

While ``null`` has the type "special", the other "null\_" values have the type indicated by their name. Quite often a null value is handled the same as an empty value, but not always. The values can be useful to clear a script-local variable, since they cannot be deleted with ``:unlet``. E.g.:

```
var theJob = job_start(...)
let the job do its work
theJob = null_job
```

The values can also be useful as the default value for an argument:

```
def MyFunc(b: blob = null_blob)
Note: compare against null, not null_blob,
to distinguish the default value from an empty blob.
if b == null
b argument was not given
```

See [null-compare](#) for more information about testing against null.

It is possible to compare ``null`` with any value, this will not give a type error. However, comparing ``null`` with a number, float or bool will always result in ``false``. This is different from legacy script, where comparing ``null`` with zero or ``false`` would return ``true``.

[vim9-false-true](#)

When converting a boolean to a string ``false`` and ``true`` are used, not ``v:false`` and ``v:true`` like in legacy script. ``v:none`` has no ``none`` replacement, it has no equivalent in other languages.

[vim9-string-index](#)

Indexing a string with `[idx]` or taking a slice with `[idx : idx]` uses character indexes instead of byte indexes. Composing characters are included.

Example:

```
echo 'bár'[1]
```

In legacy script this results in the character `0xc3` (an illegal byte), in Vim9 script this results in the string `'á'`.

A negative index is counting from the end, `"[-1]"` is the last character.

To exclude the last character use `slice()`.

To count composing characters separately use `strcharpart()` .  
If the index is out of range then an empty string results.

In legacy script `"++var"` and `"--var"` would be silently accepted and have no effect. This is an error in Vim9 script.

Numbers starting with zero are not considered to be octal, only numbers starting with `"0o"` are octal: `"0o744"`. [scriptversion-4](#)

## What to watch out for

### vim9-gotchas

Vim9 was designed to be closer to often used programming languages, but at the same time tries to support the legacy Vim commands. Some compromises had to be made. Here is a summary of what might be unexpected.

Ex command ranges need to be prefixed with a colon.

|                                                      |                                                   |
|------------------------------------------------------|---------------------------------------------------|
| <code>-&gt;</code>                                   | legacy Vim: shifts the previous line to the right |
| <code>-&gt;func()</code>                             | Vim9: method call in a continuation line          |
| <code>:-&gt;</code>                                  | Vim9: shifts the previous line to the right       |
| <code>%s/a/b</code>                                  | legacy Vim: substitute on all lines               |
| <code>x = alongname</code><br><code>% another</code> | Vim9: modulo operator in a continuation line      |
| <code>:%s/a/b</code>                                 | Vim9: substitute on all lines                     |
| <code>'t</code>                                      | legacy Vim: jump to mark t                        |
| <code>'text'-&gt;func()</code>                       | Vim9: method call                                 |
| <code>:'t</code>                                     | Vim9: jump to mark t                              |

Some Ex commands can be confused with assignments in Vim9 script:

|                             |                   |
|-----------------------------|-------------------|
| <code>g:name = value</code> | # assignment      |
| <code>:g:pattern:cmd</code> | # :global command |

To avoid confusion between a ``:global`` or ``:substitute`` command and an expression or assignment, a few separators cannot be used when these commands are abbreviated to a single character: `':'`, `'-'` and `'.'`.

|                             |                           |
|-----------------------------|---------------------------|
| <code>g:pattern:cmd</code>  | # invalid command - ERROR |
| <code>s:pattern:repl</code> | # invalid command - ERROR |
| <code>g-pattern-cmd</code>  | # invalid command - ERROR |
| <code>s-pattern-repl</code> | # invalid command - ERROR |
| <code>g.pattern.cmd</code>  | # invalid command - ERROR |
| <code>s.pattern.repl</code> | # invalid command - ERROR |

Also, there cannot be a space between the command and the separator:

|                              |                           |
|------------------------------|---------------------------|
| <code>g /pattern/cmd</code>  | # invalid command - ERROR |
| <code>s /pattern/repl</code> | # invalid command - ERROR |

Functions defined with ``:def`` compile the whole function. Legacy functions can bail out, and the following lines are not parsed:

```
func Maybe()
 if !has('feature')
 return
 endif
 use-feature
```

```
endfunc
```

Vim9 functions are compiled as a whole:

```
def Maybe()
 if !has('feature')
 return
 endif
 use-feature # May give a compilation error
enddef
```

For a workaround, split it in two functions:

```
func Maybe()
 if has('feature')
 call MaybeInner()
 endif
endfunc
if has('feature')
 def MaybeInner()
 use-feature
 enddef
endif
```

Or put the unsupported code inside an `if` with a constant expression that evaluates to false:

```
def Maybe()
 if has('feature')
 use-feature
 endif
enddef
```

The `exists\_compiled()` function can also be used for this.

[vim9-user-command](#)

Another side effect of compiling a function is that the presence of a user command is checked at compile time. If the user command is defined later an error will result. This works:

```
command -nargs=1 MyCommand echom <q-args>
def Works()
 MyCommand 123
enddef
```

This will give an error for "MyCommand" not being defined:

```
def Works()
 command -nargs=1 MyCommand echom <q-args>
 MyCommand 123
enddef
```

A workaround is to invoke the command indirectly with `:execute`:

```
def Works()
 command -nargs=1 MyCommand echom <q-args>
 execute 'MyCommand 123'
enddef
```

**Note** that for unrecognized commands there is no check for "|" and a following command. This will give an error for missing `endif`:

```
def Maybe()
 if has('feature') | use-feature | endif
enddef
```

Other differences

Patterns are used like `'magic'` is set, unless explicitly overruled.  
The `'edcompatible'` option value is not used.  
The `'gdefault'` option value is not used.

You may also find this wiki useful. It was written by an early adopter of Vim9 script: <https://github.com/lacygoill/wiki/blob/master/vim/vim9.md>

The `++` and `--` commands have been added. They are very similar to adding or subtracting one:

```
++var
var += 1
--var
var -= 1
```

Using `++var` or `--var` in an expression is not supported yet.

---

### 3. New style functions

`fast-functions`

```
:def[!] {name}([arguments])[: {return-type}]
```

Define a new function by the name `{name}`. The body of the function follows in the next lines, until the matching ``:enddef``. `E1073`

`E1011`  
The `{name}` must be less than 100 bytes long.  
`E1003 E1027 E1056 E1059`

The type of value used with ``:return`` must match `{return-type}`. When `{return-type}` is omitted or is "void" the function is not expected to return anything.

`E1077 E1123`  
`{arguments}` is a sequence of zero or more argument declarations. There are three forms:  
`{name}: {type}`  
`{name} = {value}`  
`{name}: {type} = {value}`

The first form is a mandatory argument, the caller must always provide them.  
The second and third form are optional arguments.  
When the caller omits an argument the `{value}` is used.

The function will be compiled into instructions when called, or when ``:disassemble`` or ``:defcompile`` is used. Syntax and type errors will be produced at that time.

It is possible to nest ``:def`` inside another ``:def`` or ``:function`` up to about 50 levels deep.  
`E1117`

`[!]` is used as with ``:function``. Note that script-local functions cannot be deleted or redefined

later in Vim9 script. They can only be removed by reloading the same script.

`:enddef` `:enddef` [E1057](#) [E1152](#) [E1173](#)  
End of a function defined with `:def``. It should be on a line by its own.

You may also find this wiki useful. It was written by an early adopter of Vim9 script: <https://github.com/lacygoill/wiki/blob/master/vim/vim9.md>

If the script the function is defined in is Vim9 script, then script-local variables can be accessed without the "s:" prefix. They must be defined before the function is compiled. If the script the function is defined in is legacy script, then script-local variables must be accessed with the "s:" prefix if they do not exist at the time of compiling.

[E1269](#)

Script-local variables in a [Vim9](#) script must be declared at the script level. They cannot be created in a function, also not in a legacy function.

`:defc[ompile]` `:defc` `:defcompile`  
Compile functions and classes ( `class-compile` ) defined in the current script that were not compiled yet. This will report any errors found during compilation.

`:defc[ompile] MyClass` Compile all methods in a class. `class-compile`

`:defc[ompile] {func}`

`:defc[ompile] debug {func}`

`:defc[ompile] profile {func}`

Compile function `{func}`, if needed. Use "debug" and "profile" to specify the compilation mode. This will report any errors found during compilation. `{func}` call also be "ClassName.functionName" to compile a function or method in a class. `{func}` call also be "ClassName" to compile all functions and methods in a class.

`:disa[ssemble] {func}` `:disa` `:disassemble`  
Show the instructions generated for `{func}`. This is for debugging and testing. [E1061](#)  
**Note** that for command line completion of `{func}` you can prepend "s:" to find script-local functions.

`:disa[ssemble] profile {func}`  
Like `:disassemble`` but with the instructions used for profiling.

`:disa[ssemble] debug {func}`  
Like `:disassemble`` but with the instructions used for debugging.

## Limitations



Local variables will not be visible to string evaluation. For example:

```
def MapList(): list<string>
 var list = ['aa', 'bb', 'cc', 'dd']
 return range(1, 2)->map('list[v:val]')
enddef
```

The map argument is a string expression, which is evaluated without the function scope. Instead, use a lambda:

```
def MapList(): list<string>
 var list = ['aa', 'bb', 'cc', 'dd']
 return range(1, 2)->map((_, v) => list[v])
enddef
```

For commands that are not compiled, such as `:edit`, backtick expansion can be used and it can use the local scope. Example:

```
def Replace()
 var fname = 'blah.txt'
 edit `=fname`
enddef
```

Closures defined in a loop will share the same context. For example:

```
var flist: list<func>
for i in range(5)
 var inloop = i
 flist[i] = () => inloop
endfor
echo range(5)->map((i, _) => flist[i]())
Result: [4, 4, 4, 4, 4]
```

E1271

A closure must be compiled in the context that it is defined in, so that variables in that context can be found. This mostly happens correctly, except when a function is marked for debugging with `:breakadd` after it was compiled. Make sure to define the breakpoint before compiling the outer function.

The "inloop" variable will exist only once, all closures put in the list refer to the same instance, which in the end will have the value 4. This is efficient, also when looping many times. If you do want a separate context for each closure, call a function to define it:

```
def GetClosure(i: number): func
 var infunc = i
 return () => infunc
enddef

var flist: list<func>
for i in range(5)
 flist[i] = GetClosure(i)
endfor
echo range(5)->map((i, _) => flist[i]())
Result: [0, 1, 2, 3, 4]
```

In some situations, especially when calling a Vim9 closure from legacy context, the evaluation will fail. E1248

**Note** that at the script level the loop variable will be invalid after the

loop, also when used in a closure that is called later, e.g. with a timer. This will generate error `E1302` :

```
for n in range(4)
 timer_start(500 * n, (_) => {
 echowin n
 })
endfor
```

You need to use a block and define a variable there, and use that one in the closure:

```
for n in range(4)
{
 var nr = n
 timer_start(500 * n, (_) => {
 echowin nr
 })
}
endfor
```

Using ``:echowindow`` is useful in a timer, the messages go into a popup and will not interfere with what the user is doing when it triggers.

### Converting a function from legacy to Vim9

#### `convert_legacy_function_to_vim9`

These are the most changes that need to be made to convert a legacy function to a Vim9 function:

- Change ``func`` or ``function`` to ``def``.
- Change ``endfunc`` or ``endfunction`` to ``enddef``.
- Add types to the function arguments.
- If the function returns something, add the return type.
- Change comments to start with `#` instead of `"`.

For example, a legacy function:

```
func MyFunc(text)
 " function body
endfunc
```

Becomes:

```
def MyFunc(text: string): number
 # function body
enddef
```

- Remove `"a:"` used for arguments. E.g.:

```
return len(a:text)
```

Becomes:

```
return len(text)
```

- Change ``let`` used to declare a variable to ``var``.
- Remove ``let`` used to assign a value to a variable. This is for local variables already declared and `b:` `w:` `g:` and `t:` variables.

For example, legacy function:

```
let lnum = 1
```

```
let lnum += 3
let b:result = 42
```

Becomes:

```
var lnum = 1
lnum += 3
b:result = 42
```

- Insert white space in expressions where needed.
- Change "." used for concatenation to "..".

For example, legacy function:

```
echo line(1).line(2)
```

Becomes:

```
echo line(1) .. line(2)
```

- line continuation does not always require a backslash:

```
echo ['one',
 \ 'two',
 \ 'three'
 \]
```

Becomes:

```
echo ['one',
 'two',
 'three'
]
```

#### Calling a function in an expr option

expr-option-function

The value of a few options, such as **'foldexpr'**, is an expression that is evaluated to get a value. The evaluation can have quite a bit of overhead. One way to minimize the overhead, and also to keep the option value very simple, is to define a compiled function and set the option to call it without arguments. Example:

```
vim9script
def MyFoldFunc(): any
 ... compute fold level for line v:lnum
 return level
enddef
set foldexpr=s:MyFoldFunc()
```

---

#### 4. Types

vim9-types

|       |       |       |       |
|-------|-------|-------|-------|
| E1008 | E1009 | E1010 | E1012 |
| E1013 | E1029 | E1030 |       |

The following builtin types are supported:

```
bool
number
float
string
blob
list<{type}>
dict<{type}>
```

```

job
channel
func
func: {type}
func({type}, ...)
func({type}, ...): {type}
void

```

Not supported yet:

```
tuple<a: {type}, b: {type}, ...>
```

These types can be used in declarations, but no simple value will actually have the "void" type. Trying to use a void (e.g. a function without a return value) results in error `E1031` `E1186` .

There is no array type, use `list<{type}>` instead. For a list constant an efficient implementation is used that avoids allocating a lot of small pieces of memory.

`vim9-func-declaration` `E1005` `E1007`

A partial and function can be declared in more or less specific ways:

|                                                                  |                                                                                                                                                                                                                    |
|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>func</code>                                                | any kind of function reference, no type checking for arguments or return value                                                                                                                                     |
| <code>func: void</code>                                          | any number and type of arguments, no return value                                                                                                                                                                  |
| <code>func: {type}</code>                                        | any number and type of arguments with specific return type                                                                                                                                                         |
| <code>func()</code>                                              | function with no argument, does not return a value                                                                                                                                                                 |
| <code>func(): void</code>                                        | same                                                                                                                                                                                                               |
| <code>func(): {type}</code>                                      | function with no argument and return type                                                                                                                                                                          |
| <code>func({type})</code>                                        | function with argument type, does not return a value                                                                                                                                                               |
| <code>func({type}): {type}</code>                                | function with argument type and return type                                                                                                                                                                        |
| <code>func({type})</code>                                        | function with type of optional argument, does not return a value                                                                                                                                                   |
| <code>func(...list&lt;{type}&gt;)</code>                         | function with type of list for variable number of arguments, does not return a value                                                                                                                               |
| <code>func({type}, {type}, ...list&lt;{type}&gt;): {type}</code> | function with: <ul style="list-style-type: none"> <li>- type of mandatory argument</li> <li>- type of optional argument</li> <li>- type of list for variable number of arguments</li> <li>- return type</li> </ul> |

If the return type is "void" the function does not return a value.

The reference can also be a `Partial` , in which case it stores extra arguments and/or a dictionary, which are not visible to the caller. Since they are called in the same way the declaration is the same.

Custom types can be defined with ``:type``:

```
:type MyList list<string>
```

Custom types must start with a capital letter, to avoid name clashes with builtin types added later, similarly to user functions.

```
{not implemented yet}
```

And classes and interfaces can be used as types:

```
:class MyClass
:var mine: MyClass
```

```
:interface MyInterface
:var mine: MyInterface
```

```
:class MyTemplate<Targ>
:var mine: MyTemplate<number>
:var mine: MyTemplate<string>
```

```
:class MyInterface<Targ>
:var mine: MyInterface<number>
:var mine: MyInterface<string>
```

```
{not implemented yet}
```

## Variable types and type casting

### variable-types

Variables declared in Vim9 script or in a ``:def`` function have a type, either specified explicitly or inferred from the initialization.

Global, buffer, window and tab page variables do not have a specific type, the value can be changed at any time, possibly changing the type. Therefore, in compiled code the "any" type is assumed.

This can be a problem when the "any" type is undesired and the actual type is expected to always be the same. For example, when declaring a list:

```
var l: list<number> = [1, g:two]
```

At compile time Vim doesn't know the type of "g:two" and the expression type becomes list<any>. An instruction is generated to check the list type before doing the assignment, which is a bit inefficient.

### type-casting E1104

To avoid this, use a type cast:

```
var l: list<number> = [1, <number>g:two]
```

The compiled code will then only check that "g:two" is a number and give an error if it isn't. This is called type casting.

The syntax of a type cast is: `"<" {type} ">"`. There cannot be white space after the "<" or before the ">" (to avoid them being confused with smaller-than and bigger-than operators).

The semantics is that, if needed, a runtime type check is performed. The value is not actually changed. If you need to change the type, e.g. to change it to a string, use the `string()` function. Or use `str2nr()` to convert a string to a number.

If a type is given where it is not expected you can get `E1272` .

If a type is incomplete you get **E1363** , e.g. when you have an object for which the class is not known (usually that is a null object).

## Type inference

### type-inference

In general: Whenever the type is clear it can be omitted. For example, when declaring a variable and giving it a value:

```
var name = 0 # infers number type
var name = 'hello' # infers string type
```

The type of a list and dictionary comes from the common type of the values. If the values all have the same type, that type is used for the list or dictionary. If there is a mix of types, the "any" type is used.

```
[1, 2, 3] list<number>
['a', 'b', 'c'] list<string>
[1, 'x', 3] list<any>
```

The common type of function references, if they do not all have the same number of arguments, uses "(...)" to indicate the number of arguments is not specified. For example:

```
def Foo(x: bool)
enddef
def Bar(x: bool, y: bool)
enddef
var funclist = [Foo, Bar]
echo funclist->typename()
```

Results in:

```
list<func(...)>
```

For script-local variables in Vim9 script the type is checked, also when the variable was declared in a legacy function.

When a type has been declared this is attached to a List or Dictionary. When later some expression attempts to change the type an error will be given:

```
var ll: list<number> = [1, 2, 3]
ll->extend(['x']) # Error, 'x' is not a number
```

If the type is not declared then it is allowed to change:

```
[1, 2, 3]->extend(['x']) # result: [1, 2, 3, 'x']
```

For a variable declaration an inferred type matters:

```
var ll = [1, 2, 3]
ll->extend(['x']) # Error, 'x' is not a number
```

That is because the declaration looks like a list of numbers, thus is equivalent to:

```
var ll: list<number> = [1, 2, 3]
```

If you do want a more permissive list you need to declare the type:

```
var ll: list<any> = [1, 2, 3]
ll->extend(['x']) # OK
```

## Stricter type checking

### type-checking

In legacy Vim script, where a number was expected, a string would be

automatically converted to a number. This was convenient for an actual number such as "123", but leads to unexpected problems (and no error message) if the string doesn't start with a number. Quite often this leads to hard-to-find bugs. e.g.:

```
echo 123 == '123'
```

```
1
```

With an accidental space:

```
echo 123 == ' 123'
```

```
0
```

E1206 E1210 E1212

In Vim9 script this has been made stricter. In most places it works just as before if the value used matches the expected type. There will sometimes be an error, thus breaking backwards compatibility. For example:

- Using a number other than 0 or 1 where a boolean is expected. E1023
- Using a string value when setting a number option.
- Using a number where a string is expected. E1024 E1105

One consequence is that the item type of a list or dict given to `map()` must not change, if the type was declared. This will give an error in Vim9 script:

```
var mylist: list<number> = [1, 2, 3]
echo map(mylist, (i, v) => 'item ' .. i)
E1012: Type mismatch; expected number but got string in map()
```

Instead use `mapnew()`, it creates a new list:

```
var mylist: list<number> = [1, 2, 3]
echo mapnew(mylist, (i, v) => 'item ' .. i)
['item 0', 'item 1', 'item 2']
```

If the item type was not declared or determined to be "any" it can change to a more specific type. E.g. when a list of mixed types gets changed to a list of strings:

```
var mylist = [1, 2.0, '3']
typename(mylist) == "list<any>"
map(mylist, (i, v) => 'item ' .. i)
typename(mylist) == "list<string>", no error
```

There is a subtle difference between using a list constant directly and through a variable declaration. Because of type inference, when using a list constant to initialize a variable, this also sets the declared type:

```
var mylist = [1, 2, 3]
typename(mylist) == "list<number>"
echo map(mylist, (i, v) => 'item ' .. i) # Error!
```

When using the list constant directly, the type is not declared and is allowed to change:

```
echo map([1, 2, 3], (i, v) => 'item ' .. i) # OK
```

The reasoning behind this is that when a type is declared and the list is passed around and changed, the declaration must always hold. So that you can rely on the type to match the declared type. For a constant this is not needed.

E1158

Same for `extend()` , use `extendnew()` instead, and for `flatten()` , use `flattennew()` instead. Since `flatten()` is intended to always change the type, it can not be used in Vim9 script.

Assigning to a funcref with specified arguments (see [vim9-func-declaration](#) ) does strict type checking of the arguments. For variable number of arguments the type must match:

```
var FuncRef: func(string, number, bool): number
FuncRef = (v1: string, v2: number, v3: bool) => 777 # OK
FuncRef = (v1: string, v2: number, v3: number) => 777 # Error!
variable number of arguments must have same type
var FuncVA: func(...list<string>): number
FuncVA = (...v: list<number>): number => v # Error!
FuncVA = (...v: list<any>): number => v # OK, `any` runtime check
FuncVA = (v1: string, v: string2): number => 333 # Error!
FuncVA = (v: list<string>): number => 3 # Error!
```

If the destination funcref has no specified arguments, then there is no argument type checking:

```
var FuncUnknownArgs: func: number
FuncUnknownArgs = (v): number => v # OK
FuncUnknownArgs = (v1: string, v2: string): number => 3 # OK
FuncUnknownArgs = (...v1: list<string>): number => 333 # OK
```

```
E1211 E1217 E1218 E1219 E1220 E1221
E1222 E1223 E1224 E1225 E1226 E1227
E1228 E1238 E1250 E1251 E1252 E1256
E1297 E1298 E1301
```

Types are checked for most builtin functions to make it easier to spot mistakes.

### Categories of variables, defaults and null handling

[variable-categories](#)    [null-variables](#)

There are categories of variables:

```
primitive number, float, boolean
container string, blob, list, dict
specialized function, job, channel, user-defined-object
```

When declaring a variable without an initializer, an explicit type must be provided. Each category has different default initialization semantics. Here's an example for each category:

```
var num: number # primitives default to a 0 equivalent
var cont: list<string> # containers default to an empty container
var spec: job # specialized variables default to null
```

Vim does not have a familiar null value; it has various `null_<type>` predefined values, for example `null_string` , `null_list` , `null_job` . Primitives do not have a `null_<type>`. The typical use cases for `null_<type>` are:

- to clear a variable and release its resources;
- as a default for a parameter in a function definition, see [null-compare](#) .

For a specialized variable, like ``job``, `null_<type>` is used to clear the resources. For a container variable, resources can also be cleared by assigning an empty container to the variable. For example:



```

var j: job = job_start(...)
... job does its work
j = null_job # clear the variable and release the job's resources

var l: list<any>
... add lots of stuff to list
l = [] # clear the variable and release container resources

```

Using the empty container, rather than `null_<type>`, to clear a container variable may avoid null complications as described in [null-anomalies](#) .

The initialization semantics of container variables and specialized variables differ. An uninitialized container defaults to an empty container:

```

var l1: list<string> # empty container
var l2: list<string> = [] # empty container
var l3: list<string> = null_list # null container

```

"l1" and "l2" are equivalent and indistinguishable initializations; but "l3" is a null container. A null container is similar to, but different from, an empty container, see [null-anomalies](#) .

Specialized variables default to null. These job initializations are equivalent and indistinguishable:

```

var j1: job
var j2: job = null_job
var j3 = null_job

```

When a list or dict is declared, if the item type is not specified and can not be inferred, then the type is "any":

```

var d1 = {} # type is "dict<any>"
var d2 = null_dict # type is "dict<any>"

```

Declaring a function, see [vim9-func-declaration](#) , is particularly unique.

### null-compare

For familiar null compare semantics, where a null container is not equal to an empty container, do not use `null_<type>` in a comparison:

```

vim9script
def F(arg: list<string> = null_list)
 if arg == null
 echo "null"
 else
 echo printf("not null, %empty", empty(arg) ? '' : 'not ')
 endif
enddef
F() # output: "null"
F(null_list) # output: "null"
F([]) # output: "not null, empty"
F(['']) # output: "not null, not empty"

```

The above function takes a list of strings and reports on it.

Change the above function signature to accept different types of arguments:

```

def F(arg: list<any> = null_list) # any type of list
def F(arg: any = null) # any type

```

In the above example, where the goal is to distinguish a null list from an empty list, comparing against ``null`` instead of ``null_list`` is the correct

choice. The basic reason is because "null\_list == null" and "[]" != null". Comparing to `null\_list` fails since "[]" == null\_list". In the following section there are details about comparison results.

### null-details null-anomalies

This section describes issues about using null and null\_<type>; included below are the enumerated results of null comparisons. In some cases, if familiar with vim9 null semantics, the programmer may chose to use null\_<type> in comparisons and/or other situations.

Elsewhere in the documentation it says:

Quite often a null value is handled the same as an empty value, but not always

Here's an example:

```
vim9script
var s1: list<string>
var s2: list<string> = null_list
echo s1 # output: "[]"
echo s2 # output: "[]"

echo s1 + ['a'] # output: "['a']"
echo s2 + ['a'] # output: "['a']"

echo s1->add('a') # output: "['a']"
echo s2->add('a') # E1130: Can not add to null list
```

Two values equal to a null\_<type> are not necessarily equal to each other:

```
vim9script
echo {} == null_dict # true
echo null_dict == null # true
echo {} == null # false
```

Unlike the other containers, an uninitialized string is equal to null. The **'is'** operator can be used to determine if it is a null\_string:

```
vim9script
var s1: string
var s2 = null_string
echo s1 == null # true - this is unexpected
echo s2 == null # true
echo s2 is null_string # true

var b1: blob
var b2 = null_blob
echo b1 == null # false
echo b2 == null # true
```

Any variable initialized to the null\_<type> is equal to the null\_<type> and is also equal to null. For example:

```
vim9script
var x = null_blob
echo x == null_blob # true
echo x == null # true
```

An uninitialized variable is usually equal to null; it depends on its type:

```

var s: string s == null
var b: blob b != null ***
var l: list<any> l != null ***
var d: dict<any> d != null ***
var f: func f == null
var j: job j == null
var c: channel c == null
var o: Class o == null

```

A variable initialized to empty equals `null_<type>`; but not `null`:

```

var s2: string = "" == null_string != null
var b2: blob = 0z == null_blob != null
var l2: list<any> = [] == null_list != null
var d2: dict<any> = {} == null_dict != null

```

**NOTE:** the specialized variables, like `job`, default to null value and have no corresponding empty value.

=====

## 5. Namespace, Import and Export

`vim9script`   `vim9-export`   `vim9-import`

A Vim9 script can be written to be imported. This means that some items are intentionally exported, made available to other scripts. When the exporting script is imported in another script, these exported items can then be used in that script. All the other items remain script-local in the exporting script and cannot be accessed by the importing script.

This mechanism exists for writing a script that can be sourced (imported) by other scripts, while making sure these other scripts only have access to what you want them to. This also avoids using the global namespace, which has a risk of name collisions. For example when you have two plugins with similar functionality.

You can cheat by using the global namespace explicitly. That should be done only for things that really are global.

### Namespace

`vim9-namespace`

To recognize a file that can be imported the ``vim9script`` statement must appear as the first statement in the file (see `vim9-mix` for an exception). It tells Vim to interpret the script in its own namespace, instead of the global namespace. If a file starts with:

```

vim9script
var myvar = 'yes'

```

Then "myvar" will only exist in this file. While without ``vim9script`` it would be available as ``g:myvar`` from any other script and function.

**E1101**

The variables at the file level are very much like the script-local "s:" variables in legacy Vim script, but the "s:" is omitted. And they cannot be deleted.

In Vim9 script the global "g:" namespace can still be used as before. And the "w:", "b:" and "t:" namespaces. These have in common that variables are not declared, have no specific type and they can be deleted. [E1304](#)

A side effect of `:vim9script` is that the `'coptions'` option is set to the Vim default value, like with:

```
:set cpo&vim
```

One of the effects is that `line-continuation` is always enabled. The original value of `'coptions'` is restored at the end of the script, while flags added or removed in the script are also added to or removed from the original value to get the same effect. The order of flags may change. In the `vimrc` file sourced on startup this does not happen.

#### [vim9-mix](#)

There is one way to use both legacy and Vim9 syntax in one script file:

```
" comments may go here
if !has('vim9script')
 " legacy script commands go here
 finish
endif
vim9script
Vim9 script commands go here
```

This allows for writing a script that takes advantage of the Vim9 script syntax if possible, but will also work on a `Vim version without` it.

This can only work in two ways:

1. The "if" statement evaluates to false, the commands up to ``endif`` are skipped and ``vim9script`` is then the first command actually executed.
2. The "if" statement evaluates to true, the commands up to ``endif`` are executed and ``finish`` bails out before reaching ``vim9script``.

## Export

[:export](#) [:exp](#)

Exporting an item can be written as:

```
export const EXPORTED_CONST = 1234
export var someValue = ...
export final someValue = ...
export const someValue = ...
export def MyFunc() ...
export class MyClass ...
export interface MyClass ...
```

[E1043](#) [E1044](#)

As this suggests, only constants, variables, `:def`` functions and classes can be exported.

[E1042](#)

`:export`` can only be used in Vim9 script, at the script level.

## Import

[:import](#) [:imp](#) [E1094](#) [E1047](#) [E1262](#)  
[E1048](#) [E1049](#) [E1053](#) [E1071](#) [E1088](#) [E1236](#)

The exported items can be imported in another script. The import syntax has

two forms. The simple form:

```
import {filename}
```

Where `{filename}` is an expression that must evaluate to a string. In this form the filename should end in ".vim" and the portion before ".vim" will become the script local name of the namespace. For example:

```
import "myscript.vim"
```

This makes each exported item in "myscript.vim" available as "myscript.item".

```
:import-as E1257 E1261
```

In case the name is long or ambiguous, this form can be used to specify another name:

```
import {longfilename} as {name}
```

In this form `{name}` becomes a specific script local name for the imported namespace. Therefore `{name}` must consist of letters, digits and '\_', like `internal-variables`. The `{longfilename}` expression must evaluate to any filename. For example:

```
import "thatscript.vim.v2" as that
```

```
E1060 E1258 E1259 E1260
```

Then you can use "that.item", etc. You are free to choose the name "that". Use something that will be recognized as referring to the imported script. Avoid command names, command modifiers and builtin function names, because the name will shadow them. It's better not to start the name with a capital letter, since it can then also shadow global user commands and functions. Also, you cannot use the name for something else in the script, such as a function or variable name.

In case the dot in the name is undesired, a local reference can be made for a function:

```
var LongFunc = that.LongFuncName
```

This also works for constants:

```
const MAXLEN = that.MAX_LEN_OF_NAME
```

This does not work for variables, since the value would be copied once and when changing the variable the copy will change, not the original variable. You will need to use the full name, with the dot.

``:import`` can not be used in a function. Imported items are intended to exist at the script level and only imported once.

The script name after ``import`` can be:

- A relative path, starting "." or "..". This finds a file relative to the location of the script file itself. This is useful to split up a large plugin into several files.
- An absolute path, starting with "/" on Unix or "D:/" on MS-Windows. This will rarely be used.
- A path not being relative or absolute. This will be found in the "import" subdirectories of `'runtimepath'` entries. The name will usually be longer and unique, to avoid loading the wrong file.  
**Note** that "after/import" is not used.

If the name does not end in ".vim" then the use of "as name" is required.

Once a vim9 script file has been imported, the result is cached and used the next time the same script is imported. It will not be read again.

It is not allowed to import the same script twice, also when using two different "as" names.

When using the imported name the dot and the item name must be in the same line, there can be no line break:

```
echo that.
 name # Error!
echo that
 .name # Error!
```

`import-map`

When you've imported a function from one script into a vim9 script you can refer to the imported function in a mapping by prefixing it with `<SID>` :

```
noremap <silent> ,a :call <SID>name.Function()<CR>
```

When the mapping is defined "`<SID>name.`" will be replaced with `<SNR>` and the script ID of the imported script.

An even simpler solution is using `<ScriptCmd>` :

```
noremap ,a <ScriptCmd>name.Function()<CR>
```

**Note** that this does not work for variables, only for functions.

`import-legacy` `legacy-import`

``:import`` can also be used in legacy Vim script. The imported namespace still becomes script-local, even when the "s:" prefix is not given. For example:

```
import "myfile.vim"
call s:myfile.MyFunc()
```

And using the "as name" form:

```
import "otherfile.vim9script" as that
call s:that.OtherFunc()
```

However, the namespace cannot be resolved on its own:

```
import "that.vim"
echo s:that
" ERROR: E1060: Expected dot after name: s:that
```

This also affects the use of `<SID>` in the legacy mapping context. Since `<SID>` is only a valid prefix for a function and NOT for a namespace, you cannot use it to scope a function in a script local namespace. Instead of prefixing the function with `<SID>` you should use `<ScriptCmd>` . For example:

```
noremap ,a <ScriptCmd>:call s:that.OtherFunc()<CR>
```

`:import-cycle`

The ``import`` commands are executed when encountered. If script A imports script B, and B (directly or indirectly) imports A, this will be skipped over. At this point items in A after "import B" will not have been processed and defined yet. Therefore cyclic imports can exist and not result in an error directly, but may result in an error for items in A after "import B" not being defined. This does not apply to autoload imports, see the next section.

## Importing an autoload script

`vim9-autoload` `import-autoload`

For optimal startup speed, loading scripts should be postponed until they are actually needed. Using the autoload mechanism is recommended:

E1264

1. In the plugin define user commands, functions and/or mappings that refer to items imported from an autoload script.

```
import autoload 'for/search.vim'
command -nargs=1 SearchForStuff search.Stuff(<f-args>)
```

This goes in `.../plugin/anyname.vim`. `"anyname.vim"` can be freely chosen. The `"SearchForStuff"` command is now available to the user.

The `"autoload"` argument to ``:import`` means that the script is not loaded until one of the items is actually used. The script will be found under the `"autoload"` directory in `'runtimepath'` instead of the `"import"` directory. Alternatively a relative or absolute name can be used, see below.

2. In the autoload script put the bulk of the code.

```
vim9script
export def Stuff(arg: string)
...
...
```

This goes in `.../autoload/for/search.vim`.

Putting the `"search.vim"` script under the `"/autoload/for/"` directory has the effect that `"for#search#"` will be prefixed to every exported item. The prefix is obtained from the file name, as you would to manually in a legacy autoload script. Thus the exported function can be found with `"for#search#Stuff"`, but you would normally use ``import autoload`` and not use the prefix (which has the side effect of loading the autoload script when compiling a function that encounters this name).

You can split up the functionality and import other scripts from the autoload script as you like. This way you can share code between plugins.

Searching for the autoload script in all entries in `'runtimepath'` can be a bit slow. If the plugin knows where the script is located, quite often a relative path can be used. This avoids the search and should be quite a bit faster. Another advantage is that the script name does not need to be unique. An absolute path is also possible. Examples:

```
import autoload './lib/implement.vim'
import autoload MyScriptsDir .. '/lib/implement.vim'
```

For defining a mapping that uses the imported autoload script the special key `<ScriptCmd>` is useful. It allows for a command in a mapping to use the script context of where the mapping was defined.

When compiling a ``:def`` function and a function in an autoload script is encountered, the script is not loaded until the ``:def`` function is called. This also means you get any errors only at runtime, since the argument and

return types are not known yet. If you would use the name with '#' characters then the autoload script IS loaded.

Be careful to not refer to an item in an autoload script that does trigger loading it unintentionally. For example, when setting an option that takes a function name, make sure to use a string, not a function reference:

```
import autoload 'qftf.vim'
&quickfixtextfunc = 'qftf.Func' # autoload script NOT loaded
&quickfixtextfunc = qftf.Func # autoload script IS loaded
```

On the other hand, it can be useful to load the script early, at a time when any errors should be given.

For testing the `test_override()` function can be used to have the ``import autoload`` load the script right away, so that the items and types can be checked without waiting for them to be actually used:

```
test_override('autoload', 1)
```

Reset it later with:

```
test_override('autoload', 0)
```

Or:

```
test_override('ALL', 0)
```

---

## 6. Classes and interfaces

[vim9-classes](#)

In legacy script a Dictionary could be used as a kind-of object, by adding members that are functions. However, this is quite inefficient and requires the writer to do the work of making sure all the objects have the right members. See [Dictionary-function](#) .

In [Vim9](#) script you can have classes, objects and interfaces like in most popular object-oriented programming languages. Since this is a lot of functionality it is located in a separate help file: [vim9class.txt](#) .

---

## 9. Rationale

[vim9-rationale](#)

### The `:def` command

Plugin writers have asked for much faster Vim script. Investigations have shown that keeping the existing semantics of function calls make this close to impossible, because of the overhead involved with calling a function, setting up the local function scope and executing lines. There are many details that need to be handled, such as error messages and exceptions. The need to create a dictionary for a: and l: scopes, the a:000 list and several others add too much overhead that cannot be avoided.

Therefore the ``:def`` method to define a new-style function had to be added, which allows for a function with different semantics. Most things still work as before, but some parts do not. A new way to define a function was considered the best way to separate the legacy style code from Vim9 style code.



Using "def" to define a function comes from Python. Other languages use "function" which clashes with legacy Vim script.

## Type checking

When compiling lines of Vim commands into instructions as much as possible should be done at compile time. Postponing it to runtime makes the execution slower and means mistakes are found only later. For example, when encountering the "+" character and compiling this into a generic add instruction, at runtime the instruction would have to inspect the type of the arguments and decide what kind of addition to do. And when the type is dictionary throw an error. If the types are known to be numbers then an "add number" instruction can be used, which is faster. The error can be given at compile time, no error handling is needed at runtime, since adding two numbers cannot fail.

The syntax for types, using `<type>` for compound types, is similar to Java. It is easy to understand and widely used. The type names are what were used in Vim before, with some additions such as "void" and "bool".

## Removing clutter and weirdness

Once decided that `:def`` functions have different syntax than legacy functions, we are free to add improvements to make the code more familiar for users who know popular programming languages. In other words: remove weird things that only Vim does.

We can also remove clutter, mainly things that were done to make Vim script backwards compatible with the good old Vi commands.

Examples:

- Drop `:call`` for calling a function and `:eval`` for evaluating an expression.
- Drop using a leading backslash for line continuation, automatically figure out where an expression ends.

However, this does require that some things need to change:

- Comments start with # instead of ", to avoid confusing them with strings. This is good anyway, it is also used by several popular languages.
- Ex command ranges need to be prefixed with a colon, to avoid confusion with expressions (single quote can be a string or a mark, "/" can be divide or a search command, etc.).

Goal is to limit the differences. A good criteria is that when the old syntax is accidentally used you are very likely to get an error message.

## Syntax and semantics from popular languages

Script writers have complained that the Vim script syntax is unexpectedly different from what they are used to. To reduce this complaint popular

languages are used as an example. At the same time, we do not want to abandon the well-known parts of legacy Vim script.

For many things TypeScript is followed. It's a recent language that is gaining popularity and has similarities with Vim script. It also has a mix of static typing (a variable always has a known value type) and dynamic typing (a variable can have different types, this changes at runtime). Since legacy Vim script is dynamically typed and a lot of existing functionality (esp. builtin functions) depends on that, while static typing allows for much faster execution, we need to have this mix in Vim9 script.

There is no intention to completely match TypeScript syntax and semantics. We just want to take those parts that we can use for Vim and we expect Vim users will be happy with. TypeScript is a complex language with its own history, advantages and disadvantages. To get an idea of the disadvantages read the book: "JavaScript: The Good Parts". Or find the article "TypeScript: the good parts" and read the "Things to avoid" section.

People familiar with other languages (Java, Python, etc.) will also find things in TypeScript that they do not like or do not understand. We'll try to avoid those things.

Specific items from TypeScript we avoid:

- Overloading "+", using it both for addition and string concatenation. This goes against legacy Vim script and often leads to mistakes. For that reason we will keep using ".." for string concatenation. Lua also uses ".." this way. And it allows for conversion to string for more values.
- TypeScript can use an expression like "99 || 'yes'" in a condition, but cannot assign the value to a boolean. That is inconsistent and can be annoying. Vim recognizes an expression with && or || and allows using the result as a bool. The `falsy-operator` was added for the mechanism to use a default value.
- TypeScript considers an empty string as Falsy, but an empty list or dict as Truthy. That is inconsistent. In Vim an empty list and dict are also Falsy.
- TypeScript has various "Readonly" types, which have limited usefulness, since a type cast can remove the immutable nature. Vim locks the value, which is more flexible, but is only checked at runtime.
- TypeScript has a complicated "import" statement that does not match how the Vim import mechanism works. A much simpler mechanism is used instead, which matches that the imported script is only sourced once.

## Declarations

Legacy Vim script uses `:let` for every assignment, while in Vim9 declarations are used. That is different, thus it's good to use a different command: `:var`. This is used in many languages. The semantics might be slightly different, but it's easily recognized as a declaration.

Using `:const` for constants is common, but the semantics varies. Some languages only make the variable immutable, others also make the value immutable. Since "final" is well known from Java for only making the variable immutable we decided to use that. And then `:const` can be used for making

both immutable. This was also used in legacy Vim script and the meaning is almost the same.

What we end up with is very similar to Dart:

```
:var name # mutable variable and value
:final name # immutable variable, mutable value
:const name # immutable variable and value
```

Since legacy and Vim9 script will be mixed and global variables will be shared, optional type checking is desirable. Also, type inference will avoid the need for specifying the type in many cases. The TypeScript syntax fits best for adding types to declarations:

```
var name: string # string type is specified
...
name = 'John'
const greeting = 'hello' # string type is inferred
```

This is how we put types in a declaration:

```
var mylist: list<string>
final mylist: list<string> = ['foo']
def Func(arg1: number, arg2: string): bool
```

Two alternatives were considered:

1. Put the type before the name, like Dart:

```
var list<string> mylist
final list<string> mylist = ['foo']
def Func(number arg1, string arg2) bool
```

2. Put the type after the variable name, but do not use a colon, like Go:

```
var mylist list<string>
final mylist list<string> = ['foo']
def Func(arg1 number, arg2 string) bool
```

The first is more familiar for anyone used to C or Java. The second one doesn't really have an advantage over the first, so let's discard the second.

Since we use type inference the type can be left out when it can be inferred from the value. This means that after `var` we don't know if a type or a name follows. That makes parsing harder, not only for Vim but also for humans. Also, it will not be allowed to use a variable name that could be a type name, using `var string string` is too confusing.

The chosen syntax, using a colon to separate the name from the type, adds punctuation, but it actually makes it easier to recognize the parts of a declaration.

## Expressions

Expression evaluation was already close to what other languages are doing. Some details are unexpected and can be improved. For example a boolean condition would accept a string, convert it to a number and check if the number is non-zero. This is unexpected and often leads to mistakes, since text not starting with a number would be converted to zero, which is considered false. Thus using a string for a condition would often not give an

error and be considered false. That is confusing.

In Vim9 type checking is stricter to avoid mistakes. Where a condition is used, e.g. with the `:if` command and the `||` operator, only boolean-like values are accepted:

```
true: `true`, `v:true`, `1`, `0 < 9`
false: `false`, `v:false`, `0`, `0 > 9`
```

**Note** that the number zero is false and the number one is true. This is more permissive than most other languages. It was done because many builtin functions return these values, and changing that causes more problems than it solves. After using this for a while it turned out to work well.

If you have any type of value and want to use it as a boolean, use the `!!` operator:

```
true: `!!'text'` `!![99]` `!!{'x': 1}` `!!99`
false: `!!''` `!![]` `!!{}`
```

From a language like JavaScript we have this handy construct:

```
GetName() || 'unknown'
```

However, this conflicts with only allowing a boolean for a condition.

Therefore the `??` operator was added:

```
GetName() ?? 'unknown'
```

Here you can explicitly express your intention to use the value as-is and not result in a boolean. This is called the `falsy-operator`.

## Import and Export

A problem of legacy Vim script is that by default all functions and variables are global. It is possible to make them script-local, but then they are not available in other scripts. This defies the concept of a package that only exports selected items and keeps the rest local.

In Vim9 script a mechanism very similar to the JavaScript import and export mechanism is supported. It is a variant to the existing `:source` command that works like one would expect:

- Instead of making everything global by default, everything is script-local, some of these are exported.
- When importing a script the symbols that are imported are explicitly listed, avoiding name conflicts and failures if functionality is added later.
- The mechanism allows for writing a big, long script with a very clear API: the exported functions, variables and classes.
- By using relative paths loading can be much faster for an import inside of a package, no need to search many directories.
- Once an import has been used, its items are cached and loading it again is not needed.
- The Vim-specific use of `"s:"` to make things script-local can be dropped.

When sourcing a Vim9 script (from a Vim9 or legacy script), only the items defined globally can be used, not the exported items. Alternatives considered:

- All the exported items become available as script-local items. This makes it uncontrollable what items get defined and likely soon leads to trouble.
- Use the exported items and make them global. Disadvantage is that it's then

not possible to avoid name clashes in the global namespace.

- Completely disallow sourcing a Vim9 script, require using `:import`. That makes it difficult to use scripts for testing, or sourcing them from the command line to try them out.

Note that you CAN also use `:import` in legacy Vim script, see above.

### Compiling functions early

Functions are compiled when called or when `:defcompile` is used. Why not compile them early, so that syntax and type errors are reported early?

The functions can't be compiled right away when encountered, because there may be forward references to functions defined later. Consider defining functions A, B and C, where A calls B, B calls C, and C calls A again. It's impossible to reorder the functions to avoid forward references.

An alternative would be to first scan through the file to locate items and figure out their type, so that forward references are found, and only then execute the script and compile the functions. This means the script has to be parsed twice, which is slower, and some conditions at the script level, such as checking if a feature is supported, are hard to use. An attempt was made to see if it works, but it turned out to be impossible to make work well.

It would be possible to compile all the functions at the end of the script. The drawback is that if a function never gets called, the overhead of compiling it counts anyway. Since startup speed is very important, in most cases it's better to do it later and accept that syntax and type errors are only reported then. In case these errors should be found early, e.g. when testing, a `:defcompile` command at the end of the script will help out.

### Why not use an existing embedded language?

Vim supports interfaces to Perl, Python, Lua, Tcl and a few others. But these interfaces have never become widely used, for various reasons. When Vim9 was designed a decision was made to make these interfaces lower priority and concentrate on Vim script.

Still, plugin writers may find other languages more familiar, want to use existing libraries or see a performance benefit. We encourage plugin authors to write code in any language and run it as an external process, using jobs and channels. We can try to make this easier somehow.

Using an external tool also has disadvantages. An alternative is to convert the tool into Vim script. For that to be possible without too much translation, and keeping the code fast at the same time, the constructs of the tool need to be supported. Since most languages support classes the lack of support for classes in Vim is then a problem.



This file is about indenting C programs and other files.

1. Indenting C style programs [C-indenting](#)
2. Indenting by expression [indent-expression](#)

---

## 1. Indenting C style programs [C-indenting](#)

The basics for C style indenting are explained in section [30.2](#) of the user manual.

Vim has options for automatically indenting C style program files. Many programming languages including Java and C++ follow very closely the formatting conventions established with C. These options affect only the indent and do not perform other formatting. There are additional options that affect other kinds of formatting as well as indenting, see [format-comments](#), [fo-table](#), [gq](#) and [formatting](#) for the main ones.

There are in fact four main methods available for indentation, each one overrides the previous if it is enabled, or non-empty for **'indentexpr'**:

- 'autoindent'** uses the indent from the previous line.
- 'smartindent'** is like **'autoindent'** but also recognizes some C syntax to increase/reduce the indent where appropriate.
- 'cindent'** Works more cleverly than the other two and is configurable to different indenting styles.
- 'indentexpr'** The most flexible of all: Evaluates an expression to compute the indent of a line. When non-empty this method overrides the other ones. See [indent-expression](#).

The rest of this section describes the **'cindent'** option.

**Note** that **'cindent'** indenting does not work for every code scenario. Vim is not a C compiler: it does not recognize all syntax. One requirement is that toplevel functions have a '{' in the first column. Otherwise they are easily confused with declarations.

These five options control C program indenting:

- 'cindent'** Enables Vim to perform C program indenting automatically.
- 'cinkeys'** Specifies which keys trigger reindenting in insert mode.
- 'cinoptions'** Sets your preferred indent style.
- 'cinwords'** Defines keywords that start an extra indent in the next line.
- 'cinscopedecls'** Defines strings that are recognized as a C++ scope declaration.

If **'lisp'** is not on and **'equalprg'** is empty, the "=" operator indents using Vim's built-in algorithm rather than calling an external program.

See [autocommand](#) for how to set the **'cindent'** option automatically for C code files and reset it for others.

cinkeys-format    indentkeys-format

The '**cinkeys**' option is a string that controls Vim's indenting in response to typing certain characters or commands in certain contexts. **Note** that this not only triggers C-indenting. When '**indentexpr**' is not empty '**indentkeys**' is used instead. The format of '**cinkeys**' and '**indentkeys**' is equal.

The default is "0{,0},0),0],:,0#,!^F,o,0,e" which specifies that indenting occurs as follows:

|       |                                                                                       |
|-------|---------------------------------------------------------------------------------------|
| "0{"  | if you type '{' as the first character in a line                                      |
| "0}"  | if you type '}' as the first character in a line                                      |
| "0)"  | if you type ')' as the first character in a line                                      |
| "0]"  | if you type ']' as the first character in a line                                      |
| ":"   | if you type ':' after a label or case statement                                       |
| "0#"  | if you type '#' as the first character in a line                                      |
| "!^F" | if you type <b>CTRL-F</b> (which is not inserted)                                     |
| "o"   | if you type a <b>&lt;CR&gt;</b> anywhere or use the "o" command (not in insert mode!) |
| "O"   | if you use the "O" command (not in insert mode!)                                      |
| "e"   | if you type the second 'e' for an "else" at the start of a line                       |

Characters that can precede each key:

i\_CTRL-F

- ! When a '!' precedes the key, Vim will not insert the key but will instead reindent the current line. This allows you to define a command key for reindenting the current line. **CTRL-F** is the default key for this. Be careful if you define **CTRL-I** for this because **CTRL-I** is the ASCII code for **<Tab>**.
- \* When a '\*' precedes the key, Vim will reindent the line before inserting the key. If '**cinkeys**' contains "**\*<Return>**", Vim reindents the current line before opening a new line.
- 0 When a zero precedes the key (but appears after '!' or '\*') Vim will reindent the line only if the key is the first character you type in the line. When used before "=" Vim will only reindent the line if there is only white space before the word.

When neither '!' nor '\*' precedes the key, Vim reindents the line after you type the key. So ';' sets the indentation of a line which includes the ';'.

Special key names:

- <> Angle brackets mean spelled-out names of keys. For example: "**<Up>**", "**<Ins>**" (see [key-notation](#) ).
- ^ Letters preceded by a caret (^) are control characters. For example: "**^F**" is **CTRL-F**.
- o Reindent a line when you use the "o" command or when Vim opens a new line below the current one (e.g., when you type **<Enter>** in insert mode).
- O Reindent a line when you use the "O" command.
- e Reindent a line that starts with "else" when you type the second 'e'.
- :
- Reindent a line when a ':' is typed which is after a label or case statement. Don't reindent for a ":" in "class::method" for C++. To Reindent for any ":", use "**<:>**".
- =word Reindent when typing the last character of "word". "word" may actually be part of another word. Thus "=end" would cause reindenting



when typing the "d" in "endif" or "endwhile". But not when typing "bend". Also reindent when completion produces a word that starts with "word". "0=word" reindents when there is only white space before the word.

=~word Like =word, but ignore case.

If you really want to reindent when you type 'o', 'O', 'e', 'E', '<', '>', '\*', ':', '!' or '!', use "<o>", "<O>", "<e>", "<E>", "<<>", "<>>", "<\*>", "<:>" or "<!>", respectively, for those keys.

For an emacs-style indent mode where lines aren't indented every time you press <Enter> but only if you press <Tab>, I suggest:

```
:set cinkeys=0{,0},:,0#,!<Tab>,!^F
```

You might also want to switch off 'autoindent' then.

**Note:** If you change the current line's indentation manually, Vim ignores the cindent settings for that line. This prevents vim from reindenting after you have changed the indent by typing <BS>, <Tab>, or <Space> in the indent or used **CTRL-T** or **CTRL-D**.

#### cinoptions-values

The 'cinoptions' option sets how Vim performs indentation. The value after the option character can be one of these (N is any number):

```
N indent N spaces
-N indent N spaces to the left
Ns N times 'shiftwidth' spaces
-Ns N times 'shiftwidth' spaces to the left
```

In the list below,

"N" represents a number of your choice (the number can be negative). When there is an 's' after the number, Vim multiplies the number by 'shiftwidth': "1s" is 'shiftwidth', "2s" is two times 'shiftwidth', etc. You can use a decimal point, too: "-0.5s" is minus half a 'shiftwidth'.

The examples below assume a 'shiftwidth' of 4.

#### cin->

>N Amount added for "normal" indent. Used after a line that should increase the indent (lines starting with "if", an opening brace, etc.). (default 'shiftwidth').

```
cin= cin=>2 cin=>2s
 if (cond) if (cond) if (cond)
 { { {
 foo; foo; foo;
 } } }
```

#### cin-e

eN Add N to the prevailing indent inside a set of braces if the opening brace at the End of the line (more precise: is not the first character in a line). This is useful if you want a different indent when the '{' is at the start of the line from when '{' is at the end of the line. (default 0).

```
cin= cin=e2 cin=e-2
 if (cond) { if (cond) { if (cond) {
```

```

 foo;
 }
else
{
 bar;
}

```

**cin0-n**

nN Add N to the prevailing indent for a statement after an "if", "while", etc., if it is NOT inside a set of braces. This is useful if you want a different indent when there is no '{' before the statement from when there is a '{' before it. (default 0).

```

cin0=
 if (cond)
 foo;
 else
 {
 bar;
 }

```

```

cin0=n2
 if (cond)
 foo;
 else
 {
 bar;
 }

```

```

cin0=n-2
 if (cond)
 foo;
 else
 {
 bar;
 }

```

**cin0-f**

fN Place the first opening brace of a function or other block in column N. This applies only for an opening brace that is not inside other braces and is at the start of the line. What comes after the brace is put relative to this brace. (default 0).

```

cin0=
 func()
 {
 int foo;
 }

```

```

cin0=f.5s
 func()
 {
 int foo;
 }

```

```

cin0=f1s
 func()
 {
 int foo;
 }

```

**cin0-{**

{N Place opening braces N characters from the prevailing indent. This applies only for opening braces that are inside other braces. (default 0).

```

cin0=
 if (cond)
 {
 foo;
 }

```

```

cin0={.5s
 if (cond)
 {
 foo;
 }

```

```

cin0={1s
 if (cond)
 {
 foo;
 }

```

**cin0-}**

}N Place closing braces N characters from the matching opening brace. (default 0).

```

cin0=
 if (cond)
 {
 foo;
 }

```

```

cin0={2,}-0.5s
 if (cond)
 {
 foo;
 }

```

```

cin0=}2
 if (cond)
 {
 foo;
 }

```

**cin0-^**

**^N** Add N to the prevailing indent inside a set of braces if the opening brace is in column 0. This can specify a different indent for whole of a function (some may like to set it to a negative number). (default 0).

|                                                                        |                                                                           |                                                                           |
|------------------------------------------------------------------------|---------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <pre> cino= func() {     if (cond)     {         a = b;     } } </pre> | <pre> cino=^-2 func() {     if (cond)     {         a = b;     } } </pre> | <pre> cino=^-s func() {     if (cond)     {         a = b;     } } </pre> |
|------------------------------------------------------------------------|---------------------------------------------------------------------------|---------------------------------------------------------------------------|

**LN** Controls placement of jump labels. If N is negative, the label will be placed at column 1. If N is non-negative, the indent of the label will be the prevailing indent minus N. (default -1).

|                                                                    |                                                                      |                                                                      |
|--------------------------------------------------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------|
| <pre> cino= func() {     {         stmt;     LABEL:     } } </pre> | <pre> cino=L2 func() {     {         stmt;     LABEL:     } } </pre> | <pre> cino=Ls func() {     {         stmt;     LABEL:     } } </pre> |
|--------------------------------------------------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------|

**:N** Place case labels N characters from the indent of the switch(). (default 'shiftwidth').

|                                                                           |                                                                            |
|---------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <pre> cino= switch (x) {     case 1:         a = b;     default: } </pre> | <pre> cino=:0 switch(x) {     case 1:         a = b;     default: } </pre> |
|---------------------------------------------------------------------------|----------------------------------------------------------------------------|

**=N** Place statements occurring after a case label N characters from the indent of the label. (default 'shiftwidth').

|                                            |                                                              |
|--------------------------------------------|--------------------------------------------------------------|
| <pre> cino= case 11:     a = a + 1; </pre> | <pre> cino==10 case 11: a = a + 1;         b = b + 1; </pre> |
|--------------------------------------------|--------------------------------------------------------------|

**lN** If N != 0 Vim will align with a case label instead of the statement after it in the same line.

|                                 |                                   |
|---------------------------------|-----------------------------------|
| <pre> cino= switch (a) { </pre> | <pre> cino=l1 switch (a) { </pre> |
|---------------------------------|-----------------------------------|

```

 case 1: {
 break;
 }

```

cino-b

bN If N != 0 Vim will align a final "break" with the case label, so that case..break looks like a sort of block. (default: 0). When using 1, consider adding "0=break" to 'cinkeys'.

```

cino=
switch (x)
{
 case 1:
 a = b;
 break;

 default:
 a = 0;
 break;
}

cino=b1
switch(x)
{
 case 1:
 a = b;
 break;

 default:
 a = 0;
 break;
}

```

cino-g

gN Place C++ scope declarations N characters from the indent of the block they are in. (default 'shiftwidth'). By default, a scope declaration is "public:", "protected:" or "private:". This can be adjusted with the 'cinscopedecls' option.

```

cino=
{
 public:
 a = b;
 private:
}

cino=g0
{
 public:
 a = b;
 private:
}

```

cino-h

hN Place statements occurring after a C++ scope declaration N characters from the indent of the label. (default 'shiftwidth').

```

cino=
 public:
 a = a + 1;

cino=h10
 public: a = a + 1;
 b = b + 1;

```

cino-N

NN Indent inside C++ namespace N characters extra compared to a normal block. (default 0).

```

cino=
namespace {
 void function();
}

namespace my
{

cino=N-s
namespace {
 void function();
}

namespace my
{

```

```

 void function();
 }
 void function();
 }

```

### cin0-E

EN Indent inside C++ linkage specifications (extern "C" or extern "C++") N characters extra compared to a normal block. (default 0).

```

cin0=
extern "C" {
 void function();
}

extern "C"
{
 void function();
}

cin0-E-s
extern "C" {
 void function();
}

extern "C"
{
 void function();
}

```

### cin0-p

pN Parameter declarations for K&R-style function declarations will be indented N characters from the margin. (default 'shiftwidth').

```

cin0=
func(a, b)
int a;
char b;

cin0-p0
func(a, b)
int a;
char b;

cin0-p2s
func(a, b)
int a;
char b;

```

### cin0-t

tN Indent a function return type declaration N characters from the margin. (default 'shiftwidth').

```

cin0=
int
func()

cin0-t0
int
func()

cin0-t7
int
func()

```

### cin0-i

iN Indent C++ base class declarations and constructor initializations, if they start in a new line (otherwise they are aligned at the right side of the ':'). (default 'shiftwidth').

```

cin0=
class MyClass :
 public BaseClass
{}
MyClass::MyClass() :
 BaseClass(3)
{}

cin0-i0
class MyClass :
 public BaseClass
{}
MyClass::MyClass() :
 BaseClass(3)
{}

```

### cin0+

+N Indent a continuation line (a line that spills onto the next) inside a function N additional characters. (default 'shiftwidth').

Outside of a function, when the previous line ended in a backslash, the 2 \* N is used.

```

cino= cino=+10
 a = b + 9 * a = b + 9 *
 c; c;

```

**cino-c**

cN Indent comment lines after the comment opener, when there is no other text with which to align, N characters from the comment opener. (default 3). See also [format-comments](#) .

```

cino= cino=c5
 /* /*
 text. text.
 */ */

```

**cino-C**

CN When N is non-zero, indent comment lines by the amount specified with the c flag above even if there is other text behind the comment opener. (default 0).

```

cino=c0 cino=c0,C1
 /****** /******
 text. text.
 /******/ /******/

```

(Example uses ":set comments& comments--s1:/\* comments^=s0:/\*")

**cino-/**

/N Indent comment lines N characters extra. (default 0).

```

cino= cino=/4
 a = b; a = b;
 /* comment */ /* comment */
 c = d; c = d;

```

**cino-(**

(N When in unclosed parentheses, indent N characters from the line with the unclosed parenthesis. Add a **'shiftwidth'** for every extra unclosed parentheses. When N is 0 or the unclosed parenthesis is the first non-white character in its line, line up with the next non-white character after the unclosed parenthesis. (default **'shiftwidth'** \* 2).

```

cino= cino=(0
 if (c1 && (c2 || if (c1 && (c2 ||
 c3)) c3))
 foo; foo;
 if (c1 && if (c1 &&
 (c2 || c3)) (c2 || c3))
 { {

```

**cino-u**

uN Same as (N, but for one nesting level deeper. (default **'shiftwidth'**).

|                                                                                |                                                                                  |
|--------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <pre> cino=   if (c123456789       &amp;&amp; (c22345              c3)) </pre> | <pre> cino=u2   if (c123456789       &amp;&amp; (c22345              c3)) </pre> |
|--------------------------------------------------------------------------------|----------------------------------------------------------------------------------|

cino-U

UN When N is non-zero, do not ignore the indenting specified by ( or u in case that the unclosed parenthesis is the first non-white character in its line. (default 0).

|                                                                                           |                                                                                     |
|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <pre> cino= or cino=(s   c = c1 &amp;&amp;   (     c2        c3   ) &amp;&amp; c4; </pre> | <pre> cino=(s,U1   c = c1 &amp;&amp;   (     c2        c3   ) &amp;&amp; c4; </pre> |
|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|

cino-w

WN When in unclosed parentheses and N is non-zero and either using "(0" or "u0", respectively, or using "U0" and the unclosed parenthesis is the first non-white character in its line, line up with the character immediately after the unclosed parenthesis rather than the first non-white character. (default 0).

|                                                                               |                                                                                  |
|-------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <pre> cino=(0   if ( c1       &amp;&amp; ( c2              c3))   foo; </pre> | <pre> cino=(0,w1   if ( c1       &amp;&amp; ( c2              c3))   foo; </pre> |
|-------------------------------------------------------------------------------|----------------------------------------------------------------------------------|

cino-W

WN When in unclosed parentheses and N is non-zero and either using "(0" or "u0", respectively and the unclosed parenthesis is the last non-white character in its line and it is not the closing parenthesis, indent the following line N characters relative to the outer context (i.e. start of the line or the next unclosed parenthesis). (default: 0).

|                                                                                                          |                                                                                                             |
|----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <pre> cino=(0   a_long_line(     argument,     argument);   a_short_line(argument,     argument); </pre> | <pre> cino=(0,W4   a_long_line(     argument,     argument);   a_short_line(argument,     argument); </pre> |
|----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|

cino-k

kN When in unclosed parentheses which follow "if", "for" or "while" and N is non-zero, overrides the behaviour defined by "(N": causes the indent to be N characters relative to the outer context (i.e. the line where "if", "for" or "while" is). Has no effect on deeper levels of nesting. Affects flags like "wN" only for the "if", "for" and "while" conditions. If 0, defaults to behaviour defined by the "(N" flag. (default: 0).

```

cino=(0
 if (condition1
 && condition2)
 action();
 function(argument1
 && argument2);

cino=(0,ks
 if (condition1
 && condition2)
 action();
 function(argument1
 && argument2);

```

[cino-m](#)

mN When N is non-zero, line up a line starting with a closing parenthesis with the first character of the line with the matching opening parenthesis. (default 0).

```

cino=(s
 c = c1 && (
 c2 ||
 c3
) && c4;
 if (
 c1 && c2
)
 foo;

cino=(s,m1
 c = c1 && (
 c2 ||
 c3
) && c4;
 if (
 c1 && c2
)
 foo;

```

[cino-M](#)

MN When N is non-zero, line up a line starting with a closing parenthesis with the first character of the previous line. (default 0).

```

cino=
 if (cond1 &&
 cond2
)

cino=M1
 if (cond1 &&
 cond2
)

```

[java-cinoptions](#) [java-indenting](#) [cino-j](#)

jN Indent Java anonymous classes correctly. Also works well for Javascript. The value 'N' is currently unused but must be non-zero (e.g. 'j1'). 'j1' will indent for example the following code snippet correctly:

```

object.add(new ChangeListener() {
 public void stateChanged(ChangeEvent e) {
 do_something();
 }
});

```

[javascript-cinoptions](#) [javascript-indenting](#) [cino-J](#)

JN Indent JavaScript object declarations correctly by not confusing them with labels. The value 'N' is currently unused but must be non-zero (e.g. 'J1'). If you enable this you probably also want to set [cino-j](#) .

```

var bar = {
 foo: {
 that: this,

```



```

 some: ok,
 },
 "bar":{
 a : 2,
 b: "123abc",
 x: 4,
 "y": 5
 }
}

```

- cino-)
- )N Vim searches for unclosed parentheses at most N lines away. This limits the time needed to search for parentheses. (default 20 lines).
- cino-star
- \*N Vim searches for unclosed comments at most N lines away. This limits the time needed to search for the start of a comment. If your /\* \*/ comments stop indenting after N lines this is the value you will want to change. (default 70 lines).
- cino-#
- #N When N is non-zero recognize shell/Perl comments starting with '#', do not recognize preprocessor lines; allow right-shifting lines that start with "#". When N is zero (default): don't recognize '#' comments, do recognize preprocessor lines; right-shifting lines that start with "#" does not work.
- cino-P
- PN When N is non-zero recognize C pragmas, and indent them like any other code; does not concern other preprocessor directives. When N is zero (default): don't recognize C pragmas, treating them like every other preprocessor directive.

The defaults, spelled out in full, are:

```

cinoptions=>s,e0,n0,f0,{0},0,^0,L-1,:s,=s,l0,b0,gs,hs,N0,E0,ps,ts,is,+s,
c3,C0,/0,(2s,us,U0,w0,W0,k0,m0,j0,J0,)20,*70,#0,P0

```

Vim puts a line in column 1 if:

- It starts with '#' (preprocessor directives), if '**cinkeys**' contains '#0'.
- It starts with a label (a keyword followed by ':', other than "case" and "default") and '**cinoptions**' does not contain an 'L' entry with a positive value.
- Any combination of indentations causes the line to have less than 0 indentation.

## 2. Indenting by expression

indent-expression

The basics for using flexible indenting are explained in section 30.3 of the user manual.

If you want to write your own indent file, it must set the `'indentexpr'` option. Setting the `'indentkeys'` option is often useful. See the `$VIMRUNTIME/indent/README.txt` file for hints. See the `$VIMRUNTIME/indent` directory for examples.

## REMARKS ABOUT SPECIFIC INDENT FILES

### CLOJURE

`ft-clojure-indent`    `clojure-indent`

Clojure indentation differs somewhat from traditional Lisps, due in part to the use of square and curly brackets, and otherwise by community convention. These conventions are not universally followed, so the Clojure indent script offers a few configuration options.

(If the current Vim does not include `searchpairpos()`, the indent script falls back to normal `'lisp'` indenting, and the following options are ignored.)

`g:clojure_maxlines`

Sets maximum scan distance of ``searchpairpos()`. Larger values trade performance for correctness when dealing with very long forms. A value of 0 will scan without limits. The default is 300.

`g:clojure_fuzzy_indent`  
`g:clojure_fuzzy_indent_patterns`  
`g:clojure_fuzzy_indent_blacklist`

The `'lispwords'` option is a list of comma-separated words that mark special forms whose subforms should be indented with two spaces.

For example:

```
(defn bad []
 "Incorrect indentation")

(defn good []
 "Correct indentation")
```

If you would like to specify `'lispwords'` with a `pattern` instead, you can use the fuzzy indent feature:

```
" Default
let g:clojure_fuzzy_indent = 1
let g:clojure_fuzzy_indent_patterns = ['^with', '^def', '^let']
let g:clojure_fuzzy_indent_blacklist =
 \ ['-fn$', '\v^with-(meta|out-str|loading-context)$']
```

`g:clojure_fuzzy_indent_patterns` and `g:clojure_fuzzy_indent_blacklist` are lists of patterns that will be matched against the unqualified symbol at the

head of a list. This means that a pattern like `"^foo"` will match all these candidates: `'foobar'`, `'my.ns/foobar'`, and `'#'foobar'`.

Each candidate word is tested for special treatment in this order:

1. Return true if word is literally in `'lispwords'`
2. Return false if word matches a pattern in `g:clojure_fuzzy_indent_blacklist`
3. Return true if word matches a pattern in `g:clojure_fuzzy_indent_patterns`
4. Return false and indent normally otherwise

`g:clojure_special_indent_words`

Some forms in Clojure are indented such that every subform is indented by only two spaces, regardless of `'lispwords'`. If you have a custom construct that should be indented in this idiosyncratic fashion, you can add your symbols to the default list below.

```
" Default
let g:clojure_special_indent_words =
 \ 'deftype,defrecord,reify,proxy,extend-type,extend-protocol,letfn'
```

`g:clojure_align_multiline_strings`

Align subsequent lines in multi-line strings to the column after the opening quote, instead of the same column.

For example:

```
(def default
 "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
 enim ad minim veniam, quis nostrud exercitation ullamco laboris
 nisi ut aliquip ex ea commodo consequat.")

(def aligned
 "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
 enim ad minim veniam, quis nostrud exercitation ullamco laboris
 nisi ut aliquip ex ea commodo consequat.")
```

`g:clojure_align_subforms`

By default, parenthesized compound forms that look like function calls and whose head subform is on its own line have subsequent subforms indented by two spaces relative to the opening paren:

```
(foo
 bar
 baz)
```

Setting this option to `1` changes this behaviour so that all subforms are aligned to the same column, emulating the default behaviour of `clojure-mode.el`:

```
(foo
 bar
 baz)
```

## **FORTRAN**

## **ft-fortran-indent**

Block `if`, `select case`, `select type`, `select rank`, `where`, `forall`, `type`, `interface`, `associate`, `block`, `enum`, `critical`, and `change team` constructs are indented. The indenting of subroutines, functions, modules, and program blocks is optional. Comments, labeled statements, and continuation lines are indented if the Fortran is in free source form, whereas they are not indented if the Fortran is in fixed source form because of the left margin requirements. Hence manual indent corrections will be necessary for labeled statements and continuation lines when fixed source form is being used. For further discussion of the method used for the detection of source format see [ft-fortran-syntax](#) .

### **Do loops**

All do loops are left unindented by default. Do loops can be unstructured in Fortran with (possibly multiple) loops ending on a labeled executable statement of almost arbitrary type. Correct indentation requires compiler-quality parsing. Old code with do loops ending on labeled statements of arbitrary type can be indented with elaborate programs such as Tidy. Structured do/continue loops are also left unindented because continue statements are used for purposes other than ending a do loop. Programs such as Tidy can convert structured do/continue loops to the do/enddo form. Do loops of the do/enddo variety can be indented. If you use only structured loops of the do/enddo form, you should declare this by setting the `fortran_do_enddo` variable in your `.vimrc` as follows

```
let fortran_do_enddo=1
```

in which case do loops will be indented. If all your loops are of do/enddo type only in, say, `.f90` files, then you should set a buffer flag with an autocommand such as

```
au! BufRead,BufNewFile *.f90 let b:fortran_do_enddo=1
```

to get do loops indented in `.f90` files and left alone in Fortran files with other extensions such as `.for`.

### **Program units**

Indenting of program units (subroutines, functions, modules, and program blocks) can be increased by setting the variable `fortran_indent_more` and can be decreased by setting the variable `fortran_indent_less`. These variables can be set for all fortran files in your `.vimrc` as follows

```
let fortran_indent_less=1
```

A finer level of control can be achieved by setting the corresponding buffer-local variable as follows

```
let b:fortran_indent_less=1
```

**HTML** `ft-html-indent` `html-indent` `html-indenting`

This is about variables you can set in your vimrc to customize HTML indenting.

You can set the indent for the first line after `<script>` and `<style>` "blocktags" (default "zero"):

```
:let g:html_indent_script1 = "inc"
:let g:html_indent_style1 = "inc"
```

| VALUE  | MEANING                                   |
|--------|-------------------------------------------|
| "zero" | zero indent                               |
| "auto" | auto indent (same indent as the blocktag) |
| "inc"  | auto indent + one indent step             |

You can set the indent for attributes after an open `<tag` line:

```
:let g:html_indent_attribute = 1
```

| VALUE | MEANING                                                     |
|-------|-------------------------------------------------------------|
| 1     | auto indent, one indent step more than <code>&lt;tag</code> |
| 2     | auto indent, two indent steps (default)                     |
| > 2   | auto indent, more indent steps                              |

Many tags increase the indent for what follows per default (see "Add Indent Tags" in the script). You can add further tags with:

```
:let g:html_indent_inctags = "html,body,head,tbody"
```

You can also remove such tags with:

```
:let g:html_indent_autotags = "th,td,tr,tfoot,thead"
```

Default value is empty for both variables. **Note:** the initial "inctags" are only defined once per Vim session.

User variables are only read when the script is sourced. To enable your changes during a session, without reloading the HTML file, you can manually do:

```
:call HtmlIndent_CheckUserSettings()
```

Detail:

Calculation of indent inside "blocktags" with "alien" content:

| BLOCKTAG                    | INDENT | EXPR            | WHEN APPLICABLE                       |
|-----------------------------|--------|-----------------|---------------------------------------|
| <code>&lt;script&gt;</code> | :      | {customizable}  | if first line of block                |
|                             | :      | cindent(v:lnum) | if attributes empty or contain "java" |

```

: -1 else (vbscript, tcl, ...)
<style> : {customizable} if first line of block
: GetCSSIndent() else
<!-- --> : -1

```

## MATLAB [ft-matlab-indent](#) [matlab-indent](#) [matlab-indenting](#)

The setting Function indenting format in MATLAB Editor/Debugger Language Preferences corresponds to:

```
:let g:MATLAB_function_indent = {0, 1 or 2 (default)}
```

Where 0 is for Classic, 1 for Indent nested functions and 2 for Indent all functions.

## PHP [ft-php-indent](#) [php-indent](#) [php-indenting](#)

**NOTE:** PHP files will be indented correctly only if PHP [syntax](#) is active.

If you are editing a file in Unix **'fileformat'** and '\r' characters are present before new lines, indentation won't proceed correctly ; you have to remove those useless characters first with a command like:

```
:%s /\r$/g
```

Or, you can simply `:let` the variable `PHP_removeCRwhenUnix` to 1 and the script will silently remove them when Vim loads a PHP file (at each `BufRead` ).

### OPTIONS:

PHP indenting can be altered in several ways by modifying the values of some global variables:

To not enable auto-formatting of comments by default (if you want to use your own **'formatoptions'**):

```
:let g:PHP_autoformatcomment = 0
```

Else, 't' will be removed from the **'formatoptions'** string and "qrowcb" will be added, see [fo-table](#) for more information.

-----

To add extra indentation to single-line comments:

```
:let g:PHP_outdentSLComments = N
```

[PHP\\_outdentSLComments](#)

With N being the number of **'shiftwidth'** to add.

Only single-line comments will be affected such as:

```

Comment
// Comment
/* Comment */

```

-----

`*PHP_default_indenting*`

To add extra indentation to every PHP lines with N being the number of 'shiftwidth' to add:

```
:let g:PHP_default_indenting = N
```

For example, with N = 1, this will give:

```
<?php
 if (!isset($History_lst_sel))
 if (!isset($History_lst_sel))
 if (!isset($History_lst_sel)) {
 $History_lst_sel=0;
 } else
 $foo="bar";

 $command_hist = TRUE;
?>
```

(Notice the extra indentation between the PHP container markers and the code)

-----

`PHP_outdentphpescape`

To indent PHP escape tags as the surrounding non-PHP code (only affects the PHP escape tags):

```
:let g:PHP_outdentphpescape = 0
```

-----

`*PHP_removeCRwhenUnix*`

To automatically remove '\r' characters when the 'fileformat' is set to Unix:

```
:let g:PHP_removeCRwhenUnix = 1
```

-----

`*PHP_BracesAtCodeLevel*`

To indent braces at the same level than the code they contain:

```
:let g:PHP_BracesAtCodeLevel = 1
```

This will give the following result:

```
if ($foo)
{
 foo();
}
```

Instead of:

```
if ($foo)
{
 foo();
}
```

**NOTE:** Indenting will be a bit slower if this option is used because some optimizations won't be available.

-----

`PHP_vintage_case_default_indent`

To indent 'case:' and 'default:' statements in switch() blocks:

```
:let g:PHP_vintage_case_default_indent = 1
```

In PHP braces are not required inside 'case/default' blocks therefore 'case:' and 'default:' are indented at the same level than the 'switch()' to avoid meaningless indentation. You can use the above option to return to the traditional way.

---

#### PHP\_noArrowMatching

By default the indent script will indent multi-line chained calls by matching the position of the '->':

```
$user_name_very_long->name()
 ->age()
 ->info();
```

You can revert to the classic way of indenting by setting this option to 1:

```
:let g:PHP_noArrowMatching = 1
```

You will obtain the following result:

```
$user_name_very_long->name()
->age()
->info();
```

---

#### \*PHP\_IndentFunctionCallParameters\*

Extra indentation levels to add to parameters in multi-line function calls.

```
let g:PHP_IndentFunctionCallParameters = 1
```

Function call arguments will indent 1 extra level. For two-space indentation:

```
function call_the_thing(
 $with_this,
 $and_that
) {
 $this->do_the_thing(
 $with_this,
 $and_that
);
}
```

---

#### \*PHP\_IndentFunctionDeclarationParameters\*

Extra indentation levels to add to arguments in multi-line function definitions.

```
let g:PHP_IndentFunctionDeclarationParameters = 1
```

Function arguments in declarations will indent 1 extra level. For two-space indentation:

```
function call_the_thing(
 $with_this,
```



```

 $and_that
) {
 $this->do_the_thing(
 $with_this,
 $and_that
);
}

```

## PYTHON

ft-python-indent

The amount of indent can be set with the ``g:python_indent`` Dictionary, which needs to be created before adding the items:

```
let g:python_indent = {}
```

The examples given are the defaults. Note that the dictionary values are set to an expression, so that you can change the value of `'shiftwidth'` later without having to update these values.

Indent after an open paren:

```
let g:python_indent.open_paren = 'shiftwidth() * 2'
```

Indent after a nested paren:

```
let g:python_indent.nested_paren = 'shiftwidth()'
```

Indent for a continuation line:

```
let g:python_indent.continue = 'shiftwidth() * 2'
```

By default, the closing paren on a multiline construct lines up under the first non-whitespace character of the previous line.

If you prefer that it's lined up under the first character of the line that starts the multiline construct, reset this key:

```
let g:python_indent.closed_paren_align_last_line = v:false
```

The method uses `searchpair()` to look back for unclosed parentheses. This can sometimes be slow, thus it timeouts after 150 msec. If you notice the indenting isn't correct, you can set a larger timeout in msec:

```
let g:python_indent.searchpair_timeout = 500
```

If looking back for unclosed parenthesis is still too slow, especially during a copy-paste operation, or if you don't need indenting inside multi-line parentheses, you can completely disable this feature:

```
let g:python_indent.disable_parentheses_indenting = 1
```

For backward compatibility, these variables are also supported:

```

g:pyindent_open_paren
g:pyindent_nested_paren
g:pyindent_continue
g:pyindent_searchpair_timeout
g:pyindent_disable_parentheses_indenting

```

## R

ft-r-indent

Function arguments are aligned if they span for multiple lines. If you prefer do not have the arguments of functions aligned, put in your `vimrc` :

```
let r_indent_align_args = 0
```

All lines beginning with a comment character, #, get the same indentation level of the normal R code. Users of Emacs/ESS may be used to have lines beginning with a single # indented in the 40th column, ## indented as R code, and ### not indented. If you prefer that lines beginning with comment characters are aligned as they are by Emacs/ESS, put in your `vimrc` :

```
let r_indent_ess_comments = 1
```

If you prefer that lines beginning with a single # are aligned at a column different from the 40th one, you should set a new value to the variable `r_indent_comment_column`, as in the example below:

```
let r_indent_comment_column = 30
```

Any code after a line that ends with "<-" is indented. Emacs/ESS does not indent the code if it is a top-level function. If you prefer a behavior like Emacs/ESS one in this regard, put in your `vimrc` :

```
let r_indent_ess_compatible = 1
```

Below is an example of indentation with and without this option enabled:

```
r_indent_ess_compatible = 1 ### r_indent_ess_compatible = 0
foo <- foo <-
 function(x) function(x)
{ {
 paste(x) paste(x)
}
```

The code will be indented after lines that match the pattern ``\(&|\||\|+|\-|\|*\|/|\|=|\|~|\%|\->|\)\s*$``. If you want indentation after lines that match a different pattern, you should set the appropriate value of ``r_indent_op_pattern`` in your `vimrc` .

## SHELL

## ft-sh-indent

The amount of indent applied under various circumstances in a shell file can be configured by setting the following keys in the `Dictionary` `b:sh_indent_defaults` to a specific amount or to a `Funcref` that references a function that will return the amount desired:

`b:sh_indent_options['default']` Default amount of indent.

`b:sh_indent_options['continuation-line']`  
Amount of indent to add to a continued line.

`b:sh_indent_options['case-labels']`  
Amount of indent to add for case labels.  
(not actually implemented)

`b:sh_indent_options['case-statements']`

Amount of indent to add for case statements.

```
b:sh_indent_options['case-breaks']
```

Amount of indent to add (or more likely remove) for case breaks.

## VERILOG

ft-verilog-indent

General block statements such as if, for, case, always, initial, function, specify and begin, etc., are indented. The module block statements (first level blocks) are not indented by default. you can turn on the indent with setting a variable in the .vimrc as follows:

```
let b:verilog_indent_modules = 1
```

then the module blocks will be indented. To stop this, remove the variable:

```
:unlet b:verilog_indent_modules
```

To set the variable only for Verilog file. The following statements can be used:

```
au BufReadPost * if exists("b:current_syntax")
au BufReadPost * if b:current_syntax == "verilog"
au BufReadPost * let b:verilog_indent_modules = 1
au BufReadPost * endif
au BufReadPost * endif
```

Furthermore, setting the variable b:verilog\_indent\_width to change the indenting width (default is 'shiftwidth'):

```
let b:verilog_indent_width = 4
let b:verilog_indent_width = shiftwidth() * 2
```

In addition, you can turn the verbose mode for debug issue:

```
let b:verilog_indent_verbose = 1
```

Make sure to do ":set cmdheight=2" first to allow the display of the message.

## VHDL

ft-vhdl-indent

Alignment of generic/port mapping statements are performed by default. This causes the following alignment example:

```
ENTITY sync IS
PORT (
 clk : IN STD_LOGIC;
 reset_n : IN STD_LOGIC;
 data_input : IN STD_LOGIC;
 data_out : OUT STD_LOGIC
);
END ENTITY sync;
```

To turn this off, add

```
let g:vhdl_indent_genportmap = 0
```

to the .vimrc file, which causes the previous alignment example to change:

```
ENTITY sync IS
PORT (
 clk : IN STD_LOGIC;
 reset_n : IN STD_LOGIC;
 data_input : IN STD_LOGIC;
 data_out : OUT STD_LOGIC
);
END ENTITY sync;
```

---

Alignment of right-hand side assignment "<=" statements are performed by default. This causes the following alignment example:

```
sig_out <= (bus_a(1) AND
 (sig_b OR sig_c)) OR
 (bus_a(0) AND sig_d);
```

To turn this off, add

```
let g:vhdl_indent_rhsassign = 0
```

to the .vimrc file, which causes the previous alignment example to change:

```
sig_out <= (bus_a(1) AND
 (sig_b OR sig_c)) OR
 (bus_a(0) AND sig_d);
```

---

Full-line comments (lines that begin with "--") are indented to be aligned with the very previous line's comment, PROVIDED that a whitespace follows after "--".

For example:

```
sig_a <= sig_b; -- start of a comment
 -- continuation of the comment
 -- more of the same comment
```

While in Insert mode, after typing "-- " (note the space " "), hitting **CTRL-F** will align the current "-- " with the previous line's "--".

If the very previous line does not contain "--", THEN the full-line comment will be aligned with the start of the next non-blank line that is NOT a full-line comment.

Indenting the following code:

```
sig_c <= sig_d; -- comment 0
 -- comment 1
 -- comment 2
--debug_code:
--PROCESS(debug_in)
 --BEGIN
 -- FOR i IN 15 DOWNT0 0 LOOP
 -- debug_out(8*i+7 DOWNT0 8*i) <= debug_in(15-i);
 -- END LOOP;
--END PROCESS debug_code;

 -- comment 3
sig_e <= sig_f; -- comment 4
 -- comment 5
```

results in:

```
sig_c <= sig_d; -- comment 0
 -- comment 1
 -- comment 2
--debug_code:
--PROCESS(debug_in)
--BEGIN
-- FOR i IN 15 DOWNT0 0 LOOP
-- debug_out(8*i+7 DOWNT0 8*i) <= debug_in(15-i);
-- END LOOP;
--END PROCESS debug_code;

-- comment 3
sig_e <= sig_f; -- comment 4
 -- comment 5
```

Notice that "--debug\_code:" does not align with "-- comment 2" because there is no whitespace that follows after "--" in "--debug\_code:".

Given the dynamic nature of indenting comments, indenting should be done TWICE. On the first pass, code will be indented. On the second pass, full-line comments will be indented according to the correctly indented code.

## VIM

ft-vim-indent  
g:vim\_indent

Vim scripts indentation can be configured with the `g:vim_indent` dictionary variable. It supports 3 keys, `line_continuation`, `more_in_bracket_block`, and `searchpair_timeout`.

`line_continuation` expects a number which will be added to the indent level of a continuation line starting with a backslash, and defaults to

`shiftwidth() * 3`. It also accepts a string, which is evaluated at runtime.

`more_in_bracket_block` expects a boolean value; when on, an extra `shiftwidth()` is added inside blocks surrounded with brackets. It defaults to `v:false`.

`searchpair_timeout` expects a number which will be passed to `searchpair()` as

a timeout. Increasing the value might give more accurate results, but also causes the indentation to take more time. It defaults to 100 (milliseconds).

Example of configuration:

```
let g:vim_indent = #{
 \ line_continuation: shiftwidth() * 3,
 \ more_in_bracket_block: v:false,
 \ searchpair_timeout: 100,
 \ }
```

This variable is equivalent to `g:vim_indent.line_continuation``.  
It's supported for backward compatibility.

## YAML

`ft-yaml-indent`

By default, the yaml indent script does not try to detect multiline scalars. If you want to enable this, set the following variable:

```
let g:yaml_indent_multiline_scalar = 1
```

Syntax highlighting [syntax](#) [syntax-highlighting](#) [coloring](#)

Syntax highlighting enables Vim to show parts of the text in another font or color. Those parts can be specific keywords or text matching a pattern. Vim doesn't parse the whole file (to keep it fast), so the highlighting has its limitations. Lexical highlighting might be a better name, but since everybody calls it syntax highlighting we'll stick with that.

Vim supports syntax highlighting on all terminals. But since most ordinary terminals have very limited highlighting possibilities, it works best in the GUI version, gvim.

In the User Manual:

[usr\\_06.txt](#) introduces syntax highlighting.  
[usr\\_44.txt](#) introduces writing a syntax file.

- |                                      |                                   |
|--------------------------------------|-----------------------------------|
| 1. Quick start                       | <a href="#">:syn-qstart</a>       |
| 2. Syntax files                      | <a href="#">:syn-files</a>        |
| 3. Syntax loading procedure          | <a href="#">syntax-loading</a>    |
| 4. Converting to HTML                | <a href="#">2html.vim</a>         |
| 5. Syntax file remarks               | <a href="#">:syn-file-remarks</a> |
| 6. Defining a syntax                 | <a href="#">:syn-define</a>       |
| 7. <a href="#">:syntax</a> arguments | <a href="#">:syn-arguments</a>    |
| 8. Syntax patterns                   | <a href="#">:syn-pattern</a>      |
| 9. Syntax clusters                   | <a href="#">:syn-cluster</a>      |
| 10. Including syntax files           | <a href="#">:syn-include</a>      |
| 11. Synchronizing                    | <a href="#">:syn-sync</a>         |
| 12. Listing syntax items             | <a href="#">:syntax</a>           |
| 13. Colorschemes                     | <a href="#">color-schemes</a>     |
| 14. Highlight command                | <a href="#">:highlight</a>        |
| 15. Linking groups                   | <a href="#">:highlight-link</a>   |
| 16. Cleaning up                      | <a href="#">:syn-clear</a>        |
| 17. Highlighting tags                | <a href="#">tag-highlight</a>     |
| 18. Window-local syntax              | <a href="#">:ownsyntax</a>        |
| 19. Color xterms                     | <a href="#">xterm-color</a>       |
| 20. When syntax is slow              | <a href="#">:syntime</a>          |

{Vi does not have any of these commands}

Syntax highlighting is not available when the [+syntax](#) feature has been disabled at compile time.

```
=====
1. Quick start :syn-qstart
```

[:syn-enable](#) [:syntax-enable](#)

This command switches on syntax highlighting:

```
:syntax enable
```

What this command actually does is to execute the command

```
:source $VIMRUNTIME/syntax/syntax.vim
```

If the VIM environment variable is not set, Vim will try to find the path in another way (see `$VIMRUNTIME`). Usually this works just fine. If it doesn't, try setting the VIM environment variable to the directory where the Vim stuff is located. For example, if your syntax files are in the `"/usr/vim/vim82/syntax"` directory, set `$VIMRUNTIME` to `"/usr/vim/vim82"`. You must do this in the shell, before starting Vim. This command also sources the `menu.vim` script when the GUI is running or will start soon. See `'go-M'` about avoiding that.

```
:syn-on :syntax-on
```

The ``:syntax enable`` command will keep most of your current color settings. This allows using ``:highlight`` commands to set your preferred colors before or after using this command. If you want Vim to overrule your settings with the defaults, use:

```
:syntax on
```

```
:hi-normal :highlight-normal
```

If you are running in the GUI, you can get white text on a black background with:

```
:highlight Normal guibg=Black guifg=White
```

For a color terminal see `:hi-normal-cterm`.

For setting up your own colors syntax highlighting see `syncolor`.

**NOTE:** The syntax files on MS-Windows have lines that end in `<CR><NL>`. The files for Unix end in `<NL>`. This means you should use the right type of file for your system. Although on MS-Windows the right format is automatically selected if the `'fileformats'` option is not empty.

**NOTE:** When using reverse video (`"gvim -fg white -bg black"`), the default value of `'background'` will not be set until the GUI window is opened, which is after reading the `gvimrc`. This will cause the wrong default highlighting to be used. To set the default value of `'background'` before switching on highlighting, include the `":gui"` command in the `gvimrc`:

```
:gui " open window and set default for 'background'
:syntax on " start highlighting, use 'background' to set colors
```

**NOTE:** Using `":gui"` in the `gvimrc` means that `"gvim -f"` won't start in the foreground! Use `":gui -f"` then.

```
g:syntax_on
```

You can toggle the syntax on/off with this command:

```
:if exists("g:syntax_on") | syntax off | else | syntax enable | endif
```

To put this into a mapping, you can use:

```
:map <F7> :if exists("g:syntax_on") <Bar>
 \ syntax off <Bar>
 \ else <Bar>
 \ syntax enable <Bar>
```



```
\ endif <CR>
[using the <> notation, type this literally]
```

Details:

The ":syntax" commands are implemented by sourcing a file. To see exactly how this works, look in the file:

| command        | file                             |
|----------------|----------------------------------|
| :syntax enable | \$VIMRUNTIME/syntax/syntax.vim   |
| :syntax on     | \$VIMRUNTIME/syntax/syntax.vim   |
| :syntax manual | \$VIMRUNTIME/syntax/manual.vim   |
| :syntax off    | \$VIMRUNTIME/syntax/nosyntax.vim |

Also see [syntax-loading](#) .

**NOTE:** If displaying long lines is slow and switching off syntax highlighting makes it fast, consider setting the '[synmaxcol](#)' option to a lower value.

---

## 2. Syntax files

[:syn-files](#)

The syntax and highlighting commands for one language are normally stored in a syntax file. The name convention is: "{name}.vim". Where {name} is the name of the language, or an abbreviation (to fit the name in 8.3 characters, a requirement in case the file is used on a DOS filesystem).

Examples:

|         |          |          |          |
|---------|----------|----------|----------|
| c.vim   | perl.vim | java.vim | html.vim |
| cpp.vim | sh.vim   | csh.vim  |          |

The syntax file can contain any Ex commands, just like a vimrc file. But the idea is that only commands for a specific language are included. When a language is a superset of another language, it may include the other one, for example, the cpp.vim file could include the c.vim file:

```
:so $VIMRUNTIME/syntax/c.vim
```

The .vim files are normally loaded with an autocommand. For example:

```
:au Syntax c runtime! syntax/c.vim
:au Syntax cpp runtime! syntax/cpp.vim
```

These commands are normally in the file \$VIMRUNTIME/syntax/synload.vim.

## MAKING YOUR OWN SYNTAX FILES

[mysyntaxfile](#)

When you create your own syntax files, and you want to have Vim use these automatically with ":syntax enable", do this:

1. Create your user runtime directory. You would normally use the first item of the '[runtimepath](#)' option. Example for Unix:

```
mkdir ~/.vim
```

2. Create a directory in there called "syntax". For Unix:

```
mkdir ~/.vim/syntax
```

3. Write the Vim syntax file. Or download one from the internet. Then write it in your syntax directory. For example, for the "mine" syntax:

```
:w ~/.vim/syntax/mine.vim
```

Now you can start using your syntax file manually:

```
:set syntax=mime
```

You don't have to exit Vim to use this.

If you also want Vim to detect the type of file, see [new-filetype](#) .

If you are setting up a system with many users and you don't want each user to add the same syntax file, you can use another directory from ['runtimepath'](#).

## ADDING TO AN EXISTING SYNTAX FILE

[mysyntaxfile-add](#)

If you are mostly satisfied with an existing syntax file, but would like to add a few items or change the highlighting, follow these steps:

1. Create your user directory from ['runtimepath'](#), see above.
2. Create a directory in there called "after/syntax". For Unix:

```
mkdir ~/.vim/after
mkdir ~/.vim/after/syntax
```
3. Write a Vim script that contains the commands you want to use. For example, to change the colors for the C syntax:

```
highlight cComment ctermfg=Green guifg=Green
```
4. Write that file in the "after/syntax" directory. Use the name of the syntax, with ".vim" added. For our C syntax:

```
:w ~/.vim/after/syntax/c.vim
```

That's it. The next time you edit a C file the Comment color will be different. You don't even have to restart Vim.

If you have multiple files, you can use the filetype as the directory name. All the "\*.vim" files in this directory will be used, for example:

```
~/.vim/after/syntax/c/one.vim
~/.vim/after/syntax/c/two.vim
```

## REPLACING AN EXISTING SYNTAX FILE

[mysyntaxfile-replace](#)

If you don't like a distributed syntax file, or you have downloaded a new version, follow the same steps as for [mysyntaxfile](#) above. Just make sure that you write the syntax file in a directory that is early in ['runtimepath'](#). Vim will only load the first syntax file found, assuming that it sets `b:current_syntax`.

## NAMING CONVENTIONS

[group-name](#) {[group-name](#)} E669 W18

A syntax group name is to be used for syntax items that match the same kind of thing. These are then linked to a highlight group that specifies the color. A syntax group name doesn't specify any color or attributes itself.

The name for a highlight or syntax group must consist of ASCII letters, digits, underscores, dots, or hyphens. As a regexp: "[a-zA-Z0-9\_.-]\*". However, Vim does not give an error when using other characters. The maximum length of a group name is about 200 bytes. [E1249](#)

To be able to allow each user to pick their favorite set of colors, there must be preferred names for highlight groups that are common for many languages. These are the suggested group names (if syntax highlighting works properly you can see the actual color, except for "Ignore"):

|                |                                              |
|----------------|----------------------------------------------|
| *Comment       | any comment                                  |
| *Constant      | any constant                                 |
| String         | a string constant: "this is a string"        |
| Character      | a character constant: 'c', '\n'              |
| Number         | a number constant: 234, 0xff                 |
| Boolean        | a boolean constant: TRUE, false              |
| Float          | a floating point constant: 2.3e10            |
| *Identifier    | any variable name                            |
| Function       | function name (also: methods for classes)    |
| *Statement     | any statement                                |
| Conditional    | if, then, else, endif, switch, etc.          |
| Repeat         | for, do, while, etc.                         |
| Label          | case, default, etc.                          |
| Operator       | "sizeof", "+", "*", etc.                     |
| Keyword        | any other keyword                            |
| Exception      | try, catch, throw                            |
| *PreProc       | generic Preprocessor                         |
| Include        | preprocessor #include                        |
| Define         | preprocessor #define                         |
| Macro          | same as Define                               |
| PreCondit      | preprocessor #if, #else, #endif, etc.        |
| *Type          | int, long, char, etc.                        |
| StorageClass   | static, register, volatile, etc.             |
| Structure      | struct, union, enum, etc.                    |
| Typedef        | A typedef                                    |
| *Special       | any special symbol                           |
| SpecialChar    | special character in a constant              |
| Tag            | you can use <b>CTRL-]</b> on this            |
| Delimiter      | character that needs attention               |
| SpecialComment | special things inside a comment              |
| Debug          | debugging statements                         |
| *Underlined    | text that stands out, HTML links             |
| *Ignore        | left blank, hidden <a href="#">hl-Ignore</a> |
| *Error         | any erroneous construct                      |

|          |                                                                             |
|----------|-----------------------------------------------------------------------------|
| *Todo    | anything that needs extra attention; mostly the keywords TODO FIXME and XXX |
| *Added   | added line in a diff                                                        |
| *Changed | changed line in a diff                                                      |
| *Removed | removed line in a diff                                                      |

The names marked with \* are the preferred groups; the others are minor groups. For the preferred groups, the "syntax.vim" file contains default highlighting. The minor groups are linked to the preferred groups, so they get the same highlighting. You can override these defaults by using ":highlight" commands after sourcing the "syntax.vim" file.

**Note** that highlight group names are not case sensitive. "String" and "string" can be used for the same group.

The following names are reserved and cannot be used as a group name:  
 NONE ALL ALLBUT contains contained

hl-Ignore

When using the Ignore group, you may also consider using the conceal mechanism. See [conceal](#) .

### 3. Syntax loading procedure

syntax-loading

This explains the details that happen when the command ":syntax enable" is issued. When Vim initializes itself, it finds out where the runtime files are located. This is used here as the variable `$VIMRUNTIME` .

":syntax enable" and ":syntax on" do the following:

```
Source $VIMRUNTIME/syntax/syntax.vim
|
+- Clear out any old syntax by sourcing $VIMRUNTIME/syntax/nosyntax.vim
|
+- Source first syntax/synload.vim in 'runtimepath'
|
| +- Setup the colors for syntax highlighting. If a color scheme is
| defined it is loaded again with ":colors {name}". Otherwise
| ":runtime! syntax/syncolor.vim" is used. ":syntax on" overrules
| existing colors, ":syntax enable" only sets groups that weren't
| set yet.
|
| +- Set up syntax autocmds to load the appropriate syntax file when
| the 'syntax' option is set. synload-1
|
| +- Source the user's optional file, from the mysyntaxfile variable.
| This is for backwards compatibility with Vim 5.x only. synload-2
|
+- Do ":filetype on", which does ":runtime! filetype.vim". It loads any
| filetype.vim files found. It should always Source
| $VIMRUNTIME/filetype.vim, which does the following.
|
```

- +-- Install autocmds based on suffix to set the **'filetype'** option  
 | This is where the connection between file name and file type is  
 | made for known file types. `synload-3`
- +-- Source the user's optional file, from the `myfiletypefile`  
 | variable. This is for backwards compatibility with Vim 5.x only.  
 | `synload-4`
- +-- Install one autocommand which sources `scripts.vim` when no file  
 | type was detected yet. `synload-5`
- +-- Source `$VIMRUNTIME/menu.vim`, to setup the Syntax menu. `menu.vim`
- +-- Install a FileType autocommand to set the **'syntax'** option when a file  
 | type has been detected. `synload-6`
- +-- Execute syntax autocommands to start syntax highlighting for each  
 | already loaded buffer.

Upon loading a file, Vim finds the relevant syntax file as follows:

Loading the file triggers the BufReadPost autocommands.

- +-- If there is a match with one of the autocommands from `synload-3`  
 | (known file types) or `synload-4` (user's file types), the **'filetype'**  
 | option is set to the file type.
- +-- The autocommand at `synload-5` is triggered. If the file type was not  
 | found yet, then `scripts.vim` is searched for in **'runtimepath'**. This  
 | should always load `$VIMRUNTIME/scripts.vim`, which does the following.  
 |
  - +-- Source the user's optional file, from the `myscriptsfile`  
 | variable. This is for backwards compatibility with Vim 5.x only.
  - +-- If the file type is still unknown, check the contents of the file,  
 | again with checks like `"getline(1) =~ pattern"` as to whether the  
 | file type can be recognized, and set **'filetype'**.
- +-- When the file type was determined and **'filetype'** was set, this  
 | triggers the FileType autocommand `synload-6` above. It sets  
 | **'syntax'** to the determined file type.
- +-- When the **'syntax'** option was set above, this triggers an autocommand  
 | from `synload-1` (and `synload-2`). This find the main syntax file in  
 | **'runtimepath'**, with this command:  
 | `runtime! syntax/<name>.vim`
- +-- Any other user installed FileType or Syntax autocommands are  
 | triggered. This can be used to change the highlighting for a specific  
 | syntax.

=====  
 4. Conversion to HTML `2html.vim` `convert-to-HTML`

2html is not a syntax file itself, but a script that converts the current window into HTML. Vim opens a new window in which it builds the HTML file.

After you save the resulting file, you can view it with any browser. The colors should be exactly the same as you see them in Vim. With `g:html_line_ids` you can jump to specific lines by adding (for example) #L123 or #123 to the end of the URL in your browser's address bar. And with `g:html_dynamic_folds` enabled, you can show or hide the text that is folded in Vim.

You are not supposed to set the `'filetype'` or `'syntax'` option to "2html"! Source the script to convert the current file:

```
:runtime! syntax/2html.vim
```

Many variables affect the output of 2html.vim; see below. Any of the on/off options listed below can be enabled or disabled by setting them explicitly to the desired value, or restored to their default by removing the variable using `:unlet .`

Remarks:

- Some truly ancient browsers may not show the background colors.
- From most browsers you can also print the file (in color)!
- The latest TOhtml may actually work with older versions of Vim, but some features such as conceal support will not function, and the colors may be incorrect for an old Vim without GUI support compiled in.

Here is an example how to run the script over all .c and .h files from a Unix shell:

```
for f in *.c *.h; do gvim -f +"syn on" +"run! syntax/2html.vim" +"wq" +"q" $f; done
```

To restrict the conversion to a range of lines, use a range with the `:TOhtml` command below, or set `g:html_start_line` and `g:html_end_line` to the first and last line to be converted. Example, using the last set Visual area:

```
:let g:html_start_line = line("<")
:let g:html_end_line = line(">")
:runtime! syntax/2html.vim
```

`:TOhtml`

`:[range]TOhtml` The `":TOhtml"` command is defined in a standard plugin. This command will source `2html.vim` for you. When a range is given, this command sets `g:html_start_line` and `g:html_end_line` to the start and end of the range, respectively. Default range is the entire buffer.

If the current window is part of a `diff`, unless `g:html_diff_one_file` is set, `:TOhtml` will convert all windows which are part of the diff in the current tab and place them side-by-side in a `<table>` element in the generated HTML. With `g:html_line_ids` you can

jump to lines in specific windows with (for example)  
#W1L42 for line 42 in the first diffed window, or  
#W3L87 for line 87 in the third.

Examples:

```
:10,40T0html " convert lines 10-40 to html
:'<,'>T0html " convert current/last visual selection
:T0html " convert entire buffer
```

`g:html_diff_one_file`

Default: 0.

When 0, and using `:T0html` all windows involved in a `diff` in the current tab page are converted to HTML and placed side-by-side in a `<table>` element. When 1, only the current buffer is converted.

Example:

```
let g:html_diff_one_file = 1
```

`g:html_whole_filler`

Default: 0.

When 0, if `g:html_diff_one_file` is 1, a sequence of more than 3 filler lines is displayed as three lines with the middle line mentioning the total number of inserted lines.

When 1, always display all inserted lines as if `g:html_diff_one_file` were not set.

```
:let g:html_whole_filler = 1
```

`T0html-performance` `g:html_no_progress`

Default: 0.

When 0, display a progress bar in the statusline for each major step in the `2html.vim` conversion process.

When 1, do not display the progress bar. This offers a minor speed improvement but you won't have any idea how much longer the conversion might take; for big files it can take a long time!

Example:

```
let g:html_no_progress = 1
```

You can obtain better performance improvements by also instructing Vim to not run interactively, so that too much time is not taken to redraw as the script moves through the buffer, switches windows, and the like:

```
vim -E -s -c "let g:html_no_progress=1" -c "syntax on" -c "set ft=c" -c "runtime syntax/2
```

**Note** that the `-s` flag prevents loading your `.vimrc` and any plugins, so you need to explicitly source/enable anything that will affect the HTML conversion. See `-E` and `-s-ex` for details. It is probably best to create a script to replace all the `-c` commands and use it with the `-u` flag instead of specifying each command separately.

`hl-T0htmlProgress` `T0html-progress-color`

When displayed, the progress bar will show colored boxes along the statusline

as the HTML conversion proceeds. By default, the background color as the current "DiffDelete" highlight group is used. If "DiffDelete" and "StatusLine" have the same background color, TOhtml will automatically adjust the color to differ. If you do not like the automatically selected colors, you can define your own highlight colors for the progress bar. Example:

```
hi TOhtmlProgress guifg=#c0ffee ctermbg=7
```

**g:html\_number\_lines**

Default: Current '**number**' setting.

When 0, buffer text is displayed in the generated HTML without line numbering. When 1, a column of line numbers is added to the generated HTML with the same highlighting as the line number column in Vim ( `hl-LineNr` ).

Force line numbers even if '**number**' is not set:

```
:let g:html_number_lines = 1
```

Force to omit the line numbers:

```
:let g:html_number_lines = 0
```

Go back to the default to use '**number**' by deleting the variable:

```
:unlet g:html_number_lines
```

**g:html\_line\_ids**

Default: 1 if `g:html_number_lines` is set, 0 otherwise.

When 1, adds an HTML id attribute to each line number, or to an empty `<span>` inserted for that purpose if no line numbers are shown. This ID attribute takes the form of L123 for single-buffer HTML pages, or W2L123 for diff-view pages, and is used to jump to a specific line (in a specific window of a diff view). Javascript is inserted to open any closed dynamic folds ( `g:html_dynamic_folds` ) containing the specified line before jumping. The javascript also allows omitting the window ID in the url, and the leading L. For example:

```
page.html#L123 jumps to line 123 in a single-buffer file
page.html#123 does the same
```

```
diff.html#W1L42 jumps to line 42 in the first window in a diff
diff.html#42 does the same
```

**g:html\_use\_css**

Default: 1.

When 1, generate valid HTML 5 markup with CSS styling, supported in all modern browsers and many old browsers.

When 0, generate `<font>` tags and similar outdated markup. This is not recommended but it may work better in really old browsers, email clients, forum posts, and similar situations where basic CSS support is unavailable.

Example:

```
:let g:html_use_css = 0
```

**g:html\_ignore\_conceal**

Default: 0.

When 0, concealed text is removed from the HTML and replaced with a character from `:syn-cchar` or '`listchars`' as appropriate, depending on the current value of '`conceallevel`'.

When 1, include all text from the buffer in the generated HTML, even if it is `conceal` ed.



Either of the following commands will ensure that all text in the buffer is included in the generated HTML (unless it is folded):

```
:let g:html_ignore_conceal = 1
:setl conceallevel=0
```

[g:html\\_ignore\\_folding](#)

Default: 0.

When 0, text in a closed fold is replaced by the text shown for the fold in Vim ( `fold-foldtext` ). See [g:html\\_dynamic\\_folds](#) if you also want to allow the user to expand the fold as in Vim to see the text inside.

When 1, include all text from the buffer in the generated HTML; whether the text is in a fold has no impact at all. [g:html\\_dynamic\\_folds](#) has no effect.

Either of these commands will ensure that all text in the buffer is included in the generated HTML (unless it is concealed):

```
zR
:let g:html_ignore_folding = 1
```

[g:html\\_dynamic\\_folds](#)

Default: 0.

When 0, text in a closed fold is not included at all in the generated HTML.

When 1, generate javascript to open a fold and show the text within, just like in Vim.

Setting this variable to 1 causes 2html.vim to always use CSS for styling, regardless of what [g:html\\_use\\_css](#) is set to.

This variable is ignored when [g:html\\_ignore\\_folding](#) is set.

```
:let g:html_dynamic_folds = 1
```

[g:html\\_no\\_foldcolumn](#)

Default: 0.

When 0, if [g:html\\_dynamic\\_folds](#) is 1, generate a column of text similar to Vim's `foldcolumn` ( `fold-foldcolumn` ) the user can click on to toggle folds open or closed. The minimum width of the generated text column is the current **'foldcolumn'** setting.

When 1, do not generate this column; instead, hovering the mouse cursor over folded text will open the fold as if [g:html\\_hover\\_unfold](#) were set.

```
:let g:html_no_foldcolumn = 1
```

[T0html-uncopyable-text](#) [g:html\\_prevent\\_copy](#)

Default: Empty string.

This option prevents certain regions of the generated HTML from being copied, when you select all text in document rendered in a browser and copy it. Useful for allowing users to copy-paste only the source text even if a fold column or line numbers are shown in the generated content. Specify regions to be affected in this way as follows:

```
f: fold column
n: line numbers (also within fold text)
t: fold text
d: diff filler
```

Example, to make the fold column and line numbers uncopyable:

```
:let g:html_prevent_copy = "fn"
```

The method used to prevent copying in the generated page depends on the value of `g:html_use_input_for_pc`.

`g:html_use_input_for_pc`

Default: "none"

If `g:html_prevent_copy` is non-empty, then:

When "all", read-only `<input>` elements are used in place of normal text for uncopyable regions. In some browsers, especially older browsers, after selecting an entire page and copying the selection, the `<input>` tags are not pasted with the page text. If `g:html_no_invalid` is 0, the `<input>` tags have invalid type; this works in more browsers, but the page will not validate.

**Note:** This method does NOT work in recent versions of Chrome and equivalent browsers; the `<input>` tags get pasted with the text.

When "fallback" (default value), the same `<input>` elements are generated for older browsers, but newer browsers (detected by CSS feature query) hide the `<input>` elements and instead use generated content in an `::before` pseudoelement to display the uncopyable text. This method should work with the largest number of browsers, both old and new.

When "none", the `<input>` elements are not generated at all. Only the generated-content method is used. This means that old browsers, notably Internet Explorer, will either copy the text intended not to be copyable, or the non-copyable text may not appear at all. However, this is the most standards-based method, and there will be much less markup.

`g:html_no_invalid`

Default: 0.

When 0, if `g:html_prevent_copy` is non-empty and `g:html_use_input_for_pc` is not "none", an invalid attribute is intentionally inserted into the `<input>` element for the uncopyable areas. This prevents pasting the `<input>` elements in some applications. Specifically, some versions of Microsoft Word will not paste the `<input>` elements if they contain this invalid attribute. When 1, no invalid markup is inserted, and the generated page should validate. However, `<input>` elements may be pasted into some applications and can be difficult to remove afterward.

`g:html_hover_unfold`

Default: 0.

When 0, the only way to open a fold generated by 2html.vim with `g:html_dynamic_folds` set, is to click on the generated fold column.

When 1, use CSS 2.0 to allow the user to open a fold by moving the mouse cursor over the displayed fold text. This is useful to allow users with disabled javascript to view the folded text.

**Note** that old browsers (notably Internet Explorer 6) will not support this feature. Browser-specific markup for IE6 is included to fall back to the normal CSS1 styling so that the folds show up correctly for this browser, but they will not be openable without a foldcolumn.

```
:let g:html_hover_unfold = 1
```

[g:html\\_id\\_expr](#)

Default: ""

Dynamic folding and jumping to line IDs rely on unique IDs within the document to work. If generated HTML is copied into a larger document, these IDs are no longer guaranteed to be unique. Set `g:html_id_expr` to an expression Vim can evaluate to get a unique string to append to each ID used in a given document, so that the full IDs will be unique even when combined with other content in a larger HTML document. Example, to append `_` and the buffer number to each ID:

```
:let g:html_id_expr = '"_" .. bufnr("%")'
```

To append a string `"_mystring"` to the end of each ID:

```
:let g:html_id_expr = '"_mystring"'
```

**Note:** When converting a diff view to HTML, the expression will only be evaluated for the first window in the diff, and the result used for all the windows.

[TOhtml-wrap-text](#) [g:html\\_pre\\_wrap](#)

Default: Current `'wrap'` setting.

When 0, if `g:html_no_pre` is 0 or unset, the text in the generated HTML does not wrap at the edge of the browser window.

When 1, if `g:html_use_css` is 1, the CSS 2.0 "white-space:pre-wrap" value is used, causing the text to wrap at whitespace at the edge of the browser window.

Explicitly enable text wrapping:

```
:let g:html_pre_wrap = 1
```

Explicitly disable wrapping:

```
:let g:html_pre_wrap = 0
```

Go back to default, determine wrapping from `'wrap'` setting:

```
:unlet g:html_pre_wrap
```

[g:html\\_no\\_pre](#)

Default: 0.

When 0, buffer text in the generated HTML is surrounded by `<pre>...</pre>` tags. Series of whitespace is shown as in Vim without special markup, and tab characters can be included literally (see [g:html\\_expand\\_tabs](#)).

When 1 (not recommended), the `<pre>` tags are omitted, and a plain `<div>` is used instead. Whitespace is replaced by a series of `&nbsp;` character references, and `<br>` is used to end each line. This is another way to allow text in the generated HTML to wrap (see [g:html\\_pre\\_wrap](#)) which also works in old browsers, but may cause noticeable differences between Vim's display and the rendered page generated by `2html.vim`.

```
:let g:html_no_pre = 1
```

[g:html\\_no\\_doc](#)

Default: 0.

When 1 it doesn't generate a full HTML document with a DOCTYPE, `<head>`, `<body>`, etc. If `g:html_use_css` is enabled (the default) you'll have to

define the CSS manually. The `g:html_dynamic_folds` and `g:html_line_ids` settings (off by default) also insert some JavaScript.

`g:html_no_links`

Default: 0.

Don't generate `<a>` tags for text that looks like an URL.

`g:html_no_modeline`

Default: 0.

Don't generate a modeline disabling folding.

`g:html_expand_tabs`

Default: 0 if `'tabstop'` is 8, `'expandtab'` is 0, `'vartabstop'` is not in use, and no fold column or line numbers occur in the generated HTML; 1 otherwise.

When 1, `<Tab>` characters in the buffer text are replaced with an appropriate number of space characters, or `&nbsp;` references if `g:html_no_pre` is 1. When 0, if `g:html_no_pre` is 0 or unset, `<Tab>` characters in the buffer text are included as-is in the generated HTML. This is useful for when you want to allow copy and paste from a browser without losing the actual whitespace in the source document. **Note** that this can easily break text alignment and indentation in the HTML, unless set by default.

Force `2html.vim` to keep `<Tab>` characters:

```
:let g:html_expand_tabs = 0
```

Force tabs to be expanded:

```
:let g:html_expand_tabs = 1
```

`TOhtml-encoding-detect` `TOhtml-encoding`

It is highly recommended to set your desired encoding with `g:html_use_encoding` for any content which will be placed on a web server.

If you do not specify an encoding, `2html.vim` uses the preferred IANA name for the current value of `'fileencoding'` if set, or `'encoding'` if not. `'encoding'` is always used for certain `'buftype'` values. `'fileencoding'` will be set to match the chosen document encoding.

Automatic detection works for the encodings mentioned specifically by name in `encoding-names`, but `TOhtml` will only automatically use those encodings with wide browser support. However, you can override this to support specific encodings that may not be automatically detected by default (see options below). See <http://www.iana.org/assignments/character-sets> for the IANA names.

**Note:** By default all Unicode encodings are converted to UTF-8 with no BOM in the generated HTML, as recommended by W3C:

<http://www.w3.org/International/questions/qa-choosing-encodings>  
<http://www.w3.org/International/questions/qa-byte-order-mark>

`g:html_use_encoding`

Default: none, uses IANA name for current `'fileencoding'` as above.

To overrule all automatic charset detection, set `g:html_use_encoding` to the

name of the charset to be used. It is recommended to set this variable to something widely supported, like UTF-8, for anything you will be hosting on a webserver:

```
:let g:html_use_encoding = "UTF-8"
```

You can also use this option to omit the line that specifies the charset entirely, by setting `g:html_use_encoding` to an empty string (NOT recommended):

```
:let g:html_use_encoding = ""
```

To go back to the automatic mechanism, delete the `g:html_use_encoding` variable:

```
:unlet g:html_use_encoding
```

`g:html_encoding_override`

Default: none, `autoload/tohtml.vim` contains default conversions for encodings mentioned by name at `encoding-names`.

This option allows `2html.vim` to detect the correct `'fileencoding'` when you specify an encoding with `g:html_use_encoding` which is not in the default list of conversions.

This is a dictionary of charset-encoding pairs that will replace existing pairs automatically detected by TOhtml, or supplement with new pairs.

Detect the HTML charset "windows-1252" as the encoding "8bit-cp1252":

```
:let g:html_encoding_override = {'windows-1252': '8bit-cp1252'}
```

`g:html_charset_override`

Default: none, `autoload/tohtml.vim` contains default conversions for encodings mentioned by name at `encoding-names` and which have wide browser support.

This option allows `2html.vim` to detect the HTML charset for any `'fileencoding'` or `'encoding'` which is not detected automatically. You can also use it to override specific existing encoding-charset pairs. For example, TOhtml will by default use UTF-8 for all Unicode/UCS encodings. To use UTF-16 and UTF-32 instead, use:

```
:let g:html_charset_override = {'ucs-4': 'UTF-32', 'utf-16': 'UTF-16'}
```

**Note** that documents encoded in either UTF-32 or UTF-16 have known compatibility problems with some major browsers.

`g:html_font`

Default: "monospace"

You can specify the font or fonts used in the converted document using `g:html_font`. If this option is set to a string, then the value will be surrounded with single quotes. If this option is set to a list then each list item is surrounded by single quotes and the list is joined with commas. Either way, "monospace" is added as the fallback generic family name and the entire result used as the font family (using CSS) or font face (if not using CSS).

Examples:

```
" font-family: 'Consolas', monospace;
:let g:html_font = "Consolas"
```

```
" font-family: 'DejaVu Sans Mono', 'Consolas', monospace;
:let g:html_font = ["DejaVu Sans Mono", "Consolas"]
```

[convert-to-XML](#)   [convert-to-XHTML](#)   [g:html\\_use\\_xhtml](#)

Default: 0.

When 0, generate standard HTML 4.01 (strict when possible).

When 1, generate XHTML 1.0 instead (XML compliant HTML).

```
:let g:html_use_xhtml = 1
```

---

5. Syntax file remarks

[:syn-file-remarks](#)

[b:current\\_syntax-variable](#)

Vim stores the name of the syntax that has been loaded in the "b:current\_syntax" variable. You can use this if you want to load other settings, depending on which syntax is active. Example:

```
:au BufReadPost * if b:current_syntax == "csh"
:au BufReadPost * do-some-things
:au BufReadPost * endif
```

## ABEL

[abel.vim](#)   [ft-abel-syntax](#)

ABEL highlighting provides some user-defined options. To enable them, assign any value to the respective variable. Example:

```
:let abel_obsolete_ok=1
```

To disable them use ":unlet". Example:

```
:unlet abel_obsolete_ok
```

### Variable

abel\_obsolete\_ok

abel\_cpp\_comments\_illegal

### Highlight

obsolete keywords are statements, not errors

do not interpret '/' as inline comment leader

ADA

See [ft-ada-syntax](#)

## ANT

[ant.vim](#)   [ft-ant-syntax](#)

The ant syntax file provides syntax highlighting for javascript and python by default. Syntax highlighting for other script languages can be installed by the function AntSyntaxScript(), which takes the tag name as first argument and the script syntax file name as second argument. Example:

```
:call AntSyntaxScript('perl', 'perl.vim')
```

will install syntax perl highlighting for the following ant code

```
<script language = 'perl'><![CDATA[
everything inside is highlighted as perl
]]></script>
```

See [mysyntaxfile-add](#) for installing script languages permanently.

## APACHE

apache.vim ft-apache-syntax

The apache syntax file provides syntax highlighting for Apache HTTP server version 2.2.3.

## ASSEMBLY

asm.vim asmh8300.vim nasm.vim masm.vim asm68k  
ft-asm-syntax ft-asmh8300-syntax ft-nasm-syntax  
ft-masm-syntax ft-asm68k-syntax fasm.vim

Files matching "\*.i" could be Progress or Assembly. If the automatic detection doesn't work for you, or you don't edit Progress at all, use this in your startup vimrc:

```
:let filetype_i = "asm"
```

Replace "asm" with the type of assembly you use.

There are many types of assembly languages that all use the same file name extensions. Therefore you will have to select the type yourself, or add a line in the assembly file that Vim will recognize. Currently these syntax files are included:

asm	GNU assembly (the default)
asm68k	Motorola 680x0 assembly
asmh8300	Hitachi H-8300 version of GNU assembly
ia64	Intel Itanium 64
fasm	Flat assembly ( <a href="http://flatassembler.net">http://flatassembler.net</a> )
masm	Microsoft assembly (probably works for any 80x86)
nasm	Netwide assembly
tasm	Turbo Assembly (with opcodes 80x86 up to Pentium, and MMX)
pic	PIC assembly (currently for PIC16F84)

The most flexible is to add a line in your assembly file containing:

```
asmsyntax=nasm
```

Replace "nasm" with the name of the real assembly syntax. This line must be one of the first five lines in the file. No non-white text must be immediately before or after this text. **Note** that specifying asmsyntax=foo is equivalent to setting ft=foo in a [modeline](#), and that in case of a conflict between the two settings the one from the modeline will take precedence (in particular, if you have ft=asm in the modeline, you will get the GNU syntax highlighting regardless of what is specified as asmsyntax).

The syntax type can always be overruled for a specific buffer by setting the b:asmsyntax variable:

```
:let b:asmsyntax = "nasm"
```

If b:asmsyntax is not set, either automatically or by hand, then the value of the global variable asmsyntax is used. This can be seen as a default assembly language:

```
:let asmsyntax = "nasm"
```

As a last resort, if nothing is defined, the "asm" syntax is used.

## Netwide assembler (nasm.vim) optional highlighting

To enable a feature:

```
:let {variable}=1|set syntax=nasm
```

To disable a feature:

```
:unlet {variable} |set syntax=nasm
```

Variable	Highlight
nasm_loose_syntax	unofficial parser allowed syntax not as Error (parser dependent; not recommended)
nasm_ctx_outside_macro	contexts outside macro not as Error
nasm_no_warn	potentially risky syntax not as ToDo

ASPPERL and ASPVBS

[ft-asperl-syntax](#)   [ft-aspvbs-syntax](#)

\*.asp and \*.asa files could be either Perl or Visual Basic script. Since it's hard to detect this you can set two global variables to tell Vim what you are using. For Perl script use:

```
:let g:filetype_asa = "asperl"
:let g:filetype_asp = "asperl"
```

For Visual Basic use:

```
:let g:filetype_asa = "aspvbs"
:let g:filetype_asp = "aspvbs"
```

**BAAN**

[baan.vim](#)   [baan-syntax](#)

The baan.vim gives syntax support for BaanC of release BaanIV up to SSA ERP LN for both 3 GL and 4 GL programming. Large number of standard defines/constants are supported.

Some special violation of coding standards will be signalled when one specify in ones `.vimrc` :

```
let baan_code_stds=1
```

### baan-folding

Syntax folding can be enabled at various levels through the variables mentioned below (Set those in your `.vimrc` ). The more complex folding on source blocks and SQL can be CPU intensive.

To allow any folding and enable folding at function level use:

```
let baan_fold=1
```

Folding can be enabled at source block level as if, while, for, ... The indentation preceding the begin/end keywords has to match (spaces are not considered equal to a tab).

```
let baan_fold_block=1
```

Folding can be enabled for embedded SQL blocks as SELECT, SELECTD0, SELECTEMPTY, ... The indentation preceding the begin/end keywords has to match (spaces are not considered equal to a tab).

```
let baan_fold_sql=1
```

**Note:** Block folding can result in many small folds. It is suggested to `:set`



the options `'foldminlines'` and `'foldnestmax'` in `.vimrc` or use `:setlocal` in `.../after/syntax/baan.vim` (see `after-directory`). Eg:

```
set foldminlines=5
set foldnestmax=6
```

**BASIC** `basic.vim` `vb.vim` `ft-basic-syntax` `ft-vb-syntax`

Both Visual Basic and "normal" BASIC use the extension ".bas". To detect which one should be used, Vim checks for the string "VB\_Name" in the first five lines of the file. If it is not found, filetype will be "basic", otherwise "vb". Files with the ".frm" extension will always be seen as Visual Basic.

If the automatic detection doesn't work for you or you only edit, for example, FreeBASIC files, use this in your startup vimrc:

```
:let filetype_bas = "freebasic"
```

**C** `c.vim` `ft-c-syntax`

A few things in C highlighting are optional. To enable them assign any value (including zero) to the respective variable. Example:

```
:let c_comment_strings = 1
:let c_no_bracket_error = 0
```

To disable them use `:unlet``. Example:

```
:unlet c_comment_strings
```

Setting the value to zero doesn't work!

An alternative is to switch to the C++ highlighting:

```
:set filetype=cpp
```

Variable	Highlight
<code>c_gnu</code>	GNU gcc specific items
<code>c_comment_strings</code>	strings and numbers inside a comment
<code>c_space_errors</code>	trailing white space and spaces before a <code>&lt;Tab&gt;</code>
<code>c_no_trail_space_error</code>	... but no trailing spaces
<code>c_no_tab_space_error</code>	... but no spaces before a <code>&lt;Tab&gt;</code>
<code>c_no_bracket_error</code>	don't highlight <code>{}</code> ; inside <code>[]</code> as errors
<code>c_no_curly_error</code>	don't highlight <code>{}</code> ; inside <code>[]</code> and <code>()</code> as errors; ...except <code>{ and }</code> in first column Default is to highlight them, otherwise you can't spot a missing <code>)</code> .
<code>c_curly_error</code>	highlight a missing <code>}</code> by finding all pairs; this forces syncing from the start of the file, can be slow
<code>c_no_ansi</code>	don't do standard ANSI types and constants
<code>c_ansi_typedefs</code>	... but do standard ANSI types
<code>c_ansi_constants</code>	... but do standard ANSI constants
<code>c_no_utf</code>	don't highlight <code>\u</code> and <code>\U</code> in strings
<code>c_syntax_for_h</code>	for <code>*.h</code> files use C syntax instead of C++ and use <code>objc</code> syntax instead of <code>objcpp</code>
<code>c_no_if0</code>	don't highlight <code>"#if 0"</code> blocks as comments
<code>c_no_cformat</code>	don't highlight <code>%</code> -formats in strings
<code>c_no_c99</code>	don't highlight C99 standard items

```

c_no_c11 don't highlight C11 standard items
c_no_bsd don't highlight BSD specific types
c_functions highlight function calls and definitions
c_function_pointers highlight function pointers definitions

```

When **'foldmethod'** is set to "syntax" then /\* \*/ comments and { } blocks will become a fold. If you don't want comments to become a fold use:

```

:let c_no_comment_fold = 1
"#{if 0" blocks are also folded, unless:
:let c_no_if0_fold = 1

```

If you notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the "c\_minlines" internal variable to a larger number:

```

:let c_minlines = 100

```

This will make the syntax synchronization start 100 lines before the first displayed line. The default value is 50 (15 when c\_no\_if0 is set). The disadvantage of using a larger number is that redrawing can become slow.

When using the "#if 0" / "#endif" comment highlighting, notice that this only works when the "#if 0" is within "c\_minlines" from the top of the window. If you have a long "#if 0" construct it will not be highlighted correctly.

To match extra items in comments, use the cCommentGroup cluster.

Example:

```

:au Syntax c call MyCadd()
:function MyCadd()
: syn keyword cMyItem contained Ni
: syn cluster cCommentGroup add=cMyItem
: hi link cMyItem Title
:endfun

```

ANSI constants will be highlighted with the "cConstant" group. This includes "NULL", "SIG\_IGN" and others. But not "TRUE", for example, because this is not in the ANSI standard. If you find this confusing, remove the cConstant highlighting:

```

:hi link cConstant NONE

```

If you see '{' and '}' highlighted as an error where they are OK, reset the highlighting for cErrInParen and cErrInBracket.

If you want to use folding in your C files, you can add these lines in a file in the "after" directory in **'runtimepath'**. For Unix this would be ~/.vim/after/syntax/c.vim.

```

syn sync fromstart
set foldmethod=syntax

```

## CH

ch.vim ft-ch-syntax

C/C++ interpreter. Ch has similar syntax highlighting to C and builds upon the C syntax file. See **c.vim** for all the settings that are available for C.

By setting a variable you can tell Vim to use Ch syntax for \*.h files, instead of C or C++:

```
:let ch_syntax_for_h = 1
```

## CHILL

[chill.vim](#) [ft-chill-syntax](#)

Chill syntax highlighting is similar to C. See [c.vim](#) for all the settings that are available. Additionally there is:

```
chill_space_errors like c_space_errors
chill_comment_string like c_comment_strings
chill_minlines like c_minlines
```

## CHANGELOG

[changelog.vim](#) [ft-changelog-syntax](#)

ChangeLog supports highlighting spaces at the start of a line.

If you do not like this, add following line to your `.vimrc`:

```
let g:changelog_spacing_errors = 0
```

This works the next time you edit a changelog file. You can also use `"b:changelog_spacing_errors"` to set this per buffer (before loading the syntax file).

You can change the highlighting used, e.g., to flag the spaces as an error:

```
:hi link ChangelogError Error
```

Or to avoid the highlighting:

```
:hi link ChangelogError NONE
```

This works immediately.

## CLOJURE

[ft-clojure-syntax](#)

[g:clojure\\_syntax\\_keywords](#)

Syntax highlighting of public vars in "clojure.core" is provided by default, but additional symbols can be highlighted by adding them to the `g:clojure_syntax_keywords` variable. The value should be a [Dictionary](#) of syntax group names, each containing a [List](#) of identifiers.

```
let g:clojure_syntax_keywords = {
 \ 'clojureMacro': ["defproject", "defcustom"],
 \ 'clojureFunc': ["string/join", "string/replace"]
 \ }
```

Refer to the Clojure syntax script for valid syntax group names.

There is also `b:clojure_syntax_keywords` which is a buffer-local variant of this variable intended for use by plugin authors to highlight symbols dynamically.

By setting the `b:clojure_syntax_without_core_keywords` variable, vars from "clojure.core" will not be highlighted by default. This is useful for namespaces that have set ``(:refer-clojure :only [])``

`g:clojure_fold`

Setting `g:clojure_fold` to ``1`` will enable the folding of Clojure code. Any list, vector or map that extends over more than one line can be folded using the standard Vim `fold-commands` .

`g:clojure_discard_macro`

Set this variable to ``1`` to enable basic highlighting of Clojure's "discard reader macro".

```
#_(defn foo [x]
 (println x))
```

**Note** that this option will not correctly highlight stacked discard macros (e.g. ``#_#_``).

## COBOL

`cobol.vim` `ft-cobol-syntax`

COBOL highlighting has different needs for legacy code than it does for fresh development. This is due to differences in what is being done (maintenance versus development) and other factors. To enable legacy code highlighting, add this line to your `.vimrc`:

```
:let cobol_legacy_code = 1
```

To disable it again, use this:

```
:unlet cobol_legacy_code
```

## COLD FUSION

`coldfusion.vim` `ft-coldfusion-syntax`

The ColdFusion has its own version of HTML comments. To turn on ColdFusion comment highlighting, add the following line to your startup file:

```
:let html_wrong_comments = 1
```

The ColdFusion syntax file is based on the HTML syntax file.

## CPP

`cpp.vim` `ft-cpp-syntax`

Most things are the same as `ft-c-syntax` .

Variable	Highlight
<code>cpp_no_cpp11</code>	don't highlight C++11 standard items
<code>cpp_no_cpp14</code>	don't highlight C++14 standard items
<code>cpp_no_cpp17</code>	don't highlight C++17 standard items
<code>cpp_no_cpp20</code>	don't highlight C++20 standard items

## CSH

`csh.vim` `ft-csh-syntax`

This covers the shell named "csh". **Note** that on some systems `tcsh` is actually

used.

Detecting whether a file is csh or tcsh is notoriously hard. Some systems symlink /bin/csh to /bin/tcsh, making it almost impossible to distinguish between csh and tcsh. In case VIM guesses wrong you can set the "filetype\_csh" variable. For using csh: `g:filetype_csh`

```
:let g:filetype_csh = "csh"
```

For using tcsh:

```
:let g:filetype_csh = "tcsh"
```

Any script with a tcsh extension or a standard tcsh filename (.tcshrc, tcsh.tcshrc, tcsh.login) will have filetype tcsh. All other tcsh/csh scripts will be classified as tcsh, UNLESS the "filetype\_csh" variable exists. If the "filetype\_csh" variable exists, the filetype will be set to the value of the variable.

## CYNLIB

`cynlib.vim` `ft-cynlib-syntax`

Cynlib files are C++ files that use the Cynlib class library to enable hardware modelling and simulation using C++. Typically Cynlib files have a .cc or a .cpp extension, which makes it very difficult to distinguish them from a normal C++ file. Thus, to enable Cynlib highlighting for .cc files, add this line to your .vimrc file:

```
:let cynlib_cyntax_for_cc=1
```

Similarly for cpp files (this extension is only usually used in Windows)

```
:let cynlib_cyntax_for_cpp=1
```

To disable these again, use this:

```
:unlet cynlib_cyntax_for_cc
:unlet cynlib_cyntax_for_cpp
```

## CWEB

`cweb.vim` `ft-cweb-syntax`

Files matching "\*.w" could be Progress or cweb. If the automatic detection doesn't work for you, or you don't edit Progress at all, use this in your startup vimrc:

```
:let filetype_w = "cweb"
```

## DART

`dart.vim` `ft-dart-syntax`

Dart is an object-oriented, typed, class defined, garbage collected language used for developing mobile, desktop, web, and back-end applications. Dart uses a C-like syntax derived from C, Java, and JavaScript, with features adopted from Smalltalk, Python, Ruby, and others.

More information about the language and its development environment at the official Dart language website at <https://dart.dev>

dart.vim syntax detects and highlights Dart statements, reserved words, type declarations, storage classes, conditionals, loops, interpolated values, and comments. There is no support idioms from Flutter or any other Dart framework.

Changes, fixes? Submit an issue or pull request via:

<https://github.com/pr3d4t0r/dart-vim-syntax/>

## DESKTOP

[desktop.vim](#) [ft-desktop-syntax](#)

Primary goal of this syntax file is to highlight .desktop and .directory files according to freedesktop.org standard:

<https://specifications.freedesktop.org/desktop-entry-spec/latest/>

To highlight nonstandard extensions that does not begin with X-, set

```
let g:desktop_enable_nonstd = 1
```

Note that this may cause wrong highlight.

To highlight KDE-reserved features, set

```
let g:desktop_enable_kde = 1
```

g:desktop\_enable\_kde follows g:desktop\_enable\_nonstd if not supplied

## DIFF

[diff.vim](#)

The diff highlighting normally finds translated headers. This can be slow if there are very long lines in the file. To disable translations:

```
:let diff_translations = 0
```

Also see [diff-slow](#) .

## DIRCOLORS

[dircolors.vim](#) [ft-dircolors-syntax](#)

The dircolors utility highlighting definition has one option. It exists to provide compatibility with the Slackware GNU/Linux distributions version of the command. It adds a few keywords that are generally ignored by most versions. On Slackware systems, however, the utility accepts the keywords and uses them for processing. To enable the Slackware keywords add the following line to your startup file:

```
let dircolors_is_slackware = 1
```

## DOCBOOK

[docbk.vim](#) [ft-docbk-syntax](#) [docbook](#)

### DOCBOOK XML

[docbkxml.vim](#) [ft-docbkxml-syntax](#)

### DOCBOOK SGML

[docbksgml.vim](#) [ft-docbksgml-syntax](#)

There are two types of DocBook files: SGML and XML. To specify what type you are using the "b:docbk\_type" variable should be set. Vim does this for you automatically if it can recognize the type. When Vim can't guess it the type

defaults to XML.

You can set the type manually:

```
:let docbk_type = "sgml"
```

or:

```
:let docbk_type = "xml"
```

You need to do this before loading the syntax file, which is complicated.

Simpler is setting the filetype to "docbkxml" or "docbksgml":

```
:set filetype=docbksgml
```

or:

```
:set filetype=docbkxml
```

You can specify the DocBook version:

```
:let docbk_ver = 3
```

When not set 4 is used.

## DOSBATCH

[dosbatch.vim](#) [ft-dosbatch-syntax](#)

Select the set of Windows Command interpreter extensions that should be supported with the variable `dosbatch_cmdextversion`. For versions of Windows NT (before Windows 2000) this should have the value of 1. For Windows 2000 and later it should be 2.

Select the version you want with the following line:

```
:let dosbatch_cmdextversion = 1
```

If this variable is not defined it defaults to a value of 2 to support Windows 2000 and later.

The original MS-DOS supports an idiom of using a double colon (::) as an alternative way to enter a comment line. This idiom can be used with the current Windows Command Interpreter, but it can lead to problems when used inside ( ... ) command blocks. You can find a discussion about this on Stack Overflow -

<https://stackoverflow.com/questions/12407800/which-comment-style-should-i-use-in-batch-files>

To allow the use of the :: idiom for comments in command blocks with the Windows Command Interpreter set the `dosbatch_colons_comment` variable to anything:

```
:let dosbatch_colons_comment = 1
```

If this variable is set then a :: comment that is the last line in a command block will be highlighted as an error.

There is an option that covers whether \*.btm files should be detected as type "dosbatch" (MS-DOS batch files) or type "btm" (4DOS batch files). The latter is used by default. You may select the former with the following line:

```
:let g:dosbatch_syntax_for_btm = 1
```

If this variable is undefined or zero, btm syntax is selected.

## DOXYGEN

[doxygen.vim](#) [doxygen-syntax](#)

Doxygen generates code documentation using a special documentation format (similar to Javadoc). This syntax script adds doxygen highlighting to c, cpp, idl and php files, and should also work with java.

There are a few of ways to turn on doxygen formatting. It can be done explicitly or in a modeline by appending '.doxygen' to the syntax of the file. Example:

```
:set syntax=c.doxygen
or
// vim:syntax=c.doxygen
```

It can also be done automatically for C, C++, C#, IDL and PHP files by setting the global or buffer-local variable `load_doxygen_syntax`. This is done by adding the following to your `.vimrc`.

```
:let g:load_doxygen_syntax=1
```

There are a couple of variables that have an effect on syntax highlighting, and are to do with non-standard highlighting options.

Variable	Default	Effect
<code>g:doxygen_enhanced_color</code> <code>g:doxygen_enhanced_colour</code>	0	Use non-standard highlighting for doxygen comments.
<code>doxygen_my_rendering</code>	0	Disable rendering of HTML bold, italic and <code>html_my_rendering</code> underline.
<code>doxygen_javadoc_autobrief</code>	1	Set to 0 to disable javadoc autobrief colour highlighting.
<code>doxygen_end_punctuation</code>	'[.]'	Set to regexp match for the ending punctuation of brief

There are also some highlight groups worth mentioning as they can be useful in configuration.

Highlight	Effect
<code>doxygenErrorComment</code>	The colour of an end-comment when missing punctuation in a code, verbatim or dot section
<code>doxygenLinkError</code>	The colour of an end-comment when missing the <code>\endlink</code> from a <code>\link</code> section.

## DTD

[dtd.vim](#) [ft-dtd-syntax](#)

The DTD syntax highlighting is case sensitive by default. To disable case-sensitive highlighting, add the following line to your startup file:

```
:let dtd_ignore_case=1
```

The DTD syntax file will highlight unknown tags as errors. If



this is annoying, it can be turned off by setting:

```
:let dtd_no_tag_errors=1
```

before sourcing the dtd.vim syntax file.

Parameter entity names are highlighted in the definition using the 'Type' highlighting group and 'Comment' for punctuation and '%'. Parameter entity instances are highlighted using the 'Constant' highlighting group and the 'Type' highlighting group for the delimiters % and ;. This can be turned off by setting:

```
:let dtd_no_param_entities=1
```

The DTD syntax file is also included by xml.vim to highlight included dtd's.

## EIFFEL

eiffel.vim ft-eiffel-syntax

While Eiffel is not case-sensitive, its style guidelines are, and the syntax highlighting file encourages their use. This also allows to highlight class names differently. If you want to disable case-sensitive highlighting, add the following line to your startup file:

```
:let eiffel_ignore_case=1
```

Case still matters for class names and TODO marks in comments.

Conversely, for even stricter checks, add one of the following lines:

```
:let eiffel_strict=1
:let eiffel_pedantic=1
```

Setting eiffel\_strict will only catch improper capitalization for the five predefined words "Current", "Void", "Result", "Precursor", and "NONE", to warn against their accidental use as feature or class names.

Setting eiffel\_pedantic will enforce adherence to the Eiffel style guidelines fairly rigorously (like arbitrary mixes of upper- and lowercase letters as well as outdated ways to capitalize keywords).

If you want to use the lower-case version of "Current", "Void", "Result", and "Precursor", you can use

```
:let eiffel_lower_case_predef=1
```

instead of completely turning case-sensitive highlighting off.

Support for ISE's proposed new creation syntax that is already experimentally handled by some compilers can be enabled by:

```
:let eiffel_ise=1
```

Finally, some vendors support hexadecimal constants. To handle them, add

```
:let eiffel_hex_constants=1
```

to your startup file.

**EUPHORIA** [euphoria3.vim](#) [euphoria4.vim](#) [ft-euphoria-syntax](#)

Two syntax highlighting files exist for Euphoria. One for Euphoria version 3.1.1, which is the default syntax highlighting file, and one for Euphoria version 4.0.5 or later.

Euphoria version 3.1.1 (<http://www.rapideuphoria.com/>) is still necessary for developing applications for the DOS platform, which Euphoria version 4 (<http://www.openeuphoria.org/>) does not support.

The following file extensions are auto-detected as Euphoria file type:

```
*.e, *.eu, *.ew, *.ex, *.exu, *.exw
*.E, *.EU, *.EW, *.EX, *.EXU, *.EXW
```

To select syntax highlighting file for Euphoria, as well as for auto-detecting the \*.e and \*.E file extensions as Euphoria file type, add the following line to your startup file:

```
:let g:filetype_euphoria = "euphoria3"
```

or

```
:let g:filetype_euphoria = "euphoria4"
```

Elixir and Euphoria share the \*.ex file extension. If the filetype is specifically set as Euphoria with the g:filetype\_euphoria variable, or the file is determined to be Euphoria based on keywords in the file, then the filetype will be set as Euphoria. Otherwise, the filetype will default to Elixir.

**ERLANG** [erlang.vim](#) [ft-erlang-syntax](#)

Erlang is a functional programming language developed by Ericsson. Files with the following extensions are recognized as Erlang files: erl, hrl, yaws.

The BIFs (built-in functions) are highlighted by default. To disable this, put the following line in your vimrc:

```
:let g:erlang_highlight_bifs = 0
```

To enable highlighting some special atoms, put this in your vimrc:

```
:let g:erlang_highlight_special_atoms = 1
```

**ELIXIR** [elixir.vim](#) [ft-elixir-syntax](#)

Elixir is a dynamic, functional language for building scalable and maintainable applications.

The following file extensions are auto-detected as Elixir file types:

```
*.ex, *.exs, *.eex, *.leex, *.lock
```

Elixir and Euphoria share the \*.ex file extension. If the filetype is specifically set as Euphoria with the `g:filetype_euphoria` variable, or the file is determined to be Euphoria based on keywords in the file, then the filetype will be set as Euphoria. Otherwise, the filetype will default to Elixir.

## FLEXWIKI

`flexwiki.vim` `ft-flexwiki-syntax`

FlexWiki is an ASP.NET-based wiki package available at <http://www.flexwiki.com>

**NOTE:** This site currently doesn't work, on Wikipedia is mentioned that development stopped in 2009.

Syntax highlighting is available for the most common elements of FlexWiki syntax. The associated `ftplugin` script sets some buffer-local options to make editing FlexWiki pages more convenient. FlexWiki considers a newline as the start of a new paragraph, so the `ftplugin` sets `'tw'=0` (unlimited line length), `'wrap'` (wrap long lines instead of using horizontal scrolling), `'linebreak'` (to wrap at a character in `'breakat'` instead of at the last char on screen), and so on. It also includes some keymaps that are disabled by default.

If you want to enable the keymaps that make "j" and "k" and the cursor keys move up and down by display lines, add this to your `.vimrc`:

```
:let flexwiki_maps = 1
```

## FORM

`form.vim` `ft-form-syntax`

The coloring scheme for syntax elements in the FORM file uses the default modes Conditional, Number, Statement, Comment, PreProc, Type, and String, following the language specifications in 'Symbolic Manipulation with FORM' by J.A.M. Vermaseren, CAN, Netherlands, 1991.

If you want to include your own changes to the default colors, you have to redefine the following syntax groups:

- formConditional
- formNumber
- formStatement
- formHeaderStatement
- formComment
- formPreProc
- formDirective
- formType
- formString

**Note** that the `form.vim` syntax file implements FORM preprocessor commands and

directives per default in the same syntax group.

A predefined enhanced color mode for FORM is available to distinguish between header statements and statements in the body of a FORM program. To activate this mode define the following variable in your vimrc file

```
:let form_enhanced_color=1
```

The enhanced mode also takes advantage of additional color features for a dark gvim display. Here, statements are colored LightYellow instead of Yellow, and conditionals are LightBlue for better distinction.

Both Visual Basic and FORM use the extension ".frm". To detect which one should be used, Vim checks for the string "VB\_Name" in the first five lines of the file. If it is found, filetype will be "vb", otherwise "form".

If the automatic detection doesn't work for you or you only edit, for example, FORM files, use this in your startup vimrc:

```
:let filetype_frm = "form"
```

## FORTH

[forth.vim](#) [ft-forth-syntax](#)

Files matching "\*.f" could be Fortran or Forth and those matching "\*.fs" could be F# or Forth. If the automatic detection doesn't work for you, or you don't edit F# or Fortran at all, use this in your startup vimrc:

```
:let filetype_f = "forth"
:let filetype_fs = "forth"
```

## FORTRAN

[fortran.vim](#) [ft-fortran-syntax](#)

### Default highlighting and dialect

Vim highlights according to Fortran 2023 (the most recent standard). This choice should be appropriate for most users most of the time because Fortran 2023 is almost a superset of previous versions (Fortran 2018, 2008, 2003, 95, 90, 77, and 66). A few legacy constructs deleted or declared obsolescent, respectively, in recent Fortran standards are highlighted as errors and todo items.

The syntax script no longer supports Fortran dialects. The variable `fortran_dialect` is now silently ignored. Since computers are much faster now, the variable `fortran_more_precise` is no longer needed and is silently ignored.

### Fortran source code form

Fortran code can be in either fixed or free source form. **Note** that the syntax highlighting will not be correct if the form is incorrectly set.

When you create a new Fortran file, the syntax script assumes fixed source form. If you always use free source form, then

```
:let fortran_free_source=1
```

in your .vimrc prior to the `:syntax on` command. If you always use fixed source form, then

```
:let fortran_fixed_source=1
```

in your `.vimrc` prior to the `:syntax on` command.

If the form of the source code depends, in a non-standard way, upon the file extension, then it is most convenient to set `fortran_free_source` in a `ftplugin` file. For more information on `ftplugin` files, see [ftplugin](#). Note that this will work only if the "filetype plugin indent on" command precedes the "syntax on" command in your `.vimrc` file.

When you edit an existing Fortran file, the syntax script will assume free source form if the `fortran_free_source` variable has been set, and assumes fixed source form if the `fortran_fixed_source` variable has been set. Suppose neither of these variables have been set. In that case, the syntax script attempts to determine which source form has been used by examining the file extension using conventions common to the `ifort`, `gfortran`, Cray, NAG, and PathScale compilers (`.f`, `.for`, `.f77` for fixed-source, `.f90`, `.f95`, `.f03`, `.f08` for free-source). No default is used for the `.fpp` and `.ftn` file extensions because different compilers treat them differently. If none of this works, then the script examines the first five columns of the first 500 lines of your file. If no signs of free source form are detected, then the file is assumed to be in fixed source form. The algorithm should work in the vast majority of cases. In some cases, such as a file that begins with 500 or more full-line comments, the script may incorrectly decide that the code is in fixed form. If that happens, just add a non-comment statement beginning anywhere in the first five columns of the first twenty-five lines, save (`:w`), and then reload (`:e!`) the file.

### Vendor extensions

Fixed-form Fortran requires a maximum line length of 72 characters but the script allows a maximum line length of 80 characters as do all compilers created in the last three decades. An even longer line length of 132 characters is allowed if you set the variable `fortran_extended_line_length` with a command such as

```
:let fortran_extended_line_length=1
```

placed prior to the `:syntax on` command.

If you want additional highlighting of the CUDA Fortran extensions, you should set the variable `fortran_CUDA` with a command such as

```
:let fortran_CUDA=1
```

placed prior to the `:syntax on` command.

To activate recognition of some common, non-standard, vendor-supplied intrinsics, you should set the variable `fortran_vendor_intrinsics` with a command such as

```
:let fortran_vendor_intrinsics=1
```

placed prior to the `:syntax on` command.

### Tabs in Fortran files

Tabs are not recognized by the Fortran standards. Tabs are not a good idea in fixed format Fortran source code which requires fixed column boundaries. Therefore, tabs are marked as errors. Nevertheless, some programmers like using tabs. If your Fortran files contain tabs, then you should set the variable `fortran_have_tabs` in your `.vimrc` with a command such as

```
:let fortran_have_tabs=1
```

placed prior to the `:syntax on` command. Unfortunately, the use of tabs will

mean that the syntax file will not be able to detect incorrect margins.

### Syntax folding of Fortran files

Vim will fold your file using `foldmethod=syntax`, if you set the variable `fortran_fold` in your `.vimrc` with a command such as

```
:let fortran_fold=1
```

to instruct the syntax script to define fold regions for program units, that is main programs starting with a program statement, subroutines, function subprograms, modules, submodules, blocks of comment lines, and block data units. Block, interface, associate, critical, type definition, and change team constructs will also be folded. If you also set the variable `fortran_fold_conditionals` with a command such as

```
:let fortran_fold_conditionals=1
```

then fold regions will also be defined for do loops, if blocks, select case, select type, and select rank constructs. **Note** that defining fold regions can be slow for large files.

The `syntax/fortran.vim` script contains embedded comments that tell you how to comment and/or uncomment some lines to (a) activate recognition of some non-standard, vendor-supplied intrinsics and (b) to prevent features deleted or declared obsolescent in the 2008 standard from being highlighted as todo items.

### Limitations

Parenthesis checking does not catch too few closing parentheses. Hollerith strings are not recognized. Some keywords may be highlighted incorrectly because Fortran90 has no reserved words.

For further information related to Fortran, see [ft-fortran-indent](#) and [ft-fortran-plugin](#).

## FREEBASIC

[freebasic.vim](#) [ft-freebasic-syntax](#)

FreeBASIC files will be highlighted differently for each of the four available dialects, "fb", "qb", "fblite" and "deprecated". See [ft-freebasic-plugin](#) for how to select the correct dialect.

Highlighting is further configurable via the following variables.

Variable	Highlight
<a href="#">freebasic_no_comment_fold</a>	disable multiline comment folding
<a href="#">freebasic_operators</a>	non-alpha operators
<a href="#">freebasic_space_errors</a>	trailing white space and spaces before a <Tab>
<a href="#">freebasic_type_suffixes</a>	QuickBASIC style type suffixes

## FVWM CONFIGURATION FILES

[fvwm.vim](#) [ft-fvwm-syntax](#)

In order for Vim to recognize Fvwm configuration files that do not match the patterns `fvwmrc` or `fvwm2rc`, you must put additional patterns appropriate to your system in your `myfiletypes.vim` file. For these patterns, you must set the variable `"b:fvwm_version"` to the major version number of Fvwm, and the `'filetype'` option to `fvwm`.

For example, to make Vim identify all files in /etc/X11/fvwm2/ as Fvwm2 configuration files, add the following:

```
:au! BufNewFile,BufRead /etc/X11/fvwm2/* let b:fvwm_version = 2 |
\ set filetype=fvwm
```

**GSP** [gsp.vim](#) [ft-gsp-syntax](#)

The default coloring style for GSP pages is defined by [html.vim](#), and the coloring for java code (within java tags or inline between backticks) is defined by [java.vim](#). The following HTML groups defined in [html.vim](#) are redefined to incorporate and highlight inline java code:

```
htmlString
htmlValue
htmlEndTag
htmlTag
htmlTagN
```

Highlighting should look fine most of the places where you'd see inline java code, but in some special cases it may not. To add another HTML group where you will have inline java code where it does not highlight correctly, just copy the line you want from [html.vim](#) and add `gspJava` to the `contains` clause.

The backticks for inline java are highlighted according to the `htmlError` group to make them easier to see.

**GROFF** [groff.vim](#) [ft-groff-syntax](#)

The groff syntax file is a wrapper for [nroff.vim](#), see the notes under that heading for examples of use and configuration. The purpose of this wrapper is to set up groff syntax extensions by setting the filetype from a [modeline](#) or in a personal filetype definitions file (see [filetype.txt](#)).

**HASKELL** [haskell.vim](#) [lhaskell.vim](#) [ft-haskell-syntax](#)

The Haskell syntax files support plain Haskell code as well as literate Haskell code, the latter in both Bird style and TeX style. The Haskell syntax highlighting will also highlight C preprocessor directives.

If you want to highlight delimiter characters (useful if you have a light-coloured background), add to your `.vimrc`:

```
:let hs_highlight_delimiters = 1
```

To treat True and False as keywords as opposed to ordinary identifiers, add:

```
:let hs_highlight_boolean = 1
```

To also treat the names of primitive types as keywords:

```
:let hs_highlight_types = 1
```

And to treat the names of even more relatively common types as keywords:

```
:let hs_highlight_more_types = 1
```

If you want to highlight the names of debugging functions, put in your `.vimrc`:

```
:let hs_highlight_debug = 1
```

The Haskell syntax highlighting also highlights C preprocessor directives, and flags lines that start with `#` but are not valid directives as erroneous. This interferes with Haskell's syntax for operators, as they may start with `#`. If you want to highlight those as operators as opposed to errors, put in your `.vimrc`:

```
:let hs_allow_hash_operator = 1
```

The syntax highlighting for literate Haskell code will try to automatically guess whether your literate Haskell code contains TeX markup or not, and correspondingly highlight TeX constructs or nothing at all. You can override this globally by putting in your `.vimrc`

```
:let lhs_markup = none
```

for no highlighting at all, or

```
:let lhs_markup = tex
```

to force the highlighting to always try to highlight TeX markup. For more flexibility, you may also use buffer local versions of this variable, so e.g.

```
:let b:lhs_markup = tex
```

will force TeX highlighting for a particular buffer. It has to be set before turning syntax highlighting on for the buffer or loading a file.

## HTML

`html.vim` `ft-html-syntax`

The coloring scheme for tags in the HTML file works as follows.

The `<>` of opening tags are colored differently than the `</>` of a closing tag. This is on purpose! For opening tags the 'Function' color is used, while for closing tags the 'Identifier' color is used (See `syntax.vim` to check how those are defined for you)

Known tag names are colored the same way as statements in C. Unknown tag names are colored with the same color as the `<>` or `</>` respectively which makes it easy to spot errors

**Note** that the same is true for argument (or attribute) names. Known attribute names are colored differently than unknown ones.

Some HTML tags are used to change the rendering of text. The following tags are recognized by the `html.vim` syntax coloring file and change the way normal text is shown: `<B>` `<I>` `<U>` `<EM>` `<STRONG>` (`<EM>` is used as an alias for `<I>`, while `<STRONG>` as an alias for `<B>`), `<H1>` - `<H6>`, `<HEAD>`, `<TITLE>` and `<A>`, but only if used as a link (that is, it must include a href as in `<A href="somefile.html">`).

If you want to change how such text is rendered, you must redefine the following syntax groups:



- htmlBold
- htmlBoldUnderline
- htmlBoldUnderlineItalic
- htmlUnderline
- htmlUnderlineItalic
- htmlItalic
- htmlTitle for titles
- htmlH1 - htmlH6 for headings

To make this redefinition work you must redefine them all with the exception of the last two (htmlTitle and htmlH[1-6], which are optional) and define the following variable in your vimrc (this is due to the order in which the files are read during initialization)

```
:let html_my_rendering=1
```

If you'd like to see an example download mysyntax.vim at <http://www.fleiner.com/vim/download.html>

You can also disable this rendering by adding the following line to your vimrc file:

```
:let html_no_rendering=1
```

HTML comments are rather special (see an HTML reference document for the details), and the syntax coloring scheme will highlight all errors. However, if you prefer to use the wrong style (starts with <!-- and ends with -->) you can define

```
:let html_wrong_comments=1
```

JavaScript and Visual Basic embedded inside HTML documents are highlighted as 'Special' with statements, comments, strings and so on colored as in standard programming languages. **Note** that only JavaScript and Visual Basic are currently supported, no other scripting language has been added yet.

Embedded and inlined cascading style sheets (CSS) are highlighted too.

There are several html preprocessor languages out there. html.vim has been written such that it should be trivial to include it. To do so add the following two lines to the syntax coloring file for that language (the example comes from the asp.vim file):

```
runtime! syntax/html.vim
syn cluster htmlPreproc add=asp
```

Now you just need to make sure that you add all regions that contain the preprocessor language to the cluster htmlPreproc.

The HTML syntax file provides syntax **html-folding** (see `:syn-fold` ) between start and end tags. This can be turned on by

```
:let g:html_syntax_folding = 1
:set foldmethod=syntax
```

**Note:** Syntax folding might slow down syntax highlighting significantly, especially for large files.

HTML/OS (by Aestiva) [htmllos.vim](#) [ft-htmllos-syntax](#)

The coloring scheme for HTML/OS works as follows:

Functions and variable names are the same color by default, because VIM doesn't specify different colors for Functions and Identifiers. To change this (which is recommended if you want function names to be recognizable in a different color) you need to add the following line to either your `~/.vimrc`:

```
:hi Function term=underline cterm=bold ctermfg=LightGray
```

Of course, the `ctermfg` can be a different color if you choose.

Another issue that HTML/OS runs into is that there is no special filetype to signify that it is a file with HTML/OS coding. You can change this by opening a file and turning on HTML/OS syntax by doing the following:

```
:set syntax=htmllos
```

Lastly, it should be noted that the opening and closing characters to begin a block of HTML/OS code can either be `<<` or `[[` and `>>` or `]]`, respectively.

**IA64** [ia64.vim](#) [intel-itanium](#) [ft-ia64-syntax](#)

Highlighting for the Intel Itanium 64 assembly language. See [asm.vim](#) for how to recognize this filetype.

To have `*.inc` files be recognized as IA64, add this to your `.vimrc` file:

```
:let g:filetype_inc = "ia64"
```

**INFORM** [inform.vim](#) [ft-inform-syntax](#)

Inform highlighting includes symbols provided by the Inform Library, as most programs make extensive use of it. If do not wish Library symbols to be highlighted add this to your vim startup:

```
:let inform_highlight_simple=1
```

By default it is assumed that Inform programs are Z-machine targeted, and highlights Z-machine assembly language symbols appropriately. If you intend your program to be targeted to a Glulx/Glk environment you need to add this to your startup sequence:

```
:let inform_highlight_glulx=1
```

This will highlight Glulx opcodes instead, and also adds `glk()` to the set of highlighted system functions.

The Inform compiler will flag certain obsolete keywords as errors when it encounters them. These keywords are normally highlighted as errors by Vim. To prevent such error highlighting, you must add this to your startup sequence:

```
:let inform_suppress_obsolete=1
```

By default, the language features highlighted conform to Compiler version 6.30 and Library version 6.11. If you are using an older Inform development environment, you may wish to add this to your startup sequence:

```
:let inform_highlight_old=1
```

## IDL

[idl.vim](#) [idl-syntax](#)

IDL (Interface Definition Language) files are used to define RPC calls. In Microsoft land, this is also used for defining COM interfaces and calls.

IDL's structure is simple enough to permit a full grammar based approach to rather than using a few heuristics. The result is large and somewhat repetitive but seems to work.

There are some Microsoft extensions to idl files that are here. Some of them are disabled by defining `idl_no_ms_extensions`.

The more complex of the extensions are disabled by defining `idl_no_extensions`.

Variable	Effect
<code>idl_no_ms_extensions</code>	Disable some of the Microsoft specific extensions
<code>idl_no_extensions</code>	Disable complex extensions
<code>idlsyntax_showerror</code>	Show IDL errors (can be rather intrusive, but quite helpful)
<code>idlsyntax_showerror_soft</code>	Use softer colours by default for errors

## JAVA

[java.vim](#) [ft-java-syntax](#)

The `java.vim` syntax highlighting file offers several options:

In Java 1.0.2 it was never possible to have braces inside parens, so this was flagged as an error. Since Java 1.1 this is possible (with anonymous classes), and therefore is no longer marked as an error. If you prefer the old way, put the following line into your vim startup file:

```
:let java_mark_braces_in_parens_as_errors=1
```

All identifiers in `java.lang.*` are always visible in all classes. To highlight them use:

```
:let java_highlight_java_lang_ids=1
```

You can also highlight identifiers of most standard Java packages if you download the `javaid.vim` script at <http://www.fleiner.com/vim/download.html>. If you prefer to only highlight identifiers of a certain package, say `java.io` use the following:

```
:let java_highlight_java_io=1
```

Check the `javaid.vim` file for a list of all the packages that are supported.

Function names are not highlighted, as the way to find functions depends on

how you write Java code. The syntax file knows two possible ways to highlight functions:

If you write function declarations that are always indented by either a tab, 8 spaces or 2 spaces you may want to set

```
:let java_highlight_functions="indent"
```

However, if you follow the Java guidelines about how functions and classes are supposed to be named (with respect to upper and lowercase), use

```
:let java_highlight_functions="style"
```

If both options do not work for you, but you would still want function declarations to be highlighted create your own definitions by changing the definitions in `java.vim` or by creating your own `java.vim` which includes the original one and then adds the code to highlight functions.

In Java 1.1 the functions `System.out.println()` and `System.err.println()` should only be used for debugging. Therefore it is possible to highlight debugging statements differently. To do this you must add the following definition in your startup file:

```
:let java_highlight_debug=1
```

The result will be that those statements are highlighted as 'Special' characters. If you prefer to have them highlighted differently you must define new highlightings for the following groups.:

`Debug`, `DebugSpecial`, `DebugString`, `DebugBoolean`, `DebugType` which are used for the statement itself, special characters used in debug strings, strings, boolean constants and types (`this`, `super`) respectively. I have opted to choose another background for those statements.

Javadoc is a program that takes special comments out of Java program files and creates HTML pages. The standard configuration will highlight this HTML code similarly to HTML files (see [html.vim](#)). You can even add Javascript and CSS inside this code (see below). There are four differences however:

1. The title (all characters up to the first '.' which is followed by some white space or up to the first '@') is colored differently (to change the color change the group `CommentTitle`).
2. The text is colored as 'Comment'.
3. HTML comments are colored as 'Special'
4. The special Javadoc tags (`@see`, `@param`, ...) are highlighted as specials and the argument (for `@see`, `@param`, `@exception`) as `Function`.

To turn this feature off add the following line to your startup file:

```
:let java_ignore_javadoc=1
```

If you use the special Javadoc comment highlighting described above you can also turn on special highlighting for Javascript, visual basic scripts and embedded CSS (stylesheets). This makes only sense if you actually have Javadoc comments that include either Javascript or embedded CSS. The options to use are

```
:let java_javascript=1
```

```
:let java_css=1
```

```
:let java_vb=1
```

In order to highlight nested parens with different colors define colors for `javaParen`, `javaParen1` and `javaParen2`, for example with

```
:hi link javaParen Comment
```

or

```
:hi javaParen ctermfg=blue guifg=#0000ff
```

If you notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the "java\_minlines" internal variable to a larger number:

```
:let java_minlines = 50
```

This will make the syntax synchronization start 50 lines before the first displayed line. The default value is 10. The disadvantage of using a larger number is that redrawing can become slow.

## JSON

json.vim ft-json-syntax

The json syntax file provides syntax highlighting with conceal support by default. To disable concealment:

```
let g:vim_json_conceal = 0
```

To disable syntax highlighting of errors:

```
let g:vim_json_warnings = 0
```

## LACE

lace.vim ft-lace-syntax

Lace (Language for Assembly of Classes in Eiffel) is case insensitive, but the style guide lines are not. If you prefer case insensitive highlighting, just define the vim variable 'lace\_case\_insensitive' in your startup file:

```
:let lace_case_insensitive=1
```

## LEX

lex.vim ft-lex-syntax

Lex uses brute-force synchronizing as the "^%%" section delimiter gives no clue as to what section follows. Consequently, the value for

```
:syn sync minlines=300
```

may be changed by the user if s/he is experiencing synchronization difficulties (such as may happen with large lex files).

## LIFELINES

lifelines.vim ft-lifelines-syntax

To highlight deprecated functions as errors, add in your .vimrc:

```
:let g:lifelines_deprecated = 1
```

## LISP

lisp.vim ft-lisp-syntax

The lisp syntax highlighting provides two options:

```
g:lisp_instring : If it exists, then "(...)" strings are highlighted
 as if the contents of the string were lisp.
 Useful for AutoLisp.
```

```
g:lisp_rainbow : If it exists and is nonzero, then differing levels
 of parenthesization will receive different
```

## highlighting.

The `g:lisp_rainbow` option provides 10 levels of individual colorization for the parentheses and backquoted parentheses. Because of the quantity of colorization levels, unlike non-rainbow highlighting, the rainbow mode specifies its highlighting using `ctermfg` and `guifg`, thereby bypassing the usual color scheme control using standard highlighting groups. The actual highlighting used depends on the dark/bright setting (see `'bg'`).

### LITE

`lite.vim` `ft-lite-syntax`

There are two options for the lite syntax highlighting.

If you like SQL syntax highlighting inside Strings, use this:

```
:let lite_sql_query = 1
```

For syncing, `minlines` defaults to 100. If you prefer another value, you can set `"lite_minlines"` to the value you desire. Example:

```
:let lite_minlines = 200
```

### LPC

`lpc.vim` `ft-lpc-syntax`

LPC stands for a simple, memory-efficient language: Lars Pensjö C. The file name of LPC is usually `*.c`. Recognizing these files as LPC would bother users writing only C programs. If you want to use LPC syntax in Vim, you should set a variable in your `.vimrc` file:

```
:let lpc_syntax_for_c = 1
```

If it doesn't work properly for some particular C or LPC files, use a modeline. For a LPC file:

```
// vim:set ft=lpc:
```

For a C file that is recognized as LPC:

```
// vim:set ft=c:
```

If you don't want to set the variable, use the modeline in EVERY LPC file.

There are several implementations for LPC, we intend to support most widely used ones. Here the default LPC syntax is for MudOS series, for MudOS v22 and before, you should turn off the sensible modifiers, and this will also assert the new efuns after v22 to be invalid, don't set this variable when you are using the latest version of MudOS:

```
:let lpc_pre_v22 = 1
```

For LpMud 3.2 series of LPC:

```
:let lpc_compat_32 = 1
```

For LPC4 series of LPC:

```
:let lpc_use_lpc4_syntax = 1
```

For uLPC series of LPC:

uLPC has been developed to Pike, so you should use Pike syntax instead, and the name of your source file should be \*.pike

## LUA

lua.vim ft-lua-syntax

The Lua syntax file can be used for versions 4.0, 5.0, 5.1 and 5.2 (5.2 is the default). You can select one of these versions using the global variables lua\_version and lua\_subversion. For example, to activate Lua 5.1 syntax highlighting, set the variables like this:

```
:let lua_version = 5
:let lua_subversion = 1
```

## MAIL

mail.vim ft-mail.vim

Vim highlights all the standard elements of an email (headers, signatures, quoted text and URLs / email addresses). In keeping with standard conventions, signatures begin in a line containing only "---" followed optionally by whitespaces and end with a newline.

Vim treats lines beginning with ']', '}', '|', '>' or a word followed by '>' as quoted text. However Vim highlights headers and signatures in quoted text only if the text is quoted with '>' (optionally followed by one space).

By default mail.vim synchronises syntax to 100 lines before the first displayed line. If you have a slow machine, and generally deal with emails with short headers, you can change this to a smaller value:

```
:let mail_minlines = 30
```

## MAKE

make.vim ft-make-syntax

In makefiles, commands are usually highlighted to make it easy for you to spot errors. However, this may be too much coloring for you. You can turn this feature off by using:

```
:let make_no_commands = 1
```

## MAPLE

maple.vim ft-maple-syntax

Maple V, by Waterloo Maple Inc, supports symbolic algebra. The language supports many packages of functions which are selectively loaded by the user. The standard set of packages' functions as supplied in Maple V release 4 may be

highlighted at the user's discretion. Users may place in their .vimrc file:

```
:let mvpkg_all= 1
```

to get all package functions highlighted, or users may select any subset by choosing a variable/package from the table below and setting that variable to 1, also in their .vimrc file (prior to sourcing \$VIMRUNTIME/syntax/syntax.vim).

Table of Maple V Package Function Selectors

mv_DEtools	mv_genfunc	mv_networks	mv_process
mv_Galois	mv_geometry	mv_numapprox	mv_simplex
mv_GaussInt	mv_grobner	mv_numtheory	mv_stats
mv_LREtools	mv_group	mv_orthopoly	mv_student
mv_combinat	mv_inttrans	mv_padic	mv_sumtools
mv_combstruct	mv_liesymm	mv_plots	mv_tensor
mv_diffforms	mv_linalg	mv_plottools	mv_totorder
mv_finance	mv_logic	mv_powseries	

## MARKDOWN

ft-markdown-syntax

If you have long regions there might be wrong highlighting. At the cost of slowing down displaying, you can have the engine look further back to sync on the start of a region, for example 500 lines (default is 50):

```
:let g:markdown_minlines = 500
```

If you want to enable fenced code block syntax highlighting in your markdown documents you can enable like this:

```
:let g:markdown_fenced_languages = ['html', 'python', 'bash=sh']
```

To disable markdown syntax concealing add the following to your vimrc:

```
:let g:markdown_syntax_conceal = 0
```

## MATHEMATICA

mma.vim

ft-mma-syntax

ft-mathematica-syntax

Empty \*.m files will automatically be presumed to be Matlab files unless you have the following in your .vimrc:

```
let filetype_m = "mma"
```

## MODULA2

modula2.vim

ft-modula2-syntax

Vim will recognise comments with dialect tags to automatically select a given dialect.

The syntax for a dialect tag comment is:

```
taggedComment :=
```



```

 '(*!' dialectTag '*)'
 ;

 dialectTag :=
 m2pim | m2iso | m2r10
 ;

 reserved words
 m2pim = 'm2pim', m2iso = 'm2iso', m2r10 = 'm2r10'

```

A dialect tag comment is recognised by Vim if it occurs within the first 200 lines of the source file. Only the very first such comment is recognised, any additional dialect tag comments are ignored.

Example:

```

 DEFINITION MODULE FooLib; (*!m2pim*)
 ...

```

Variable `g:modula2_default_dialect` sets the default Modula-2 dialect when the dialect cannot be determined from the contents of the Modula-2 file: if defined and set to 'm2pim', the default dialect is PIM.

Example:

```

 let g:modula2_default_dialect = 'm2pim'

```

Highlighting is further configurable for each dialect via the following variables.

Variable	Highlight
<code>modula2_iso_allow_lowline</code>	allow low line in identifiers
<code>modula2_iso_disallow_octals</code>	disallow octal integer literals
<code>modula2_iso_disallow_synonyms</code>	disallow "@", "&" and "~" synonyms
<code>modula2_pim_allow_lowline</code>	allow low line in identifiers
<code>modula2_pim_disallow_octals</code>	disallow octal integer literals
<code>modula2_pim_disallow_synonyms</code>	disallow "&" and "~" synonyms
<code>modula2_r10_allow_lowline</code>	allow low line in identifiers

**MOO** `moo.vim` `ft-moo-syntax`

If you use C-style comments inside expressions and find it mangles your highlighting, you may want to use extended (slow!) matches for C-style comments:

```

 :let moo_extended_cstyle_comments = 1

```

To disable highlighting of pronoun substitution patterns inside strings:

```

 :let moo_no_pronoun_sub = 1

```

To disable highlighting of the regular expression operator '%|', and matching '%(' and '%)' inside strings:

```
:let moo_no_regexp = 1
```

Unmatched double quotes can be recognized and highlighted as errors:

```
:let moo_unmatched_quotes = 1
```

To highlight builtin properties (.name, .location, .programmer etc.):

```
:let moo_builtin_properties = 1
```

Unknown builtin functions can be recognized and highlighted as errors. If you use this option, add your own extensions to the mooKnownBuiltinFunction group. To enable this option:

```
:let moo_unknown_builtin_functions = 1
```

An example of adding sprintf() to the list of known builtin functions:

```
:syn keyword mooKnownBuiltinFunction sprintf contained
```

## MSQL

[msql.vim](#) [ft-msql-syntax](#)

There are two options for the msql syntax highlighting.

If you like SQL syntax highlighting inside Strings, use this:

```
:let msql_sql_query = 1
```

For syncing, minlines defaults to 100. If you prefer another value, you can set "msql\_minlines" to the value you desire. Example:

```
:let msql_minlines = 200
```

## N1QL

[n1ql.vim](#) [ft-n1ql-syntax](#)

N1QL is a SQL-like declarative language for manipulating JSON documents in Couchbase Server databases.

Vim syntax highlights N1QL statements, keywords, operators, types, comments, and special values. Vim ignores syntactical elements specific to SQL or its many dialects, like COLUMN or CHAR, that don't exist in N1QL.

## NCF

[ncf.vim](#) [ft-ncf-syntax](#)

There is one option for NCF syntax highlighting.

If you want to have unrecognized (by ncf.vim) statements highlighted as errors, use this:

```
:let ncf_highlight_unknowns = 1
```

If you don't want to highlight these errors, leave it unset.

## **NROFF**

`nroff.vim` `ft-nroff-syntax`

The nroff syntax file works with AT&T n/troff out of the box. You need to activate the GNU groff extra features included in the syntax file before you can use them.

For example, Linux and BSD distributions use groff as their default text processing package. In order to activate the extra syntax highlighting features for groff, arrange for files to be recognized as groff (see `ft-groff-syntax`) or add the following option to your start-up files:

```
:let nroff_is_groff = 1
```

Groff is different from the old AT&T n/troff that you may still find in Solaris. Groff macro and request names can be longer than 2 characters and there are extensions to the language primitives. For example, in AT&T troff you access the year as a 2-digit number with the request `\(yr`. In groff you can use the same request, recognized for compatibility, or you can use groff's native syntax, `\[yr]`. Furthermore, you can use a 4-digit year directly: `\[year]`. Macro requests can be longer than 2 characters, for example, GNU mm accepts the requests `".VERBON"` and `".VERBOFF"` for creating verbatim environments.

In order to obtain the best formatted output g/troff can give you, you should follow a few simple rules about spacing and punctuation.

1. Do not leave empty spaces at the end of lines.
2. Leave one space and one space only after an end-of-sentence period, exclamation mark, etc.
3. For reasons stated below, it is best to follow all period marks with a carriage return.

The reason behind these unusual tips is that g/n/troff have a line breaking algorithm that can be easily upset if you don't follow the rules given above.

Unlike TeX, troff fills text line-by-line, not paragraph-by-paragraph and, furthermore, it does not have a concept of glue or stretch, all horizontal and vertical space input will be output as is.

Therefore, you should be careful about not using more space between sentences than you intend to have in your final document. For this reason, the common practice is to insert a carriage return immediately after all punctuation marks. If you want to have "even" text in your final processed output, you need to maintain regular spacing in the input text. To mark both trailing spaces and two or more spaces after a punctuation as an error, use:

```
:let nroff_space_errors = 1
```

Another technique to detect extra spacing and other errors that will interfere with the correct typesetting of your file, is to define an eye-catching highlighting definition for the syntax groups "nroffDefinition" and "nroffDefSpecial" in your configuration files. For example:

```
hi def nroffDefinition term=italic cterm=italic gui=reverse
hi def nroffDefSpecial term=italic,bold cterm=italic,bold
\ gui=reverse,bold
```

If you want to navigate preprocessor entries in your source file as easily as with section markers, you can activate the following option in your .vimrc file:

```
let b:preprocs_as_sections = 1
```

As well, the syntax file adds an extra paragraph marker for the extended paragraph macro (.XP) in the ms package.

Finally, there is a `groff.vim` syntax file that can be used for enabling groff syntax highlighting either on a file basis or globally by default.

## OCAML

`ocaml.vim` `ft-ocaml-syntax`

The OCaml syntax file handles files having the following prefixes: .ml, .mli, .mll and .mly. By setting the following variable

```
:let ocaml_revised = 1
```

you can switch from standard OCaml-syntax to revised syntax as supported by the camlp4 preprocessor. Setting the variable

```
:let ocaml_noend_error = 1
```

prevents highlighting of "end" as error, which is useful when sources contain very long structures that Vim does not synchronize anymore.

## PAPP

`papp.vim` `ft-papp-syntax`

The PApp syntax file handles .papp files and, to a lesser extent, .pxml and .pxsl files which are all a mixture of perl/xml/html/other using xml as the top-level file format. By default everything inside phtml or pxml sections is treated as a string with embedded preprocessor commands. If you set the variable:

```
:let papp_include_html=1
```

in your startup file it will try to syntax-highlight html code inside phtml sections, but this is relatively slow and much too colourful to be able to edit sensibly. ;)

The newest version of the papp.vim syntax file can usually be found at <http://papp.plan9.de>.

## PASCAL

pascal.vim ft-pascal-syntax

Files matching "\*.p" could be Progress or Pascal and those matching "\*.pp" could be Puppet or Pascal. If the automatic detection doesn't work for you, or you only edit Pascal files, use this in your startup vimrc:

```
:let filetype_p = "pascal"
:let filetype_pp = "pascal"
```

The Pascal syntax file has been extended to take into account some extensions provided by Turbo Pascal, Free Pascal Compiler and GNU Pascal Compiler. Delphi keywords are also supported. By default, Turbo Pascal 7.0 features are enabled. If you prefer to stick with the standard Pascal keywords, add the following line to your startup file:

```
:let pascal_traditional=1
```

To switch on Delphi specific constructions (such as one-line comments, keywords, etc):

```
:let pascal_delphi=1
```

The option pascal\_symbol\_operator controls whether symbol operators such as +, \*, .., etc. are displayed using the Operator color or not. To colorize symbol operators, add the following line to your startup file:

```
:let pascal_symbol_operator=1
```

Some functions are highlighted by default. To switch it off:

```
:let pascal_no_functions=1
```

Furthermore, there are specific variables for some compilers. Besides pascal\_delphi, there are pascal\_gpc and pascal\_fpc. Default extensions try to match Turbo Pascal.

```
:let pascal_gpc=1
```

or

```
:let pascal_fpc=1
```

To ensure that strings are defined on a single line, you can define the pascal\_one\_line\_string variable.

```
:let pascal_one_line_string=1
```

If you dislike <Tab> chars, you can set the pascal\_no\_tabs variable. Tabs will be highlighted as Error.

```
:let pascal_no_tabs=1
```

## PERL

perl.vim ft-perl-syntax

There are a number of possible options to the perl syntax highlighting.

Inline POD highlighting is now turned on by default. If you don't wish to have the added complexity of highlighting POD embedded within Perl files, you may set the 'perl\_include\_pod' option to 0:

```
:let perl_include_pod = 0
```

To reduce the complexity of parsing (and increase performance) you can switch off two elements in the parsing of variable names and contents.

To handle package references in variable and function names not differently from the rest of the name (like 'PkgName::' in '\$PkgName::VarName'):

```
:let perl_no_scope_in_variables = 1
```

(In Vim 6.x it was the other way around: "perl\_want\_scope\_in\_variables" enabled it.)

If you do not want complex things like '@{\${"foo"}}' to be parsed:

```
:let perl_no_extended_vars = 1
```

(In Vim 6.x it was the other way around: "perl\_extended\_vars" enabled it.)

The coloring strings can be changed. By default strings and qq friends will be highlighted like the first line. If you set the variable perl\_string\_as\_statement, it will be highlighted as in the second line.

```
"hello world!"; qq|hello world|;
^.....^NN^.....^N (unlet perl_string_as_statement)
S^.....^SNSSS^.....^SN (let perl_string_as_statement)
```

(^ = perlString, S = perlStatement, N = None at all)

The syncing has 3 options. The first two switch off some triggering of synchronization and should only be needed in case it fails to work properly. If while scrolling all of a sudden the whole screen changes color completely then you should try and switch off one of those. Let me know if you can figure out the line that causes the mistake.

One triggers on "^\\s\*sub\\s\*" and the other on "^[\$@%]" more or less.

```
:let perl_no_sync_on_sub
:let perl_no_sync_on_global_var
```

Below you can set the maximum distance VIM should look for starting points for

its attempts in syntax highlighting.

```
:let perl_sync_dist = 100
```

If you want to use folding with perl, set perl\_fold:

```
:let perl_fold = 1
```

If you want to fold blocks in if statements, etc. as well set the following:

```
:let perl_fold_blocks = 1
```

Subroutines are folded by default if 'perl\_fold' is set. If you do not want this, you can set 'perl\_nofold\_subs':

```
:let perl_nofold_subs = 1
```

Anonymous subroutines are not folded by default; you may enable their folding via 'perl\_fold\_anonymous\_subs':

```
:let perl_fold_anonymous_subs = 1
```

Packages are also folded by default if 'perl\_fold' is set. To disable this behavior, set 'perl\_nofold\_packages':

```
:let perl_nofold_packages = 1
```

PHP3 and PHP4                    `php.vim`   `php3.vim`   `ft-php-syntax`   `ft-php3-syntax`

[Note: Previously this was called "php3", but since it now also supports php4 it has been renamed to "php"]

There are the following options for the php syntax highlighting.

If you like SQL syntax highlighting inside Strings:

```
let php_sql_query = 1
```

For highlighting the Baselib methods:

```
let php_baselib = 1
```

Enable HTML syntax highlighting inside strings:

```
let php_htmlInStrings = 1
```

Using the old colorstyle:

```
let php_oldStyle = 1
```

Enable highlighting ASP-style short tags:

```
let php_esp_tags = 1
```

Disable short tags:

```
let php_noShortTags = 1
```

For highlighting parent error ] or ):

```
let php_parent_error_close = 1
```

For skipping a php end tag, if there exists an open ( or [ without a closing one:

```
let php_parent_error_open = 1
```

Enable folding for classes and functions:

```
let php_folding = 1
```

Selecting syncing method:

```
let php_sync_method = x
```

x = -1 to sync by search (default),  
x > 0 to sync at least x lines backwards,  
x = 0 to sync from start.

## PLAINTEX

[plaintex.vim](#) [ft-plaintex-syntax](#)

TeX is a typesetting language, and plaintex is the file type for the "plain" variant of TeX. If you never want your \*.tex files recognized as plain TeX, see [ft-tex-plugin](#).

This syntax file has the option

```
let g:plaintex_delimiters = 1
```

if you want to highlight brackets "[]" and braces "{}".

## PPWIZARD

[ppwiz.vim](#) [ft-ppwiz-syntax](#)

PPWizard is a preprocessor for HTML and OS/2 INF files

This syntax file has the options:

- `ppwiz_highlight_defs` : Determines highlighting mode for PPWizard's definitions. Possible values are

`ppwiz_highlight_defs = 1` : PPWizard #define statements retain the colors of their contents (e.g. PPWizard macros and variables).

`ppwiz_highlight_defs = 2` : Preprocessor #define and #evaluate statements are shown in a single color with the exception of line continuation symbols.



The default setting for `ppwiz_highlight_defs` is 1.

- `ppwiz_with_html` : If the value is 1 (the default), highlight literal HTML code; if 0, treat HTML code like ordinary text.

## **PHTML** `phtml.vim` `ft-phtml-syntax`

There are two options for the phtml syntax highlighting.

If you like SQL syntax highlighting inside Strings, use this:

```
:let phtml_sql_query = 1
```

For syncing, `minlines` defaults to 100. If you prefer another value, you can set "`phtml_minlines`" to the value you desire. Example:

```
:let phtml_minlines = 200
```

## **POSTSCRIPT** `postscr.vim` `ft-postscr-syntax`

There are several options when it comes to highlighting PostScript.

First which version of the PostScript language to highlight. There are currently three defined language versions, or levels. Level 1 is the original and base version, and includes all extensions prior to the release of level 2. Level 2 is the most common version around, and includes its own set of extensions prior to the release of level 3. Level 3 is currently the highest level supported. You select which level of the PostScript language you want highlighted by defining the `postscr_level` variable as follows:

```
:let postscr_level=2
```

If this variable is not defined it defaults to 2 (level 2) since this is the most prevalent version currently.

**Note:** Not all PS interpreters will support all language features for a particular language level. In particular the `%!PS-Adobe-3.0` at the start of PS files does NOT mean the PostScript present is level 3 PostScript!

If you are working with Display PostScript, you can include highlighting of Display PS language features by defining the `postscr_display` variable as follows:

```
:let postscr_display=1
```

If you are working with Ghostscript, you can include highlighting of Ghostscript specific language features by defining the variable `postscr_ghostscript` as follows:

```
:let postscr_ghostscript=1
```

PostScript is a large language, with many predefined elements. While it is useful to have all these elements highlighted, on slower machines this can cause Vim to slow down. In an attempt to be machine friendly font names and character encodings are not highlighted by default. Unless you are working explicitly with either of these this should be ok. If you want them to be highlighted you should set one or both of the following variables:

```
:let postscr_fonts=1
:let postscr_encodings=1
```

There is a stylistic option to the highlighting of and, or, and not. In PostScript the function of these operators depends on the types of their operands - if the operands are booleans then they are the logical operators, if they are integers then they are binary operators. As binary and logical operators can be highlighted differently they have to be highlighted one way or the other. By default they are treated as logical operators. They can be highlighted as binary operators by defining the variable `postscr_andornot_binary` as follows:

```
:let postscr_andornot_binary=1
```

```
PRINTCAP + TERMCAP ptcap.vim ft-printcap-syntax
 ft-ptcap-syntax ft-termcap-syntax
```

This syntax file applies to the `printcap` and `termcap` databases.

In order for Vim to recognize `printcap`/`termcap` files that do not match the patterns `*printcap*`, or `*termcap*`, you must put additional patterns appropriate to your system in your `myfiletypefile` file. For these patterns, you must set the variable `"b:ptcap_type"` to either `"print"` or `"term"`, and then the `'filetype'` option to `ptcap`.

For example, to make Vim identify all files in `/etc/termcaps/` as `termcap` files, add the following:

```
:au BufNewFile,BufRead /etc/termcaps/* let b:ptcap_type = "term" |
\ set filetype=ptcap
```

If you notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the `"ptcap_minlines"` internal variable to a larger number:

```
:let ptcap_minlines = 50
```

(The default is 20 lines.)

```
PROGRESS progress.vim ft-progress-syntax
```

Files matching `*.w` could be `Progress` or `cweb`. If the automatic detection doesn't work for you, or you don't edit `cweb` at all, use this in your startup `vimrc`:

```
:let filetype_w = "progress"
```

The same happens for "\*.i", which could be assembly, and "\*.p", which could be Pascal. Use this if you don't use assembly and Pascal:

```
:let filetype_i = "progress"
:let filetype_p = "progress"
```

## PYTHON

[python.vim](#) [ft-python-syntax](#)

There are six options to control Python syntax highlighting.

For highlighted numbers:

```
:let python_no_number_highlight = 1
```

For highlighted builtin functions:

```
:let python_no_builtin_highlight = 1
```

For highlighted standard exceptions:

```
:let python_no_exception_highlight = 1
```

For highlighted doctests and code inside:

```
:let python_no_doctest_highlight = 1
```

or

```
:let python_no_doctest_code_highlight = 1
```

The first option implies the second one.

For highlighted trailing whitespace and mix of spaces and tabs:

```
:let python_space_error_highlight = 1
```

If you want all possible Python highlighting:

```
:let python_highlight_all = 1
```

This has the same effect as setting `python_space_error_highlight` and unsetting all the other ones.

If you use Python 2 or straddling code (Python 2 and 3 compatible), you can enforce the use of an older syntax file with support for Python 2 and up to Python 3.5.

```
:let python_use_python2_syntax = 1
```

This option will exclude all modern Python 3.6 or higher features.

**Note:** Only existence of these options matters, not their value.

You can replace 1 above with anything.

## QUAKE

[quake.vim](#) [ft-quake-syntax](#)

The Quake syntax definition should work for most FPS (First Person Shooter) based on one of the Quake engines. However, the command names vary a bit between the three games (Quake, Quake 2, and Quake 3 Arena) so the syntax definition checks for the existence of three global variables to allow users to specify what commands are legal in their files. The three variables can be set for the following effects:

set to highlight commands only available in Quake:

```
:let quake_is_quake1 = 1
```

set to highlight commands only available in Quake 2:  
`:let quake_is_quake2 = 1`

set to highlight commands only available in Quake 3 Arena:  
`:let quake_is_quake3 = 1`

Any combination of these three variables is legal, but might highlight more commands than are actually available to you by the game.

## R r.vim ft-r-syntax

The parsing of R code for syntax highlight starts 40 lines backwards, but you can set a different value in your `vimrc`. Example:

```
let r_syntax_minlines = 60
```

You can also turn off syntax highlighting of R0xygen:

```
let r_syntax_hl_roxygen = 0
```

enable folding of code delimited by parentheses, square brackets and curly braces:

```
let r_syntax_folding = 1
```

and highlight as functions all keywords followed by an opening parenthesis:

```
let r_syntax_fun_pattern = 1
```

## R MARKDOWN rmd.vim ft-rmd-syntax

To disable syntax highlight of YAML header, add to your `vimrc`:

```
let rmd_syn_hl_yaml = 0
```

To disable syntax highlighting of citation keys:

```
let rmd_syn_hl_citations = 0
```

To highlight R code in knitr chunk headers:

```
let rmd_syn_hl_chunk = 1
```

By default, chunks of R code will be highlighted following the rules of R language. Moreover, whenever the buffer is saved, Vim scans the buffer and highlights other languages if they are present in new chunks. LaTeX code also is automatically recognized and highlighted when the buffer is saved. This behavior can be controlled with the variables ``rmd_dynamic_fenced_languages``, and ``rmd_include_latex`` whose valid values are:

```
let rmd_dynamic_fenced_languages = 0 " No autodetection of languages
let rmd_dynamic_fenced_languages = 1 " Autodetection of languages
let rmd_include_latex = 0 " Don't highlight LaTeX code
let rmd_include_latex = 1 " Autodetect LaTeX code
let rmd_include_latex = 2 " Always include LaTeX highlighting
```

If the value of ``rmd_dynamic_fenced_languages`` is 0, you still can set the list of languages whose chunks of code should be properly highlighted, as in the example:

```
let rmd_fenced_languages = ['r', 'python']
```

## R RESTRUCTURED TEXT

[rrst.vim](#) [ft-rrst-syntax](#)

To highlight R code in knitr chunk headers, add to your `vimrc` :

```
let rrst_syn_hl_chunk = 1
```

## READLINE

[readline.vim](#) [ft-readline-syntax](#)

The readline library is primarily used by the BASH shell, which adds quite a few commands and options to the ones already available. To highlight these items as well you can add the following to your `vimrc` or just type it in the command line before loading a file with the readline syntax:

```
let readline_has_bash = 1
```

This will add highlighting for the commands that BASH (version 2.05a and later, and part earlier) adds.

## REGO

[rego.vim](#) [ft-rego-syntax](#)

Rego is a query language developed by Styra. It is mostly used as a policy language for kubernetes, but can be applied to almost anything. Files with the following extensions are recognized as rego files: `.rego`.

## RESTRUCTURED TEXT

[rst.vim](#) [ft-rst-syntax](#)

Syntax highlighting is enabled for code blocks within the document for a select number of file types. See `$VIMRUNTIME/syntax/rst.vim` for the default syntax list.

To set a user-defined list of code block syntax highlighting:

```
let rst_syntax_code_list = ['vim', 'lisp', ...]
```

To assign multiple code block types to a single syntax, define `'rst_syntax_code_list'` as a mapping:

```
let rst_syntax_code_list = {
 \ 'cpp': ['cpp', 'c++'],
 \ 'bash': ['bash', 'sh'],
 ...
\ }
```

To use color highlighting for emphasis text:

```
let rst_use_emphasis_colors = 1
```

To enable folding of sections:

```
let rst_fold_enabled = 1
```

**Note** that folding can cause performance issues on some platforms.

## REXX

rex.vim ft-rexx-syntax

If you notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the "rex\_minlines" internal variable to a larger number:

```
:let rex_minlines = 50
```

This will make the syntax synchronization start 50 lines before the first displayed line. The default value is 10. The disadvantage of using a larger number is that redrawing can become slow.

Vim tries to guess what type a ".r" file is. If it can't be detected (from comment lines), the default is "r". To make the default rexx add this line to your .vimrc: `g:filetype_r`

```
:let g:filetype_r = "r"
```

## RUBY

ruby.vim ft-ruby-syntax

Ruby: Operator highlighting	ruby_operators
Ruby: Whitespace errors	ruby_space_errors
Ruby: Folding	ruby_fold ruby_foldable_groups
Ruby: Reducing expensive operations	ruby_no_expensive ruby_minlines
Ruby: Spellchecking strings	ruby_spellcheck_strings

ruby\_operators

Ruby: Operator highlighting

Operators can be highlighted by defining "ruby\_operators":

```
:let ruby_operators = 1
```

ruby\_space\_errors

Ruby: Whitespace errors

Whitespace errors can be highlighted by defining "ruby\_space\_errors":

```
:let ruby_space_errors = 1
```

This will highlight trailing whitespace and tabs preceded by a space character as errors. This can be refined by defining "ruby\_no\_trail\_space\_error" and "ruby\_no\_tab\_space\_error" which will ignore trailing whitespace and tabs after spaces respectively.

ruby\_fold ruby\_foldable\_groups

Ruby: Folding

Folding can be enabled by defining "ruby\_fold":

```
:let ruby_fold = 1
```

This will set the value of '**foldmethod**' to "syntax" locally to the current buffer or window, which will enable syntax-based folding when editing Ruby filetypes.

Default folding is rather detailed, i.e., small syntax units like "if", "do", "%w[]" may create corresponding fold levels.

You can set "ruby\_foldable\_groups" to restrict which groups are foldable:

```
:let ruby_foldable_groups = 'if case %'
```

The value is a space-separated list of keywords:

keyword	meaning
ALL	Most block syntax (default)
NONE	Nothing
if	"if" or "unless" block
def	"def" block
class	"class" block
module	"module" block
do	"do" block
begin	"begin" block
case	"case" block
for	"for", "while", "until" loops
{	Curly bracket block or hash literal
[	Array literal
%	Literal with "%" notation, e.g.: %w(String), %!STRING!
/	Regexp
string	String and shell command output (surrounded by ', ', ` `)
:	Symbol
#	Multiline comment
<<	Here documents
__END__	Source code after "__END__" directive

ruby\_no\_expensive

### Ruby: Reducing expensive operations

By default, the "end" keyword is colorized according to the opening statement of the block it closes. While useful, this feature can be expensive; if you experience slow redrawing (or you are on a terminal with poor color support) you may want to turn it off by defining the "ruby\_no\_expensive" variable:

```
:let ruby_no_expensive = 1
```

In this case the same color will be used for all control keywords.

ruby\_minlines

If you do want this feature enabled, but notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the "ruby\_minlines" variable to a value larger than 50:

```
:let ruby_minlines = 100
```

Ideally, this value should be a number of lines large enough to embrace your largest class or module.

ruby\_spellcheck\_strings

### Ruby: Spellchecking strings

Ruby syntax will perform spellchecking of strings if you define "ruby\_spellcheck\_strings":

```
:let ruby_spellcheck_strings = 1
```

### SCHEME

scheme.vim ft-scheme-syntax

By default only R7RS keywords are highlighted and properly indented.

scheme.vim also supports extensions of the CHICKEN Scheme->C compiler. Define b:is\_chicken or g:is\_chicken, if you need them.

### SDL

sdl.vim ft-sdl-syntax

The SDL highlighting probably misses a few keywords, but SDL has so many of them it's almost impossible to cope.

The new standard, SDL-2000, specifies that all identifiers are case-sensitive (which was not so before), and that all keywords can be used either completely lowercase or completely uppercase. To have the highlighting reflect this, you can set the following variable:

```
:let sdl_2000=1
```

This also sets many new keywords. If you want to disable the old keywords, which is probably a good idea, use:

```
:let SDL_no_96=1
```

The indentation is probably also incomplete, but right now I am very satisfied with it for my own projects.

### SED

sed.vim ft-sed-syntax

To make tabs stand out from regular blanks (accomplished by using Todo highlighting on the tabs), define "g:sed\_highlight\_tabs" by putting

```
:let g:sed_highlight_tabs = 1
```

in the vimrc file. (This special highlighting only applies for tabs inside search patterns, replacement texts, addresses or text included by an Append/Change/Insert command.) If you enable this option, it is also a good idea to set the tab width to one character; by doing that, you can easily count the number of tabs in a string.

GNU sed allows comments after text on the same line. BSD sed only allows comments where "#" is the first character of the line. To enforce BSD-style comments, i.e. mark end-of-line comments as errors, use:



```
:let g:sed_dialect = "bsd"
```

**Note** that there are other differences between GNU sed and BSD sed which are not (yet) affected by this setting.

Bugs:

The transform command (y) is treated exactly like the substitute command. This means that, as far as this syntax file is concerned, transform accepts the same flags as substitute, which is wrong. (Transform accepts no flags.) I tolerate this bug because the involved commands need very complex treatment (95 patterns, one for each plausible pattern delimiter).

## SGML

[sgml.vim](#) [ft-sgml-syntax](#)

The coloring scheme for tags in the SGML file works as follows.

The `<>` of opening tags are colored differently than the `</>` of a closing tag. This is on purpose! For opening tags the 'Function' color is used, while for closing tags the 'Type' color is used (See `syntax.vim` to check how those are defined for you)

Known tag names are colored the same way as statements in C. Unknown tag names are not colored which makes it easy to spot errors.

**Note** that the same is true for argument (or attribute) names. Known attribute names are colored differently than unknown ones.

Some SGML tags are used to change the rendering of text. The following tags are recognized by the `sgml.vim` syntax coloring file and change the way normal text is shown: `<varname>` `<emphasis>` `<command>` `<function>` `<literal>` `<replaceable>` `<ulink>` and `<link>`.

If you want to change how such text is rendered, you must redefine the following syntax groups:

- `sgmlBold`
- `sgmlBoldItalic`
- `sgmlUnderline`
- `sgmlItalic`
- `sgmlLink` for links

To make this redefinition work you must redefine them all and define the following variable in your `vimrc` (this is due to the order in which the files are read during initialization)

```
let sgml_my_rendering=1
```

You can also disable this rendering by adding the following line to your `vimrc` file:

```
let sgml_no_rendering=1
```

(Adapted from the `html.vim` help text by Claudio Fleiner <[claudio@fleiner.com](mailto:claudio@fleiner.com)>)

**SH**                    `ft-posix-syntax`   `ft-dash-syntax`  
                      `sh.vim`     `ft-sh-syntax`     `ft-bash-syntax`     `ft-ksh-syntax`

This covers syntax highlighting for the older Unix (Bourne) `sh`, and newer shells such as `bash`, `dash`, `posix`, and the Korn shells.

Vim attempts to determine which shell type is in use by specifying that various filenames are of specific types, e.g.:

```
ksh : .kshrc* *.ksh
bash: .bashrc* bashrc bash.bashrc .bash_profile* *.bash
```

See `$VIMRUNTIME/filetype.vim` for the full list of patterns. If none of these cases pertain, then the first line of the file is examined (ex. looking for `/bin/sh` `/bin/ksh` `/bin/bash`). If the first line specifies a shelltype, then that shelltype is used. However some files (ex. `.profile`) are known to be shell files but the type is not apparent. Furthermore, on many systems `sh` is symbolically linked to `"bash"` (Linux, Windows+cygwin) or `"ksh"` (Posix).

One may specify a global default by instantiating one of the following variables in your `<.vimrc>`:

```
ksh:
 let g:is_kornshell = 1
posix: (using this is nearly the same as setting g:is_kornshell to 1)
 let g:is_posix = 1
bash:
 let g:is_bash = 1
sh: (default) Bourne shell
 let g:is_sh = 1
```

(dash users should use `posix`)

If there's no `"#! ..."` line, and the user hasn't availed himself/herself of a default `sh.vim` syntax setting as just shown, then `syntax/sh.vim` will assume the Bourne shell syntax. No need to quote RFCs or market penetration statistics in error reports, please -- just select the default version of the `sh` your system uses and install the associated `"let..."` in your `<.vimrc>`.

The `syntax/sh.vim` file provides several levels of syntax-based folding:

```
let g:sh_fold_enabled= 0 (default, no syntax folding)
let g:sh_fold_enabled= 1 (enable function folding)
let g:sh_fold_enabled= 2 (enable heredoc folding)
let g:sh_fold_enabled= 4 (enable if/do/for folding)
```

then various syntax items (ie. HereDocuments and function bodies) become syntax-foldable (see `:syn-fold`). You also may add these together to get multiple types of folding:

```
let g:sh_fold_enabled= 3 (enables function and heredoc folding)
```

If you notice highlighting errors while scrolling backwards which are fixed when one redraws with **CTRL-L**, try setting the "sh\_minlines" internal variable to a larger number. Example:

```
let sh_minlines = 500
```

This will make syntax synchronization start 500 lines before the first displayed line. The default value is 200. The disadvantage of using a larger number is that redrawing can become slow.

If you don't have much to synchronize on, displaying can be very slow. To reduce this, the "sh\_maxlines" internal variable can be set. Example:

```
let sh_maxlines = 100
```

The default is to use the twice sh\_minlines. Set it to a smaller number to speed up displaying. The disadvantage is that highlight errors may appear.

syntax/sh.vim tries to flag certain problems as errors; usually things like unmatched "]", "done", "fi", etc. If you find the error handling problematic for your purposes, you may suppress such error highlighting by putting the following line in your .vimrc:

```
let g:sh_no_error= 1
```

[sh-embed](#)    [sh-awk](#)

## Sh: EMBEDDING LANGUAGES

You may wish to embed languages into sh. I'll give an example courtesy of Lorance Stinson on how to do this with awk as an example. Put the following file into \$HOME/.vim/after/syntax/sh/awkembed.vim:

```
" AWK Embedding:
" =====
" Shamelessly ripped from asperl.vim by Aaron Hope.
if exists("b:current_syntax")
 unlet b:current_syntax
endif
syn include @AWKScript syntax/awk.vim
syn region AWKScriptCode matchgroup=AWKCommand start+=[\]\@<!' + skip+=\|' + end+='+ co
syn region AWKScriptEmbedded matchgroup=AWKCommand start+=\
```

This code will then let the awk code in the single quotes:

```
awk '...awk code here...'
```

be highlighted using the awk highlighting syntax. Clearly this may be extended to other languages.

## SPEEDUP

(AspenTech plant simulator)

[spup.vim](#)    [ft-spup-syntax](#)

The Speedup syntax file has some options:

- `strict_subsections` : If this variable is defined, only keywords for sections and subsections will be highlighted as statements but not other keywords (like `WITHIN` in the `OPERATION` section).
- `highlight_types` : Definition of this variable causes stream types like temperature or pressure to be highlighted as `Type`, not as a plain Identifier. Included are the types that are usually found in the `DECLARE` section; if you defined own types, you have to include them in the syntax file.
- `oneline_comments` : This value ranges from 1 to 3 and determines the highlighting of `#` style comments.

`oneline_comments = 1` : Allow normal Speedup code after an even number of `#`s.

`oneline_comments = 2` : Show code starting with the second `#` as error. This is the default setting.

`oneline_comments = 3` : Show the whole line as error if it contains more than one `#`.

Since especially `OPERATION` sections tend to become very large due to `PRESET`ting variables, syncing may be critical. If your computer is fast enough, you can increase `minlines` and/or `maxlines` near the end of the syntax file.

## SQL

`sql.vim`    `ft-sql-syntax`  
`sqlinformix.vim`    `ft-sqlinformix-syntax`  
`sqlanywhere.vim`    `ft-sqlanywhere-syntax`

While there is an ANSI standard for SQL, most database engines add their own custom extensions. Vim currently supports the Oracle and Informix dialects of SQL. Vim assumes `*.sql` files are Oracle SQL by default.

Vim currently has SQL support for a variety of different vendors via syntax scripts. You can change Vim's default from Oracle to any of the current SQL supported types. You can also easily alter the SQL dialect being used on a buffer by buffer basis.

For more detailed instructions see [ft\\_sql.txt](#) .

## SQUIRREL

`squirrel.vim`    `ft-squirrel-syntax`

Squirrel is a high level imperative, object-oriented programming language, designed to be a light-weight scripting language that fits in the size, memory bandwidth, and real-time requirements of applications like video games. Files with the following extensions are recognized as squirrel files: `.nut`.

## TCSH

[tcsh.vim](#) [ft-tcsh-syntax](#)

This covers the shell named "tcsh". It is a superset of csh. See [csh.vim](#) for how the filetype is detected.

Tcsh does not allow \" in strings unless the "backslash\_quote" shell variable is set. If you want VIM to assume that no backslash quote constructs exist add this line to your .vimrc:

```
:let tcsh_backslash_quote = 0
```

If you notice highlighting errors while scrolling backwards, which are fixed when redrawing with **CTRL-L**, try setting the "tcsh\_minlines" internal variable to a larger number:

```
:let tcsh_minlines = 1000
```

This will make the syntax synchronization start 1000 lines before the first displayed line. If you set "tcsh\_minlines" to "fromstart", then synchronization is done from the start of the file. The default value for tcsh\_minlines is 100. The disadvantage of using a larger number is that redrawing can become slow.

## TEX

[tex.vim](#) [ft-tex-syntax](#) [latex-syntax](#)  
[syntax-tex](#) [syntax-latex](#)

### Tex Contents

Tex: Want Syntax Folding?	<a href="#">tex-folding</a>
Tex: No Spell Checking Wanted	<a href="#">g:tex_nospell</a>
Tex: Don't Want Spell Checking In Comments?	<a href="#">tex-nospell</a>
Tex: Want Spell Checking in Verbatim Zones?	<a href="#">tex-verb</a>
Tex: Run-on Comments or MathZones	<a href="#">tex-runon</a>
Tex: Slow Syntax Highlighting?	<a href="#">tex-slow</a>
Tex: Want To Highlight More Commands?	<a href="#">tex-morecommands</a>
Tex: Excessive Error Highlighting?	<a href="#">tex-error</a>
Tex: Need a new Math Group?	<a href="#">tex-math</a>
Tex: Starting a New Style?	<a href="#">tex-style</a>
Tex: Taking Advantage of Conceal Mode	<a href="#">tex-conceal</a>
Tex: Selective Conceal Mode	<a href="#">g:tex_conceal</a>
Tex: Controlling iskeyword	<a href="#">g:tex_isk</a>
Tex: Fine Subscript and Superscript Control	<a href="#">tex-supersub</a>
Tex: Match Check Control	<a href="#">tex-matchcheck</a>

[tex-folding](#) [g:tex\\_fold\\_enabled](#)

[Tex: Want Syntax Folding?](#)

As of version 28 of [<syntax/tex.vim>](#), syntax-based folding of parts, chapters, sections, subsections, etc are supported. Put

```
let g:tex_fold_enabled=1
```

in your [.<.vimrc>](#), and `:set fdm=syntax`. I suggest doing the latter via a modeline at the end of your LaTeX file:

```
% vim: fdm=syntax
```

If your system becomes too slow, then you might wish to look into [https://vimhelp.org/vim\\_faq.txt.html#faq-29.7](https://vimhelp.org/vim_faq.txt.html#faq-29.7)

`g:tex_nospell`

Tex: No Spell Checking Wanted

If you don't want spell checking anywhere in your LaTeX document, put `let g:tex_nospell=1` into your `.vimrc`. If you merely wish to suppress spell checking inside comments only, see `g:tex_comment_nospell` .

`tex-nospell` `g:tex_comment_nospell`

Tex: Don't Want Spell Checking In Comments?

Some folks like to include things like source code in comments and so would prefer that spell checking be disabled in comments in LaTeX files. To do this, put the following in your `<.vimrc>`:

```
let g:tex_comment_nospell= 1
```

If you want to suppress spell checking everywhere inside your LaTeX document, see `g:tex_nospell` .

`tex-verb` `g:tex_verbspell`

Tex: Want Spell Checking in Verbatim Zones?

Often verbatim regions are used for things like source code; seldom does one want source code spell-checked. However, for those of you who do want your verbatim zones spell-checked, put the following in your `<.vimrc>`:

```
let g:tex_verbspell= 1
```

`tex-runon` `tex-stopzone`

Tex: Run-on Comments or MathZones

The `<syntax/tex.vim>` highlighting supports TeX, LaTeX, and some AmsTeX. The highlighting supports three primary zones/regions: normal, texZone, and texMathZone. Although considerable effort has been made to have these zones terminate properly, zones delineated by `$.$.` and `$$$.$$` cannot be synchronized as there's no difference between start and end patterns. Consequently, a special "TeX comment" has been provided

```
%stopzone
```

which will forcibly terminate the highlighting of either a texZone or a texMathZone.

`tex-slow` `tex-sync`

Tex: Slow Syntax Highlighting?

If you have a slow computer, you may wish to reduce the values for

```
:syn sync maxlines=200
```

```
:syn sync minlines=50
```

(especially the latter). If your computer is fast, you may wish to increase them. This primarily affects synchronizing (i.e. just what group, if any, is the text at the top of the screen supposed to be in?).

Another cause of slow highlighting is due to syntax-driven folding; see `tex-folding` for a way around this.

`g:tex_fast`

Finally, if syntax highlighting is still too slow, you may set

```
:let g:tex_fast= ""
```

in your `.vimrc`. Used this way, the `g:tex_fast` variable causes the syntax highlighting script to avoid defining any regions and associated synchronization. The result will be much faster syntax highlighting; the price: you will no longer have as much highlighting or any syntax-based folding, and you will be missing syntax-based error checking.

You may decide that some syntax is acceptable; you may use the following table selectively to enable just some syntax highlighting:

```
b : allow bold and italic syntax
c : allow texComment syntax
m : allow texMatcher syntax (ie. {...} and [...])
M : allow texMath syntax
p : allow parts, chapter, section, etc syntax
r : allow texRefZone syntax (nocite, bibliography, label, pageref, eqref)
s : allow superscript/subscript regions
S : allow texStyle syntax
v : allow verbatim syntax
V : allow texNewEnv and texNewCmd syntax
```

As an example, let `g:tex_fast= "M"` will allow math-associated highlighting but suppress all the other region-based syntax highlighting. (also see: `g:tex_conceal` and `tex-supersub` )

`tex-morecommands` `tex-package`

Tex: Want To Highlight More Commands?

LaTeX is a programmable language, and so there are thousands of packages full of specialized LaTeX commands, syntax, and fonts. If you're using such a package you'll often wish that the distributed `syntax/tex.vim` would support it. However, clearly this is impractical. So please consider using the techniques in `mysyntaxfile-add` to extend or modify the highlighting provided by `syntax/tex.vim`. Please consider uploading any extensions that you write, which typically would go in `$HOME/after/syntax/tex/[pkgname].vim`, to <http://vim.sf.net/>.

I've included some support for various popular packages on my website:

<http://www.drchip.org/astronaut/vim/index.html#LATEXPKGS>

The syntax files there go into your `.../after/syntax/tex/` directory.

`tex-error` `g:tex_no_error`

Tex: Excessive Error Highlighting?

The `<tex.vim>` supports lexical error checking of various sorts. Thus, although the error checking is oftentimes very useful, it can indicate

errors where none actually are. If this proves to be a problem for you, you may put in your `<.vimrc>` the following statement:

```
let g:tex_no_error=1
```

and all error checking by `<syntax/tex.vim>` will be suppressed.

tex-math

### Tex: Need a new Math Group?

If you want to include a new math group in your LaTeX, the following code shows you an example as to how you might do so:

```
call TexNewMathZone(sfx,mathzone,starform)
```

You'll want to provide the new math group with a unique suffix (currently, A-L and V-Z are taken by `<syntax/tex.vim>` itself).

As an example, consider how `eqnarray` is set up by `<syntax/tex.vim>`:

```
call TexNewMathZone("D","eqnarray",1)
```

You'll need to change "mathzone" to the name of your new math group, and then to the call to it in `.vim/after/syntax/tex.vim`.

The "starform" variable, if true, implies that your new math group has a starred form (ie. `eqnarray*`).

tex-style b:tex\_stylish

### Tex: Starting a New Style?

One may use `"\makeatletter"` in `*.tex` files, thereby making the use of "@" in commands available. However, since the `*.tex` file doesn't have one of the following suffices: `sty cls clo dtx ltx`, the syntax highlighting will flag such use of @ as an error. To solve this:

```
:let b:tex_stylish = 1
:set ft=tex
```

Putting `"let g:tex_stylish=1"` into your `<.vimrc>` will make `<syntax/tex.vim>` always accept such use of @.

tex-cchar tex-cole tex-conceal

### Tex: Taking Advantage of Conceal Mode

If you have `'conceallevel'` set to 2 and if your encoding is utf-8, then a number of character sequences can be translated into appropriate utf-8 glyphs, including various accented characters, Greek characters in MathZones, and superscripts and subscripts in MathZones. Not all characters can be made into superscripts or subscripts; the constraint is due to what utf-8 supports. In fact, only a few characters are supported as subscripts.

One way to use this is to have vertically split windows (see `CTRL-W_v`); one with `'conceallevel'` at 0 and the other at 2; and both using `'scrollbind'`.

g:tex\_conceal

### Tex: Selective Conceal Mode

You may selectively use conceal mode by setting `g:tex_conceal` in your `<.vimrc>`. By default, `g:tex_conceal` is set to "admg" to enable concealment for the following sets of characters:



a = accents/ligatures  
b = bold and italic  
d = delimiters  
m = math symbols  
g = Greek  
s = superscripts/subscripts

By leaving one or more of these out, the associated conceal-character substitution will not be made.

[g:tex\\_isk](#)   [g:tex\\_stylish](#)

### Tex: Controlling iskeyword

Normally, LaTeX keywords support 0-9, a-z, A-z, and 192-255 only. Latex keywords don't support the underscore - except when in \*.sty files. The syntax highlighting script handles this with the following logic:

- \* If [g:tex\\_stylish](#) exists and is 1  
    then the file will be treated as a "sty" file, so the "\_" will be allowed as part of keywords (regardless of [g:tex\\_isk](#))
- \* Else if the file's suffix is sty, cls, clo, dtx, or ltx,  
    then the file will be treated as a "sty" file, so the "\_" will be allowed as part of keywords (regardless of [g:tex\\_isk](#))
- \* If [g:tex\\_isk](#) exists, then it will be used for the local **'iskeyword'**
- \* Else the local **'iskeyword'** will be set to 48-57,a-z,A-Z,192-255

[tex-supersub](#)   [g:tex\\_superscripts](#)   [g:tex\\_subscripts](#)

### Tex: Fine Subscript and Superscript Control

See [tex-conceal](#) for how to enable concealed character replacement.

See [g:tex\\_conceal](#) for selectively concealing accents, bold/italic, math, Greek, and superscripts/subscripts.

One may exert fine control over which superscripts and subscripts one wants syntax-based concealment for (see [:syn-cchar](#)). Since not all fonts support all characters, one may override the concealed-replacement lists; by default these lists are given by:

```
let g:tex_superscripts= "[0-9a-zA-W.,;+<>/()=]"
let g:tex_subscripts= "[0-9aehijklmnoprstuvx,+/(().)]"
```

For example, I use Luxi Mono Bold; it doesn't support subscript characters for "hklmpst", so I put

```
let g:tex_subscripts= "[0-9aeijoruvx,+/(().)]"
```

in ~/.vim/ftplugin/tex/tex.vim in order to avoid having inscrutable utf-8 glyphs appear.

[tex-matchcheck](#)   [g:tex\\_matchcheck](#)

### Tex: Match Check Control

Sometimes one actually wants mismatched parentheses, square braces, and or curly braces; for example, `\text{(1,10]}` is a range from but not including 1 to and including 10. This wish, of course, conflicts with the desire to provide delimiter mismatch detection. To accommodate these conflicting goals, `syntax/tex.vim` provides

```
g:tex_matchcheck = '[([]'
```

which is shown along with its default setting. So, if one doesn't want `[]` and `()` to be checked for mismatches, try using

```
let g:tex_matchcheck= '[{}]'
```

If you don't want matching to occur inside bold and italicized regions,

```
let g:tex_excludematcher= 1
```

will prevent the `texMatcher` group from being included in those regions.

## TF

`tf.vim` `ft-tf-syntax`

There is one option for the `tf` syntax highlighting.

For syncing, `minlines` defaults to 100. If you prefer another value, you can set `"tf_minlines"` to the value you desire. Example:

```
:let tf_minlines = your choice
```

## VIM

`vim.vim`

`ft-vim-syntax`

`g:vimsyn_minlines`

`g:vimsyn_maxlines`

There is a trade-off between more accurate syntax highlighting versus screen updating speed. To improve accuracy, you may wish to increase the `g:vimsyn_minlines` variable. The `g:vimsyn_maxlines` variable may be used to improve screen updating rates (see `:syn-sync` for more on this).

```
g:vimsyn_minlines : used to set synchronization minlines
g:vimsyn_maxlines : used to set synchronization maxlines
```

(`g:vim_minlines` and `g:vim_maxlines` are deprecated variants of these two options)

`g:vimsyn_embed`

The `g:vimsyn_embed` option allows users to select what, if any, types of embedded script highlighting they wish to have.

```
g:vimsyn_embed == 0 : don't support any embedded scripts
g:vimsyn_embed =~ 'l' : support embedded lua
g:vimsyn_embed =~ 'm' : support embedded mzscheme
g:vimsyn_embed =~ 'p' : support embedded perl
g:vimsyn_embed =~ 'P' : support embedded python
g:vimsyn_embed =~ 'r' : support embedded ruby
g:vimsyn_embed =~ 't' : support embedded tcl
```

By default, `g:vimsyn_embed` is a string supporting interpreters that your vim itself supports. Concatenate multiple characters to support multiple types of embedded interpreters; ie. `g:vimsyn_embed="mp"` supports embedded `mzscheme` and embedded `perl`.

`g:vimsyn_folding`

Some folding is now supported with syntax/vim.vim:

```
g:vimsyn_folding == 0 or doesn't exist: no syntax-based folding
g:vimsyn_folding =~ 'a' : augroups
g:vimsyn_folding =~ 'f' : fold functions
g:vimsyn_folding =~ 'l' : fold lua script
g:vimsyn_folding =~ 'm' : fold mzscheme script
g:vimsyn_folding =~ 'p' : fold perl script
g:vimsyn_folding =~ 'P' : fold python script
g:vimsyn_folding =~ 'r' : fold ruby script
g:vimsyn_folding =~ 't' : fold tcl script
```

Not all error highlighting that syntax/vim.vim does may be correct; Vim script is a difficult language to highlight correctly. A way to suppress error highlighting is to put the following line in your `vimrc` :

```
let g:vimsyn_noerror = 1
```

## WDL

`wdl.vim` `wdl-syntax`

The Workflow Description Language is a way to specify data processing workflows with a human-readable and writeable syntax. This is used a lot in bioinformatics. More info on the spec can be found here:

<https://github.com/openwdl/wdl>

## XF86CONFIG

`xf86conf.vim` `ft-xf86conf-syntax`

The syntax of XF86Config file differs in XFree86 v3.x and v4.x. Both variants are supported. Automatic detection is used, but is far from perfect. You may need to specify the version manually. Set the variable `xf86conf_xfree86_version` to 3 or 4 according to your XFree86 version in your `.vimrc`. Example:

```
:let xf86conf_xfree86_version=3
```

When using a mix of versions, set the `b:xf86conf_xfree86_version` variable.

**Note** that spaces and underscores in option names are not supported. Use "SyncOnGreen" instead of "\_\_s yn con gr\_e\_e\_n" if you want the option name highlighted.

## XML

`xml.vim` `ft-xml-syntax`

Xml namespaces are highlighted by default. This can be inhibited by setting a global variable:

```
:let g:xml_namespace_transparent=1
```

The xml syntax file provides syntax `folding` (see `xml-folding` `:syn-fold` ) between start and end tags. This can be turned on by

```
:let g:xml_syntax_folding = 1
:set foldmethod=syntax
```

**Note:** Syntax folding might slow down syntax highlighting significantly, especially for large files.

X Pixmaps (XPM)

`xpm.vim` `ft-xpm-syntax`

`xpm.vim` creates its syntax items dynamically based upon the contents of the XPM file. Thus if you make changes e.g. in the color specification strings, you have to source it again e.g. with `":set syn=xpm"`.

To copy a pixel with one of the colors, yank a "pixel" with `"yl"` and insert it somewhere else with `"P"`.

Do you want to draw with the mouse? Try the following:

```
:function! GetPixel()
: let c = getline(".")[col(".") - 1]
: echo c
: exe "noremap <LeftMouse> <LeftMouse>r" .. c
: exe "noremap <LeftDrag> <LeftMouse>r" .. c
:endifunction
:noremap <RightMouse> <LeftMouse>:call GetPixel()<CR>
:set guicursor=n:hor20 " to see the color beneath the cursor
```

This turns the right button into a pipette and the left button into a pen. It will work with XPM files that have one character per pixel only and you must not click outside of the pixel strings, but feel free to improve it.

It will look much better with a font in a quadratic cell size, e.g. for X:

```
:set guifont=-*-clean-medium-r-*-*8-*-*-*-*80-*
```

**YAML**

`yaml.vim` `ft-yaml-syntax`

`g:yaml_schema` `b:yaml_schema`

A YAML schema is a combination of a set of tags and a mechanism for resolving non-specific tags. For user this means that YAML parser may, depending on plain scalar contents, treat plain scalar (which can actually be only string and nothing else) as a value of the other type: null, boolean, floating-point, integer. ``g:yaml_schema`` option determines according to which schema values will be highlighted specially. Supported schemas are

Schema	Description
failsafe	No additional highlighting.
json	Supports JSON-style numbers, booleans and null.
core	Supports more number, boolean and null styles.
pyyaml	In addition to core schema supports highlighting timestamps, but there are some differences in what is recognized as numbers and many additional boolean values not present in core schema.

Default schema is ``core``.

**Note** that schemas are not actually limited to plain scalars, but this is the only difference between schemas defined in YAML specification and the only difference defined in the syntax file.

## ZSH

zsh.vim ft-zsh-syntax

The syntax script for zsh allows for syntax-based folding:

```
:let g:zsh_fold_enable = 1
```

---

## 6. Defining a syntax

:syn-define E410

Vim understands three types of syntax items:

### 1. Keyword

It can only contain keyword characters, according to the characters specified with `:syn-iskeyword` or the `'iskeyword'` option. It cannot contain other syntax items. It will only match with a complete word (there are no keyword characters before or after the match). The keyword "if" would match in "if(a=b)", but not in "ifdef x", because "(" is not a keyword character and "d" is.

### 2. Match

This is a match with a single regexp pattern.

### 3. Region

This starts at a match of the "start" regexp pattern and ends with a match with the "end" regexp pattern. Any other text can appear in between. A "skip" regexp pattern can be used to avoid matching the "end" pattern.

Several syntax ITEMS can be put into one syntax GROUP. For a syntax group you can give highlighting attributes. For example, you could have an item to define a `"/ * .. */` comment and another one that defines a `"// .."` comment, and put them both in the "Comment" group. You can then specify that a "Comment" will be in bold font and have a blue color. You are free to make one highlight group for one syntax item, or put all items into one group. This depends on how you want to specify your highlighting attributes. Putting each item in its own group results in having to specify the highlighting for a lot of groups.

**Note** that a syntax group and a highlight group are similar. For a highlight group you will have given highlight attributes. These attributes will be used for the syntax group with the same name.

In case more than one item matches at the same position, the one that was defined LAST wins. Thus you can override previously defined syntax items by using an item that matches the same text. But a keyword always goes before a match or region. And a keyword with matching case always goes before a keyword with ignoring case.

## PRIORITY

:syn-priority

When several syntax items may match, these rules are used:

1. When multiple Match or Region items start in the same position, the item defined last has priority.
2. A Keyword has priority over Match and Region items.
3. An item that starts in an earlier position has priority over items that start in later positions.

## DEFINING CASE

:syn-case E390

```
:sy[ntax] case [match | ignore]
```

This defines if the following ":syntax" commands will work with matching case, when using "match", or with ignoring case, when using "ignore". **Note** that any items before this are not affected, and all items until the next ":syntax case" command are affected.

```
:sy[ntax] case
```

Show either "syntax case match" or "syntax case ignore".

## DEFINING FOLDLEVEL

:syn-foldlevel

```
:sy[ntax] foldlevel start
```

```
:sy[ntax] foldlevel minimum
```

This defines how the foldlevel of a line is computed when using foldmethod=syntax (see [fold-syntax](#) and [:syn-fold](#)):

start:            Use level of item containing start of line.

minimum:         Use lowest local-minimum level of items on line.

The default is "start". Use "minimum" to search a line horizontally for the lowest level contained on the line that is followed by a higher level. This produces more natural folds when syntax items may close and open horizontally within a line.

```
:sy[ntax] foldlevel
```

Show the current foldlevel method, either "syntax foldlevel start" or "syntax foldlevel minimum".

{not meaningful when Vim was compiled without |+folding| feature}

## SPELL CHECKING

:syn-spell

```
:sy[ntax] spell toplevel
```

```
:sy[ntax] spell notoplevel
```

```
:sy[ntax] spell default
```

This defines where spell checking is to be done for text that is not in a syntax item:

toplevel:         Text is spell checked.

notoplevel:       Text is not spell checked.

default:            When there is a @Spell cluster no spell checking.

For text in syntax items use the @Spell and @NoSpell clusters `spell-syntax` . When there is no @Spell and no @NoSpell cluster then spell checking is done for "default" and "toplevel".

To activate spell checking the **'spell'** option must be set.

`:sy[ntax] spell`  
Show the current syntax spell checking method, either "syntax spell  
toplevel", "syntax spell notoplevel" or "syntax spell default".

### SYNTAX ISKEYWORD SETTING

`:syn-iskeyword`

`:sy[ntax] iskeyword [clear | {option}]`  
This defines the keyword characters. It's like the **'iskeyword'** option for but only applies to syntax highlighting.

`clear:`            Syntax specific iskeyword setting is disabled and the  
                    buffer-local **'iskeyword'** setting is used.  
`{option}`          Set the syntax **'iskeyword'** option to a new value.

Example:

```
:syntax iskeyword @,48-57,192-255,$, _
```

This would set the syntax specific iskeyword option to include all alphabetic characters, plus the numeric characters, all accented characters and also includes the "\_" and the "\$".

If no argument is given, the current value will be output.

Setting this option influences what `/\k` matches in syntax patterns and also determines where `:syn-keyword` will be checked for a new match.

It is recommended when writing syntax files, to use this command to set the correct value for the specific syntax language and not change the **'iskeyword'** option.

### DEFINING KEYWORDS

`:syn-keyword`

`:sy[ntax] keyword {group-name} [{options}] {keyword} .. [{options}]`

This defines a number of keywords.

`{group-name}`      Is a syntax group name such as "Comment".  
`[{options}]`       See `:syn-arguments` below.  
`{keyword} ..`      Is a list of keywords which are part of this group.

Example:

```
:syntax keyword Type int long char
```

The `{options}` can be given anywhere in the line. They will apply to

all keywords given, also for options that come after a keyword.

These examples do exactly the same:

```
:syntax keyword Type contained int long char
:syntax keyword Type int long contained char
:syntax keyword Type int long char contained
```

E789 E890

When you have a keyword with an optional tail, like Ex commands in Vim, you can put the optional characters inside [], to define all the variations at once:

```
:syntax keyword vimCommand ab[breviate] n[ext]
```

Don't forget that a keyword can only be recognized if all the characters are included in the **'iskeyword'** option. If one character isn't, the keyword will never be recognized.

Multi-byte characters can also be used. These do not have to be in **'iskeyword'**.

See [:syn-iskeyword](#) for defining syntax specific iskeyword settings.

A keyword always has higher priority than a match or region, the keyword is used if more than one item matches. Keywords do not nest and a keyword can't contain anything else.

**Note** that when you have a keyword that is the same as an option (even one that isn't allowed here), you can not use it. Use a match instead.

The maximum length of a keyword is 80 characters.

The same keyword can be defined multiple times, when its containment differs. For example, you can define the keyword once not contained and use one highlight group, and once contained, and use a different highlight group. Example:

```
:syn keyword vimCommand tag
:syn keyword vimSetting contained tag
```

When finding "tag" outside of any syntax item, the "vimCommand" highlight group is used. When finding "tag" in a syntax item that contains "vimSetting", the "vimSetting" group is used.

## DEFINING MATCHES

[:syn-match](#)

```
:sy[ntax] match {group-name} [{options}]
 [excludenl]
 [keepend]
 {pattern}
 [{options}]
```

This defines one match.

```
{group-name}
[options]
[excludenl]
```

A syntax group name such as "Comment".

See [:syn-arguments](#) below.

Don't make a pattern with the end-of-line "\$" extend a containing match or region. Must be given before the pattern. [:syn-excludenl](#)



keepend Don't allow contained matches to go past a match with the end pattern. See [:syn-keepend](#) .

`{pattern}` The search pattern that defines the match. See [:syn-pattern](#) below. **Note** that the pattern may match more than one line, which makes the match depend on where Vim starts searching for the pattern. You need to make sure syncing takes care of this.

Example (match a character constant):  
`:syntax match Character /\.'/'hs=s+1,he=e-1`

**DEFINING REGIONS**                    [:syn-region](#)    [:syn-start](#)    [:syn-skip](#)    [:syn-end](#)  
E398            E399

```
:sy[ntax] region {group-name} [{options}]
 [matchgroup={group-name}]
 [keepend]
 [extend]
 [excludenl]
 start={start-pattern} ..
 [skip={skip-pattern}]
 end={end-pattern} ..
 [{options}]
```

This defines one region. It may span several lines.

`{group-name}`                    A syntax group name such as "Comment".  
`[{options}]`                    See [:syn-arguments](#) below.  
`[matchgroup={group-name}]`    The syntax group to use for the following start or end pattern matches only. Not used for the text in between the matched start and end patterns. Use NONE to reset to not using a different group for the start or end match. See [:syn-matchgroup](#) .

keepend Don't allow contained matches to go past a match with the end pattern. See [:syn-keepend](#) .

extend Override a "keepend" for an item this region is contained in. See [:syn-extend](#) .

excludenl Don't make a pattern with the end-of-line "\$" extend a containing match or item. Only useful for end patterns. Must be given before the patterns it applies to. [:syn-excludenl](#)

start={start-pattern} The search pattern that defines the start of the region. See [:syn-pattern](#) below.

skip={skip-pattern} The search pattern that defines text inside the region where not to look for the end pattern. See [:syn-pattern](#) below.

end={end-pattern} The search pattern that defines the end of the region. See [:syn-pattern](#) below.

Example:

```
:syntax region String start="+\" skip=+\\\" end="+\"
```

The start/skip/end patterns and the options can be given in any order. There can be zero or one skip pattern. There must be one or more start and end patterns. This means that you can omit the skip pattern, but you must give at least one start and one end pattern. It is allowed to have white space before and after the equal sign (although it mostly looks better without white space).

When more than one start pattern is given, a match with one of these is sufficient. This means there is an OR relation between the start patterns. The last one that matches is used. The same is true for the end patterns.

The search for the end pattern starts right after the start pattern. Offsets are not used for this. This implies that the match for the end pattern will never overlap with the start pattern.

The skip and end pattern can match across line breaks, but since the search for the pattern can start in any line it often does not do what you want. The skip pattern doesn't avoid a match of an end pattern in the next line. Use single-line patterns to avoid trouble.

**Note:** The decision to start a region is only based on a matching start pattern. There is no check for a matching end pattern. This does NOT work:

```
:syn region First start="(\" end=":"
:syn region Second start="(\" end=";"
```

The Second always matches before the First (last defined pattern has higher priority). The Second region then continues until the next ';', no matter if there is a ':' before it. Using a match does work:

```
:syn match First \"(_\\.\\{-}:\"
:syn match Second \"(_\\.\\{-};\"
```

This pattern matches any character or line break with \"\_\\.\" and repeats that with \"\\{-}\" (repeat as few as possible).

**:syn-keepend**

By default, a contained match can obscure a match for the end pattern. This is useful for nesting. For example, a region that starts with \"{\" and ends with \"}\", can contain another region. An encountered \"}\" will then end the contained region, but not the outer region:

```
{ starts outer \"{ }\" region
 { starts contained \"{ }\" region
 } ends contained \"{ }\" region
} ends outer \"{ }\" region
```

If you don't want this, the "keepend" argument will make the matching of an end pattern of the outer region also end any contained item. This makes it impossible to nest the same region, but allows for contained items to highlight parts of the end pattern, without causing that to skip the match with the end pattern. Example:

```
:syn match vimComment +\"[^\"]\+\"$+
:syn region vimCommand start="set" end=\"$\" contains=vimComment keepend
```

The "keepend" makes the vimCommand always end at the end of the line, even though the contained vimComment includes a match with the <EOL>.

When "keepend" is not used, a match with an end pattern is retried after each contained match. When "keepend" is included, the first encountered match with an end pattern is used, truncating any contained matches.

#### :syn-extend

The "keepend" behavior can be changed by using the "extend" argument. When an item with "extend" is contained in an item that uses "keepend", the "keepend" is ignored and the containing region will be extended.

This can be used to have some contained items extend a region while others don't. Example:

```
:syn region htmlRef start=+<a>+ end=++ keepend contains=htmlItem,htmlScript
:syn match htmlItem +<[^>]*>+ contained
:syn region htmlScript start=+<script+ end=+</script[^>]*>+ contained extend
```

Here the htmlItem item does not make the htmlRef item continue further, it is only used to highlight the <> items. The htmlScript item does extend the htmlRef item.

Another example:

```
:syn region xmlFold start="<a>" end="" fold transparent keepend extend
```

This defines a region with "keepend", so that its end cannot be changed by contained items, like when the "</a>" is matched to highlight it differently. But when the xmlFold region is nested (it includes itself), the "extend" applies, so that the "</a>" of a nested region only ends that region, and not the one it is contained in.

#### :syn-excludenl

When a pattern for a match or end pattern of a region includes a '\$' to match the end-of-line, it will make a region item that it is contained in continue on the next line. For example, a match with "\\\$" (backslash at the end of the line) can make a region continue that would normally stop at the end of the line. This is the default behavior. If this is not wanted, there are two ways to avoid it:

1. Use "keepend" for the containing item. This will keep all contained matches from extending the match or region. It can be used when all contained items must not extend the containing item.
2. Use "excludenl" in the contained item. This will keep that match from extending the containing match or region. It can be used if only some contained items must not extend the containing item. "excludenl" must be given before the pattern it applies to.

#### :syn-matchgroup

"matchgroup" can be used to highlight the start and/or end pattern differently than the body of the region. Example:

```
:syntax region String matchgroup=Quote start=+"+ skip=+\\\"+ end=+"+
```

This will highlight the quotes with the "Quote" group, and the text in between with the "String" group.

The "matchgroup" is used for all start and end patterns that follow, until the next "matchgroup". Use "matchgroup=NONE" to go back to not using a matchgroup.

In a start or end pattern that is highlighted with "matchgroup" the contained items of the region are not used. This can be used to avoid that a contained item matches in the start or end pattern match. When using "transparent", this does not apply to a start or end pattern match that is highlighted with "matchgroup".

Here is an example, which highlights three levels of parentheses in different colors:

```
:sy region par1 matchgroup=par1 start=/(/ end=)/ contains=par2
:sy region par2 matchgroup=par2 start=/(/ end=)/ contains=par3 contained
:sy region par3 matchgroup=par3 start=/(/ end=)/ contains=par1 contained
:hi par1 ctermfg=red guifg=red
:hi par2 ctermfg=blue guifg=blue
:hi par3 ctermfg=darkgreen guifg=darkgreen
```

E849

The maximum number of syntax groups is 19999.

=====  
7. :syntax arguments

:syn-arguments

The :syntax commands that define syntax items take a number of arguments. The common ones are explained here. The arguments may be given in any order and may be mixed with patterns.

Not all commands accept all arguments. This table shows which arguments can not be used for all commands:

E395

	contains	oneline	fold	display	extend	concealends
:syntax keyword	-	-	-	-	-	-
:syntax match	yes	-	yes	yes	yes	-
:syntax region	yes	yes	yes	yes	yes	yes

These arguments can be used for all three commands:

- conceal
- cchar
- contained
- containedin
- nextgroup
- transparent
- skipwhite
- skipnl
- skipempty

conceal

conceal :syn-conceal

When the "conceal" argument is given, the item is marked as concealable. Whether or not it is actually concealed depends on the value of the '**conceallevel**' option. The '**concealcursor**' option is used to decide whether concealable items in the current line are displayed unconcealed to be able to edit the line.

Another way to conceal text is with `matchadd()` .

concealends

:syn-concealends

When the "concealends" argument is given, the start and end matches of the region, but not the contents of the region, are marked as concealable. Whether or not they are actually concealed depends on the setting on the '**conceallevel**' option. The ends of a region can only be concealed separately in this way when they have their own highlighting via "matchgroup"

cchar :syn-cchar  
E844

The "cchar" argument defines the character shown in place of the item when it is concealed (setting "cchar" only makes sense when the conceal argument is given.) If "cchar" is not set then the default conceal character defined in the '**listchars**' option is used. The character cannot be a control character such as Tab. Example:

```
:syntax match Entity "&" conceal cchar=&
See hl-Conceal for highlighting.
```

contained :syn-contained

When the "contained" argument is given, this item will not be recognized at the top level, but only when it is mentioned in the "contains" field of another match. Example:

```
:syntax keyword Todo TODO contained
:syntax match Comment "//.*" contains=Todo
```

display :syn-display

If the "display" argument is given, this item will be skipped when the detected highlighting will not be displayed. This will speed up highlighting, by skipping this item when only finding the syntax state for the text that is to be displayed.

Generally, you can use "display" for match and region items that meet these conditions:

- The item does not continue past the end of a line. Example for C: A region for a "/\*" comment can't contain "display", because it continues on the next line.
- The item does not contain items that continue past the end of the line or make it continue on the next line.
- The item does not change the size of any item it is contained in. Example for C: A match with "\\\$" in a preprocessor match can't have "display", because it may make that preprocessor match shorter.
- The item does not allow other items to match that didn't match otherwise, and that item may extend the match too far. Example for C: A match for a "/" comment can't use "display", because a "/\*" inside that comment would match then and start a comment which extends past the end of the line.

Examples, for the C language, where "display" can be used:

- match with a number
- match with a label

transparent :syn-transparent

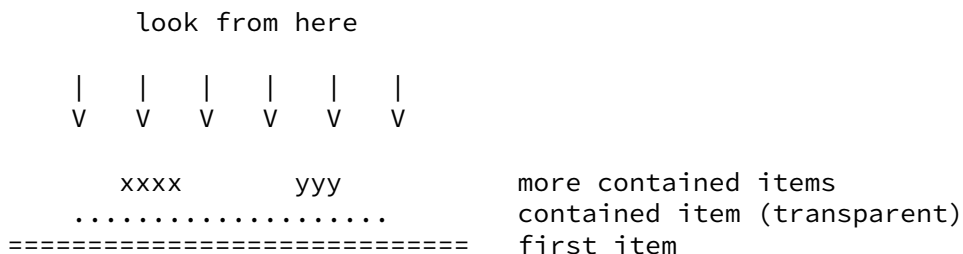
If the "transparent" argument is given, this item will not be highlighted itself, but will take the highlighting of the item it is contained in. This is useful for syntax items that don't need any highlighting but are used only to skip over a part of the text.

The "contains=" argument is also inherited from the item it is contained in, unless a "contains" argument is given for the transparent item itself. To avoid that unwanted items are contained, use "contains=NONE". Example, which highlights words in strings, but makes an exception for "vim":

```
:syn match myString /^[^']*'/ contains=myWord,myVim
:syn match myWord /\<[a-z]*\>/ contained
:syn match myVim /\<vim\>/ transparent contained contains=NONE
:hi link myString String
:hi link myWord Comment
```

Since the "myVim" match comes after "myWord" it is the preferred match (last match in the same position overrules an earlier one). The "transparent" argument makes the "myVim" match use the same highlighting as "myString". But it does not contain anything. If the "contains=NONE" argument would be left out, then "myVim" would use the contains argument from myString and allow "myWord" to be contained, which will be highlighted as a Comment. This happens because a contained match doesn't match inside itself in the same position, thus the "myVim" match doesn't overrule the "myWord" match here.

When you look at the colored text, it is like looking at layers of contained items. The contained item is on top of the item it is contained in, thus you see the contained item. When a contained item is transparent, you can look through, thus you see the item it is contained in. In a picture:



The 'x', 'y' and '=' represent a highlighted syntax item. The '.' represent a transparent group.

What you see is:

```
=====xxxx=====yyy=====
```

Thus you look through the transparent "....".

oneline :syn-oneline

The "oneline" argument indicates that the region does not cross a line boundary. It must match completely in the current line. However, when the region has a contained item that does cross a line boundary, it continues on the next line anyway. A contained item can be used to recognize a line

continuation pattern. But the "end" pattern must still match in the first line, otherwise the region doesn't even start.

When the start pattern includes a "\n" to match an end-of-line, the end pattern must be found in the same line as where the start pattern ends. The end pattern may also include an end-of-line. Thus the "oneline" argument means that the end of the start pattern and the start of the end pattern must be within one line. This can't be changed by a skip pattern that matches a line break.

fold :syn-fold

The "fold" argument makes the fold level increase by one for this item.

Example:

```
:syn region myFold start="{ " end="}" transparent fold
:syn sync fromstart
:set foldmethod=syntax
```

This will make each {} block form one fold.

The fold will start on the line where the item starts, and end where the item ends. If the start and end are within the same line, there is no fold.

The '**foldnestmax**' option limits the nesting of syntax folds.

See [:syn-foldlevel](#) to control how the foldlevel of a line is computed from its syntax items.

{not available when Vim was compiled without |+folding| feature}

:syn-contains E405 E406 E407 E408 E409  
contains={group-name},..

The "contains" argument is followed by a list of syntax group names. These groups will be allowed to begin inside the item (they may extend past the containing group's end). This allows for recursive nesting of matches and regions. If there is no "contains" argument, no groups will be contained in this item. The group names do not need to be defined before they can be used here.

contains=ALL

If the only item in the contains list is "ALL", then all groups will be accepted inside the item.

contains=ALLBUT,{group-name},..

If the first item in the contains list is "ALLBUT", then all groups will be accepted inside the item, except the ones that are listed. Example:

```
:syntax region Block start="{ " end="}" ... contains=ALLBUT,Function
```

contains=TOP

If the first item in the contains list is "TOP", then all groups will be accepted that don't have the "contained" argument.

contains=TOP,{group-name},..

Like "TOP", but excluding the groups that are listed.

contains=CONTAINED

If the first item in the contains list is "CONTAINED", then all groups will be accepted that have the "contained" argument.

contains=CONTAINED,{group-name},..

Like "CONTAINED", but excluding the groups that are listed.

The {group-name} in the "contains" list can be a pattern. All group names that match the pattern will be included (or excluded, if "ALLBUT" is used). The pattern cannot contain white space or a ','. Example:

```
... contains=Comment.*,Keyw[0-3]
```

The matching will be done at moment the syntax command is executed. Groups that are defined later will not be matched. Also, if the current syntax command defines a new group, it is not matched. Be careful: When putting syntax commands in a file you can't rely on groups NOT being defined, because the file may have been sourced before, and ":syn clear" doesn't remove the group names.

The contained groups will also match in the start and end patterns of a region. If this is not wanted, the "matchgroup" argument can be used :syn-matchgroup . The "ms=" and "me=" offsets can be used to change the region where contained items do match. Note that this may also limit the area that is highlighted

containedin={group-name}...

:syn-containedin

The "containedin" argument is followed by a list of syntax group names. The item will be allowed to begin inside these groups. This works as if the containing item has a "contains=" argument that includes this item.

The {group-name}... can be used just like for "contains", as explained above.

This is useful when adding a syntax item afterwards. An item can be told to be included inside an already existing item, without changing the definition of that item. For example, to highlight a word in a C comment after loading the C syntax:

```
:syn keyword myword HELP containedin=cComment contained
```

Note that "contained" is also used, to avoid that the item matches at the top level.

Matches for "containedin" are added to the other places where the item can appear. A "contains" argument may also be added as usual. Don't forget that keywords never contain another item, thus adding them to "containedin" won't work.

nextgroup={group-name},..

:syn-nextgroup

The "nextgroup" argument is followed by a list of syntax group names, separated by commas (just like with "contains", so you can also use patterns).



If the "nextgroup" argument is given, the mentioned syntax groups will be tried for a match, after the match or region ends. If none of the groups have a match, highlighting continues normally. If there is a match, this group will be used, even when it is not mentioned in the "contains" field of the current group. This is like giving the mentioned group priority over all other groups. Example:

```
:syntax match ccFoobar "Foo.\{-}Bar" contains=ccFoo
:syntax match ccFoo "Foo" contained nextgroup=ccFiller
:syntax region ccFiller start="." matchgroup=ccBar end="Bar" contained
```

This will highlight "Foo" and "Bar" differently, and only when there is a "Bar" after "Foo". In the text line below, "f" shows where ccFoo is used for highlighting, and "bbb" where ccBar is used.

```
Foo asdfasd Bar asdf Foo asdf Bar asdf
fff bbb fff bbb
```

**Note** the use of ".\{-}" to skip as little as possible until the next Bar. when ".\*" would be used, the "asdf" in between "Bar" and "Foo" would be highlighted according to the "ccFoobar" group, because the ccFoobar match would include the first "Foo" and the last "Bar" in the line (see [pattern](#) ).

```
skipwhite :syn-skipwhite
skipnl :syn-skipnl
skipempty :syn-skipempty
```

These arguments are only used in combination with "nextgroup". They can be used to allow the next group to match after skipping some text:

```
skipwhite skip over space and tab characters
skipnl skip over the end of a line
skipempty skip over empty lines (implies a "skipnl")
```

When "skipwhite" is present, the white space is only skipped if there is no next group that matches the white space.

When "skipnl" is present, the match with nextgroup may be found in the next line. This only happens when the current item ends at the end of the current line! When "skipnl" is not present, the nextgroup will only be found after the current item in the same line.

When skipping text while looking for a next group, the matches for other groups are ignored. Only when no next group matches, other items are tried for a match again. This means that matching a next group and skipping white space and <EOL>s has a higher priority than other items.

Example:

```
:syn match ifstart "\<if.*" nextgroup=ifline skipwhite skipempty
:syn match ifline "[^ \t].*" nextgroup=ifline skipwhite skipempty contained
:syn match ifline "endif" contained
```

**Note** that the "[^ \t].\*" match matches all non-white text. Thus it would also match "endif". Therefore the "endif" match is put last, so that it takes precedence.

Note that this example doesn't work for nested "if"s. You need to add "contains" arguments to make that work (omitted for simplicity of the example).

## IMPLICIT CONCEAL

:syn-conceal-implicit

```
:sy[ntax] conceal [on|off]
```

This defines if the following ":syntax" commands will define keywords, matches or regions with the "conceal" flag set. After ":syn conceal on", all subsequent ":syn keyword", ":syn match" or ":syn region" defined will have the "conceal" flag set implicitly. ":syn conceal off" returns to the normal state where the "conceal" flag must be given explicitly.

```
:sy[ntax] conceal
```

Show either "syntax conceal on" or "syntax conceal off".

---

## 8. Syntax patterns

:syn-pattern E401 E402

In the syntax commands, a pattern must be surrounded by two identical characters. This is like it works for the ":s" command. The most common to use is the double quote. But if the pattern contains a double quote, you can use another character that is not used in the pattern. Examples:

```
:syntax region Comment start="/*" end="*/"
:syntax region String start="+"" end="+"" skip="+\\\""
```

See [pattern](#) for the explanation of what a pattern is. Syntax patterns are always interpreted like the 'magic' option is set, no matter what the actual value of 'magic' is. And the patterns are interpreted like the 'l' flag is not included in 'coptions'. This was done to make syntax files portable and independent of 'compatible' and 'magic' settings.

Try to avoid patterns that can match an empty string, such as "[a-z]\*". This slows down the highlighting a lot, because it matches everywhere.

:syn-pattern-offset

The pattern can be followed by a character offset. This can be used to change the highlighted part, and to change the text area included in the match or region (which only matters when trying to match other items). Both are relative to the matched pattern. The character offset for a skip pattern can be used to tell where to continue looking for an end pattern.

The offset takes the form of "{what}={offset}"

The {what} can be one of seven strings:

ms	Match Start	offset for the start of the matched text
me	Match End	offset for the end of the matched text
hs	Highlight Start	offset for where the highlighting starts
he	Highlight End	offset for where the highlighting ends
rs	Region Start	offset for where the body of a region starts
re	Region End	offset for where the body of a region ends
lc	Leading Context	offset past "leading context" of pattern

The `{offset}` can be:

s start of the matched pattern  
s+{nr} start of the matched pattern plus {nr} chars to the right  
s-{nr} start of the matched pattern plus {nr} chars to the left  
e end of the matched pattern  
e+{nr} end of the matched pattern plus {nr} chars to the right  
e-{nr} end of the matched pattern plus {nr} chars to the left  
{nr} (for "lc" only): start matching {nr} chars right of the start

Examples: "ms=s+1", "hs=e-2", "lc=3".

Although all offsets are accepted after any pattern, they are not always meaningful. This table shows which offsets are actually used:

	ms	me	hs	he	rs	re	lc
match item	yes	yes	yes	yes	-	-	yes
region item start	yes	-	yes	-	yes	-	yes
region item skip	-	yes	-	-	-	-	yes
region item end	-	yes	-	yes	-	yes	yes

Offsets can be concatenated, with a ',' in between. Example:

```
:syn match String /"[^"]*" /hs=s+1,he=e-1
```

```
some "string" text
 ^^^^^^^ highlighted
```

#### Notes:

- There must be no white space between the pattern and the character offset(s).
- The highlighted area will never be outside of the matched text.
- A negative offset for an end pattern may not always work, because the end pattern may be detected when the highlighting should already have stopped.
- Before Vim 7.2 the offsets were counted in bytes instead of characters. This didn't work well for multibyte characters, so it was changed with the Vim 7.2 release.
- The start of a match cannot be in a line other than where the pattern matched. This doesn't work: "a\n"ms=e. You can make the highlighting start in another line, this does work: "a\n"hs=e.

Example (match a comment but don't highlight the /\* and \*/):

```
:syntax region Comment start="/*"hs=e+1 end="*/"he=s-1
```

```
/* this is a comment */
 ^^^^^^^^^^^^^^^^^^^^^ highlighted
```

A more complicated Example:

```
:syn region Exa matchgroup=Foo start="foo"hs=s+2,rs=e+2 matchgroup=Bar end="bar"me=e-1,he
```

```
abcfoostringbarabc
 mmmmmmmmmmmmm match
 sssrrreee highlight start/region/end ("Foo", "Exa" and "Bar")
```

Leading context `:syn-lc` `:syn-leading` `:syn-context`

**Note:** This is an obsolete feature, only included for backwards compatibility with previous Vim versions. It's now recommended to use the `/\@<=` construct in the pattern. You can also often use `/\zs` .

The "lc" offset specifies leading context -- a part of the pattern that must be present, but is not considered part of the match. An offset of "lc=n" will cause Vim to step back n columns before attempting the pattern match, allowing characters which have already been matched in previous patterns to also be used as leading context for this match. This can be used, for instance, to specify that an "escaping" character must not precede the match:

```
:syn match ZNoBackslash "[^\]z"ms=s+1
:syn match WNoBackslash "[^\]w"lc=1
:syn match Underline "_\+"
```

```

 ---ZZZZ ---WWWWW
 ^^^ ^^^ matches Underline
 ^ ^ matches ZNoBackslash
 ^^^^ matches WNoBackslash

```

The "ms" offset is automatically set to the same value as the "lc" offset, unless you set "ms" explicitly.

### Multi-line patterns

`:syn-multi-line`

The patterns can include "\n" to match an end-of-line. Mostly this works as expected, but there are a few exceptions.

When using a start pattern with an offset, the start of the match is not allowed to start in a following line. The highlighting can start in a following line though. Using the "\zs" item also requires that the start of the match doesn't move to another line.

The skip pattern can include the "\n", but the search for an end pattern will continue in the first character of the next line, also when that character is matched by the skip pattern. This is because redrawing may start in any line halfway a region and there is no check if the skip pattern started in a previous line. For example, if the skip pattern is "a\nb" and an end pattern is "b", the end pattern does match in the second line of this:

```

 x x a
 b x x

```

Generally this means that the skip pattern should not match any characters after the "\n".

### External matches

`:syn-ext-match`

These extra regular expression items are available in region patterns:

```

\z(\) \z(\z(\) E50 E52 E879
 Marks the sub-expression as "external", meaning that it can be
 accessed from another pattern match. Currently only usable in

```

defining a syntax region start pattern.

```
\z1 ... \z9 /\z1 /\z2 /\z3 /\z4 /\z5
 /\z6 /\z7 /\z8 /\z9 E66 E67
Matches the same string that was matched by the corresponding
sub-expression in a previous start pattern match.
```

Sometimes the start and end patterns of a region need to share a common sub-expression. A common example is the "here" document in Perl and many Unix shells. This effect can be achieved with the "\z" special regular expression items, which marks a sub-expression as "external", in the sense that it can be referenced from outside the pattern in which it is defined. The here-document example, for instance, can be done like this:

```
:syn region hereDoc start="<<\z(\I|i*)" end="^\z1$"
```

As can be seen here, the \z actually does double duty. In the start pattern, it marks the "\(\I|i\*)" sub-expression as external; in the end pattern, it changes the \z1 back-reference into an external reference referring to the first external sub-expression in the start pattern. External references can also be used in skip patterns:

```
:syn region foo start="start \z(\I|i*)" skip="not end \z1" end="end \z1"
```

**Note** that normal and external sub-expressions are completely orthogonal and indexed separately; for instance, if the pattern "\z(..)\(..\)" is applied to the string "aabb", then \1 will refer to "bb" and \z1 will refer to "aa". **Note** also that external sub-expressions cannot be accessed as back-references within the same pattern like normal sub-expressions. If you want to use one sub-expression as both a normal and an external sub-expression, you can nest the two, as in "\(\z(..)\)".

**Note** that only matches within a single line can be used. Multi-line matches cannot be referred to.

```
=====
9. Syntax clusters :syn-cluster E400
```

```
:sy[ntax] cluster {cluster-name} [contains={group-name}..]
 [add={group-name}..]
 [remove={group-name}..]
```

This command allows you to cluster a list of syntax groups together under a single name.

```
contains={group-name}..
 The cluster is set to the specified list of groups.
add={group-name}..
 The specified groups are added to the cluster.
remove={group-name}..
 The specified groups are removed from the cluster.
```

A cluster so defined may be referred to in a contains=.., containedin=.., nextgroup=.., add=.. or remove=.. list with a "@" prefix. You can also use this notation to implicitly declare a cluster before specifying its contents.

Example:

```
:syntax match Thing "# [^#]\+ #" contains=@ThingMembers
:syntax cluster ThingMembers contains=ThingMember1,ThingMember2
```

As the previous example suggests, modifications to a cluster are effectively retroactive; the membership of the cluster is checked at the last minute, so to speak:

```
:syntax keyword A aaa
:syntax keyword B bbb
:syntax cluster AandB contains=A
:syntax match Stuff "(aaa bbb)" contains=@AandB
:syntax cluster AandB add=B " now both keywords are matched in Stuff
```

This also has implications for nested clusters:

```
:syntax keyword A aaa
:syntax keyword B bbb
:syntax cluster SmallGroup contains=B
:syntax cluster BigGroup contains=A,@SmallGroup
:syntax match Stuff "(aaa bbb)" contains=@BigGroup
:syntax cluster BigGroup remove=B " no effect, since B isn't in BigGroup
:syntax cluster SmallGroup remove=B " now bbb isn't matched within Stuff
```

E848

The maximum number of clusters is 9767.

---

## 10. Including syntax files

:syn-include E397

It is often useful for one language's syntax file to include a syntax file for a related language. Depending on the exact relationship, this can be done in two different ways:

- If top-level syntax items in the included syntax file are to be allowed at the top level in the including syntax, you can simply use the `:runtime` command:

```
" In cpp.vim:
:runtime! syntax/c.vim
:unlet b:current_syntax
```

- If top-level syntax items in the included syntax file are to be contained within a region in the including syntax, you can use the `:syntax include` command:

```
:sy[ntax] include [@{grouplist-name}] {file-name}
```

All syntax items declared in the included file will have the "contained" flag added. In addition, if a group list is specified, all top-level syntax items in the included file will be added to that list.

```
" In perl.vim:
:syntax include @Pod <sfile>:p:h/pod.vim
:syntax region perlPOD start="^=head" end="^=cut" contains=@Pod
```

When `{file-name}` is an absolute path (starts with `"/"`, `"c:"`, `"$VAR"` or `"<sfile>"`) that file is sourced. When it is a relative path (e.g., `"syntax/pod.vim"`) the file is searched for in `'runtimepath'`. All matching files are loaded. Using a relative path is recommended, because it allows a user to replace the included file with their own version, without replacing the file that does the `":syn include"`.

E847

The maximum number of includes is 999.

---

## 11. Synchronizing

`:syn-sync` E403 E404

Vim wants to be able to start redrawing in any position in the document. To make this possible it needs to know the syntax state at the position where redrawing starts.

```
:sy[ntax] sync [ccomment [group-name] | minlines={N} | ...]
```

There are four ways to synchronize:

1. Always parse from the start of the file.  
`:syn-sync-first`
2. Based on C-style comments. Vim understands how C-comments work and can figure out if the current line starts inside or outside a comment.  
`:syn-sync-second`
3. Jumping back a certain number of lines and start parsing there.  
`:syn-sync-third`
4. Searching backwards in the text for a pattern to sync on.  
`:syn-sync-fourth`

`:syn-sync-maxlines` `:syn-sync-minlines`

For the last three methods, the line range where the parsing can start is limited by `"minlines"` and `"maxlines"`.

If the `"minlines={N}"` argument is given, the parsing always starts at least that many lines backwards. This can be used if the parsing may take a few lines before it's correct, or when it's not possible to use syncing.

If the `"maxlines={N}"` argument is given, the number of lines that are searched for a comment or syncing pattern is restricted to N lines backwards (after adding `"minlines"`). This is useful if you have few things to sync on and a slow machine. Example:

```
:syntax sync maxlines=500 ccomment
```

`:syn-sync-linebreaks`

When using a pattern that matches multiple lines, a change in one line may cause a pattern to no longer match in a previous line. This means has to start above where the change was made. How many lines can be specified with the `"linebreaks"` argument. For example, when a pattern may include one line break use this:

```
:syntax sync linebreaks=1
```

The result is that redrawing always starts at least one line before where a

change was made. The default value for "linebreaks" is zero. Usually the value for "minlines" is bigger than "linebreaks".

First syncing method: `:syn-sync-first`

`:syntax sync fromstart`

The file will be parsed from the start. This makes syntax highlighting accurate, but can be slow for long files. Vim caches previously parsed text, so that it's only slow when parsing the text for the first time. However, when making changes some part of the text needs to be parsed again (worst case: to the end of the file).

Using "fromstart" is equivalent to using "minlines" with a very large number.

Second syncing method: `:syn-sync-second` `:syn-sync-ccomment`

For the second method, only the "ccomment" argument needs to be given.

Example:

`:syntax sync ccomment`

When Vim finds that the line where displaying starts is inside a C-style comment, the last region syntax item with the group-name "Comment" will be used. This requires that there is a region with the group-name "Comment"! An alternate group name can be specified, for example:

`:syntax sync ccomment javaComment`

This means that the last item specified with "syn region javaComment" will be used for the detected C comment region. This only works properly if that region does have a start pattern "\/\*" and an end pattern "\*\/\*".

The "maxlines" argument can be used to restrict the search to a number of lines. The "minlines" argument can be used to at least start a number of lines back (e.g., for when there is some construct that only takes a few lines, but it hard to sync on).

**Note:** Syncing on a C comment doesn't work properly when strings are used that cross a line and contain a "\*/". Since letting strings cross a line is a bad programming habit (many compilers give a warning message), and the chance of a "\*/" appearing inside a comment is very small, this restriction is hardly ever noticed.

Third syncing method: `:syn-sync-third`

For the third method, only the "minlines={N}" argument needs to be given. Vim will subtract {N} from the line number and start parsing there. This means {N} extra lines need to be parsed, which makes this method a bit slower.

Example:

`:syntax sync minlines=50`

"lines" is equivalent to "minlines" (used by older versions).



Fourth syncing method:

`:syn-sync-fourth`

The idea is to synchronize on the end of a few specific regions, called a sync pattern. Only regions can cross lines, so when we find the end of some region, we might be able to know in which syntax item we are. The search starts in the line just above the one where redrawing starts. From there the search continues backwards in the file.

This works just like the non-syncing syntax items. You can use contained matches, nextgroup, etc. But there are a few differences:

- Keywords cannot be used.
- The syntax items with the "sync" keyword form a completely separated group of syntax items. You can't mix syncing groups and non-syncing groups.
- The matching works backwards in the buffer (line by line), instead of forwards.
- A line continuation pattern can be given. It is used to decide which group of lines need to be searched like they were one line. This means that the search for a match with the specified items starts in the first of the consecutive lines that contain the continuation pattern.
- When using "nextgroup" or "contains", this only works within one line (or group of continued lines).
- When using a region, it must start and end in the same line (or group of continued lines). Otherwise the end is assumed to be at the end of the line (or group of continued lines).
- When a match with a sync pattern is found, the rest of the line (or group of continued lines) is searched for another match. The last match is used. This is used when a line can contain both the start and the end of a region (e.g., in a C-comment like `/* this */`, the last `*/` is used).

There are two ways how a match with a sync pattern can be used:

1. Parsing for highlighting starts where redrawing starts (and where the search for the sync pattern started). The syntax group that is expected to be valid there must be specified. This works well when the regions that cross lines cannot contain other regions.
2. Parsing for highlighting continues just after the match. The syntax group that is expected to be present just after the match must be specified. This can be used when the previous method doesn't work well. It's much slower, because more text needs to be parsed.

Both types of sync patterns can be used at the same time.

Besides the sync patterns, other matches and regions can be specified, to avoid finding unwanted matches.

[The reason that the sync patterns are given separately, is that mostly the search for the sync point can be much simpler than figuring out the highlighting. The reduced number of patterns means it will go (much) faster.]

```
 syn-sync-grouphere E393 E394
:syntax sync match {sync-group-name} grouphere {group-name} "pattern" ..
```

Define a match that is used for syncing. `{group-name}` is the name of a syntax group that follows just after the match. Parsing

of the text for highlighting starts just after the match. A region must exist for this `{group-name}`. The first one defined will be used. "NONE" can be used for when there is no syntax group after the match.

#### syn-sync-grouphere

```
:syntax sync match {sync-group-name} grouphere {group-name} "pattern" ..
```

Like "grouphere", but `{group-name}` is the name of a syntax group that is to be used at the start of the line where searching for the sync point started. The text between the match and the start of the sync pattern searching is assumed not to change the syntax highlighting. For example, in C you could search backwards for `"/*` and `*/"`. If `"/*` is found first, you know that you are inside a comment, so the "grouphere" is "cComment". If `*/"` is found first, you know that you are not in a comment, so the "grouphere" is "NONE". (in practice it's a bit more complicated, because the `"/*` and `*/"` could appear inside a string. That's left as an exercise to the reader...).

```
:syntax sync match ..
:syntax sync region ..
```

Without a "grouphere" argument. Define a region or match that is skipped while searching for a sync point.

#### syn-sync-linecont

```
:syntax sync linecont {pattern}
```

When `{pattern}` matches in a line, it is considered to continue in the next line. This means that the search for a sync point will consider the lines to be concatenated.

If the "maxlines={N}" argument is given too, the number of lines that are searched for a match is restricted to N. This is useful if you have very few things to sync on and a slow machine. Example:

```
:syntax sync maxlines=100
```

You can clear all sync settings with:

```
:syntax sync clear
```

You can clear specific sync patterns with:

```
:syntax sync clear {sync-group-name} ..
```

---

12. Listing syntax items

```
:syntax :sy :syn :syn-list
```

This command lists all the syntax items:

```
:sy[ntax] [list]
```

To show the syntax items for one syntax group:

```
:sy[ntax] list {group-name}
```

To list the syntax groups in one cluster:

E392

```
:sy[ntax] list @{{cluster-name}}
```

See above for other arguments for the ":syntax" command.

**Note** that the ":syntax" command can be abbreviated to ":sy", although ":syn" is mostly used, because it looks better.

---

### 13. Colorschemes

### color-schemes

In the next section you can find information about individual highlight groups and how to specify colors for them. Most likely you want to just select a set of colors by using the `:colorscheme` command, for example:

```
colorscheme pablo
```

```
 :colo :colorscheme E185
:colo[rscheme] Output the name of the currently active color scheme.
 This is basically the same as
 :echo g:colors_name
In case g:colors_name has not been defined :colo will
output "default". When compiled without the +eval
feature it will output "unknown".
```

```
:colo[rscheme] {name} Load color scheme {name}. This searches 'runtimepath'
 for the file "colors/{name}.vim". The first one that
 is found is loaded.
 Also searches all plugins in 'packpath', first below
 "start" and then under "opt".
```

Doesn't work recursively, thus you can't use  
":colorscheme" in a color scheme script.

You have two options for customizing a color scheme. For changing the appearance of specific colors, you can redefine a color name before loading the scheme. The desert scheme uses the khaki color for the cursor. To use a darker variation of the same color:

```
let v:colornames['khaki'] = '#bdb76b'
colorscheme desert
```

For further customization, such as changing `:highlight-link` associations, use another name, e.g. `"~/vim/colors/mine.vim"`, and use `:runtime` to load the original color scheme:

```
runtime colors/evening.vim
hi Statement ctermfg=Blue guifg=Blue
```

Before the color scheme will be loaded all default color list scripts (`colors/lists/default.vim`) will be executed and then the `ColorSchemePre` autocommand event is triggered. After the color scheme has been loaded the `ColorScheme` autocommand event is triggered.

### colorscheme-override

If a color scheme is almost right, you can add modifications on top of it by using the `ColorScheme` autocommand. For example, to remove the background color (can make it transparent in some terminals):

```
augroup my_colorschemes
 au!
 au Colorscheme pablo hi Normal ctermbg=NONE
augroup END
```

Change a couple more colors:

```
augroup my_colorschemes
 au!
 au Colorscheme pablo hi Normal ctermbg=NONE
 \ | highlight Special ctermfg=63
 \ | highlight Identifier ctermfg=44
augroup END
```

If you make a lot of changes it might be better to copy the distributed colorscheme to your home directory and change it:

```
!cp $VIMRUNTIME/colors/pablo.vim ~/.vim/colors
:edit ~/.vim/colors/pablo.vim
```

With Vim 9.0 the collection of color schemes was updated and made work in many different terminals. One change was to often define the Normal highlight group to make sure the colors work well. In case you prefer the old version, you can find them here:

[https://github.com/vim/colorschemes/blob/master/legacy\\_colors/](https://github.com/vim/colorschemes/blob/master/legacy_colors/)

For info about writing a color scheme file:

```
:edit $VIMRUNTIME/colors/README.txt
```

---

#### 14. Highlight command

```
:highlight :hi E28 E411 E415
```

There are three types of highlight groups:

- The ones used for specific languages. For these the name starts with the name of the language. Many of these don't have any attributes, but are linked to a group of the second type.
- The ones used for all syntax languages.
- The ones used for the **'highlight'** option.

`hitest.vim`

You can see all the groups currently active with this command:

```
:so $VIMRUNTIME/syntax/hitest.vim
```

This will open a new window containing all highlight group names, displayed in their own color.

`:hi[highlight]` List all the current highlight groups that have attributes set.

`:hi[highlight] {group-name}` List one highlight group.

`:hi[highlight] clear` `highlight-clear` `:hi-clear` Reset all highlighting to the defaults. Removes all

highlighting for groups added by the user.  
Uses the current value of **'background'** to decide which default colors to use.  
If there was a default link, restore it. `:hi-link`

```
:hi[ghlight] clear {group-name}
:hi[ghlight] {group-name} NONE
 Disable the highlighting for one highlight group. It
 is _not_ set back to the default colors.

:hi[ghlight] [default] {group-name} {key}={arg} ..
 Add a highlight group, or change the highlighting for
 an existing group. If a given color name is not
 recognized, each colors/lists/default.vim found on
 'runtimepath' will be loaded.
 See highlight-args for the {key}={arg} arguments.
 See :highlight-default for the optional [default]
 argument.
```

Normally a highlight group is added once when starting up. This sets the default values for the highlighting. After that, you can use additional highlight commands to change the arguments that you want to set to non-default values. The value "NONE" can be used to switch the value off or go back to the default value.

A simple way to change colors is with the `:colorscheme` command. This loads a file with `":highlight"` commands such as this:

```
:hi Comment gui=bold
```

**Note** that all settings that are not included remain the same, only the specified field is used, and settings are merged with previous ones. So, the result is like this single command has been used:

```
:hi Comment term=bold ctermfg=Cyan guifg=#80a0ff gui=bold
```

`:highlight-verbose`

When listing a highlight group and **'verbose'** is non-zero, the listing will also tell where it was last set. Example:

```
:verbose hi Comment
Comment xxx term=bold ctermfg=4 guifg=Blue
Last set from /home/mool/vim/vim7/runtime/syntax/syncolor.vim
```

When `":hi clear"` is used then the script where this command is used will be mentioned for the default values. See `:verbose-cmd` for more information.

`highlight-args` E416 E417 E423

There are three types of terminals for highlighting:

```
term a normal terminal (vt100, xterm)
cterm a color terminal (MS-Windows console, color-xterm, these have the "Co"
 termcap entry)
gui the GUI
```

For each type the highlighting can be given. This makes it possible to use the same syntax file on all terminals, and use the optimal highlighting.

## 1. highlight arguments for normal terminals

```
term={attr-list} bold underline undercurl
 underdouble underdotted
 underdashed inverse italic
 standout nocombine strikethrough
 attr-list highlight-term E418
attr-list is a comma-separated list (without spaces) of the
following items (in any order):
 bold
 underline
 undercurl not always available
 underdouble not always available
 underdotted not always available
 underdashed not always available
 strikethrough not always available
 reverse
 inverse same as reverse
 italic
 standout
 nocombine override attributes instead of combining them
 NONE no attributes used (used to reset it)
```

**Note** that "bold" can be used here and by using a bold font. They have the same effect.

### underline-codes

"undercurl" is a curly underline. When "undercurl" is not possible then "underline" is used. In general "undercurl" and "strikethrough" are only available in the GUI and some terminals. The color is set with `highlight-guisp` or `highlight-ctermul`. You can try these termcap entries to make undercurl work in a terminal:

```
let &t_Cs = "\e[4:3m"
let &t_Ce = "\e[4:0m"
```

"underdouble" is a double underline, "underdotted" is a dotted underline and "underdashed" is a dashed underline. These are only supported by some terminals. If your terminal supports them you may have to specify the codes like this:

```
let &t_Us = "\e[4:2m"
let &t_ds = "\e[4:4m"
let &t_Ds = "\e[4:5m"
```

They are reset with `t_Ce`, the same as curly underline (undercurl). When `t_Us`, `t_ds` or `t_Ds` is not set then underline will be used as a fallback.

```
start={term-list} highlight-start E422
stop={term-list} term-list highlight-stop
These lists of terminal codes can be used to get
non-standard attributes on a terminal.
```

The escape sequence specified with the "start" argument is written before the characters in the highlighted

area. It can be anything that you want to send to the terminal to highlight this area. The escape sequence specified with the "stop" argument is written after the highlighted area. This should undo the "start" argument. Otherwise the screen will look messed up.

The `{term-list}` can have two forms:

1. A string with escape sequences.  
This is any string of characters, except that it can't start with "t\_" and blanks are not allowed. The `<>` notation is recognized here, so you can use things like "`<Esc>`" and "`<Space>`". Example:  
start=`<Esc>`[27h;`<Esc>`[`<Space>`r;
2. A list of terminal codes.  
Each terminal code has the form "t\_xx", where "xx" is the name of the termcap entry. The codes have to be separated with commas. White space is not allowed. Example:  
start=t\_C1,t\_BL  
The terminal codes must exist for this to work.

## 2. highlight arguments for color terminals

`cterm={attr-list}` highlight-cterm  
See above for the description of `{attr-list}` `attr-list`.  
The "cterm" argument is likely to be different from "term", when colors are used. For example, in a normal terminal comments could be underlined, in a color terminal they can be made Blue.  
**Note:** Some terminals (e.g., DOS console) can't mix these attributes with coloring. To be portable, use only one of "cterm=" OR "ctermfg=" OR "ctermbg=".

`ctermfg={color-nr}` highlight-ctermfg E421  
`ctermbg={color-nr}` highlight-ctermbg  
`ctermul={color-nr}` highlight-ctermul

These give the foreground (ctermfg), background (ctermbg) and underline (ctermul) color to use in the terminal.

The `{color-nr}` argument is a color number. Its range is zero to (not including) the number given by the termcap entry "Co". The actual color with this number depends on the type of terminal and its settings. Sometimes the color also depends on the settings of "cterm". For example, on some systems "cterm=bold ctermfg=3" gives another color, on others you just get color 3.

For an xterm this depends on your resources, and is a bit unpredictable. See your xterm documentation for the defaults. The colors for a color-xterm can be changed from the `.Xdefaults` file. Unfortunately this means that it's not possible to get the same colors for each user. See `xterm-color` for info about color xterms.

tmux

When using tmux you may want to use this in the tmux config:

```
tmux colors
```

```
set -s default-terminal "tmux-256color"
set -as terminal-overrides ",*-256color:Tc"
```

More info at:

<https://github.com/tmux/tmux/wiki/FAQ#how-do-i-use-a-256-colour-terminal>  
<https://github.com/tmux/tmux/wiki/FAQ#how-do-i-use-rgb-colour>

The MS-Windows standard colors are fixed (in a console window), so these have been used for the names. But the meaning of color names in X11 are fixed, so these color settings have been used, to make the highlighting settings portable (complicated, isn't it?). The following names are recognized, with the color number used:

			cterm-colors
NR-16	NR-8	COLOR NAME	
0	0	Black	
1	4	DarkBlue	
2	2	DarkGreen	
3	6	DarkCyan	
4	1	DarkRed	
5	5	DarkMagenta	
6	3	Brown, DarkYellow	
7	7	LightGray, LightGrey, Gray, Grey	
8	0*	DarkGray, DarkGrey	
9	4*	Blue, LightBlue	
10	2*	Green, LightGreen	
11	6*	Cyan, LightCyan	
12	1*	Red, LightRed	
13	5*	Magenta, LightMagenta	
14	3*	Yellow, LightYellow	
15	7*	White	

The number under "NR-16" is used for 16-color terminals ('t\_Co' greater than or equal to 16). The number under "NR-8" is used for 8-color terminals ('t\_Co' less than 16). The '\*' indicates that the bold attribute is set for ctermfg. In many 8-color terminals (e.g., "linux"), this causes the bright colors to appear. This doesn't work for background colors! Without the '\*' the bold attribute is removed. If you want to set the bold attribute in a different way, put a "cterm=" argument AFTER the "ctermfg=" or "ctermbg=" argument. Or use a number instead of a color name.

The case of the color names is ignored, however Vim will use lower case color names when reading from the `v:colornames` dictionary. **Note** that for 16 color ansi style terminals (including xterms), the numbers in the NR-8 column is used. Here '\*' means 'add 8' so that Blue is 12, DarkGray is 8 etc.

**Note** that for some color terminals these names may result in the wrong colors!

You can also use "NONE" to remove the color.

`:hi-normal-cterm`

When setting the "ctermfg" or "ctermbg" colors for the Normal group,



these will become the colors used for the non-highlighted text.  
Example:

```
:highlight Normal ctermfg=grey ctermbg=darkblue
```

When setting the "ctermbg" color for the Normal group, the 'background' option will be adjusted automatically, under the condition that the color is recognized and 'background' was not set explicitly. This causes the highlight groups that depend on 'background' to change! This means you should set the colors for Normal first, before setting other colors.

When a color scheme is being used, changing 'background' causes it to be reloaded, which may reset all colors (including Normal). First delete the "g:colors\_name" variable when you don't want this.

When you have set "ctermfg" or "ctermbg" for the Normal group, Vim needs to reset the color when exiting. This is done with the "op" termcap entry `t_op`. If this doesn't work correctly, try setting the 't\_op' option in your .vimrc.

E419 E420 E453

When Vim knows the normal foreground, background and underline colors, "fg", "bg" and "ul" can be used as color names. This only works after setting the colors for the Normal group and for the MS-Windows console. Example, for reverse video:

```
:highlight Visual ctermfg=bg ctermbg=fg
```

Note that the colors are used that are valid at the moment this command is given. If the Normal group colors are changed later, the "fg" and "bg" colors will not be adjusted.

ctermfont={font-nr}

highlight-ctermfont

This gives the alternative font number to use in the terminal. The available fonts depend on the terminal, and if the terminal is not set up for alternative fonts this simply won't do anything. The range of {font-nr} is 0-10 where 0 resets the font to the default font, 1-9 selects one of the 9 alternate fonts, and 10 selects the Fraktur font. For more information see your terminal's handling of SGR parameters 10-20. `t_CF`

### 3. highlight arguments for the GUI

gui={attr-list}

highlight-gui

These give the attributes to use in the GUI mode.

See `attr-list` for a description.

Note that "bold" can be used here and by using a bold font. They have the same effect.

Note that the attributes are ignored for the "Normal" group.

font={font-name}

highlight-font

font-name is the name of a font, as it is used on the system Vim runs on. For X11 this is a complicated name, for example:

```
font=-misc-fixed-bold-r-normal--14-130-75-75-c-70-iso8859-1
```

The font-name "NONE" can be used to revert to the default font.

When setting the font for the "Normal" group, this becomes the default font (until the 'guifont' option is changed; the last one set is used).

The following only works with Motif, not with other GUIs:  
When setting the font for the "Menu" group, the menus will be changed.  
When setting the font for the "Tooltip" group, the tooltips will be changed.

All fonts used, except for Menu and Tooltip, should be of the same character size as the default font! Otherwise redrawing problems will occur.

To use a font name with an embedded space or other special character, put it in single quotes. The single quote cannot be used then.

Example:

```
:hi comment font='Monospace 10'
```

```
guifg={color-name} highlight-guifg
guibg={color-name} highlight-guibg
guisp={color-name} highlight-guisp
```

These give the foreground (guifg), background (guibg) and special (guisp) color to use in the GUI. "guisp" is used for undercurl and strikethrough.

There are a few special names:

NONE	no color (transparent)	E1361
bg	use normal background color	
background	use normal background color	
fg	use normal foreground color	
foreground	use normal foreground color	

To use a color name with an embedded space or other special character, put it in single quotes. The single quote cannot be used then.

Example:

```
:hi comment guifg='salmon pink'
```

```
gui-colors
Suggested color names (these are available on most systems):
```

Red	LightRed	DarkRed	
Green	LightGreen	DarkGreen	SeaGreen
Blue	LightBlue	DarkBlue	SlateBlue
Cyan	LightCyan	DarkCyan	
Magenta	LightMagenta	DarkMagenta	
Yellow	LightYellow	Brown	DarkYellow
Gray	LightGray	DarkGray	
Black	White		
Orange	Purple	Violet	

In the Win32 GUI version, additional system colors are available. See [win32-colors](#) .

You can also specify a color by its Red, Green and Blue values.

The format is "#rrggbb", where

"rr"	is the Red value
"gg"	is the Green value
"bb"	is the Blue value

All values are hexadecimal, range from "00" to "ff". Examples:

```
:highlight Comment guifg=#11f0c3 guibg=#ff00ff
```

If you are authoring a color scheme and use the same hexadecimal value repeatedly, you can define a (lower case) name for it in [v:colornames](#) .

For example:

```
provide a default value for this color but allow the user to
override it.
:call extend(v:colornames, {'alt_turquoise': '#11f0c3'}, 'keep')
:highlight Comment guifg=alt_turquoise guibg=magenta
```

If you are using a color scheme that relies on named colors and you would like to adjust the precise appearance of those colors, you can do so by overriding the values in `v:colornames` prior to loading the scheme:

```
let v:colornames['alt_turquoise'] = '#22f0d3'
colorscheme alt
```

If you want to develop a color list that can be relied on by others, it is best to prefix your color names. By convention these color lists are placed in the `colors/lists` directory. You can see an example in `'$VIMRUNTIME/colors/lists/csscolors.vim'`. This list would be sourced by a color scheme using:

```
:runtime colors/lists/csscolors.vim
:highlight Comment guifg=css_turquoise
```

These are the default highlighting groups. These groups are used by the `'highlight'` option default. `Note` that the highlighting depends on the value of `'background'`. You can see the current settings with the `":highlight"` command.

When possible the name is highlighted in the used colors. If this makes it unreadable use Visual selection.

ColorColumn	Used for the columns set with <code>'colorcolumn'</code> .	<code>hl-ColorColumn</code>
Conceal	Placeholder characters substituted for concealed text (see <code>'conceallevel'</code> ).	<code>hl-Conceal</code>
Cursor	Character under the cursor.	<code>hl-Cursor</code>
lCursor	Character under the cursor when <code>language-mapping</code> is used (see <code>'guicursor'</code> ).	<code>hl-lCursor</code>
CursorIM	Like Cursor, but used when in IME mode.	<code>hl-CursorIM</code>
CursorColumn	Screen column that the cursor is in when <code>'cursorcolumn'</code> is set.	<code>CursorIM</code> <code>hl-CursorColumn</code>
CursorLine	Screen line that the cursor is in when <code>'cursorline'</code> is set.	<code>hl-CursorLine</code>
Directory	Directory names (and other special names in listings).	<code>hl-Directory</code>
DiffAdd	Diff mode: Added line. <code>diff.txt</code>	<code>hl-DiffAdd</code>
DiffChange	Diff mode: Changed line. <code>diff.txt</code>	<code>hl-DiffChange</code>

DiffDelete	Diff mode: Deleted line. <code>diff.txt</code>	<code>hl-DiffDelete</code>
DiffText	Diff mode: Changed text within a changed line. <code>diff.txt</code>	<code>hl-DiffText</code>
EndOfBuffer	Filler lines (~) after the last line in the buffer. By default, this is highlighted like <code>hl-NonText</code> .	<code>hl-EndOfBuffer</code>
ErrorMsg	Error messages on the command line.	<code>hl-ErrorMsg</code>
VertSplit	Column separating vertically split windows.	<code>hl-VertSplit</code>
Folded	Line used for closed folds.	<code>hl-Folded</code>
FoldColumn	<code>'foldcolumn'</code>	<code>hl-FoldColumn</code>
SignColumn	Column where <code>signs</code> are displayed.	<code>hl-SignColumn</code>
IncSearch	<code>'incsearch'</code> highlighting; also used for the text replaced with <code>":s///c"</code> .	<code>hl-IncSearch</code>
LineNr	Line number for <code>":number"</code> and <code>":#"</code> commands, and when <code>'number'</code> or <code>'relativenumber'</code> option is set.	<code>hl-LineNr</code>
LineNrAbove	Line number for when the <code>'relativenumber'</code> option is set, above the cursor line.	<code>hl-LineNrAbove</code>
LineNrBelow	Line number for when the <code>'relativenumber'</code> option is set, below the cursor line.	<code>hl-LineNrBelow</code>
CursorLineNr	Like LineNr when <code>'cursorline'</code> is set and <code>'cursorlineopt'</code> contains "number" or is "both", for the cursor line.	<code>hl-CursorLineNr</code>
CursorLineFold	Like FoldColumn when <code>'cursorline'</code> is set for the cursor line.	<code>hl-CursorLineFold</code>
CursorLineSign	Like SignColumn when <code>'cursorline'</code> is set for the cursor line.	<code>hl-CursorLineSign</code>
MatchParen	Character under the cursor or just before it, if it is a paired bracket, and its match. <code>pi_paren.txt</code>	<code>hl-MatchParen</code>
MessageWindow	Messages popup window used by <code>':echowindow'</code> . If not defined <code>hl-WarningMsg</code> is used.	<code>hl-MessageWindow</code>
ModeMsg	<code>'showmode'</code> message (e.g., <code>-- INSERT --</code> ).	<code>hl-ModeMsg</code>
MoreMsg	<code>more-prompt</code>	<code>hl-MoreMsg</code>
NonText	'@' at the end of the window, "<<<" at the start of the window for <code>'smoothscroll'</code> , characters from <code>'showbreak'</code> and other characters that do not really exist in the text, such as the ">" displayed when a double-wide character doesn't fit at the end of the line.	<code>hl-NonText</code>
Normal	Normal text.	<code>hl-Normal</code>

Pmenu	Popup menu: Normal item.	<a href="#">hl-Pmenu</a>
PmenuSel	Popup menu: Selected item.	<a href="#">hl-PmenuSel</a>
PmenuKind	Popup menu: Normal item "kind".	<a href="#">hl-PmenuKind</a>
PmenuKindSel	Popup menu: Selected item "kind".	<a href="#">hl-PmenuKindSel</a>
PmenuExtra	Popup menu: Normal item "extra text".	<a href="#">hl-PmenuExtra</a>
PmenuExtraSel	Popup menu: Selected item "extra text".	<a href="#">hl-PmenuExtraSel</a>
PmenuSbar	Popup menu: Scrollbar.	<a href="#">hl-PmenuSbar</a>
PmenuThumb	Popup menu: Thumb of the scrollbar.	<a href="#">hl-PmenuThumb</a>
PopupNotification		<a href="#">hl-PopupNotification</a>
	Popup window created with <a href="#">popup_notification()</a> . If not defined <a href="#">hl-WarningMsg</a> is used.	
Question	<a href="#">hit-enter</a> prompt and yes/no questions.	<a href="#">hl-Question</a>
QuickFixLine	Current <a href="#">quickfix</a> item in the quickfix window.	<a href="#">hl-QuickFixLine</a>
Search	Last search pattern highlighting (see <a href="#">'hlsearch'</a> ). Also used for similar items that need to stand out.	<a href="#">hl-Search</a>
CurSearch	Current match for the last search pattern (see <a href="#">'hlsearch'</a> ). <b>Note:</b> This is correct after a search, but may get outdated if changes are made or the screen is redrawn.	<a href="#">hl-CurSearch</a>
SpecialKey	Meta and special keys listed with ":map", also for text used to show unprintable characters in the text, <a href="#">'listchars'</a> . Generally: Text that is displayed differently from what it really is.	<a href="#">hl-SpecialKey</a>
SpellBad	Word that is not recognized by the spellchecker. <a href="#">spell</a> This will be combined with the highlighting used otherwise.	<a href="#">hl-SpellBad</a>
SpellCap	Word that should start with a capital. <a href="#">spell</a> This will be combined with the highlighting used otherwise.	<a href="#">hl-SpellCap</a>
SpellLocal	Word that is recognized by the spellchecker as one that is used in another region. <a href="#">spell</a> This will be combined with the highlighting used otherwise.	<a href="#">hl-SpellLocal</a>
SpellRare	Word that is recognized by the spellchecker as one that is hardly ever used. <a href="#">spell</a> This will be combined with the highlighting used otherwise.	<a href="#">hl-SpellRare</a>
StatusLine	Status line of current window.	<a href="#">hl-StatusLine</a>
StatusLineNC	status lines of not-current windows	<a href="#">hl-StatusLineNC</a>

**Note:** If this is equal to "StatusLine", Vim will use "^^^" in the status line of the current window.

StatusLineTerm	Status line of current window, if it is a <code>terminal</code> window.	<code>hl-StatusLineTerm</code>
StatusLineTermNC	Status lines of not-current windows that is a <code>terminal</code> window.	<code>hl-StatusLineTermNC</code>
TabLine	Tab pages line, not active tab page label.	<code>hl-TabLine</code>
TabLineFill	Tab pages line, where there are no labels.	<code>hl-TabLineFill</code>
TabLineSel	Tab pages line, active tab page label.	<code>hl-TabLineSel</code>
Terminal	<code>terminal</code> window (see <code>terminal-size-color</code> ).	<code>hl-Terminal</code>
Title	Titles for output from <code>:set all</code> , <code>:autocmd</code> etc.	<code>hl-Title</code>
Visual	Visual mode selection.	<code>hl-Visual</code>
VisualNOS	Visual mode selection when vim is "Not Owning the Selection". Only X11 Gui's <code>gui-x11</code> and <code>xterm-clipboard</code> supports this.	<code>hl-VisualNOS</code>
WarningMsg	Warning messages.	<code>hl-WarningMsg</code>
WildMenu	Current match in <code>'wildmenu'</code> completion.	<code>hl-WildMenu</code>

`hl-User1` `hl-User1..9` `hl-User9`

The `'statusline'` syntax allows the use of 9 different highlights in the statusline and ruler (via `'rulerformat'`). The names are User1 to User9.

For the GUI you can use the following groups to set the colors for the menu, scrollbars and tooltips. They don't have defaults. This doesn't work for the Win32 GUI. Only three highlight arguments have any effect here: font, guibg, and guifg.

Menu	Current font, background and foreground colors of the menus. Also used for the toolbar. Applicable highlight arguments: font, guibg, guifg.	<code>hl-Menu</code>
	<b>NOTE:</b> For Motif the font argument actually specifies a fontset at all times, no matter if <code>'guifontset'</code> is empty, and as such it is tied to the current <code>:language</code> when set.	

Scrollbar	Current background and foreground of the main window's scrollbars. Applicable highlight arguments: guibg, guifg.	<code>hl-Scrollbar</code>
-----------	---------------------------------------------------------------------------------------------------------------------	---------------------------

Tooltip	Current font, background and foreground of the tooltips. Applicable highlight arguments: font, guibg, guifg.	<code>hl-Tooltip</code>
---------	-----------------------------------------------------------------------------------------------------------------	-------------------------

**NOTE:** For Motif the font argument actually specifies a fontset at all times, no matter if 'guifontset' is empty, and as such it is tied to the current `:language` when set.

---

15. Linking groups `:hi-link` `:highlight-link` E412 E413

When you want to use the same highlighting for several syntax groups, you can do this more easily by linking the groups into one common highlight group, and give the color attributes only for that group.

To set a link:

```
:hi[ghlight][!] [default] link {from-group} {to-group}
```

To remove a link:

```
:hi[ghlight][!] [default] link {from-group} NONE
```

**Notes:**

E414

- If the `{from-group}` and/or `{to-group}` doesn't exist, it is created. You don't get an error message for a non-existing group.
- As soon as you use a `":highlight"` command for a linked group, the link is removed.
- If there are already highlight settings for the `{from-group}`, the link is not made, unless the `''` is given. For a `":highlight link"` command in a sourced file, you don't get an error message. This can be used to skip links for groups that already have settings.

`:hi-default` `:highlight-default`

The `[default]` argument is used for setting the default highlighting for a group. If highlighting has already been specified for the group the command will be ignored. Also when there is an existing link.

Using `[default]` is especially useful to overrule the highlighting of a specific syntax file. For example, the C syntax file contains:

```
:highlight default link cComment Comment
```

If you like Question highlighting for C comments, put this in your vimrc file:

```
:highlight link cComment Question
```

Without the "default" in the C syntax file, the highlighting would be overruled when the syntax file is loaded.

To have a link survive `':highlight clear'`, which is useful if you have highlighting for a specific filetype and you want to keep it when selecting another color scheme, put a command like this in the `"after/syntax/{filetype}.vim"` file:

```
highlight! default link cComment Question
```

---

16. Cleaning up `:syn-clear` E391

If you want to clear the syntax stuff for the current buffer, you can use this

command:

```
:syntax clear
```

This command should be used when you want to switch off syntax highlighting, or when you want to switch to using another syntax. It's normally not needed in a syntax file itself, because syntax is cleared by the autocommands that load the syntax file.

The command also deletes the "b:current\_syntax" variable, since no syntax is loaded after this command.

To clean up specific syntax groups for the current buffer:

```
:syntax clear {group-name} ..
```

This removes all patterns and keywords for {group-name}.

To clean up specific syntax group lists for the current buffer:

```
:syntax clear @{grouplist-name} ..
```

This sets {grouplist-name}'s contents to an empty list.

```
:syntax-off :syn-off
```

If you want to disable syntax highlighting for all buffers, you need to remove the autocommands that load the syntax files:

```
:syntax off
```

What this command actually does, is executing the command

```
:source $VIMRUNTIME/syntax/nosyntax.vim
```

See the "nosyntax.vim" file for details. **Note** that for this to work \$VIMRUNTIME must be valid. See `$VIMRUNTIME` .

```
:syntax-reset :syn-reset
```

If you have changed the colors and messed them up, use this command to get the defaults back:

```
:syntax reset
```

It is a bit of a wrong name, since it does not reset any syntax items, it only affects the highlighting.

This doesn't change the colors for the **'highlight'** option.

**Note** that the syntax colors that you set in your vimrc file will also be reset back to their Vim default.

**Note** that if you are using a color scheme, the colors defined by the color scheme for syntax highlighting will be lost.

What this actually does is:

```
let g:syntax_cmd = "reset"
runtime! syntax/syncolor.vim
```

**Note** that this uses the **'runtimepath'** option.

```
syncolor
```

If you want to use different colors for syntax highlighting, you can add a Vim script file to set these colors. Put this file in a directory in



'**runtimepath**' which comes after \$VIMRUNTIME, so that your settings overrule the default colors. This way these colors will be used after the ":syntax reset" command.

For Unix you can use the file ~/.vim/after/syntax/syncolor.vim. Example:

```
if &background == "light"
 highlight comment ctermfg=darkgreen guifg=darkgreen
else
 highlight comment ctermfg=green guifg=green
endif
```

E679

Do make sure this syncolor.vim script does not use a "syntax on", set the '**background**' option or uses a "colorscheme" command, because it results in an endless loop.

**Note** that when a color scheme is used, there might be some confusion whether your defined colors are to be used or the colors from the scheme. This depends on the color scheme file. See `:colorscheme` .

syntax\_cmd

The "syntax\_cmd" variable is set to one of these values when the syntax/syncolor.vim files are loaded:

"on"	`:syntax on` command. Highlight colors are overruled but links are kept
"enable"	`:syntax enable` command. Only define colors for groups that don't have highlighting yet. Use `:highlight default` .
"reset"	`:syntax reset` command or loading a color scheme. Define all the colors.
"skip"	Don't define colors. Used to skip the default settings when a syncolor.vim file earlier in ' <b>runtimepath</b> ' has already set them.

---

## 17. Highlighting tags

tag-highlight

If you want to highlight all the tags in your file, you can use the following mappings.

```
<F11> -- Generate tags.vim file, and highlight tags.
<F12> -- Just highlight tags based on existing tags.vim file.

:map <F11> :sp tags<CR>:%s/^\([^ :]*\)\=\\(\[^\]*\).*\/syntax keyword Tag \2/<CR>:v
:map <F12> :so tags.vim<CR>
```

**WARNING:** The longer the tags file, the slower this will be, and the more memory Vim will consume.

Only highlighting typedefs, unions and structs can be done too. For this you must use Universal Ctags (found at <https://ctags.io>) or Exuberant ctags (found at <http://ctags.sf.net>).

Put these lines in your Makefile:

```
Make a highlight file for types. Requires Universal/Exuberant ctags and awk
types: types.vim
types.vim: *. [ch]
 ctags --c-kinds=gstu -o- *. [ch] | \
 awk 'BEGIN{printf("syntax keyword Type\t")}\
 {printf("%s ", $$1)}END{print ""}' > $@
```

And put these lines in your .vimrc:

```
" load the types.vim highlighting file, if it exists
autocmd BufRead,BufNewFile *. [ch] let fname = expand('<afile>:p:h') .. '/types.vim'
autocmd BufRead,BufNewFile *. [ch] if filereadable(fname)
autocmd BufRead,BufNewFile *. [ch] exe 'so ' .. fname
autocmd BufRead,BufNewFile *. [ch] endif
```

---

## 18. Window-local syntax

`:ownsyntax`

Normally all windows on a buffer share the same syntax settings. It is possible, however, to set a particular window on a file to have its own private syntax setting. A possible example would be to edit LaTeX source with conventional highlighting in one window, while seeing the same source highlighted differently (so as to hide control sequences and indicate bold, italic etc regions) in another. The `'scrollbind'` option is useful here.

To set the current window to have the syntax "foo", separately from all other windows on the buffer:

```
:ownsyntax foo
```

`w:current_syntax`

This will set the "w:current\_syntax" variable to "foo". The value of "b:current\_syntax" does not change. This is implemented by saving and restoring "b:current\_syntax", since the syntax files do set "b:current\_syntax". The value set by the syntax file is assigned to "w:current\_syntax".

**Note:** This resets the `'spell'`, `'spellcapcheck'` and `'spellfile'` options.

Once a window has its own syntax, syntax commands executed from other windows on the same buffer (including `:syntax clear`) have no effect. Conversely, syntax commands executed from that window do not affect other windows on the same buffer.

A window with its own syntax reverts to normal behavior when another buffer is loaded into that window or the file is reloaded.

When splitting the window, the new window will use the original syntax.

---

## 19. Color xterms

`xterm-color`    `color-xterm`

Most color xterms have only eight colors. If you don't get colors with the default setup, it should work with these lines in your .vimrc:

```
:if &term =~ "xterm"
: if has("terminfo")
: set t_Co=8
```

```

: set t_Sf=<Esc>[3%p1%dm
: set t_Sb=<Esc>[4%p1%dm
: else
: set t_Co=8
: set t_Sf=<Esc>[3%dm
: set t_Sb=<Esc>[4%dm
: endif
:endif
 [<Esc> is a real escape, type CTRL-V <Esc>]

```

You might want to change the first "if" to match the name of your terminal, e.g. "dtterm" instead of "xterm".

**Note:** Do these settings BEFORE doing ":syntax on". Otherwise the colors may be wrong.

**xterm rxvt**

The above settings have been mentioned to work for xterm and rxvt too. But for using 16 colors in an rxvt these should work with terminfo:

```

:set t_AB=<Esc>[%?%p1%{8}%<%t25;%p1%{40}%+%e5;%p1%{32}%+%;%dm
:set t_AF=<Esc>[%?%p1%{8}%<%t22;%p1%{30}%+%e1;%p1%{22}%+%;%dm

```

**colortest.vim**

To test your color setup, a file has been included in the Vim distribution. To use it, execute this command:

```

:runtime syntax/colortest.vim

```

Some versions of xterm (and other terminals, like the Linux console) can output lighter foreground colors, even though the number of colors is defined at 8. Therefore Vim sets the "cterm=bold" attribute for light foreground colors, when '**t\_Co**' is 8.

**xfree-xterm**

To get 16 colors or more, get the newest xterm version (which should be included with XFree86 3.3 and later). You can also find the latest version at:

<http://invisible-island.net/xterm/xterm.html>

Here is a good way to configure it. This uses 88 colors and enables the termcap-query feature, which allows Vim to ask the xterm how many colors it supports.

```

./configure --disable-bold-color --enable-88-color --enable-tcap-query

```

If you only get 8 colors, check the xterm compilation settings.

(Also see [UTF8-xterm](#) for using this xterm with UTF-8 character encoding).

This xterm should work with these lines in your .vimrc (for 16 colors):

```

:if has("terminfo")
: set t_Co=16
: set t_AB=<Esc>[%?%p1%{8}%<%t%p1%{40}%+%e%p1%{92}%+%;%dm
: set t_AF=<Esc>[%?%p1%{8}%<%t%p1%{30}%+%e%p1%{82}%+%;%dm
:else
: set t_Co=16
: set t_Sf=<Esc>[3%dm
: set t_Sb=<Esc>[4%dm
:endif
 [<Esc> is a real escape, type CTRL-V <Esc>]

```

Without `+terminfo`, Vim will recognize these settings, and automatically translate cterm colors of 8 and above to "`<Esc>[9%dm`" and "`<Esc>[10%dm`". Colors above 16 are also translated automatically.

For 256 colors this has been reported to work:

```
:set t_AB=<Esc>[48;5;%dm
:set t_AF=<Esc>[38;5;%dm
```

Or just set the TERM environment variable to "xterm-color" or "xterm-16color" and try if that works.

You probably want to use these X resources (in your `~/.Xdefaults` file):

```
XTerm*color0: #000000
XTerm*color1: #c00000
XTerm*color2: #008000
XTerm*color3: #808000
XTerm*color4: #0000c0
XTerm*color5: #c000c0
XTerm*color6: #008080
XTerm*color7: #c0c0c0
XTerm*color8: #808080
XTerm*color9: #ff6060
XTerm*color10: #00ff00
XTerm*color11: #ffff00
XTerm*color12: #8080ff
XTerm*color13: #ff40ff
XTerm*color14: #00ffff
XTerm*color15: #ffffff
Xterm*cursorColor: Black
```

[Note: The `cursorColor` is required to work around a bug, which changes the cursor color to the color of the last drawn text. This has been fixed by a newer version of xterm, but not everybody is using it yet.]

To get these right away, reload the `.Xdefaults` file to the X Option database Manager (you only need to do this when you just changed the `.Xdefaults` file):

```
xrdb -merge ~/.Xdefaults
```

`xterm-blink` `xterm-blinking-cursor`

To make the cursor blink in an xterm, see `tools/blink.c`. Or use Thomas Dickey's xterm above patchlevel 107 (see above for where to get it), with these resources:

```
XTerm*cursorBlink: on
XTerm*cursorOnTime: 400
XTerm*cursorOffTime: 250
XTerm*cursorColor: White
```

`hpterm-color`

These settings work (more or less) for an hpterm, which only supports 8 foreground colors:

```
:if has("terminfo")
: set t_Co=8
```

```

: set t_Sf=<Esc>[&v%p1%dS
: set t_Sb=<Esc>[&v7S
:else
: set t_Co=8
: set t_Sf=<Esc>[&v%dS
: set t_Sb=<Esc>[&v7S
:endif
[<Esc> is a real escape, type CTRL-V <Esc>]

```

#### Eterm enlightened-terminal

These settings have been reported to work for the Enlightened terminal emulator, or Eterm. They might work for all xterm-like terminals that use the bold attribute to get bright colors. Add an ":if" like above when needed.

```

:set t_Co=16
:set t_AF=^[[%?%p1%{8}%<%t3%p1%d%e%p1%{22}%+%d;1%;m
:set t_AB=^[[%?%p1%{8}%<%t4%p1%d%e%p1%{32}%+%d;1%;m

```

#### TTpro-telnet

These settings should work for TTpro telnet. Tera Term Pro is a freeware / open-source program for MS-Windows.

```

set t_Co=16
set t_AB=^[[%?%p1%{8}%<%t%p1%{40}%+%e%p1%{32}%+5%;%;%dm
set t_AF=^[[%?%p1%{8}%<%t%p1%{30}%+%e%p1%{22}%+1%;%;%dm

```

Also make sure TTpro's Setup / Window / Full Color is enabled, and make sure that Setup / Font / Enable Bold is NOT enabled.

(info provided by John Love-Jensen <eljay@Adobe.COM>)

## ===== 20. When syntax is slow

### :syntime

This is aimed at authors of a syntax file.

If your syntax causes redrawing to be slow, here are a few hints on making it faster. To see slowness switch on some features that usually interfere, such as '**relativenumber**' and **foldings** .

**Note:** This is only available when compiled with the **+profile** feature. You may need to build Vim with "huge" features.

To find out what patterns are consuming most time, get an overview with this sequence:

```

:syntime on
[redraw the text at least once with CTRL-L]
:syntime report

```

This will display a list of syntax patterns that were used, sorted by the time it took to match them against the text.

```

:syntime on Start measuring syntax times. This will add some
 overhead to compute the time spent on syntax pattern
 matching.

```

```

:syntime off Stop measuring syntax times.

```

:syntime clear           Set all the counters to zero, restart measuring.

:syntime report         Show the syntax items used since ":syntime on" in the current window. Use a wider display to see more of the output.

The list is sorted by total time. The columns are:

TOTAL	Total time in seconds spent on matching this pattern.
COUNT	Number of times the pattern was used.
MATCH	Number of times the pattern actually matched
SLOWEST	The longest time for one try.
AVERAGE	The average time for one try.
NAME	Name of the syntax item. <a href="#">Note</a> that this is not unique.
PATTERN	The pattern being used.

Pattern matching gets slow when it has to try many alternatives. Try to include as much literal text as possible to reduce the number of ways a pattern does NOT match.

When using the "\@<=" and "\@<!" items, add a maximum size to avoid trying at all positions in the current and previous line. For example, if the item is literal text specify the size of that text (in bytes):

"<\@<=span"       Matches "span" in "<span". This tries matching with "<" in many places.

"<\@1<=span"     Matches the same, but only tries one byte before "span".

Displaying text with properties attached. [textprop](#) [text-properties](#)

- 1. Introduction [text-prop-intro](#)
- 2. Functions [text-prop-functions](#)
- 3. When text changes [text-prop-changes](#)

{not able to use text properties when the [+textprop](#) feature was disabled at compile time}

---

## 1. Introduction [text-prop-intro](#)

Text properties can be attached to text in a buffer. They will move with the text: If lines are deleted or inserted the properties move with the text they are attached to. Also when inserting/deleting text in the line before the text property. And when inserting/deleting text inside the text property, it will increase/decrease in size.

The main use for text properties is to highlight text. This can be seen as a replacement for syntax highlighting. Instead of defining patterns to match the text, the highlighting is set by a script, possibly using the output of an external parser. This only needs to be done once, not every time when redrawing the screen, thus can be much faster, after the initial cost of attaching the text properties.

Text properties can also be used for other purposes to identify text. For example, add a text property on a function name, so that a search can be defined to jump to the next/previous function.

A text property is attached at a specific line and column, and has a specified length. The property can span multiple lines.

A text property has these fields:

"id"	a number to be used as desired
"type"	the name of a property type

## Property Types

[E971](#)

A text property normally has the name of a property type, which defines how to highlight the text. The property type can have these entries:

"highlight"	name of the highlight group to use
"combine"	when omitted or TRUE the text property highlighting is combined with any syntax highlighting; when FALSE the text property highlighting replaces the syntax highlighting

"priority"	when properties overlap, the one with the highest priority will be used.
"start_incl"	when TRUE inserts at the start position will be included in the text property
"end_incl"	when TRUE inserts at the end position will be included in the text property

### Example

Suppose line 11 in a buffer has this text (excluding the indent):

```
The number 123 is smaller than 4567.
```

To highlight the numbers in this text:

```
call prop_type_add('number', {'highlight': 'Constant'})
call prop_add(11, 12, {'length': 3, 'type': 'number'})
call prop_add(11, 32, {'length': 4, 'type': 'number'})
```

Try inserting or deleting lines above the text, you will see that the text properties stick to the text, thus the line number is adjusted as needed.

Setting "start\_incl" and "end\_incl" is useful when white space surrounds the text, e.g. for a function name. Using false is useful when the text starts and/or ends with a specific character, such as the quote surrounding a string.

```
func FuncName(arg)
 ^^^^^^^^^ property with start_incl and end_incl set

var = "text";
 ^^^^^^^ property with start_incl and end_incl not set
```

Nevertheless, when text is inserted or deleted the text may need to be parsed and the text properties updated. But this can be done asynchronously.

### Internal error E967

If you see E967, please report the bug. You can do this at Github:  
<https://github.com/vim/vim/issues/new>

```
=====
2. Functions text-prop-functions
```

Manipulating text property types:

prop_type_add({name}, {props})	define a new property type
prop_type_change({name}, {props})	change an existing property type
prop_type_delete({name} [, {props}])	delete a property type
prop_type_get({name} [, {props}])	get property type values
prop_type_list([{props}])	get list of property types

Manipulating text properties:



`prop_add({lnum}, {col}, {props})`      add a text property  
`prop_add_list({props}, [{item}, ...])`      add a text property at multiple positions.  
`prop_clear({lnum} [, {lnum-end} [, {bufnr}]])`      remove all text properties  
`prop_find({props} [, {direction}])`      search for a text property  
`prop_list({lnum} [, {props}])`      text properties in {lnum}  
`prop_remove({props} [, {lnum} [, {lnum-end}]])`      remove a text property

`prop_add()`    E965

`prop_add({lnum}, {col}, {props})`  
 Attach a text property at position {lnum}, {col}. {col} is counted in bytes, use one for the first column.  
 If {lnum} is invalid an error is given.    E966  
 If {col} is invalid an error is given.    E964

{props} is a dictionary with these fields:

type	name of the text property type
length	length of text in bytes, can only be used for a property that does not continue in another line; can be zero
end_lnum	line number for the end of text (inclusive)
end_col	column just after the text; not used when "length" is present; when {col} and "end_col" are equal, and "end_lnum" is omitted or equal to {lnum}, this is a zero-width text property
bufnr	buffer to add the property to; when omitted the current buffer is used
id	user defined ID for the property; must be a number, should be positive; when using "text" then "id" must not be present and will be set automatically to a negative number; otherwise zero is used
text	text to be displayed before {col}, or above/below the line if {col} is zero; prepend and/or append spaces for padding with highlighting; cannot be used with "length", "end_lnum" and "end_col" See <a href="#">virtual-text</a> for more information.
text_align	when "text" is present and {col} is zero; specifies where to display the text: <ul style="list-style-type: none"> <li>after    after the end of the line</li> <li>right    right aligned in the window (unless the text wraps to the next screen line)</li> <li>below    in the next screen line</li> <li>above    just above the line</li> </ul> When omitted "after" is used. Only one "right" property can fit in each line, if

E1305

E1294

there are two or more these will go in a separate line (still right aligned).

`text_padding_left` E1296  
used when "text" is present and `{col}` is zero; padding between the end of the text line (leftmost column for "above" and "below") and the virtual text, not highlighted

`text_wrap` when "text" is present and `{col}` is zero, specifies what happens if the text doesn't fit:

- `wrap` wrap the text to the next line
- `truncate` truncate the text to make it fit

When omitted "truncate" is used.

Note that this applies to the individual text property, the 'wrap' option sets the overall behavior

All fields except "type" are optional.

It is an error when both "length" and "end\_lnum" or "end\_col" are given. Either use "length" or "end\_col" for a property within one line, or use "end\_lnum" and "end\_col" for a property that spans more than one line. When neither "length" nor "end\_col" are given the property will be zero-width. That means it will move with the text, as a kind of mark. One character will be highlighted, if the type specifies highlighting. The property can end exactly at the last character of the text, or just after it. In the last case, if text is appended to the line, the text property size will increase, also when the property type does not have "end\_incl" set.

"type" will first be looked up in the buffer the property is added to. When not found, the global property types are used. If not found an error is given.

#### virtual-text

When "text" is used and the column is non-zero then this text will be displayed at the specified start location of the text property. The text of the buffer line will be shifted to make room. This is called "virtual text".

When the column is zero the virtual text will appear above, after or below the buffer text. The "text\_align" and "text\_wrap" arguments determine how it is displayed.

To separate the virtual text from the buffer text prepend and/or append spaces to the "text" field or use the "text\_padding\_left" value.

Make sure to use a highlight that makes clear to the user that this is virtual text, otherwise it will be very confusing that the text cannot be edited. When using "above" you need to make clear this text belongs to the text line below it, when using "below" you need to make sure it belongs to the text line above it.

The text will be displayed but it is not part of the actual

buffer line, the cursor cannot be placed on it. A mouse click in the text will move the cursor to the first character after the text, or the last character of the line.

Any Tab and other control character in the text will be changed to a space (Rationale: otherwise the size of the text is difficult to compute).

A negative "id" will be chosen and is returned.

Before text properties with text were supported it was possible to use a negative "id", even though this was very rare. Now that negative "id"s are reserved for text properties with text an error is given when using a negative "id". When a text property with text already exists using a negative "id" results in **E1293** . If a negative "id" was used and later a text property with text is added results in **E1339** .

Can also be used as a **method** :

```
GetLnum()->prop_add(col, props)
```

```
prop_add_list()
```

```
prop_add_list({props}, [{item}, ...])
```

Similar to prop\_add(), but attaches a text property at multiple positions in a buffer.

**{props}** is a dictionary with these fields:

bufnr	buffer to add the property to; when omitted the current buffer is used
id	user defined ID for the property; must be a number; when omitted zero is used
type	name of the text property type

All fields except "type" are optional.

The second argument is a List of items, where each **{item}** is a list that specifies the starting and ending position of the text: **[{lnum}, {col}, {end-lnum}, {end-col}]**  
or: **[{lnum}, {col}, {end-lnum}, {end-col}, {id}]**

The first two items **{lnum}** and **{col}** specify the starting position of the text where the property will be attached. The next two items **{end-lnum}** and **{end-col}** specify the position just after the text.

An optional fifth item **{id}** can be used to give a different ID to a property. When omitted the ID from **{props}** is used, falling back to zero if none are present.

It is not possible to add a text property with a "text" field here.

Example:

```
call prop_add_list(#{type: 'MyProp', id: 2},
 \ [[1, 4, 1, 7],
 \ [1, 15, 1, 20],
 \ [2, 30, 3, 30]]
```

Can also be used as a `method` :

```
GetProp()->prop_add_list([[1, 1, 1, 2], [1, 4, 1, 8]])
```

```
prop_clear({lnum} [, {lnum-end} [, {props}]]) prop_clear()
```

Remove all text properties from line `{lnum}`.

When `{lnum-end}` is given, remove all text properties from line `{lnum}` to `{lnum-end}` (inclusive).

When `{props}` contains a "bufnr" item use this buffer, otherwise use the current buffer.

Can also be used as a `method` :

```
GetLnum()->prop_clear()
```

```
prop_find({props} [, {direction}]) prop_find()
```

Search for a text property as specified with `{props}`:

<code>id</code>	property with this ID
<code>type</code>	property with this type name
<code>both</code>	"id" and "type" must both match
<code>bufnr</code>	buffer to search in; when present a start position with "lnum" and "col" must be given; when omitted the current buffer is used
<code>lnum</code>	start in this line (when omitted start at the cursor)
<code>col</code>	start at this column (when omitted and "lnum" is given: use column 1, otherwise start at the cursor)
<code>skipstart</code>	do not look for a match at the start position

A property matches when either "id" or "type" matches. `{direction}` can be "f" for forward and "b" for backward. When omitted forward search is performed.

If a match is found then a Dict is returned with the entries as with `prop_list()`, and additionally an "lnum" entry.

If no match is found then an empty Dict is returned.

```
prop_list({lnum} [, {props}]) prop_list()
```

Returns a List with all the text properties in line `{lnum}`.

The following optional items are supported in `{props}`:

<code>bufnr</code>	use this buffer instead of the current buffer
<code>end_lnum</code>	return text properties in all the lines between <code>{lnum}</code> and <code>{end_lnum}</code> (inclusive). A negative value is used as an offset from the last buffer line; -1 refers to the last buffer line.
<code>types</code>	List of property type names. Return only text

ids properties that match one of the type names.  
List of property identifiers. Return only text properties with one of these identifiers.

The properties are ordered by starting column and priority.  
Each property is a Dict with these entries:

lnum	starting line number. Present only when returning text properties between {lnum} and {end_lnum}.
col	starting column
length	length in bytes, one more if line break is included
id	property ID
text	text to be displayed before {col}. Only present for <code>virtual-text</code> properties.
text_align	alignment property of <code>virtual-text</code> .
text_padding_left	left padding used for virtual text.
text_wrap	specifies whether <code>virtual-text</code> is wrapped.
type	name of the property type, omitted if the type was deleted
type_bufnr	buffer number for which this type was defined; 0 if the type is global
start	when TRUE property starts in this line
end	when TRUE property ends in this line

When "start" is zero the property started in a previous line, the current one is a continuation.

When "end" is zero the property continues in the next line. The line break after this line is included.

Returns an empty list on error.

Examples:

```
" get text properties placed in line 5
echo prop_list(5)
" get text properties placed in line 20 in buffer 4
echo prop_list(20, {'bufnr': 4})
" get all the text properties between line 1 and 20
echo prop_list(1, {'end_lnum': 20})
" get all the text properties of type 'myprop'
echo prop_list(1, {'types': ['myprop'],
 \ 'end_lnum': -1})
" get all the text properties of type 'prop1' or 'prop2'
echo prop_list(1, {'types': ['prop1', 'prop2'],
 \ 'end_lnum': -1})
" get all the text properties with ID 8
echo prop_list(1, {'ids': [8], 'end_lnum': line('$')})
" get all the text properties with ID 10 and 20
echo prop_list(1, {'ids': [10, 20], 'end_lnum': -1})
" get text properties with type 'myprop' and ID 100
" in buffer 4.
echo prop_list(1, {'bufnr': 4, 'types': ['myprop'],
 \ 'ids': [100], 'end_lnum': -1})
```

Can also be used as a [method](#) :  
[GetLnum\(\)->prop\\_list\(\)](#)

[prop\\_remove\(\)](#) [E968](#) [E860](#)  
`prop_remove({props} [, {lnum} [, {lnum-end}]])`  
Remove a matching text property from line `{lnum}`. When `{lnum-end}` is given, remove matching text properties from line `{lnum}` to `{lnum-end}` (inclusive).  
When `{lnum}` is omitted remove matching text properties from all lines (this requires going over all lines, thus will be a bit slow for a buffer with many lines).

`{props}` is a dictionary with these fields:

<code>id</code>	remove text properties with this ID
<code>type</code>	remove text properties with this type name
<code>types</code>	remove text properties with type names in this List
<code>both</code>	"id" and "type"/"types" must both match
<code>bufnr</code>	use this buffer instead of the current one
<code>all</code>	when TRUE remove all matching text properties, not just the first one

Only one of "type" and "types" may be supplied. [E1295](#)

A property matches when either "id" or one of the supplied types matches.

If buffer "bufnr" does not exist you get an error message.  
If buffer "bufnr" is not loaded then nothing happens.

Returns the number of properties that were removed.

Can also be used as a [method](#) :  
[GetProps\(\)->prop\\_remove\(\)](#)

`prop_type_add({name}, {props})` [prop\\_type\\_add\(\)](#) [E969](#) [E970](#)  
Add a text property type `{name}`. If a property type with this name already exists an error is given. Nothing is returned.  
`{props}` is a dictionary with these optional fields:

<code>bufnr</code>	define the property only for this buffer; this avoids name collisions and automatically clears the property types when the buffer is deleted.
<code>highlight</code>	name of highlight group to use
<code>priority</code>	when a character has multiple text properties the one with the highest priority will be used; negative values can be used, the default priority is zero
<code>combine</code>	when omitted or TRUE combine the highlight with any syntax highlight; when FALSE syntax highlight will not be used
<code>override</code>	when TRUE the highlight overrides any other, including ' <a href="#">cursorline</a> ' and Visual
<code>start_incl</code>	when TRUE inserts at the start position will

end\_incl be included in the text property  
when TRUE inserts at the end position will be  
included in the text property

Can also be used as a `method` :  
`GetPropName()->prop_type_add(props)`

`prop_type_change({name}, {props})` `prop_type_change()`  
Change properties of an existing text property type. If a property with this name does not exist an error is given. The `{props}` argument is just like `prop_type_add()`.

Can also be used as a `method` :  
`GetPropName()->prop_type_change(props)`

`prop_type_delete({name} [, {props}])` `prop_type_delete()`  
Remove the text property type `{name}`. When text properties using the type `{name}` are still in place, they will not have an effect and can no longer be removed by name.

`{props}` can contain a "bufnr" item. When it is given, delete a property type from this buffer instead of from the global property types.

When text property type `{name}` is not found there is no error.

Can also be used as a `method` :  
`GetPropName()->prop_type_delete()`

`prop_type_get({name} [, {props}])` `prop_type_get()`  
Returns the properties of property type `{name}`. This is a dictionary with the same fields as was given to `prop_type_add()`. When the property type `{name}` does not exist, an empty dictionary is returned.

`{props}` can contain a "bufnr" item. When it is given, use this buffer instead of the global property types.

Can also be used as a `method` :  
`GetPropName()->prop_type_get()`

`prop_type_list([{props}])` `prop_type_list()`  
Returns a list with all property type names.

`{props}` can contain a "bufnr" item. When it is given, use this buffer instead of the global property types.

---

### 3. When text changes

### text-prop-changes

Vim will do its best to keep the text properties on the text where it was attached. When inserting or deleting text the properties after the change

will move accordingly.

When text is deleted and a text property no longer includes any text, it is deleted. However, a text property that was defined as zero-width will remain, unless the whole line is deleted.

E275

When a buffer is unloaded, all the text properties are gone. There is no way to store the properties in a file. You can only re-create them. When a buffer is hidden the text is preserved and so are the text properties. It is not possible to add text properties to an unloaded buffer.

When using replace mode, the text properties stay on the same character positions, even though the characters themselves change.

To update text properties after the text was changed, install a callback with `listener_add()`. E.g, if your plugin does spell checking, you can have the callback update spelling mistakes in the changed text. Vim will move the properties below the changed text, so that they still highlight the same text, thus you don't need to update these.

Text property columns are not updated or copied:

- When setting the line with `setline()` or through an interface, such as Lua, Tcl or Python. Vim does not know what text got inserted or deleted.
- With a command like `:move`, which takes a line of text out of context.



Filetypes [filetype](#) [file-type](#)

1. Filetypes [filetypes](#)
2. Filetype plugin [filetype-plugins](#)
3. Docs for the default filetype plugins. [ftplugin-docs](#)

Also see [autocmd.txt](#) .

---

## 1. Filetypes [filetypes](#) [file-types](#)

Vim can detect the type of file that is edited. This is done by checking the file name and sometimes by inspecting the contents of the file for specific text.

To enable file type detection, use this command in your vimrc:  
`:filetype on`

Each time a new or existing file is edited, Vim will try to recognize the type of the file and set the **'filetype'** option. This will trigger the FileType event, which can be used to set the syntax highlighting, set options, etc.

**NOTE:** Filetypes and **'compatible'** don't work together well, since being Vi compatible means options are global. Resetting **'compatible'** is recommended, if you didn't do that already.

Detail: The `":filetype on"` command will load one of these files:

Amiga	\$VIMRUNTIME/filetype.vim
Mac	\$VIMRUNTIME:filetype.vim
MS-Windows	\$VIMRUNTIME\filetype.vim
Unix	\$VIMRUNTIME/filetype.vim
VMS	\$VIMRUNTIME/filetype.vim

This file is a Vim script that defines autocommands for the BufNewFile and BufRead events. If the file type is not found by the name, the file \$VIMRUNTIME/scripts.vim is used to detect it from the contents of the file.

When the GUI is running or will start soon, the [menu.vim](#) script is also sourced. See ['go-M'](#) about avoiding that.

To add your own file types, see [new-filetype](#) below. To search for help on a filetype prepend "ft-" and optionally append "--syntax", "--indent" or "--plugin". For example:

```
:help ft-vim-indent
:help ft-vim-syntax
:help ft-man-plugin
```

If the file type is not detected automatically, or it finds the wrong type,

you can either set the **'filetype'** option manually, or add a modeline to your file. Example, for an IDL file use the command:

```
:set filetype=idl
```

or add this **modeline** to the file:

```
/* vim: set filetype=idl : */
```

**:filetype-plugin-on**

You can enable loading the plugin files for specific file types with:

```
:filetype plugin on
```

If filetype detection was not switched on yet, it will be as well.

This actually loads the file "ftplugin.vim" in **'runtimepath'**.

The result is that when a file is edited its plugin file is loaded (if there is one for the detected filetype). **filetype-plugin**

**:filetype-plugin-off**

You can disable it again with:

```
:filetype plugin off
```

The filetype detection is not switched off then. But if you do switch off filetype detection, the plugins will not be loaded either.

This actually loads the file "ftplugof.vim" in **'runtimepath'**.

**:filetype-indent-on**

You can enable loading the indent file for specific file types with:

```
:filetype indent on
```

If filetype detection was not switched on yet, it will be as well.

This actually loads the file "indent.vim" in **'runtimepath'**.

The result is that when a file is edited its indent file is loaded (if there is one for the detected filetype). **indent-expression**

**:filetype-indent-off**

You can disable it again with:

```
:filetype indent off
```

The filetype detection is not switched off then. But if you do switch off filetype detection, the indent files will not be loaded either.

This actually loads the file "indoff.vim" in **'runtimepath'**.

This disables auto-indenting for files you will open. It will keep working in already opened files. Reset **'autoindent'**, **'cindent'**, **'smartindent'** and/or **'indentexpr'** to disable indenting in an opened file.

**:filetype-off**

To disable file type detection, use this command:

```
:filetype off
```

This will keep the flags for "plugin" and "indent", but since no file types are being detected, they won't work until the next **":filetype on"**.

Overview:

**:filetype-overview**

<b>command</b>	<b>detection</b>	<b>plugin</b>	<b>indent</b>
<b>:filetype on</b>	on	unchanged	unchanged
<b>:filetype off</b>	off	unchanged	unchanged
<b>:filetype plugin on</b>	on	on	unchanged
<b>:filetype plugin off</b>	unchanged	off	unchanged

```

:filetype indent on on unchanged on
:filetype indent off unchanged unchanged off
:filetype plugin indent on on on on
:filetype plugin indent off unchanged off off

```

To see the current status, type:

```
:filetype
```

The output looks something like this:

```
filetype detection:ON plugin:ON indent:OFF
```

The file types are also used for syntax highlighting. If the ":syntax on" command is used, the file type detection is installed too. There is no need to do ":filetype on" after ":syntax on".

To disable one of the file types, add a line in your filetype file, see [remove-filetype](#) .

### filetype-detect

To detect the file type again:

```
:filetype detect
```

Use this if you started with an empty file and typed text that makes it possible to detect the file type. For example, when you entered this in a shell script: "#!/bin/csh".

When filetype detection was off, it will be enabled first, like the "on" argument was used.

### filetype-override

When the same extension is used for multiple filetypes, Vim tries to guess what kind of file it is. This doesn't always work. A number of global variables can be used to override the filetype used for certain extensions:

file name	variable	
*.asa	g:filetype_asa	ft-asperl-syntax ft-aspvbs-syntax
*.asm	g:asmsyntax	ft-asm-syntax
*.asp	g:filetype_asp	ft-asperl-syntax ft-aspvbs-syntax
*.bas	g:filetype_bas	ft-basic-syntax
*.cfg	g:filetype_cfg	
*.cls	g:filetype_cls	
*.csh	g:filetype_csh	ft-csh-syntax
*.dat	g:filetype_dat	
*.def	g:filetype_def	
*.f	g:filetype_f	ft-forth-syntax
*.frm	g:filetype_frm	ft-form-syntax
*.fs	g:filetype_fs	ft-forth-syntax
*.h	g:c_syntax_for_h	ft-c-syntax
*.i	g:filetype_i	ft-progress-syntax
*.inc	g:filetype_inc	
*.lsl	g:filetype_lsl	
*.m	g:filetype_m	ft-mathematica-syntax
*.mod	g:filetype_mod	
*.p	g:filetype_p	ft-pascal-syntax
*.pl	g:filetype_pl	

```

*.pp g:filetype_pp ft-pascal-syntax
*.prg g:filetype_prg
*.r g:filetype_r
*.sig g:filetype_sig
*.sql g:filetype_sql ft-sql-syntax
*.src g:filetype_src
*.sys g:filetype_sys
*.sh g:bash_is_sh ft-sh-syntax
*.tex g:tex_flavor ft-tex-plugin
*.typ g:filetype_typ
*.v g:filetype_v
*.w g:filetype_w ft-cweb-syntax

```

For a few filetypes the global variable is used only when the filetype could not be detected:

```

*.r g:filetype_r ft-rexx-syntax

```

### filetype-ignore

To avoid that certain files are being inspected, the `g:ft_ignore_pat` variable is used. The default value is set like this:

```

:let g:ft_ignore_pat = '\.(Z\|gz\|bz2\|zip\|tgz\)$'

```

This means that the contents of compressed files are not inspected.

### new-filetype

If a file type that you want to use is not detected yet, there are four ways to add it. In any way, it's better not to modify the `$VIMRUNTIME/filetype.vim` file. It will be overwritten when installing a new version of Vim.

A. If you want to overrule all default file type checks.

This works by writing one file for each filetype. The disadvantage is that there can be many files. The advantage is that you can simply drop this file in the right directory to make it work.

### ftdetect

1. Create your user runtime directory. You would normally use the first item of the `'runtimepath'` option. Then create the directory "ftdetect" inside it. Example for Unix:

```

:!mkdir ~/.vim
:!mkdir ~/.vim/ftdetect

```

2. Create a file that contains an autocommand to detect the file type. Example:

```

au BufRead,BufNewFile *.mine set filetype=mine

```

**Note** that there is no "augroup" command, this has already been done when sourcing your file. You could also use the pattern "\*" and then check the contents of the file to recognize it.

Write this file as "mine.vim" in the "ftdetect" directory in your user runtime directory. For example, for Unix:

```

:w ~/.vim/ftdetect/mine.vim

```

3. To use the new filetype detection you must restart Vim.

The files in the "ftdetect" directory are used after all the default checks, thus they can overrule a previously detected file type. But you can also use `:setfiletype` to keep a previously detected filetype.

- B. If you want to detect your file after the default file type checks.

This works like A above, but instead of setting **'filetype'** unconditionally use `":setfiletype"`. This will only set **'filetype'** if no file type was detected yet. Example:

```
au BufRead,BufNewFile *.txt setfiletype text
```

You can also use the already detected file type in your command. For example, to use the file type "mypascal" when "pascal" has been detected:

```
au BufRead,BufNewFile * if &ft == 'pascal' | set ft=mypascal
 | endif
```

- C. If your file type can be detected by the file name.
1. Create your user runtime directory. You would normally use the first item of the **'runtimepath'** option. Example for Unix:

```
:!mkdir ~/.vim
```

2. Create a file that contains autocommands to detect the file type.

Example:

```
" my filetype file
if exists("did_load_filetypes")
 finish
endif
augroup filetypedetect
 au! BufRead,BufNewFile *.mine setfiletype mine
 au! BufRead,BufNewFile *.xyz setfiletype drawing
augroup END
```

Write this file as "filetype.vim" in your user runtime directory. For example, for Unix:

```
:w ~/.vim/filetype.vim
```

3. To use the new filetype detection you must restart Vim.

Your filetype.vim will be sourced before the default FileType autocommands have been installed. Your autocommands will match first, and the `":setfiletype"` command will make sure that no other autocommands will set **'filetype'** after this.

[new-filetype-scripts](#)

- D. If your filetype can only be detected by inspecting the contents of the file.

1. Create your user runtime directory. You would normally use the first item of the **'runtimepath'** option. Example for Unix:

```
:!mkdir ~/.vim
```

2. Create a vim script file for doing this. Example:

```
if did_filetype() " filetype already set..
 finish " ..don't do these checks
endif
if getline(1) =~ '^#!.*\<mine\>'
 setfiletype mine
elseif getline(1) =~? '\<drawing\>'
 setfiletype drawing
```

```
endif
```

See \$VIMRUNTIME/scripts.vim for more examples.

Write this file as "scripts.vim" in your user runtime directory. For example, for Unix:

```
:w ~/.vim/scripts.vim
```

3. The detection will work right away, no need to restart Vim.

Your scripts.vim is loaded before the default checks for file types, which means that your rules override the default rules in \$VIMRUNTIME/scripts.vim.

#### remove-filetype

If a file type is detected that is wrong for you, install a filetype.vim or scripts.vim to catch it (see above). You can set **'filetype'** to a non-existing name to avoid that it will be set later anyway:

```
:set filetype=ignored
```

If you are setting up a system with many users, and you don't want each user to add/remove the same filetypes, consider writing the filetype.vim and scripts.vim files in a runtime directory that is used for everybody. Check the **'runtimepath'** for a directory to use. If there isn't one, set **'runtimepath'** in the `system-vimrc`. Be careful to keep the default directories!

#### autocmd-osfiletypes

**NOTE:** this code is currently disabled, as the RISC OS implementation was removed. In the future this will use the **'filetype'** option.

On operating systems which support storing a file type with the file, you can specify that an autocommand should only be executed if the file is of a certain type.

The actual type checking depends on which platform you are running Vim on; see your system's documentation for details.

To use osfiletype checking in an autocommand you should put a list of types to match in angle brackets in place of a pattern, like this:

```
:au BufRead *.html,<&faf;HTML> runtime! syntax/html.vim
```

This will match:

- Any file whose name ends in ".html"
- Any file whose type is "&faf" or "HTML", where the meaning of these types depends on which version of Vim you are using. Unknown types are considered NOT to match.

You can also specify a type and a pattern at the same time (in which case they must both match):

```
:au BufRead <&fff>diff*
```

This will match files of type "&fff" whose names start with "diff".

### plugin-details

The "plugin" directory can be in any of the directories in the 'runtimepath' option. All of these directories will be searched for plugins and they are all loaded. For example, if this command:

```
set runtimepath
```

produces this output:

```
runtimepath=/etc/vim,~/.vim,/usr/local/share/vim/vim82
```

then Vim will load all plugins in these directories and below:

```
/etc/vim/plugin/
~/.vim/plugin/
/usr/local/share/vim/vim82/plugin/
```

Note that the last one is the value of \$VIMRUNTIME which has been expanded.

Note that when using a plugin manager or packages many directories will be added to 'runtimepath'. These plugins each require their own directory, don't put them directly in ~/.vim/plugin.

What if it looks like your plugin is not being loaded? You can find out what happens when Vim starts up by using the -V argument:

```
vim -V2
```

You will see a lot of messages, in between them is a remark about loading the plugins. It starts with:

```
Searching for "plugin/**/*.vim" in
```

There you can see where Vim looks for your plugin scripts.

---

## 2. Filetype plugin filetype-plugins

When loading filetype plugins has been enabled :filetype-plugin-on , options will be set and mappings defined. These are all local to the buffer, they will not be used for other files.

Defining mappings for a filetype may get in the way of the mappings you define yourself. There are a few ways to avoid this:

1. Set the "maplocalleader" variable to the key sequence you want the mappings to start with. Example:

```
:let maplocalleader = ","
```

All mappings will then start with a comma instead of the default, which is a backslash. Also see <LocalLeader> .

2. Define your own mapping. Example:

```
:map ,p <Plug>MailQuote
```

You need to check the description of the plugin file below for the functionality it offers and the string to map to.

You need to define your own mapping before the plugin is loaded (before editing a file of that type). The plugin will then skip installing the default mapping.

```
no_mail_maps g:no_mail_maps
```

3. Disable defining mappings for a specific filetype by setting a variable, which contains the name of the filetype. For the "mail" filetype this would be:

```
:let no_mail_maps = 1
```

```
no_plugin_maps g:no_plugin_maps
```

4. Disable defining mappings for all filetypes by setting a variable:

```
:let no_plugin_maps = 1
```

```
ftplugin-override
```

If a global filetype plugin does not do exactly what you want, there are three ways to change this:

1. Add a few settings.

You must create a new filetype plugin in a directory early in `'runtimepath'`. For Unix, for example you could use this file:

```
vim ~/.vim/ftplugin/fortran.vim
```

You can set those settings and mappings that you would like to add. **Note** that the global plugin will be loaded after this, it may override the settings that you do here. If this is the case, you need to use one of the following two methods.

2. Make a copy of the plugin and change it.

You must put the copy in a directory early in `'runtimepath'`. For Unix, for example, you could do this:

```
cp $VIMRUNTIME/ftplugin/fortran.vim ~/.vim/ftplugin/fortran.vim
```

Then you can edit the copied file to your liking. Since the `b:did_ftplugin` variable will be set, the global plugin will not be loaded.

A disadvantage of this method is that when the distributed plugin gets improved, you will have to copy and modify it again.

3. Override the settings after loading the global plugin.

You must create a new filetype plugin in a directory from the end of `'runtimepath'`. For Unix, for example, you could use this file:

```
vim ~/.vim/after/ftplugin/fortran.vim
```

In this file you can change just those settings that you want to change.

- ```
=====
```
3. Docs for the default filetype plugins.

```
ftplugin-docs
```

```
plugin_exec g:plugin_exec
```

Enable executing of external commands. This was done historically for e.g. the perl filetype plugin (and a few others) to set the search path.

Disabled by default for security reasons:

```
:let g:plugin_exec = 1
```

It is also possible to enable this only for certain filetypes:


```
:let g:<filetype>_exec = 1
```

So to enable this only for ruby, set the following variable:

```
:let g:ruby_exec = 1
```

If both, the global ``plugin_exec`` and the ``<filetype>_exec`` specific variable are set, the filetype specific variable should have precedent.

AWK

ft-awk-plugin

Support for features specific to GNU Awk, like `@include`, can be enabled by setting:

```
:let g:awk_is_gawk = 1
```

CHANGELOG

ft-changelog-plugin

Allows for easy entrance of Changelog entries in Changelog files. There are some commands, mappings, and variables worth exploring:

Options:

| | |
|------------------------------|--|
| <code>'comments'</code> | is made empty to not mess up formatting. |
| <code>'textwidth'</code> | is set to 78, which is standard. |
| <code>'formatoptions'</code> | the 't' flag is added to wrap when inserting text. |

Commands:

| | |
|--------------------------------|---|
| <code>NewChangelogEntry</code> | Adds a new Changelog entry in an intelligent fashion (see below). |
|--------------------------------|---|

Local mappings:

| | |
|------------------------------|---|
| <code><Leader>o</code> | Starts a new Changelog entry in an equally intelligent fashion (see below). |
|------------------------------|---|

Global mappings:

| | |
|------------------------------|--|
| | NOTE: The global mappings are accessed by sourcing the <code>ftplugin/changelog.vim</code> file first, e.g. with <pre>runtime ftplugin/changelog.vim</pre> in your <code>.vimrc</code> . |
| <code><Leader>o</code> | Switches to the ChangeLog buffer opened for the current directory, or opens it in a new buffer if it exists in the current directory. Then it does the same as the local <code><Leader>o</code> described above. |

Variables:

| | |
|-------------------------------------|---|
| <code>g:changelog_timeformat</code> | Deprecated; use <code>g:changelog_dateformat</code> instead. |
| <code>g:changelog_dateformat</code> | The date (and time) format used in ChangeLog entries. The format accepted is the same as for the <code>strftime()</code> function. The default is <code>"%Y-%m-%d"</code> which is the standard format for many ChangeLog layouts. |
| <code>g:changelog_username</code> | The name and email address of the user. The default is deduced from environment variables and system files. It searches <code>/etc/passwd</code> for the comment part of the current user, which informally contains the real name of the user up to the first separating |

comma. then it checks the \$NAME environment variable and finally runs `whoami` and `hostname` to build an email address. The final form is

```
Full Name <user@host>
```

g:changelog_new_date_format

The format to use when creating a new date-entry. The following table describes special tokens in the string:

| | |
|----|---|
| %% | insert a single '%' character |
| %d | insert the date from above |
| %u | insert the user from above |
| %p | insert result of b:changelog_entry_prefix |
| %c | where to position cursor when done |

The default is "%d %u\n\n\t* %p%c\n\n", which produces something like (| is where cursor will be, unless at the start of the line where it denotes the beginning of the line)

```
|2003-01-14 Full Name <user@host>
|
| * prefix|
```

g:changelog_new_entry_format

The format used when creating a new entry. The following table describes special tokens in the string:

| | |
|----|---|
| %p | insert result of b:changelog_entry_prefix |
| %c | where to position cursor when done |

The default is "\t*%c", which produces something similar to

```
| * prefix|
```

g:changelog_date_entry_search

The search pattern to use when searching for a date-entry. The same tokens that can be used for g:changelog_new_date_format can be used here as well. The default is '^\\s*%d_s*%u' which finds lines matching the form

```
|2003-01-14 Full Name <user@host>
```

and some similar formats.

g:changelog_date_end_entry_search

The search pattern to use when searching for the end of a date-entry. The same tokens that can be used for g:changelog_new_date_format can be used here as well. The default is '^\\s*\$' which finds lines that contain only whitespace or are completely empty.

b:changelog_name

b:changelog_name

Name of the ChangeLog file to look for. The default is 'ChangeLog'.

`b:changelog_path`

Path of the ChangeLog to use for the current buffer. The default is empty, thus looking for a file named `b:changelog_name` in the same directory as the current buffer. If not found, the parent directory of the current buffer is searched. This continues recursively until a file is found or there are no more parent directories to search.

`b:changelog_entry_prefix`

Name of a function to call to generate a prefix to a new entry. This function takes no arguments and should return a string containing the prefix. Returning an empty prefix is fine. The default generates the shortest path between the ChangeLog's pathname and the current buffers pathname. In the future, it will also be possible to use other variable contexts for this variable, for example, `g:`.

The Changelog entries are inserted where they add the least amount of text. After figuring out the current date and user, the file is searched for an entry beginning with the current date and user and if found adds another item under it. If not found, a new entry and item is prepended to the beginning of the Changelog.

ASCIIDOC

[ft-asciidoc-plugin](#)

To enable `ft-asciidoc` use this:

```
let g:asciidoc_folding = 1
```

To disable nesting of folded headers use this:

```
let g:asciidoc_foldnested = 0
```

To disable folding everything under the title use this:

```
let asciidoc_fold_under_title = 0
```

FORTRAN

[ft-fortran-plugin](#)

Options:

'expandtab' is switched on to avoid tabs as required by the Fortran standards unless the user has set `fortran_have_tabs` in `.vimrc`.
'textwidth' is set to 80 for fixed source format whereas it is set to 132 for free source format. Setting the `fortran_extended_line_length` variable increases the width to 132 for fixed source format.
'formatoptions' is set to break code and comment lines and to preserve long lines. You can format comments with `gq`.

For further discussion of `fortran_have_tabs` and the method used for the detection of source format see [ft-fortran-syntax](#).

FREEBASIC

[ft-freebasic-plugin](#)

This plugin aims to treat the four FreeBASIC dialects, "fb", "qb", "fblite" and "deprecated", as distinct languages.

The dialect will be set to the first name found in `g:freebasic_forcelang`, any `#lang` directive or `$lang` metacommand in the file being edited, or finally `g:freebasic_lang`. These global variables conceptually map to the fbc options `-forcelang` and `-lang`. If no dialect is explicitly specified "fb" will be used.

For example, to set the dialect to a default of "fblite" but still allow for any `#lang` directive overrides, use the following command:

```
let g:freebasic_lang = "fblite"
```

GIT COMMIT

[ft-gitcommit-plugin](#)

One command, `:DiffGitCached`, is provided to show a diff of the current commit in the preview window. It is equivalent to calling `"git diff --cached"` plus any arguments given to the command.

GPROF

[ft-gprof-plugin](#)

The `gprof` filetype plugin defines a mapping `<C-]>` to jump from a function entry in the `gprof` flat profile or from a function entry in the call graph to the details of that function in the call graph.

The mapping can be disabled with:

```
let g:no_gprof_maps = 1
```

JSON-FORMAT

[ft-json-plugin](#)

JSON filetype can be extended to use `'formatexpr'` and `"json.FormatExpr()"` function for json formatting (using `gq`).

Add following lines to `$HOME/.vim/ftplugin/json.vim`:

```
vim9script
import autoload 'dist/json.vim'
setl formatexpr=json.FormatExpr()
```

MAIL

[ft-mail-plugin](#)

Options:

- `'modeline'` is switched off to avoid the danger of trojan horses, and to avoid that a Subject line with "Vim:" in it will cause an error message.
- `'textwidth'` is set to 72. This is often recommended for e-mail.
- `'formatoptions'` is set to break text lines and to repeat the comment leader in new lines, so that a leading ">" for quotes is repeated. You can also format quoted text with `gq`.

Local mappings:

```
<LocalLeader>q or \\MailQuote
```

Quotes the text selected in Visual mode, or from the cursor position to the end of the file in Normal mode. This means "> " is inserted in each line.

MAN

```
ft-man-plugin :Man man.vim
```

This plugin displays a manual page in a nice way. See `find-manpage` in the user manual for more information.

To start using the `:Man` command before any manual page has been loaded, source this script from your startup `vimrc` file:

```
runtime ftplugin/man.vim
```

Options:

'iskeyword' The '.' character is added to support the use of **CTRL-]** on the manual page name.

Commands:

```
Man {name} Display the manual page for {name} in a window.
```

```
Man {number} {name}
```

Display the manual page for {name} in a section {number}.

Global mapping:

```
<Leader>K Displays the manual page for the word under the cursor.
```

```
<Plug>ManPreGetPage
```

idem, allows for using a mapping:

```
nmap <F1> <Plug>ManPreGetPage
```

Local mappings:

```
CTRL-] Jump to the manual page for the word under the cursor.
```

```
CTRL-T Jump back to the previous manual page.
```

```
q Same as the :quit command.
```

To use a vertical split instead of horizontal:

```
let g:ft_man_open_mode = 'vert'
```

To use a new tab:

```
let g:ft_man_open_mode = 'tab'
```

To enable `foldings`, use this:

```
let g:ft_man_folding_enable = 1
```

If you do not like the default folding, use an `autocommand` to add your desired folding style instead. For example:

```
autocmd FileType man setlocal foldmethod=indent foldenable
```

If you would like `:Man {number} {name}` to behave like `man {number} {name}` by not running `man {name}` if no page is found, then use this:

```
let g:ft_man_no_sect_fallback = 1
```

You may also want to set **'keywordprg'** to make the `K` command open a manual page in a Vim window:

```
set keywordprg=:Man
```

MANPAGER

manpager.vim

The `:Man` command allows you to turn Vim into a manpager (that syntax highlights manpages and follows linked manpages on hitting **CTRL-]**).

For bash,zsh,ksh or dash, add to the config file (`.bashrc`,`.zshrc`, ...)

```
export MANPAGER="vim +MANPAGER --not-a-term -"
```

For (t)csch, add to the config file

```
setenv MANPAGER "vim +MANPAGER --not-a-term -"
```

For fish, add to the config file

```
set -x MANPAGER "vim +MANPAGER --not-a-term -"
```

MARKDOWN

ft-markdown-plugin

To enable folding use this:

```
let g:markdown_folding = 1
```

'**expandtab**' will be set by default. If you do not want that use this:

```
let g:markdown_recommended_style = 0
```

PDF

ft-pdf-plugin

Two maps, `<C-]>` and `<C-T>`, are provided to simulate a tag stack for navigating the PDF. The following are treated as tags:

- The byte offset after "startxref" to the xref table
- The byte offset after the /Prev key in the trailer to an earlier xref table
- A line of the form "0123456789 00000 n" in the xref table
- An object reference like "1 0 R" anywhere in the PDF

These maps can be disabled with

```
:let g:no_pdf_maps = 1
```

PYTHON

ft-python-plugin PEP8

By default the following options are set, in accordance with PEP8:

```
setlocal expandtab shiftwidth=4 softtabstop=4 tabstop=8
```

To disable this behavior, set the following variable in your vimrc:

```
let g:python_recommended_style = 0
```

QF QUICKFIX

qf.vim ft-qf-plugin

The "qf" filetype is used for the quickfix window, see [quickfix-window](#) .

The quickfix filetype plugin includes configuration for displaying the command that produced the quickfix list in the [status-line](#) . To disable this setting, configure as follows:

```
:let g:qf_disable_statusline = 1
```

R MARKDOWN

ft-rmd-plugin

By default ftplugin/html.vim is not sourced. If you want it sourced, add to your `vimrc` :

```
let rmd_include_html = 1
```

The `'formatexpr'` option is set dynamically with different values for R code and for Markdown code. If you prefer that `'formatexpr'` is not set, add to your `vimrc` :

```
let rmd_dynamic_comments = 0
```

R RESTRUCTURED TEXT

ft-rrst-plugin

The `'formatexpr'` option is set dynamically with different values for R code and for ReStructured text. If you prefer that `'formatexpr'` is not set, add to your `vimrc` :

```
let rrst_dynamic_comments = 0
```

RESTRUCTUREDTEXT

ft-rst-plugin

The following formatting setting are optionally available:

```
setlocal expandtab shiftwidth=3 softtabstop=3 tabstop=8
```

To enable this behavior, set the following variable in your `vimrc`:

```
let g:rst_style = 1
```

RNOWEB

ft-rnoweb-plugin

The `'formatexpr'` option is set dynamically with different values for R code and for LaTeX code. If you prefer that `'formatexpr'` is not set, add to your `vimrc` :

```
let rnw_dynamic_comments = 0
```

RPM SPEC

ft-spec-plugin

Since the text for this plugin is rather long it has been put in a separate file: [pi_spec.txt](#) .

RUST

ft-rust

Since the text for this plugin is rather long it has been put in a separate file: `ft_rust.txt` .

SQL

`ft-sql`

Since the text for this plugin is rather long it has been put in a separate file: `ft_sql.txt` .

TEX

`ft-tex-plugin` `g:tex_flavor`

If the first line of a *.tex file has the form

```
%<format>
```

then this determined the file type: `plaintex` (for plain TeX), `context` (for ConTeXt), or `tex` (for LaTeX). Otherwise, the file is searched for keywords to choose `context` or `tex`. If no keywords are found, it defaults to `plaintex`.

You can change the default by defining the variable `g:tex_flavor` to the format (not the file type) you use most. Use one of these:

```
let g:tex_flavor = "plain"
let g:tex_flavor = "context"
let g:tex_flavor = "latex"
```

Currently no other formats are recognized.

VIM

`ft-vim-plugin`

The Vim filetype plugin defines mappings to move to the start and end of functions with `[[` and `]]`. Move around comments with `]"` and `["`.

The mappings can be disabled with:

```
let g:no_vim_maps = 1
```

ZIMBU

`ft-zimbu-plugin`

The Zimbu filetype plugin defines mappings to move to the start and end of functions with `[[` and `]]`.

The mappings can be disabled with:

```
let g:no_zimbu_maps = 1
```


This subject is introduced in section 30.1 of the user manual.

- | | |
|---------------------------------------|--------------------------|
| 1. Using QuickFix commands | quickfix |
| 2. The error window | quickfix-window |
| 3. Using more than one list of errors | quickfix-error-lists |
| 4. Using :make | :make_makeprg |
| 5. Using :grep | grep |
| 6. Selecting a compiler | compiler-select |
| 7. The error format | error-file-format |
| 8. The directory stack | quickfix-directory-stack |
| 9. Specific error file formats | errorformats |
| 10. Customizing the quickfix window | quickfix-window-function |

The quickfix commands are not available when the +quickfix feature was disabled at compile time.

1. Using QuickFix commands quickfix Quickfix E42

Vim has a special mode to speedup the edit-compile-edit cycle. This is inspired by the quickfix option of the Manx's Aztec C compiler on the Amiga. The idea is to save the error messages from the compiler in a file and use Vim to jump to the errors one by one. You can examine each problem and fix it, without having to remember all the error messages.

In Vim the quickfix commands are used more generally to find a list of positions in files. For example, :vimgrep finds pattern matches. You can use the positions in a script with the getqflist() function. Thus you can do a lot more than the edit/compile/fix cycle!

If you have the error messages in a file you can start Vim with:

```
vim -q filename
```

From inside Vim an easy way to run a command and handle the output is with the :make command (see below).

The 'errorformat' option should be set to match the error messages from your compiler (see errorformat below).

quickfix-ID

Each quickfix list has a unique identifier called the quickfix ID and this number will not change within a Vim session. The getqflist() function can be used to get the identifier assigned to a list. There is also a quickfix list number which may change whenever more than ten lists are added to a quickfix stack.

location-list E776

A location list is a window-local quickfix list. You get one after commands

like `:lvimgrep`, `:lgrep`, `:lhelpgrep`, `:lmake`, etc., which create a location list instead of a quickfix list as the corresponding `:vimgrep`, `:grep`, `:helpgrep`, `:make` do.

location-list-file-window

A location list is associated with a window and each window can have a separate location list. A location list can be associated with only one window. The location list is independent of the quickfix list.

When a window with a location list is split, the new window gets a copy of the location list. When there are no longer any references to a location list, the location list is destroyed.

quickfix-changedtick

Every quickfix and location list has a read-only `changedtick` variable that tracks the total number of changes made to the list. Every time the quickfix list is modified, this count is incremented. This can be used to perform an action only when the list has changed. The `getqflist()` and `getloclist()` functions can be used to query the current value of `changedtick`. You cannot change the `changedtick` variable.

The following quickfix commands can be used. The location list commands are similar to the quickfix commands, replacing the 'c' prefix in the quickfix command with 'l'.

E924

If the current window was closed by an `autocommand` while processing a location list command, it will be aborted.

E925 E926

If the current quickfix or location list was changed by an `autocommand` while processing a quickfix or location list command, it will be aborted.

:cc

`:cc[!] [nr]`
`:[nr]cc[!]`

Display error `[nr]`. If `[nr]` is omitted, the same error is displayed again. Without `[!]` this doesn't work when jumping to another buffer, the current buffer has been changed, there is the only window for the buffer and both `'hidden'` and `'autowrite'` are off. When jumping to another buffer with `[!]` any changes to the current buffer are lost, unless `'hidden'` is set or there is another window for this buffer. The `'switchbuf'` settings are respected when jumping to a buffer. When used in the quickfix window the line number can be used, including "." for the current line and "\$" for the last line.

:ll

`:ll[!] [nr]`
`:[nr]ll[!]`

Same as `":cc"`, except the location list for the current window is used instead of the quickfix list.

:cn :cne :cnext E553

`:[count]cn[ext][!]`

Display the `[count]` next error in the list that includes a file name. If there are no file names at

all, go to the [count] next error. See :cc for [!] and 'switchbuf'.

:[count]lne[xt][!] :lne :lnext
Same as ":cnext", except the location list for the current window is used instead of the quickfix list.

:[count]cN[ext][!] :cp :cprevious :cprev :cN :cNext
:[count]cp[revious][!] Display the [count] previous error in the list that includes a file name. If there are no file names at all, go to the [count] previous error. See :cc for [!] and 'switchbuf'.

:[count]lN[ext][!] :lp :lprevious :lprev :lN :lNext
:[count]lp[revious][!] Same as ":cNext" and ":cprevious", except the location list for the current window is used instead of the quickfix list.

:[count]cabo[ve] :cabo :cabove
Go to the [count] error above the current line in the current buffer. If [count] is omitted, then 1 is used. If there are no errors, then an error message is displayed. Assumes that the entries in a quickfix list are sorted by their buffer number and line number. If there are multiple errors on the same line, then only the first entry is used. If [count] exceeds the number of entries above the current line, then the first error in the file is selected.

:[count]lab[ove] :lab :labove
Same as ":cabove", except the location list for the current window is used instead of the quickfix list.

:[count]cbel[ow] :cbel :cbelow
Go to the [count] error below the current line in the current buffer. If [count] is omitted, then 1 is used. If there are no errors, then an error message is displayed. Assumes that the entries in a quickfix list are sorted by their buffer number and line number. If there are multiple errors on the same line, then only the first entry is used. If [count] exceeds the number of entries below the current line, then the last error in the file is selected.

:[count]lbel[ow] :lbel :lbelow
Same as ":cbelow", except the location list for the current window is used instead of the quickfix list.

:[count]cbe[fore] :cbe :cbefore
Go to the [count] error before the current cursor position in the current buffer. If [count] is omitted, then 1 is used. If there are no errors, then an error message is displayed. Assumes that the

entries in a quickfix list are sorted by their buffer, line and column numbers. If [count] exceeds the number of entries before the current position, then the first error in the file is selected.

- `: [count] lbe[fore]` `:lbe` `:lbefore`
Same as `":cbefore"`, except the location list for the current window is used instead of the quickfix list.
- `: [count] caf[ter]` `:caf` `:cafter`
Go to the [count] error after the current cursor position in the current buffer. If [count] is omitted, then 1 is used. If there are no errors, then an error message is displayed. Assumes that the entries in a quickfix list are sorted by their buffer, line and column numbers. If [count] exceeds the number of entries after the current position, then the last error in the file is selected.
- `: [count] laf[ter]` `:laf` `:lafter`
Same as `":cafter"`, except the location list for the current window is used instead of the quickfix list.
- `: [count] cnf[ile][!]` `:cnf` `:cnfile`
Display the first error in the [count] next file in the list that includes a file name. If there are no file names at all or if there is no next file, go to the [count] next error. See `:cc` for [!] and `'switchbuf'`.
- `: [count] lnf[ile][!]` `:lnf` `:lnfile`
Same as `":cnfile"`, except the location list for the current window is used instead of the quickfix list.
- `: [count] cNf[ile][!]` `:cpf` `:cpfile` `:cNf` `:cNfile`
`: [count] cpf[ile][!]`
Display the last error in the [count] previous file in the list that includes a file name. If there are no file names at all or if there is no next file, go to the [count] previous error. See `:cc` for [!] and `'switchbuf'`.
- `: [count] lNf[ile][!]` `:lpf` `:lpfile` `:lNf` `:lNfile`
`: [count] lpf[ile][!]`
Same as `":cNfile"` and `":cpfile"`, except the location list for the current window is used instead of the quickfix list.
- `: cr[ewind][!] [nr]` `:crewind` `:cr`
Display error [nr]. If [nr] is omitted, the FIRST error is displayed. See `:cc`.
- `: lr[ewind][!] [nr]` `:lrewind` `:lr`
Same as `":crewind"`, except the location list for the current window is used instead of the quickfix list.

| | | |
|---|---|--|
| <code>:cfir[st][!] [nr]</code> | Same as <code>":crewind"</code> . | <code>:cfirst</code> <code>:cfir</code> |
| <code>:lfir[st][!] [nr]</code> | Same as <code>":lrewind"</code> . | <code>:lfirst</code> <code>:lfir</code> |
| <code>:cla[st][!] [nr]</code> | Display error <code>[nr]</code> . If <code>[nr]</code> is omitted, the LAST error is displayed. See <code>:cc</code> . | <code>:clast</code> <code>:cla</code> |
| <code>:lla[st][!] [nr]</code> | Same as <code>":clast"</code> , except the location list for the current window is used instead of the quickfix list. | <code>:llast</code> <code>:lla</code> |
| <code>:cq[uit][!]</code>
<code>:{N}cq[uit][!]</code>
<code>:cq[uit][!] {N}</code> | Quit Vim with error code <code>{N}</code> . <code>{N}</code> defaults to one. Useful when Vim is called from another program: e.g., a compiler will not compile the same file again, <code>`git commit`</code> will abort the committing process, <code>`fc`</code> (built-in for shells like bash and zsh) will not execute the command, etc. <code>{N}</code> can also be zero, in which case Vim exits normally.
WARNING: All changes in files are lost! Also when the <code>[!]</code> is not used. It works like <code>":qall!"</code> <code>:qall</code> , except that Vim returns a non-zero exit code. | <code>:cq</code> <code>:cquit</code> |
| <code>:cf[ile][!] [errorfile]</code> | Read the error file and jump to the first error. This is done automatically when Vim is started with the <code>-q</code> option. You can use this command when you keep Vim running while compiling. If you give the name of the errorfile, the <code>'errorfile'</code> option will be set to <code>[errorfile]</code> . See <code>:cc</code> for <code>[!]</code> . If the encoding of the error file differs from the <code>'encoding'</code> option, you can use the <code>'makeencoding'</code> option to specify the encoding. | <code>:cf</code> <code>:cfi</code> <code>:cfile</code> |
| <code>:lf[ile][!] [errorfile]</code> | Same as <code>":cfile"</code> , except the location list for the current window is used instead of the quickfix list. You can not use the <code>-q</code> command-line option to set the location list. | <code>:lf</code> <code>:lfi</code> <code>:lfile</code> |
| <code>:cg[etfile] [errorfile]</code> | Read the error file. Just like <code>":cfile"</code> but don't jump to the first error. If the encoding of the error file differs from the <code>'encoding'</code> option, you can use the <code>'makeencoding'</code> option to specify the encoding. | <code>:cg</code> <code>:cgetfile</code> |

`:lg[etfile] [errorfile]` :lg :lge :lgetfile
 Same as `":cgetfile"`, except the location list for the current window is used instead of the quickfix list.

`:caddf[ile] [errorfile]` :caddf :caddfile
 Read the error file and add the errors from the errorfile to the current quickfix list. If a quickfix list is not present, then a new list is created. If the encoding of the error file differs from the **'encoding'** option, you can use the **'makeencoding'** option to specify the encoding.

`:laddf[ile] [errorfile]` :laddf :laddfile
 Same as `":caddfile"`, except the location list for the current window is used instead of the quickfix list.

`:cb[uffer][!] [bufnr]` :cb :cbuffer E681
 Read the error list from the current buffer. When `[bufnr]` is given it must be the number of a loaded buffer. That buffer will then be used instead of the current buffer. A range can be specified for the lines to be used. Otherwise all lines in the buffer are used. See `:cc` for `[!]`.

`:lb[uffer][!] [bufnr]` :lb :lbuffer
 Same as `":cbuffer"`, except the location list for the current window is used instead of the quickfix list.

`:cgetb[uffer] [bufnr]` :cgetb :cgetbuffer
 Read the error list from the current buffer. Just like `":cbuffer"` but don't jump to the first error.

`:lgetb[uffer] [bufnr]` :lgetb :lgetbuffer
 Same as `":cgetbuffer"`, except the location list for the current window is used instead of the quickfix list.

`:cad[dbuffer] [bufnr]` :cad :cadd :caddbuffer
 Read the error list from the current buffer and add the errors to the current quickfix list. If a quickfix list is not present, then a new list is created. Otherwise, same as `":cbuffer"`.

`:laddb[uffer] [bufnr]` :laddb :laddbuffer
 Same as `":caddbuffer"`, except the location list for the current window is used instead of the quickfix list.

`:cex[pr][!] {expr}` :cex :cexpr E777
 Create a quickfix list using the result of `{expr}` and jump to the first error.

If `{expr}` is a String, then each newline terminated line in the String is processed using the global value of `'errorformat'` and the result is added to the quickfix list.

If `{expr}` is a List, then each String item in the list is processed and added to the quickfix list. Non String items in the List are ignored.

See `:cc` for `[!]`.

Examples:

```
:cexpr system('grep -n xyz *')
:cexpr getline(1, '$')
```

```
:lex[pr][!] {expr}           :lex    :lexpr
Same as :cexpr, except the location list for the
current window is used instead of the quickfix list.
```

```
:cgete[xpr] {expr}          :cgete  :cgetexpr
Create a quickfix list using the result of {expr}.
Just like :cexpr, but don't jump to the first error.
```

```
:lgete[xpr] {expr}          :lgete  :lgetexpr
Same as :cgetexpr, except the location list for the
current window is used instead of the quickfix list.
```

```
:cadde[xpr] {expr}          :cadde  :caddexpr
Evaluate {expr} and add the resulting lines to the
current quickfix list. If a quickfix list is not
present, then a new list is created. The current
cursor position will not be changed. See :cexpr for
more information.
```

Example:

```
:g/mypattern/caddexpr expand("%") .. ":" .. line(".") .. ":" .. getline(".")
```

```
:lad[dexpr] {expr}          :lad     :add     :laddexpr
Same as ":caddexpr", except the location list for the
current window is used instead of the quickfix list.
```

```
:cl[ist] [from] [, [to]]    :cl      :clist
List all errors that are valid quickfix-valid.
If numbers [from] and/or [to] are given, the respective
range of errors is listed. A negative number counts
from the last error backwards, -1 being the last error.
The :filter command can be used to display only the
quickfix entries matching a supplied pattern. The
pattern is matched against the filename, module name,
pattern and text of the entry.
```

```
:cl[ist] +{count}           List the current and next {count} valid errors. This
is similar to ":clist from from+count", where "from"
is the current error position.
```

```
:cl[ist]! [from] [, [to]]  List all errors.
```

`:cl[ist]! +{count}` List the current and next `{count}` error lines. This is useful to see unrecognized lines after the current one. For example, if `":clist"` shows:

```
8384 testje.java:252: error: cannot find symbol
```

Then using `":cl! +3"` shows the reason:

```
8384 testje.java:252: error: cannot find symbol
```

```
8385:   ZexitCode = Fmainx();
```

```
8386:           ^
```

```
8387:   symbol:   method Fmainx()
```

`:lli[st] [from] [, [to]]` `:lli` `:lolist`
Same as `":clist"`, except the location list for the current window is used instead of the quickfix list.

`:lli[st]! [from] [, [to]]`
List all the entries in the location list for the current window.

If you insert or delete lines, mostly the correct error location is still found because hidden marks are used. Sometimes, when the mark has been deleted for some reason, the message "line changed" is shown to warn you that the error location may not be correct. If you quit Vim and start again the marks are lost and the error locations may not be correct anymore.

Two autocommands are available for running commands before and after a quickfix command (`':make'`, `':grep'` and so on) is executed. See [QuickFixCmdPre](#) and [QuickFixCmdPost](#) for details.

QuickFixCmdPost-example

When `'encoding'` differs from the locale, the error messages may have a different encoding from what Vim is using. To convert the messages you can use this code:

```
function QfMakeConv()  
  let qflist = getqflist()  
  for i in qflist  
    let i.text = iconv(i.text, "cp936", "utf-8")  
  endfor  
  call setqflist(qflist)  
endfunction
```

```
au QuickfixCmdPost make call QfMakeConv()
```

Another option is using `'makeencoding'`.

quickfix-title

Every quickfix and location list has a title. By default the title is set to the command that created the list. The `getqflist()` and `getloclist()` functions can be used to get the title of a quickfix and a location list respectively. The `setqflist()` and `setloclist()` functions can be used to modify the title of a quickfix and location list respectively. Examples:

```
call setqflist([], 'a', {'title' : 'Cmd output'})  
echo getqflist({'title' : 1})  
call setloclist(3, [], 'a', {'title' : 'Cmd output'})  
echo getloclist(3, {'title' : 1})
```


quickfix-index

When you jump to a quickfix/location list entry using any of the quickfix commands (e.g. `:cc`, `:cnext`, `:cprev`, etc.), that entry becomes the currently selected entry. The index of the currently selected entry in a quickfix/location list can be obtained using the `getqflist()/getloclist()` functions. Examples:

```
echo getqflist({'idx' : 0}).idx
echo getqflist({'id' : qfid, 'idx' : 0}).idx
echo getloclist(2, {'idx' : 0}).idx
```

For a new quickfix list, the first entry is selected and the index is 1. Any entry in any quickfix/location list can be set as the currently selected entry using the `setqflist()` function. Examples:

```
call setqflist([], 'a', {'idx' : 12})
call setqflist([], 'a', {'id' : qfid, 'idx' : 7})
call setloclist(1, [], 'a', {'idx' : 7})
```

quickfix-size

You can get the number of entries (size) in a quickfix and a location list using the `getqflist()` and `getloclist()` functions respectively. Examples:

```
echo getqflist({'size' : 1})
echo getloclist(5, {'size' : 1})
```

quickfix-context

Any Vim type can be associated as a context with a quickfix or location list. The `setqflist()` and the `setloclist()` functions can be used to associate a context with a quickfix and a location list respectively. The `getqflist()` and the `getloclist()` functions can be used to retrieve the context of a quickfix and a location list respectively. This is useful for a Vim plugin dealing with multiple quickfix/location lists.

Examples:

```
let somectx = {'name' : 'Vim', 'type' : 'Editor'}
call setqflist([], 'a', {'context' : somectx})
echo getqflist({'context' : 1})

let newctx = ['red', 'green', 'blue']
call setloclist(2, [], 'a', {'id' : qfid, 'context' : newctx})
echo getloclist(2, {'id' : qfid, 'context' : 1})
```

quickfix-parse

You can parse a list of lines using `'errorformat'` without creating or modifying a quickfix list using the `getqflist()` function. Examples:

```
echo getqflist({'lines' : ["F1:10:Line10", "F2:20:Line20"]})
echo getqflist({'lines' : systemlist('grep -Hn quickfix *')})
```

This returns a dictionary where the "items" key contains the list of quickfix entries parsed from lines. The following shows how to use a custom

`'errorformat'` to parse the lines without modifying the `'errorformat'` option:

```
echo getqflist({'efm' : '%f#%l#%m', 'lines' : ['F1#10#Line']})
```

EXECUTE A COMMAND IN ALL THE BUFFERS IN QUICKFIX OR LOCATION LIST:

```
:cdo
```

`:cdo[!] {cmd}` Execute `{cmd}` in each valid entry in the quickfix list. It works like doing this:

```
:cfirst
:{cmd}
:cnext
:{cmd}
etc.
```

When the current file can't be `abandon` ed and the `[!]` is not present, the command fails.

When going to the next entry fails execution stops.

The last buffer (or where an error occurred) becomes the current buffer.

`{cmd}` can contain `|` to concatenate several commands.

Only valid entries in the quickfix list are used.

A range can be used to select entries, e.g.:

```
:10,$cdo cmd
```

To skip entries 1 to 9.

Note: While this command is executing, the Syntax autocommand event is disabled by adding it to `'eventignore'`. This considerably speeds up editing each buffer.

Also see `:bufdo` , `:tabdo` , `:argdo` , `:windo` , `:ldo` , `:cfdo` and `:lfdo` .

`:cfdo`

`:cfdo[!] {cmd}` Execute `{cmd}` in each file in the quickfix list. It works like doing this:

```
:cfirst
:{cmd}
:cnfile
:{cmd}
etc.
```

Otherwise it works the same as ``:cdo``.

`:ldo`

`:ld[o][!] {cmd}` Execute `{cmd}` in each valid entry in the location list for the current window.

It works like doing this:

```
:lfirst
:{cmd}
:lnext
:{cmd}
etc.
```

Only valid entries in the location list are used.

Otherwise it works the same as ``:cdo``.

`:lfdo`

`:lfdo[!] {cmd}` Execute `{cmd}` in each file in the location list for the current window.

It works like doing this:

```
:lfirst
:{cmd}
```

```
:lnfile
:{cmd}
etc.
```

Otherwise it works the same as `:ldo`.

FILTERING A QUICKFIX OR LOCATION LIST:

`cfilter-plugin` `:Cfilter` `:Lfilter`

If you have too many entries in a quickfix list, you can use the `cfilter` plugin to reduce the number of entries. Load the plugin with:

```
packadd cfilter
```

Then you can use the following commands to filter a quickfix/location list:

```
:Cfilter[!] /{pat}/
:Lfilter[!] /{pat}/
```

The `:Cfilter` command creates a new quickfix list from the entries matching `{pat}` in the current quickfix list. `{pat}` is a Vim `regular-expression` pattern. Both the file name and the text of the entries are matched against `{pat}`. If the optional `!` is supplied, then the entries not matching `{pat}` are used. The pattern can be optionally enclosed using one of the following characters: `'`, `"`, `/`. If the pattern is empty, then the last used search pattern is used.

The `:Lfilter` command does the same as `:Cfilter` but operates on the current location list.

The current quickfix/location list is not modified by these commands, so you can go back to the unfiltered list using the `:colder|/|:lolder` command.

=====

2. The error window

`quickfix-window`

```
:cope[n] [height] :cope :copen w:quickfix_title
Open a window to show the current list of errors.
```

When `[height]` is given, the window becomes that high (if there is room). When `[height]` is omitted the window is made ten lines high.

If there already is a quickfix window, it will be made the current window. It is not possible to open a second quickfix window. If `[height]` is given the existing window will be resized to it.

`quickfix-buffer`

The window will contain a special buffer, with `'buftype'` equal to "quickfix". Don't change this! The window will have the `w:quickfix_title` variable set which will indicate the command that produced the quickfix list. This can be used to compose a custom status line if the value of `'statusline'` is adjusted properly. Whenever this buffer is modified by a

quickfix command or function, the `b:changedtick` variable is incremented. You can get the number of this buffer using the `getqflist()` and `getloclist()` functions by passing the "qfbufnr" item. For a location list, this buffer is wiped out when the location list is removed.

| | |
|----------------------------------|---|
| <code>:lop[en] [height]</code> | <code>:lop</code> <code>:lopen</code>
Open a window to show the location list for the current window. Works only when the location list for the current window is present. You can have more than one location window opened at a time. Otherwise, it acts the same as <code>:copen</code> . |
| <code>:ccl[ose]</code> | <code>:ccl</code> <code>:cclose</code>
Close the quickfix window. |
| <code>:lcl[ose]</code> | <code>:lcl</code> <code>:lclose</code>
Close the window showing the location list for the current window. |
| <code>:cw[indow] [height]</code> | <code>:cw</code> <code>:cwindow</code>
Open the quickfix window when there are recognized errors. If the window is already open and there are no recognized errors, close the window. |
| <code>:lw[indow] [height]</code> | <code>:lw</code> <code>:lwindow</code>
Same as <code>:cwindow</code> , except use the window showing the location list for the current window. |
| <code>:cbo[ttom]</code> | <code>:cbo</code> <code>:cbottom</code>
Put the cursor in the last line of the quickfix window and scroll to make it visible. This is useful for when errors are added by an asynchronous callback. Only call it once in a while if there are many updates to avoid a lot of redrawing. |
| <code>:lbo[ttom]</code> | <code>:lbo</code> <code>:lbottom</code>
Same as <code>:cbottom</code> , except use the window showing the location list for the current window. |

Normally the quickfix window is at the bottom of the screen. If there are vertical splits, it's at the bottom of the rightmost column of windows. To make it always occupy the full width:

```
:botright cwindow
```

You can move the window around with `window-moving` commands.

For example, to move it to the top: **CTRL-W K**

The `'winfixheight'` option will be set, which means that the window will mostly keep its height, ignoring `'winheight'` and `'equalalways'`. You can change the height manually (e.g., by dragging the status line above it with the mouse).

In the quickfix window, each line is one error. The line number is equal to the error number. The current entry is highlighted with the `QuickFixLine` highlighting. You can change it to your liking, e.g.:

```
:hi QuickFixLine ctermbg=Yellow guibg=Yellow
```

You can use `:.cc` to jump to the error under the cursor. Hitting the `<Enter>` key or double-clicking the mouse on a line has the same effect. The file containing the error is opened in the window above the quickfix window. If there already is a window for that file, it is used instead. If the buffer in the used window has changed, and the error is in another file, jumping to the error will fail. You will first have to make sure the window contains a buffer which can be abandoned.

When you select a file from the quickfix window, the following steps are used to find a window to edit the file:

1. If a window displaying the selected file is present in the current tabpage (starting with the window before the quickfix window), then that window is used.
2. If the above step fails and if `'switchbuf'` contains `"usetab"` and a window displaying the selected file is present in any one of the tabpages (starting with the first tabpage) then that window is used.
3. If the above step fails then a window in the current tabpage displaying a buffer with `'buftype'` not set (starting with the window before the quickfix window) is used.
4. If the above step fails and if `'switchbuf'` contains `"uselast"`, then the previously accessed window is used.
5. If the above step fails then the window before the quickfix window is used. If there is no previous window, then the window after the quickfix window is used.
6. If the above step fails, then a new horizontally split window above the quickfix window is used.

```
CTRL-W_<Enter> CTRL-W_<CR>
```

You can use `CTRL-W <Enter>` to open a new window and jump to the error there.

When the quickfix window has been filled, two autocommand events are triggered. First the `'filetype'` option is set to `"qf"`, which triggers the FileType event (also see `qf.vim`). Then the BufReadPost event is triggered, using `"quickfix"` for the buffer name. This can be used to perform some action on the listed errors. Example:

```
au BufReadPost quickfix setlocal modifiable
    \ | silent exe 'g/^s//\=line(".") .. " "/'
    \ | setlocal nomodifiable
```

This prepends the line number to each line. **Note** the use of `"\="` in the substitute string of the `:"s"` command, which is used to evaluate an expression.

The BufWinEnter event is also triggered, again using `"quickfix"` for the buffer name.

Note: When adding to an existing quickfix list the autocommand are not triggered.

Note: Making changes in the quickfix window has no effect on the list of errors. `'modifiable'` is off to avoid making changes. If you delete or insert lines anyway, the relation between the text and the error number is messed up. If you really want to do this, you could write the contents of the quickfix

window to a file and use ":cfile" to have it parsed and used as the new error list.

location-list-window

The location list window displays the entries in a location list. When you open a location list window, it is created below the current window and displays the location list for the current window. The location list window is similar to the quickfix window, except that you can have more than one location list window open at a time. When you use a location list command in this window, the displayed location list is used.

When you select a file from the location list window, the following steps are used to find a window to edit the file:

1. If a non-quickfix window associated with the location list is present in the current tabpage, then that window is used.
2. If the above step fails and if the file is already opened in another window in the current tabpage, then that window is used.
3. If the above step fails and 'switchbuf' contains "usetab" and if the file is opened in a window in any one of the tabpages, then that window is used.
4. If the above step fails then a window in the current tabpage showing a buffer with 'buftype' not set is used.
5. If the above step fails, then the file is edited in a new window.

In all of the above cases, if the location list for the selected window is not yet set, then it is set to the location list displayed in the location list window.

quickfix-window-ID

You can use the `getqflist()` and `getloclist()` functions to obtain the window ID of the quickfix window and location list window respectively (if present). Examples:

```
echo getqflist({'winid' : 1}).winid
echo getloclist(2, {'winid' : 1}).winid
```

getqflist-examples

The `getqflist()` and `getloclist()` functions can be used to get the various attributes of a quickfix and location list respectively. Some examples for using these functions are below:

```
" get the title of the current quickfix list
:echo getqflist({'title' : 0}).title

" get the identifier of the current quickfix list
:let qfid = getqflist({'id' : 0}).id

" get the identifier of the fourth quickfix list in the stack
:let qfid = getqflist({'nr' : 4, 'id' : 0}).id

" check whether a quickfix list with a specific identifier exists
:if getqflist({'id' : qfid}).id == qfid

" get the index of the current quickfix list in the stack
:let qfnum = getqflist({'nr' : 0}).nr
```

```

" get the items of a quickfix list specified by an identifier
:echo getqflist({'id' : qfid, 'items' : 0}).items

" get the number of entries in a quickfix list specified by an id
:echo getqflist({'id' : qfid, 'size' : 0}).size

" get the context of the third quickfix list in the stack
:echo getqflist({'nr' : 3, 'context' : 0}).context

" get the number of quickfix lists in the stack
:echo getqflist({'nr' : '$'}).nr

" get the number of times the current quickfix list is changed
:echo getqflist({'changedtick' : 0}).changedtick

" get the current entry in a quickfix list specified by an identifier
:echo getqflist({'id' : qfid, 'idx' : 0}).idx

" get all the quickfix list attributes using an identifier
:echo getqflist({'id' : qfid, 'all' : 0})

" parse text from a List of lines and return a quickfix list
:let myList = ["a.java:10:L10", "b.java:20:L20"]
:echo getqflist({'lines' : myList}).items

" parse text using a custom 'efm' and return a quickfix list
:echo getqflist({'lines' : ['a.c#10#Line 10'], 'efm': '%f#%l#%m'}).items

" get the quickfix list window id
:echo getqflist({'winid' : 0}).winid

" get the quickfix list window buffer number
:echo getqflist({'qfbufnr' : 0}).qfbufnr

" get the context of the current location list
:echo getloclist(0, {'context' : 0}).context

" get the location list window id of the third window
:echo getloclist(3, {'winid' : 0}).winid

" get the location list window buffer number of the third window
:echo getloclist(3, {'qfbufnr' : 0}).qfbufnr

" get the file window id of a location list window (winnr: 4)
:echo getloclist(4, {'filewinid' : 0}).filewinid

```

setqflist-examples

The `setqflist()` and `setloclist()` functions can be used to set the various attributes of a quickfix and location list respectively. Some examples for using these functions are below:

```

" create an empty quickfix list with a title and a context
:let t = 'Search results'

```

```

:let c = {'cmd' : 'grep'}
:callsetqfllist([], ' ', {'title' : t, 'context' : c})

" set the title of the current quickfix list
:callsetqfllist([], 'a', {'title' : 'Mytitle'})

" change the current entry in the list specified by an identifier
:callsetqfllist([], 'a', {'id' : qfid, 'idx' : 10})

" set the context of a quickfix list specified by an identifier
:callsetqfllist([], 'a', {'id' : qfid, 'context' : {'val' : 100}})

" create a new quickfix list from a command output
:callsetqfllist([], ' ', {'lines' : systemlist('grep -Hn main *.c')})

" parse text using a custom efm and add to a particular quickfix list
:callsetqfllist([], 'a', {'id' : qfid,
    \ 'lines' : ["a.c#10#L10", "b.c#20#L20"], 'efm': '%f#%l#%m'})

" add items to the quickfix list specified by an identifier
:let newItems = [{'filename' : 'a.txt', 'lnum' : 10, 'text' : "Apple"},
    \ {'filename' : 'b.txt', 'lnum' : 20, 'text' : "Orange"}]
:callsetqfllist([], 'a', {'id' : qfid, 'items' : newItems})

" empty a quickfix list specified by an identifier
:callsetqfllist([], 'r', {'id' : qfid, 'items' : []})

" free all the quickfix lists in the stack
:callsetqfllist([], 'f')

" set the title of the fourth quickfix list
:callsetqfllist([], 'a', {'nr' : 4, 'title' : 'SomeTitle'})

" create a new quickfix list at the end of the stack
:callsetqfllist([], ' ', {'nr' : '$',
    \ 'lines' : systemlist('grep -Hn class *.java')})

" create a new location list from a command output
:callsetloclist(0, [], ' ', {'lines' : systemlist('grep -Hn main *.c')})

" replace the location list entries for the third window
:callsetloclist(3, [], 'r', {'items' : newItems})

```

3. Using more than one list of errors

[quickfix-error-lists](#)

So far has been assumed that there is only one list of errors. Actually the ten last used lists are remembered. When starting a new list, the previous ones are automatically kept. Two commands can be used to access older error lists. They set one of the existing error lists as the current one.

```

:col[der] [count]          :colder :col E380
                          Go to older error list. When [count] is given, do
                          this [count] times. When already at the oldest error

```


list, an error message is given.

`:lol[der] [count]` `:lolder` `:lol`
Same as `:colder`, except use the location list for the current window instead of the quickfix list.

`:cnew[er] [count]` `:cnewer` `:cnew` `E381`
Go to newer error list. When `[count]` is given, do this `[count]` times. When already at the newest error list, an error message is given.

`:lnew[er] [count]` `:lnewer` `:lnew`
Same as `:cnewer`, except use the location list for the current window instead of the quickfix list.

`:[count]chi[story]` `:chistory` `:chi`
Show the list of error lists. The current list is marked with `>`. The output looks like:
`error list 1 of 3; 43 errors :make`
`> error list 2 of 3; 0 errors :helpgrep tag`
`error list 3 of 3; 15 errors :grep ex_help *.c`

When `[count]` is given, then the count'th quickfix list is made the current list. Example:
`" Make the 4th quickfix list current`
`:4chistory`

`:[count]lhi[story]` `:lhistory` `:lhi`
Show the list of location lists, otherwise like `:chistory`.

When adding a new error list, it becomes the current list.

When `:"colder"` has been used and `:"make"` or `:"grep"` is used to add a new error list, one newer list is overwritten. This is especially useful if you are browsing with `:"grep"` `grep`. If you want to keep the more recent error lists, use `:"cnewer 99"` first.

To get the number of lists in the quickfix and location list stack, you can use the `getqflist()` and `getloclist()` functions respectively with the list number set to the special value `'$'`. Examples:

```
echo getqflist({'nr' : '$'}).nr
echo getloclist(3, {'nr' : '$'}).nr
```

To get the number of the current list in the stack:

```
echo getqflist({'nr' : 0}).nr
```

=====
4. Using `:make` `:make_makeprg`

`:mak[e][!] [arguments]` `:mak` `:make`
1. All relevant `QuickFixCmdPre` autocommands are executed.
2. If the `'autowrite'` option is on, write any changed buffers

3. An errorfile name is made from **'makeef'**. If **'makeef'** doesn't contain "##", and a file with this name already exists, it is deleted.
 4. The program given with the **'makeprg'** option is started (default "make") with the optional **[arguments]** and the output is saved in the errorfile (for Unix it is also echoed on the screen).
 5. The errorfile is read using **'errorformat'**.
 6. All relevant **QuickFixCmdPost** autocommands are executed. See example below.
 7. If **[!]** is not given the first error is jumped to.
 8. The errorfile is deleted.
 9. You can now move through the errors with commands like **:cnext** and **:cprevious**, see above.
- This command does not accept a comment, any " characters are considered part of the arguments. If the encoding of the program output differs from the **'encoding'** option, you can use the **'makeencoding'** option to specify the encoding.

:lmak **:lmake**

:lmak[e][!] **[arguments]**

Same as **":make"**, except the location list for the current window is used instead of the quickfix list.

The **":make"** command executes the command given with the **'makeprg'** option. This is done by passing the command to the shell given with the **'shell'** option. This works almost like typing

":!{makeprg} [arguments] {shellpipe} {errorfile}".

{makeprg} is the string given with the **'makeprg'** option. Any command can be used, not just "make". Characters '%' and '#' are expanded as usual on a command-line. You can use "%<" to insert the current file name without extension, or "#<" to insert the alternate file name without extension, for example:

```
:set makeprg=make\ #<.o
```

[arguments] is anything that is typed after **":make"**.

{shellpipe} is the **'shellpipe'** option.

{errorfile} is the **'makeef'** option, with ## replaced to make it unique.

The placeholder "\$*" can be used for the argument list in **{makeprg}** if the command needs some additional characters after its arguments. The \$* is replaced then by all arguments. Example:

```
:set makeprg=latex\ \\\nonstopmode\ \\\input\{\$*}
```

or simpler

```
:let &mp = 'latex \\\nonstopmode \\\input\{\$*}'
```

"\$*" can be given multiple times, for example:

```
:set makeprg=gcc\ -o\ \$*\ $*
```

The **'shellpipe'** option defaults to ">" for the Amiga and ">%s 2>&1" for Win32. This means that the output of the compiler is saved in a file and not shown on

the screen directly. For Unix "| tee" is used. The compiler output is shown on the screen and saved in a file the same time. Depending on the shell used "|& tee" or "2>&1| tee" is the default, so stderr output will be included.

If **'shellpipe'** is empty, the `{errorfile}` part will be omitted. This is useful for compilers that write to an errorfile themselves (e.g., Manx's Amiga C).

Using QuickFixCmdPost to fix the encoding

It may be that **'encoding'** is set to an encoding that differs from the messages your build program produces. This example shows how to fix this after Vim has read the error messages:

```
function QfMakeConv()
  let qflist = getqflist()
  for i in qflist
    let i.text = iconv(i.text, "cp936", "utf-8")
  endfor
  call setqflist(qflist)
endfunction

au QuickfixCmdPost make call QfMakeConv()
```

(Example by Faque Cheng)

Another option is using **'makeencoding'**.

5. Using `:vimgrep` and `:grep`

`grep` `lid`

Vim has two ways to find matches for a pattern: Internal and external. The advantage of the internal `grep` is that it works on all systems and uses the powerful Vim search patterns. An external `grep` program can be used when the Vim `grep` does not do what you want.

The internal method will be slower, because files are read into memory. The advantages are:

- Line separators and encoding are automatically recognized, as if a file is being edited.
- Uses Vim search patterns. Multi-line patterns can be used.
- When plugins are enabled: compressed and remote files can be searched.

`gzip` `netrw`

To be able to do this Vim loads each file as if it is being edited. When there is no match in the file the associated buffer is wiped out again. The **'hidden'** option is ignored here to avoid running out of memory or file descriptors when searching many files. However, when the `:hide` command modifier is used the buffers are kept loaded. This makes following searches in the same files a lot faster.

Note that `:copen` (or `:lopen` for `:lgrep`) may be used to open a buffer containing the search results in linked form. The `:silent` command may be used to suppress the default full screen `grep` output. The `":grep!"` form of the `:grep` command doesn't jump to the first match automatically. These

commands can be combined to create a NewGrep command:

```
command! -nargs=+ NewGrep execute 'silent grep! <args>' | copen 42
```

5.1 using Vim's internal grep

```
                                :vim   :vimgrep  E682  E683
:vim[grep][!] /{pattern}/{g}[j][f] {file} ...
    Search for {pattern} in the files {file} ... and set
    the error list to the matches. Files matching
    'wildignore' are ignored; files in 'suffixes' are
    searched last.

    {pattern} is a Vim search pattern. Instead of
    enclosing it in / any non-ID character (see
    'isident' ) can be used, so long as it does not
    appear in {pattern}.
    'ignorecase' applies. To overrule it put /\c in the
    pattern to ignore case or /\C to match case.
    'smartcase' is not used.
    If {pattern} is empty (e.g. // is specified), the last
    used search pattern is used. last-pattern

    Flags:
    'g' Without the 'g' flag each line is added only
        once. With 'g' every match is added.

    'j' Without the 'j' flag Vim jumps to the first
        match. With 'j' only the quickfix list is
        updated. With the [!] any changes in the current
        buffer are abandoned.

    'f' When the 'f' flag is specified, fuzzy string
        matching is used to find matching lines. In this
        case, {pattern} is treated as a literal string
        instead of a regular expression. See
        fuzzy-matching for more information about fuzzy
        matching strings.

    QuickFixCmdPre and QuickFixCmdPost are triggered.
    A file that is opened for matching may use a buffer
    number, but it is reused if possible to avoid
    consuming buffer numbers.
```

```
:{count}vim[grep] ...
```

When a number is put before the command this is used as the maximum number of matches to find. Use `":1vimgrep pattern file"` to find only the first. Useful if you only want to check if there is a match and quit quickly when it's found.

Every second or so the searched file name is displayed to give you an idea of the progress made.

Examples:

```
:vimgrep /an error/ *.c
:vimgrep /\<FileName\>/ *.h include/*
:vimgrep /myfunc/ **/*.c
```

For the use of "*" see [starstar-wildcard](#) .

```
:vim[grep][!] {pattern} {file} ...
```

Like above, but instead of enclosing the pattern in a non-ID character use a white-separated pattern. The pattern must start with an ID character.

Example:

```
:vimgrep Error *.c
```

[:lv](#) [:lvimgrep](#)

```
:lv[imgrep][!] /{pattern}/[g][j][f] {file} ...
```

```
:lv[imgrep][!] {pattern} {file} ...
```

Same as ":vimgrep", except the location list for the current window is used instead of the quickfix list.

[:vimgrepa](#) [:vimgrepadd](#)

```
:vimgrepa[dd][!] /{pattern}/[g][j][f] {file} ...
```

```
:vimgrepa[dd][!] {pattern} {file} ...
```

Just like ":vimgrep", but instead of making a new list of errors the matches are appended to the current list.

[:lvimgrepa](#) [:lvimgrepadd](#)

```
:lvimgrepa[dd][!] /{pattern}/[g][j][f] {file} ...
```

```
:lvimgrepa[dd][!] {pattern} {file} ...
```

Same as ":vimgrepadd", except the location list for the current window is used instead of the quickfix list.

5.2 External grep

Vim can interface with "grep" and grep-like programs (such as the GNU id-utils) in a similar way to its compiler integration (see [:make](#) above).

[Unix trivia: The name for the Unix "grep" command comes from ":g/re/p", where "re" stands for Regular Expression.]

[:gr](#) [:grep](#)

```
:gr[ep][!] [arguments] Just like ":make", but use 'grepgrg' instead of 'makeprg' and 'grepformat' instead of 'errorformat'.
```

When **'grepgrg'** is "internal" this works like

```
:vimgrep . Note that the pattern needs to be
```

enclosed in separator characters then.

If the encoding of the program output differs from the **'encoding'** option, you can use the **'makeencoding'** option to specify the encoding.

[:lgr](#) [:lgrep](#)

```
:lgr[ep][!] [arguments] Same as ":grep", except the location list for the current window is used instead of the quickfix list.
```

`:grepa` `:grepadd`

`:grepa[dd][!]` `[arguments]`

Just like `":grep"`, but instead of making a new list of errors the matches are appended to the current list.
Example:

```
:call setqflist([])
:bufdo grepadd! something %
```

The first command makes a new error list which is empty. The second command executes `"grepadd"` for each listed buffer. **Note** the use of `!` to avoid that `":grepadd"` jumps to the first error, which is not allowed with `:bufdo`.

An example that uses the argument list and avoids errors for files without matches:

```
:silent argdo try
\ | grepadd! something %
\ | catch /E480:/
\ | endtry"
```

If the encoding of the program output differs from the `'encoding'` option, you can use the `'makeencoding'` option to specify the encoding.

`:lgrepa` `:lgrepadd`

`:lgrepa[dd][!]` `[arguments]`

Same as `":grepadd"`, except the location list for the current window is used instead of the quickfix list.

5.3 Setting up external grep

If you have a standard `"grep"` program installed, the `:grep` command may work well with the defaults. The syntax is very similar to the standard command:

```
:grep foo *.c
```

Will search all files with the `.c` extension for the substring `"foo"`. The arguments to `:grep` are passed straight to the `"grep"` program, so you can use whatever options your `"grep"` supports.

By default, `:grep` invokes `grep` with the `-n` option (show file and line numbers). You can change this with the `'grepprg'` option. You will need to set `'grepprg'` if:

- a) You are using a program that isn't called `"grep"`
- b) You have to call `grep` with a full path
- c) You want to pass other options automatically (e.g. case insensitive search.)

Once `"grep"` has executed, Vim parses the results using the `'grepformat'` option. This option works in the same way as the `'errorformat'` option – see that for details. You may need to change `'grepformat'` from the default if your `grep` outputs in a non-standard format, or you are using some other program with a special format.

Once the results are parsed, Vim loads the first file containing a match and jumps to the appropriate line, in the same way that it jumps to a compiler error in `quickfix` mode. You can then use the `:cnext`, `:clist`, etc. commands to see the other matches.

5.4 Using `:grep` with `id-utils`

You can set up `:grep` to work with the GNU `id-utils` like this:

```
:set grepprg=lid\ -Rgrep\ -s
:set grepformat=%f:%l:%m
```

then

```
:grep (regexp)
```

works just as you'd expect.
(provided you remembered to `mkid` first :)

5.5 Browsing source code with `:vimgrep` or `:grep`

Using the stack of error lists that Vim keeps, you can browse your files to look for functions and the functions they call. For example, suppose that you have to add an argument to the `read_file()` function. You enter this command:

```
:vimgrep /\<read_file\>/ *.c
```

You use `":cn"` to go along the list of matches and add the argument. At one place you have to get the new argument from a higher level function `msg()`, and need to change that one too. Thus you use:

```
:vimgrep /\<msg\>/ *.c
```

While changing the `msg()` functions, you find another function that needs to get the argument from a higher level. You can again use `":vimgrep"` to find these functions. Once you are finished with one function, you can use

```
:colder
```

to go back to the previous one.

This works like browsing a tree: `":vimgrep"` goes one level deeper, creating a list of branches. `":colder"` goes back to the previous level. You can mix this use of `":vimgrep"` and `":colder"` to browse all the locations in a tree-like way. If you do this consistently, you will find all locations without the need to write down a "todo" list.

```
=====
6. Selecting a compiler compiler-select
```

```
:comp :compiler E666
:comp[iler][!] {name} Set options to work with compiler {name}.
```

Without the "!" options are set for the current buffer. With "!" global options are set.

If you use ":compiler foo" in "file.foo" and then ":compiler! bar" in another buffer, Vim will keep on using "foo" in "file.foo".

{not available when compiled without the `+eval` feature}

The Vim plugins in the "compiler" directory will set options to use the selected compiler. For `:compiler`` local options are set, for `:compiler!`` global options.

`current_compiler`

To support older Vim versions, the plugins always use "current_compiler" and not "b:current_compiler". What the command actually does is the following:

- Delete the "current_compiler" and "b:current_compiler" variables.
- Define the "CompilerSet" user command. With "!" it does ":set", without "!" it does ":setlocal".
- Execute ":runtime! compiler/{name}.vim". The plugins are expected to set options with "CompilerSet" and set the "current_compiler" variable to the name of the compiler.
- Delete the "CompilerSet" user command.
- Set "b:current_compiler" to the value of "current_compiler".
- Without "!" the old value of "current_compiler" is restored.

For writing a compiler plugin, see [write-compiler-plugin](#) .

DOTNET

`compiler-dotnet`

The .NET CLI compiler outputs both errors and warnings by default. The output may be limited to include only errors, by setting the `g:dotnet_errors_only` variable to `v:true` .

The associated project name is included in each error and warning. To suppress the project name, set the `g:dotnet_show_project_file` variable to `v:false` .

Example: limit output to only display errors, and suppress the project name:

```
let dotnet_errors_only = v:true
let dotnet_show_project_file = v:false
compiler dotnet
```

GCC

`quickfix-gcc`

`compiler-gcc`

There's one variable you can set for the GCC compiler:

`g:compiler_gcc_ignore_unmatched_lines`

Ignore lines that don't match any patterns defined for GCC. Useful if output from commands run from make are generating false

positives.

MANX AZTEC C

[quickfix-manx](#) [compiler-manx](#)

To use Vim with Manx's Aztec C compiler on the Amiga you should do the following:

- Set the CCEDIT environment variable with the command:
`mset "CCEDIT=vim -q"`
- Compile with the `-qf` option. If the compiler finds any errors, Vim is started and the cursor is positioned on the first error. The error message will be displayed on the last line. You can go to other errors with the commands mentioned above. You can fix the errors and write the file(s).
- If you exit Vim normally the compiler will re-compile the same file. If you exit with the `:cq` command, the compiler will terminate. Do this if you cannot fix the error, or if another file needs to be compiled first.

There are some restrictions to the Quickfix mode on the Amiga. The compiler only writes the first 25 errors to the errorfile (Manx's documentation does not say how to get more). If you want to find the others, you will have to fix a few errors and exit the editor. After recompiling, up to 25 remaining errors will be found.

If Vim was started from the compiler, the `:sh` and some `:!` commands will not work, because Vim is then running in the same process as the compiler and stdin (standard input) will not be interactive.

PERL

[quickfix-perl](#) [compiler-perl](#)

The Perl compiler plugin doesn't actually compile, but invokes Perl's internal syntax checking feature and parses the output for possible errors so you can correct them in quick-fix mode.

Warnings are forced regardless of "no warnings" or "\$^W = 0" within the file being checked. To disable this set `g:perl_compiler_force_warnings` to a zero value. For example:

```
let g:perl_compiler_force_warnings = 0
```

PYUNIT COMPILER

[compiler-pyunit](#)

This is not actually a compiler, but a unit testing framework for the Python language. It is included into standard Python distribution starting from version 2.0. For older versions, you can get it from <http://pyunit.sourceforge.net>.

When you run your tests with the help of the framework, possible errors are parsed by Vim and presented for you in quick-fix mode.

Unfortunately, there is no standard way to run the tests. The `alltests.py` script seems to be used quite often, that's all. Useful values for the `'makeprg'` options therefore are:
`setlocal makeprg=./alltests.py " Run a testsuite`

```
setlocal makeprg=python\ %:S " Run a single testcase
```

Also see http://vim.sourceforge.net/tip_view.php?tip_id=280.

TEX COMPILER

compiler-tex

Included in the distribution compiler for TeX (`$VIMRUNTIME/compiler/tex.vim`) uses `make` command if possible. If the compiler finds a file named "Makefile" or "makefile" in the current directory, it supposes that you want to process your *TeX files with `make`, and the makefile does the right work. In this case compiler sets '**errorformat**' for *TeX output and leaves '**makeprg**' untouched. If neither "Makefile" nor "makefile" is found, the compiler will not use `make`. You can force the compiler to ignore makefiles by defining `b:tex_ignore_makefile` or `g:tex_ignore_makefile` variable (they are checked for existence only).

If the compiler chose not to use `make`, it needs to choose a right program for processing your input. If `b:tex_flavor` or `g:tex_flavor` (in this precedence) variable exists, it defines TeX flavor for `:make` (actually, this is the name of executed command), and if both variables do not exist, it defaults to "latex". For example, while editing `chapter2.tex` \input-ed from `mypaper.tex` written in AMS-TeX:

```
:let b:tex_flavor = 'amstex'  
:compiler tex  
[editing...]  
:make mypaper
```

Note that you must specify a name of the file to process as an argument (to process the right file when editing \input-ed or \include-ed file; portable solution for substituting % for no arguments is welcome). This is not in the semantics of `make`, where you specify a target, not source, but you may specify filename without extension ".tex" and mean this as "make filename.dvi or filename.pdf or filename.some_result_extension according to compiler".

Note: `tex` command line syntax is set to usable both for MikTeX (suggestion by Srinath Avadhanula) and teTeX (checked by Artem Chuprina). Suggestion from [errorformat-LaTeX](#) is too complex to keep it working for different shells and OSes and also does not allow to use other available TeX options, if any. If your TeX doesn't support "--interaction=nonstopmode", please report it with different means to express \nonstopmode from the command line.

=====
7. The error format error-file-format

```
errorformat E372 E373 E374  
            E375 E376 E377 E378
```

The '**errorformat**' option specifies a list of formats that are recognized. The first format that matches with an error message is used. You can add several formats for different messages your compiler produces, or even entries for multiple compilers. See [efm-entries](#) .

Each entry in '**errorformat**' is a scanf-like string that describes the format.

First, you need to know how scanf works. Look in the documentation of your C compiler. Below you find the % items that Vim understands. Others are invalid.

Special characters in **'errorformat'** are comma and backslash. See [efm-entries](#) for how to deal with them. **Note** that a literal "%" is matched by "%%", thus it is not escaped with a backslash. Keep in mind that in the `:make` and `:grep` output all NUL characters are replaced with SOH (0x01).

Note: By default the difference between upper and lowercase is ignored. If you want to match case, add "\C" to the pattern `/\C`.

Vim will read lines of any length, but only the first 4095 bytes are used, the rest is ignored. Items can only be 1023 bytes long.

Basic items

| | |
|----------|---|
| %f | file name (finds a string) |
| %b | buffer number (finds a number) |
| %o | module name (finds a string) |
| %l | line number (finds a number) |
| %e | end line number (finds a number) |
| %c | column number (finds a number representing character column of the error, byte index, a <tab> is 1 character column) |
| %v | virtual column number (finds a number representing screen column of the error (1 <tab> == 8 screen columns)) |
| %k | end column number (finds a number representing the character column of the error, byte index, or a number representing screen end column of the error if it's used with %v) |
| %t | error type (finds a single character):
e - error message
w - warning message
i - info message
n - note message |
| %n | error number (finds a number) |
| %m | error message (finds a string) |
| %r | matches the "rest" of a single-line file message %O/P/Q |
| %p | pointer line (finds a sequence of '-', '.', ' ' or tabs and uses the length for the column number) |
| %*{conv} | any scanf non-assignable conversion |
| %% | the single '%' character |
| %s | search text (finds a string) |

The "%f" conversion may depend on the current **'isfname'** setting. "~/" is expanded to the home directory and environment variables are expanded.

The "%f" and "%m" conversions have to detect the end of the string. This normally happens by matching following characters and items. When nothing is following the rest of the line is matched. If "%f" is followed by a '%' or a

backslash, it will look for a sequence of `'isfname'` characters.

On MS-Windows a leading "C:" will be included in "%f", even when using "%f:". This means that a file name which is a single alphabetical letter will not be detected.

The "%b" conversion is used to parse a buffer number. This is useful for referring to lines in a scratch buffer or a buffer with no name. If a buffer with the matching number doesn't exist, then that line is used as a non-error line.

The "%p" conversion is normally followed by a "^". It's used for compilers that output a line like:

```
or
-----^
```

to indicate the column of the error. This is to be used in a multi-line error message. See [errorformat-javac](#) for a useful example.

The "%s" conversion specifies the text to search for, to locate the error line. The text is used as a literal string. The anchors "^" and "\$" are added to the text to locate the error line exactly matching the search text and the text is prefixed with the "\V" atom to make it "very nomagic". The "%s" conversion can be used to locate lines without a line number in the error output. Like the output of the "grep" shell command. When the pattern is present the line number will not be used.

The "%o" conversion specifies the module name in quickfix entry. If present it will be used in quickfix error window instead of the filename. The module name is used only for displaying purposes, the file name is used when jumping to the file.

Changing directory

The following uppercase conversion characters specify the type of special format strings. At most one of them may be given as a prefix at the beginning of a single comma-separated format pattern.

Some compilers produce messages that consist of directory names that have to be prepended to each file name read by %f (example: GNU make). The following codes can be used to scan these directory names; they will be stored in an internal directory stack. E379

| | |
|----|---|
| %D | "enter directory" format string; expects a following %f that finds the directory name |
| %X | "leave directory" format string; expects following %f |

When defining an "enter directory" or "leave directory" format, the "%D" or "%X" has to be given at the start of that substring. Vim tracks the directory changes and prepends the current directory to each erroneous file found with a relative path. See [quickfix-directory-stack](#) for details, tips and limitations.

Multi-line messages

[errorformat-multi-line](#)

It is possible to read the output of programs that produce multi-line messages, i.e. error strings that consume more than one line. Possible prefixes are:

| | |
|----|---|
| %E | start of a multi-line error message |
| %W | start of a multi-line warning message |
| %I | start of a multi-line informational message |
| %N | start of a multi-line note message |
| %A | start of a multi-line message (unspecified type) |
| %> | for next line start with current pattern again <code>efm-%></code> |
| %C | continuation of a multi-line message |
| %Z | end of a multi-line message |

These can be used with '+' and '-', see `efm-ignore` below.

Using "\n" in the pattern won't work to match multi-line messages.

Example: Your compiler happens to write out errors in the following format (leading line numbers not being part of the actual output):

```
1 Error 275
2 line 42
3 column 3
4 ' ' expected after '--'
```

The appropriate error format string has to look like this:

```
:set efm=%EError\ %n,%Cline\ %l,%Ccolumn\ %c,%Z%m
```

And the `:clist` error message generated for this error is:

```
1:42 col 3 error 275: ' ' expected after '--'
```

Another example: Think of a Python interpreter that produces the following error message (line numbers are not part of the actual output):

```
1 =====
2 FAIL: testGetTypeIdCachesResult (dbfacadeTest.DjsDBFacadeTest)
3 -----
4 Traceback (most recent call last):
5   File "unittests/dbfacadeTest.py", line 89, in testFoo
6     self.assertEqual(34, dtid)
7   File "/usr/lib/python2.2/unittest.py", line 286, in
8     failUnlessEqual
9     raise self.failureException, \
10  AssertionError: 34 != 33
11
12 -----
13 Ran 27 tests in 0.063s
```

Say you want `:clist` write the relevant information of this message only, namely:

```
5 unittests/dbfacadeTest.py:89: AssertionError: 34 != 33
```

Then the error format string could be defined as follows:

```
:set efm=%C\ %.%#,%A\ \ File\ \"%f\"\\,\ line\ %l%.%#,%Z[%^\ ]%\\@=%m
```

Note that the %C string is given before the %A here: since the expression '%.##' (which stands for the regular expression '.*') matches every line starting with a space, followed by any characters to the end of the line, it also hides line 7 which would trigger a separate error message otherwise. Error format strings are always parsed pattern by pattern until the first match occurs.

efm-%>

The %> item can be used to avoid trying patterns that appear earlier in 'errorformat'. This is useful for patterns that match just about anything. For example, if the error looks like this:

```
Error in line 123 of foo.c:
unknown variable "i"
```

This can be found with:

```
:set efm=xxx,%E%>Error in line %l of %f:,%Z%m
```

Where "xxx" has a pattern that would also match the second line.

Important: There is no memory of what part of the errorformat matched before; every line in the error file gets a complete new run through the error format lines. For example, if one has:

```
setlocal efm=aa,bb,cc,dd,ee
```

Where aa, bb, etc. are error format strings. Each line of the error file will be matched to the pattern aa, then bb, then cc, etc. Just because cc matched the previous error line does not mean that dd will be tried first on the current line, even if cc and dd are multi-line errorformat strings.

Separate file name

errorformat-separate-filename

These prefixes are useful if the file name is given once and multiple messages follow that refer to this file name.

| | |
|----|--|
| %O | single-line file message: overread the matched part |
| %P | single-line file message: push file %f onto the stack |
| %Q | single-line file message: pop the last file from stack |

Example: Given a compiler that produces the following error logfile (without leading line numbers):

```
1 [a1.tt]
2 (1,17) error: ';' missing
3 (21,2) warning: variable 'z' not defined
4 (67,3) error: end of file found before string ended
5
6 [a2.tt]
7
8 [a3.tt]
9 NEW compiler v1.1
10 (2,2) warning: variable 'x' not defined
11 (67,3) warning: 's' already defined
```

This logfile lists several messages for each file enclosed in [...] which are properly parsed by an error format like this:

```
:set efm=%+P[%f],(%l\\,%c)%*[\ ]%t%*[^:]:\ %m,%-Q
```

A call of `:clist` writes them accordingly with their correct filenames:

```
2 a1.tt:1 col 17 error: ';' missing
3 a1.tt:21 col 2 warning: variable 'z' not defined
4 a1.tt:67 col 3 error: end of file found before string ended
8 a3.tt:2 col 2 warning: variable 'x' not defined
9 a3.tt:67 col 3 warning: 's' already defined
```

Unlike the other prefixes that all match against whole lines, %P, %Q and %O can be used to match several patterns in the same line. Thus it is possible to parse even nested files like in the following line:

```
{"file1" {"file2" error1} error2 {"file3" error3 {"file4" error4 error5}}}
```

The %O then parses over strings that do not contain any push/pop file name information. See [errorformat-LaTeX](#) for an extended example.

Ignoring and using whole messages

`efm-ignore`

The codes '+' or '-' can be combined with the uppercase codes above; in that case they have to precede the letter, e.g. '%+A' or '%-G':

| | |
|----|--|
| %- | do not include the matching multi-line in any output |
| %+ | include the whole matching line in the %m error string |

One prefix is only useful in combination with '+' or '-', namely %G. It parses over lines containing general information like compiler version strings or other headers that can be skipped.

| | |
|-----|---------------------|
| %-G | ignore this message |
| %+G | general message |

Pattern matching

The scanf()-like "%*[]" notation is supported for backward-compatibility with previous versions of Vim. However, it is also possible to specify (nearly) any Vim supported regular expression in format strings.

Since meta characters of the regular expression language can be part of ordinary matching strings or file names (and therefore internally have to be escaped), meta symbols have to be written with leading '%':

| | |
|---|---|
| %\
%.
%#
% [^]
%\$
%[
%~ | The single '\ ' character. Note that this has to be escaped ("%\\") in ":set errorformat=" definitions.
The single '.' character.
The single '*'(!) character.
The single '^' character. Note that this is not useful, the pattern already matches start of line.
The single '\$' character. Note that this is not useful, the pattern already matches end of line.
The single '[' character for a [] character range.
The single '~' character. |
|---|---|

When using character classes in expressions (see [/\i](#) for an overview), terms containing the "\+" quantifier can be written in the scanf() "%*" notation. Example: "%\\d%\\+" ("\d+", "any number") is equivalent to "%*\\d". Important **note**: The `\(...\)` grouping of sub-matches can not be used in format

specifications because it is reserved for internal conversions.

Multiple entries in **'errorformat'**

efm-entries

To be able to detect output from several compilers, several format patterns may be put in **'errorformat'**, separated by commas (note: blanks after the comma are ignored). The first pattern that has a complete match is used. If no match is found, matching parts from the last one will be used, although the file name is removed and the error message is set to the whole message. If there is a pattern that may match output from several compilers (but not in a right way), put it after one that is more restrictive.

To include a comma in a pattern precede it with a backslash (you have to type two in a `":set"` command). To include a backslash itself give two backslashes (you have to type four in a `":set"` command). You also need to put a backslash before a space for `":set"`.

Valid matches

quickfix-valid

If a line does not completely match one of the entries in **'errorformat'**, the whole line is put in the error message and the entry is marked "not valid" These lines are skipped with the `":cn"` and `":cp"` commands (unless there is no valid line at all). You can use `":cl!"` to display all the error messages.

If the error format does not contain a file name Vim cannot switch to the correct file. You will have to do this by hand.

Examples

The format of the file from the Amiga Aztec compiler is:

```
filename>linenumber:columnnumber:errortype:errornumber:errormessage

filename      name of the file in which the error was detected
linenumber    line number where the error was detected
columnnumber  column number where the error was detected
errortype     type of the error, normally a single 'E' or 'W'
errornumber   number of the error (for lookup in the manual)
errormessage  description of the error
```

This can be matched with this **'errorformat'** entry:

```
%f>%l:%c:%t:%n:%m
```

Some examples for C compilers that produce single-line error outputs:

```
%f:%l:\ %t%*[^0123456789]%n:\ %m      for Manx/Aztec C error messages
                                       (scanf() doesn't understand [0-9])
%f\ %l\ %t%*[^0-9]%n:\ %m            for SAS C
\"%f\"\\,%*[^0-9]%l:\ %m              for generic C compilers
%f:%l:\ %m                            for GCC
%f:%l:\ %m,%Dgmake[%*\\d]:\ Entering\ directory\ `f',
%Dgmake[%*\\d]:\ Leaving\ directory\ `f'
```


| | |
|--|--|
| <code>%f(%l)\ :\ %*[^:]:\ %m</code> | for GCC with gmake (concat the lines!) |
| <code>%f(%l)\ :\ %t%*[^0-9]%n:\ %m</code> | old SCO C compiler (pre-OS5) |
| <code>%f:%l:\ %m,In\ file\ included\ from\ %f:%l:,\ ^I\ ^Ifrom\ %f:%l%
%m</code> | idem, with error type and number |
| | for GCC, with some extras |

Extended examples for the handling of multi-line messages are given below, see [errorformat-Jikes](#) and [errorformat-LaTeX](#) .

Note the backslash in front of a space and double quote. It is required for the `:set` command. There are two backslashes in front of a comma, one for the `:set` command and one to avoid recognizing the comma as a separator of error formats.

Filtering messages

If you have a compiler that produces error messages that do not fit in the format string, you could write a program that translates the error messages into this format. You can use this program with the `:make` command by changing the `'makeprg'` option. For example:

```
:set mp=make\ \\\|&\ error_filter
```

The backslashes before the pipe character are required to avoid it to be recognized as a command separator. The backslash before each space is required for the set command.

8. The directory stack quickfix-directory-stack

Quickfix maintains a stack for saving all used directories parsed from the make output. For GNU-make this is rather simple, as it always prints the absolute path of all directories it enters and leaves. Regardless if this is done via a `'cd'` command in the makefile or with the parameter `"-C dir"` (change to directory before reading the makefile). It may be useful to use the switch `"-w"` to force GNU-make to print out the working directory before and after processing.

Maintaining the correct directory is more complicated if you don't use GNU-make. AIX-make for example doesn't print any information about its working directory. Then you need to enhance the makefile. In the makefile of LessTif there is a command which echoes `"Making {target} in {dir}"`. The special problem here is that it doesn't print information on leaving the directory and that it doesn't print the absolute path.

To solve the problem with relative paths and missing `"leave directory"` messages Vim uses the following algorithm:

- 1) Check if the given directory is a subdirectory of the current directory. If this is true, store it as the current directory.
- 2) If it is not a subdir of the current directory, try if this is a subdirectory of one of the upper directories.
- 3) If the directory still isn't found, it is assumed to be a subdirectory of Vim's current directory.

Additionally it is checked for every file, if it really exists in the identified directory. If not, it is searched in all other directories of the directory stack (NOT the directory subtree!). If it is still not found, it is assumed that it is in Vim's current directory.

There are limitations in this algorithm. These examples assume that make just prints information about entering a directory in the form "Making all in dir".

1) Assume you have following directories and files:

```
./dir1
./dir1/file1.c
./file1.c
```

If make processes the directory "./dir1" before the current directory and there is an error in the file "./file1.c", you will end up with the file "./dir1/file.c" loaded by Vim.

This can only be solved with a "leave directory" message.

2) Assume you have following directories and files:

```
./dir1
./dir1/dir2
./dir2
```

You get the following:

| Make output | Directory interpreted by Vim |
|--------------------|------------------------------|
| Making all in dir1 | ./dir1 |
| Making all in dir2 | ./dir1/dir2 |
| Making all in dir2 | ./dir1/dir2 |

This can be solved by printing absolute directories in the "enter directory" message or by printing "leave directory" messages.

To avoid this problem, ensure to print absolute directory names and "leave directory" messages.

Examples for Makefiles:

Unix:

```
libs:
    for dn in $(LIBDIRS); do
        (cd $$dn; echo "Entering dir '$$(pwd)'; make); \
        echo "Leaving dir"; \
    done
```

Add

```
%DEntering\ dir\ '%f',%XLeaving\ dir
```

to your **'errorformat'** to handle the above output.

Note that Vim doesn't check if the directory name in a "leave directory" messages is the current directory. This is why you could just use the message "Leaving dir".

9. Specific error file formats

errorformats

errorformat-Jikes

Jikes(TM), a source-to-bytecode Java compiler published by IBM Research, produces simple multi-line error messages.

An **'errorformat'** string matching the produced messages is shown below. The following lines can be placed in the user's `vimrc` to overwrite Vim's recognized default formats, or see `:set+=` how to install this format additionally to the default.

```
:set efm=%A%f:%l:%c:%*\d:%*\d:,\n    \%C%*\s%trror:%m,\n    \+%C%*[^:]*%trror:%m,\n    \%C%*\s%tarning:%m,\n    \%C%m
```

Jikes(TM) produces a single-line error message when invoked with the option "+E", and can be matched with the following:

```
:setl efm=%f:%l:%v:%*\d:%*\d:%*\s%m
```

errorformat-javac

This **'errorformat'** has been reported to work well for javac, which outputs a line with "^" to indicate the column of the error:

```
:setl efm=%A%f:%l:\ %m,%-Z%p^,%-C%.%#
```

or:

```
:setl efm=%A%f:%l:\ %m,%+Z%p^,%+C%.%#,%-G%.%#
```

Here is an alternative from Michael F. Lamb for Unix that filters the errors first:

```
:setl errorformat=%Z%f:%l:\ %m,%A%p^,%-G%*[^sl]%.%#\n:setl makeprg=javac\ %:S\ 2>&1\ \\\|\ vim-javac-filter
```

You need to put the following in "vim-javac-filter" somewhere in your path (e.g., in ~/bin) and make it executable:

```
#!/bin/sed -f\n/^\$/s/\t/ /g;/:[0-9]\+:/[h;d];/^[ \t]*\^/G;
```

In English, that sed script:

- Changes single tabs to single spaces and
- Moves the line with the filename, line number, error message to just after the pointer line. That way, the unused error text between doesn't break vim's notion of a "multi-line message" and also doesn't force us to include it as a "continuation of a multi-line message."

errorformat-ant

For ant (<http://jakarta.apache.org/>) the above errorformat has to be modified to honour the leading [javac] in front of each javac output line:

```
:set efm=%A\ %#[javac]\ %f:%l:\ %m,%-Z\ %#[javac]\ %p^,%-C%.%#
```

The **'errorformat'** can also be configured to handle ant together with either

javac or jikes. If you're using jikes, you should tell ant to use jikes' +E command line switch which forces jikes to generate one-line error messages. This is what the second line (of a build.xml file) below does:

```
<property name = "build.compiler"      value = "jikes"/>
<property name = "build.compiler.emacs" value = "true"/>
```

The **'errorformat'** which handles ant with both javac and jikes is:

```
:set efm=\ %#[javac]\ %%%f:%l:%c:%*\d:%*\d:\ %t%[%^:]%#:%m,
\%A\ %#[javac]\ %f:%l:\ %m,%-Z\ %#[javac]\ %p^,%-C%.%#
```

errorformat-jade

parsing jade (see <http://www.jclark.com/>) errors is simple:

```
:set efm=jade:%f:%l:%c:%t:%m
```

errorformat-LaTeX

The following is an example how an **'errorformat'** string can be specified for the (La)TeX typesetting system which displays error messages over multiple lines. The output of ":clist" and ":cc" etc. commands displays multi-lines in a single line, leading white space is removed. It should be easy to adopt the above LaTeX errorformat to any compiler output consisting of multi-line errors.

The commands can be placed in a **vimrc** file or some other Vim script file, e.g. a script containing LaTeX related stuff which is loaded only when editing LaTeX sources.

Make sure to copy all lines of the example (in the given order), afterwards remove the comment lines. For the '\ ' notation at the start of some lines see [line-continuation](#) .

First prepare **'makeprg'** such that LaTeX will report multiple errors; do not stop when the first error has occurred:

```
:set makeprg=latex\ \\\nonstopmode\ \\\input\\\{$*}
```

Start of multi-line error messages:

```
:set efm=%E!\ LaTeX\ %trror:\ %m,
\%E!\ %m,
```

Start of multi-line warning messages; the first two also include the line number. Meaning of some regular expressions:
- "%.%#" (".*") matches a (possibly empty) string
- "%*\d" ("\d+") matches a number

```
\%+WLaTeX\ %.%#Warning:\ %.%#line\ %l%.%#,
\%+W%.%#\ at\ lines\ %l--%*\d,
\%WLaTeX\ %.%#Warning:\ %m,
```

Possible continuations of error/warning messages; the first one also includes the line number:

```
\%Cl.%l\ %m,
\%+C\ \ %m.,
\%+C%.%#-%.%#,
\%+C%.%#[ ]%.%#,
\%+C[ ]%.%#,
\%+C%.%#[{ }\\\%.%#,
\%+C<%.%#>%.%#,
\%C\ \ %m,
```

Lines that match the following patterns do not contain any

important information; do not include them in messages:

```
\%-GSee\ the\ LaTeX%m,  
\%-GType\ \ H\ <return>%m,  
\%-G\ ...%.%#,  
\%-G%.%#\ (C)\ %.%#,  
\%-G(see\ the\ transcript%.%#),
```

Generally exclude any empty or whitespace-only line from being displayed:

```
\%-G\\s%#,
```

The LaTeX output log does not specify the names of erroneous source files per line; rather they are given globally, enclosed in parentheses.

The following patterns try to match these names and store them in an internal stack. The patterns possibly scan over the same input line (one after another), the trailing "%r" conversion indicates the "rest" of the line that will be parsed in the next go until the end of line is reached.

Overread a file name enclosed in '('...')'; do not push it on a stack since the file apparently does not contain any error:

```
\%+O(%f)%r,
```

Push a file name onto the stack. The name is given after '(':

```
\%+P(%f)%r,  
\%+P\ \%\\=(%f)%r,  
\%+P%*[^()](%f)%r,  
\%+P[%\\d%[^()]]#(%f)%r,
```

Pop the last stored file name when a ')' is scanned:

```
\%+Q)%r,  
\%+Q%*[^()])%r,  
\%+Q[%\\d%*[^()])%r
```

Note that in some cases file names in the LaTeX output log cannot be parsed properly. The parser might have been messed up by unbalanced parentheses then. The above example tries to catch the most relevant cases only. You can customize the given setting to suit your own purposes, for example, all the annoying "Overfull ..." warnings could be excluded from being recognized as an error.

Alternatively to filtering the LaTeX compiler output, it is also possible to directly read the *.log file that is produced by the [La]TeX compiler. This contains even more useful information about possible error causes. However, to properly parse such a complex file, an external filter should be used. See the description further above how to make such a filter known by Vim.

errorformat-Perl

In \$VIMRUNTIME/tools you can find the efm_perl.pl script, which filters Perl error messages into a format that quickfix mode will understand. See the start of the file about how to use it. (This script is deprecated, see [compiler-perl](#) .)

=====
10. Customizing the quickfix window

quickfix-window-function

The default format for the lines displayed in the quickfix window and location list window is:

```
<filename>|<lnum> col <col>|<text>
```

The values displayed in each line correspond to the "bufnr", "lnum", "col" and "text" fields returned by the `getqflist()` function.

For some quickfix/location lists, the displayed text needs to be customized. For example, if only the filename is present for a quickfix entry, then the two "|" field separator characters after the filename are not needed. Another use case is to customize the path displayed for a filename. By default, the complete path (which may be too long) is displayed for files which are not under the current directory tree. The file path may need to be simplified to a common parent directory.

The displayed text can be customized by setting the **'quickfixtextfunc'** option to a Vim function. This function will be called with a dict argument and should return a List of strings to be displayed in the quickfix or location list window. The dict argument will have the following fields:

quickfix	set to 1 when called for a quickfix list and 0 when called for a location list.
winid	for a location list, set to the id of the window with the location list. For a quickfix list, set to 0. Can be used in <code>getloclist()</code> to get the location list entry.
id	quickfix or location list identifier
start_idx	index of the first entry for which text should be returned
end_idx	index of the last entry for which text should be returned

The function should return a single line of text to display in the quickfix window for each entry from `start_idx` to `end_idx`. The function can obtain information about the entries using the `getqflist()` function and specifying the quickfix list identifier "id". For a location list, `getloclist()` function can be used with the "winid" argument. If an empty list is returned, then the default format is used to display all the entries. If an item in the returned list is an empty string, then the default format is used to display the corresponding entry.

If a quickfix or location list specific customization is needed, then the **'quickfixtextfunc'** attribute of the list can be set using the `setqflist()` or `setloclist()` function. This overrides the global **'quickfixtextfunc'** option.

The example below displays the list of old files (`v:oldfiles`) in a quickfix window. As there is no line, column number and error text information associated with each entry, the **'quickfixtextfunc'** function returns only the filename.

Example:

```
" create a quickfix list from v:oldfiles
call setqflist([], ' ', {'lines' : v:oldfiles, 'efm' : '%f',
                        \ 'quickfixtextfunc' : 'QfOldFiles'})

func QfOldFiles(info)
  " get information about a range of quickfix entries
  let items = getqflist({'id' : a:info.id, 'items' : 1}).items
```

```
let l = []
for idx in range(a:info.start_idx - 1, a:info.end_idx - 1)
    " use the simplified file name
    call add(l, framemodify(bufname(items[idx].bufnr), ':p:.'))
endfor
return l
endfunc
```


ADA FILE TYPE PLUG-INS REFERENCE MANUAL

ADA

ada.vim

1. Syntax Highlighting	ft-ada-syntax
2. File type Plug-in	ft-ada-plugin
3. Omni Completion	ft-ada-omni
3.1 Omni Completion with "gnat xref"	gnat-xref
3.2 Omni Completion with "ctags"	ada-ctags
4. Compiler Support	ada-compiler
4.1 GNAT	compiler-gnat
4.2 Dec Ada	compiler-decada
5. References	ada-reference
5.1 Options	ft-ada-options
5.2 Commands	ft-ada-commands
5.3 Variables	ft-ada-variables
5.4 Constants	ft-ada-constants
5.5 Functions	ft-ada-functions
6. Extra Plug-ins	ada-extra-plugins

1. Syntax Highlighting

ft-ada-syntax

This mode is designed for the 2005 edition of Ada ("Ada 2005"), which includes support for objected-programming, protected types, and so on. It handles code written for the original Ada language ("Ada83", "Ada87", "Ada95") as well, though code which uses Ada 2005-only keywords will be wrongly colored (such code should be fixed anyway). For more information about Ada, see <http://www.adapower.com>.

The Ada mode handles a number of situations cleanly.

For example, it knows that the "-" in "-5" is a number, but the same character in "A-5" is an operator. Normally, a "with" or "use" clause referencing another compilation unit is coloured the same way as C's "#include" is coloured. If you have "Conditional" or "Repeat" groups coloured differently, then "end if" and "end loop" will be coloured as part of those respective groups.

You can set these to different colours using vim's "highlight" command (e.g., to change how loops are displayed, enter the command ":hi Repeat" followed by the colour specification; on simple terminals the colour specification ctermfg=White often shows well).

There are several options you can select in this Ada mode. See [ft-ada-options](#) for a complete list.

To enable them, assign a value to the option. For example, to turn one on:

```
> let g:ada_standard_types = 1
```

To disable them use `":unlet"`. Example:

```
> unlet g:ada_standard_types
```

You can just use `":"` and type these into the command line to set these temporarily before loading an Ada file. You can make these option settings permanent by adding the `"let"` command(s), without a colon, to your `"~/.vimrc"` file.

Even on a slow (90Mhz) PC this mode works quickly, but if you find the performance unacceptable, turn on `g:ada_withuse_ordinary` .

Syntax folding instructions (`fold-syntax`) are added when `g:ada_folding` is set.

2. File type Plug-in

`ft-ada-indent` `ft-ada-plugin`

The Ada plug-in provides support for:

- auto indenting (`indent.txt`)
- insert completion (`i_CTRL-N`)
- user completion (`i_CTRL-X_CTRL-U`)
- tag searches (`tagsrch.txt`)
- Quick Fix (`quickfix.txt`)
- backspace handling (`'backspace'`)
- comment handling (`'comments'` , `'commentstring'`)

The plug-in only activates the features of the Ada mode whenever an Ada file is opened and adds Ada related entries to the main and pop-up menu.

3. Omni Completion

`ft-ada-omni`

The Ada omni-completions (`i_CTRL-X_CTRL-O`) uses tags database created either by `"gnat xref -v"` or the "Universal Ctags" (<https://ctags.io>). The complete function will automatically detect which tool was used to create the tags file.

3.1 Omni Completion with "gnat xref"

`gnat-xref`

GNAT XREF uses the compiler internal information (ali-files) to produce the tags file. This has the advantage to be 100% correct and the option of deep nested analysis. However the code must compile, the generator is quite slow and the created tags file contains only the basic Ctags information for each entry - not enough for some of the more advanced Vim code browser plug-ins.

NOTE: `"gnat xref -v"` is very tricky to use as it has almost no diagnostic output - If nothing is printed then usually the parameters are wrong.

Here some important tips:

- 1) You need to compile your code first and use the "-a0" option to point to your .ali files.
- 2) "gnat xref -v ../Include/adacl.ads" won't work - use the "gnat xref -v -aI../Include adacl.ads" instead.
- 3) "gnat xref -v -aI../Include *.ad?" won't work - use "cd ../Include" and then "gnat xref -v *.ad?"
- 4) Project manager support is completely broken - don't even try "gnat xref -Padacl.gpr".
- 5) Vim is faster when the tags file is sorted - use "sort --unique --ignore-case --output=tags tags" .
- 6) Remember to insert "!_TAG_FILE_SORTED 2 %sort ui" as first line to mark the file assorted.

3.2 Omni Completion with "ctags"

ada-ctags

Universal/Exuberant Ctags use their own multi-language code parser. The parser is quite fast, produces a lot of extra information and can run on files which currently do not compile.

There are also lots of other Vim-tools which use Universal/Exuberant Ctags. Universal Ctags is preferred, Exuberant Ctags is no longer being developed.

You will need to install Universal Ctags which is available from <https://ctags.io>

The Ada parser for Universal/Exuberant Ctags is fairly new - don't expect complete support yet.

4. Compiler Support

ada-compiler

The Ada mode supports more than one Ada compiler and will automatically load the compiler set in `g:ada_default_compiler` whenever an Ada source is opened. The provided compiler plug-ins are split into the actual compiler plug-in and a collection of support functions and variables. This allows the easy development of specialized compiler plug-ins fine tuned to your development environment.

4.1 GNAT

compiler-gnat

GNAT is the only free (beer and speech) Ada compiler available. There are several versions available which differ in the licence terms used.

The GNAT compiler plug-in will perform a compile on pressing <F7> and then immediately shows the result. You can set the project file to be used by setting:

```
> call g:gnat.Set_Project_File ('my_project.gpr')
```

Setting a project file will also create a Vim session (`views-sessions`) so - like with the GPS - opened files, window positions etc. will be remembered separately for all projects.

GNAT OBJECT gnat_members

`g:gnat.Make()` g:gnat.Make()
Calls `g:gnat.Make_Command` and displays the result inside a `quickfix` window.

`g:gnat.Pretty()` g:gnat.Pretty()
Calls `g:gnat.Pretty_Program`

`g:gnat.Find()` g:gnat.Find()
Calls `g:gnat.Find_Program`

`g:gnat.Tags()` g:gnat.Tags()
Calls `g:gnat.Tags_Command`

`g:gnat.Set_Project_File([file])` g:gnat.Set_Project_File()
Set gnat project file and load associated session. An open project will be closed and the session written. If called without file name the file selector opens for selection of a project file. If called with an empty string then the project and associated session are closed.

`g:gnat.Project_File` string g:gnat.Project_File
Current project file.

`g:gnat.Make_Command` string g:gnat.Make_Command
External command used for `g:gnat.Make()` ('makeprg').

`g:gnat.Pretty_Program` string g:gnat.Pretty_Program
External command used for `g:gnat.Pretty()`

`g:gnat.Find_Program` string g:gnat.Find_Program
External command used for `g:gnat.Find()`

`g:gnat.Tags_Command` string g:gnat.Tags_Command
External command used for `g:gnat.Tags()`

`g:gnat.Error_Format` string `g:gnat.Error_Format`
 Error format (`'errorformat'`)

4.2 Dec Ada

`compiler-hpada` `compiler-decada`
`compiler-vaxada` `compiler-compaqada`

Dec Ada (also known by - in chronological order - VAX Ada, Dec Ada, Compaq Ada and HP Ada) is a fairly dated Ada 83 compiler. Support is basic: `<F7>` will compile the current unit.

The Dec Ada compiler expects the package name and not the file name to be passed as a parameter. The compiler plug-in supports the usual file name convention to convert the file into a unit name. Both '-' and '_' are allowed as separators.

`decada_members`
DEC ADA OBJECT

`g:decada.Make()` function `g:decada.Make()`
 Calls `g:decada.Make_Command` and displays the result inside a `quickfix` window.

`g:decada.Unit_Name()` function `g:decada.Unit_Name()`
 Get the Unit name for the current file.

`g:decada.Make_Command` string `g:decada.Make_Command`
 External command used for `g:decada.Make()` (`'makeprg'`).

`g:decada.Error_Format` string `g:decada.Error_Format`
 Error format (`'errorformat'`).

5. References

`ada-reference`

5.1 Options

`ft-ada-options`

`g:ada_standard_types` bool (true when exists) `g:ada_standard_types`
 Highlight types in package Standard (e.g., "Float").

`g:ada_space_errors`
`g:ada_no_trail_space_error`
`g:ada_no_tab_space_error`
`g:ada_all_tab_usage`

`g:ada_space_errors` `bool` (true when exists)
Highlight extraneous errors in spaces ...
`g:ada_no_trail_space_error`
- but ignore trailing spaces at the end of a line
`g:ada_no_tab_space_error`
- but ignore tabs after spaces
`g:ada_all_tab_usage`
- highlight all tab use

[g:ada_line_errors](#)

`g:ada_line_errors` `bool` (true when exists)
Highlight lines which are too long. **Note:** This highlighting option is quite CPU intensive.

[g:ada_rainbow_color](#)

`g:ada_rainbow_color` `bool` (true when exists)
Use rainbow colours for '(' and ')'. You need the `rainbow_parenthesis` for this to work.

[g:ada_folding](#)

`g:ada_folding` `set` ('sigpft')
Use folding for Ada sources.
's': activate syntax folding on load
 'p': fold packages
 'f': fold functions and procedures
 't': fold types
 'c': fold conditionals
'g': activate gnat pretty print folding on load
 'i': lone **'is'** folded with line above
 'b': lone **'begin'** folded with line above
 'p': lone **'private'** folded with line above
 'x': lone **'exception'** folded with line above
'i': activate indent folding on load

Note: Syntax folding is in an early (unusable) stage and indent or gnat pretty folding is suggested.

For gnat pretty folding to work the following settings are suggested: `-cl3 -M79 -c2 -c3 -c4 -A1 -A2 -A3 -A4 -A5`

For indent folding to work the following settings are suggested: `shiftwidth=3 softtabstop=3`

[g:ada_abbrev](#)

`g:ada_abbrev` `bool` (true when exists)
Add some abbreviations. This feature is more or less superseded by the various completion methods.

[g:ada_withuse_ordinary](#)

`g:ada_withuse_ordinary` `bool` (true when exists)
Show "with" and "use" as ordinary keywords (when used to reference other compilation units they're normally highlighted specially).

`g:ada_begin_preproc` bool (true when exists) `g:ada_begin_preproc`
Show all begin-like keywords using the colouring of C
preprocessor commands.

`g:ada_omni_with_keywords`
`g:ada_omni_with_keywords`
Add Keywords, Pragmas, Attributes to omni-completions
(`compl-omni`). **Note:** You can always complete then with user
completion (`i_CTRL-X_CTRL-U`).

`g:ada_extended_tagging` enum ('jump', 'list') `g:ada_extended_tagging`
use extended tagging, two options are available
'jump': use tjump to jump.
'list': add tags quick fix list.
Normal tagging does not support function or operator
overloading as these features are not available in C and
tagging was originally developed for C.

`g:ada_extended_completion` `g:ada_extended_completion`
`g:ada_extended_completion`
Uses extended completion for <C-N> and <C-R> completions
(`i_CTRL-N`). In this mode the '.' is used as part of the
identifier so that 'Object.Method' or 'Package.Procedure' are
completed together.

`g:ada_gnat_extensions` bool (true when exists) `g:ada_gnat_extensions`
Support GNAT extensions.

`g:ada_with_gnat_project_files` bool (true when exists) `g:ada_with_gnat_project_files`
Add gnat project file keywords and Attributes.

`g:ada_default_compiler` string `g:ada_default_compiler`
set default compiler. Currently supported are **'gnat'** and
'decada'.

An "exists" type is a boolean considered true when the variable is defined and
false when the variable is undefined. The value to which the variable is set
makes no difference.

5.2 Commands

`:AdaRainbow` `ft-ada-commands` `:AdaRainbow`
Toggles rainbow colour (`g:ada_rainbow_color`) mode for
'(' and ')'.
`ft-ada-commands`

`:AdaLines` `:AdaLines`
Toggles line error (`g:ada_line_errors`) display.

<code>:AdaSpaces</code>	Toggles space error (<code>g:ada_space_errors</code>) display.	<code>:AdaSpaces</code>
<code>:AdaTagDir</code>	Creates tags file for the directory of the current file.	<code>:AdaTagDir</code>
<code>:AdaTagFile</code>	Creates tags file for the current file.	<code>:AdaTagFile</code>
<code>:AdaTypes</code>	Toggles standard types (<code>g:ada_standard_types</code>) colour.	<code>:AdaTypes</code>
<code>:GnatFind</code>	Calls <code>g:gnat.Find()</code>	<code>:GnatFind</code>
<code>:GnatPretty</code>	Calls <code>g:gnat.Pretty()</code>	<code>:GnatPretty</code>
<code>:GnatTags</code>	Calls <code>g:gnat.Tags()</code>	<code>:GnatTags</code>

5.3 Variables

`ft-ada-variables`

<code>g:gnat</code>	object	<code>g:gnat</code>
	Control object which manages GNAT compiles. The object is created when the first Ada source code is loaded provided that <code>g:ada_default_compiler</code> is set to ' gnat '. See <code>gnat_members</code> for details.	
<code>g:decada</code>	object	<code>g:decada</code>
	Control object which manages Dec Ada compiles. The object is created when the first Ada source code is loaded provided that <code>g:ada_default_compiler</code> is set to ' decada '. See <code>decada_members</code> for details.	

5.4 Constants

`ft-ada-constants`

All constants are locked. See `:lockvar` for details.

<code>g:ada#WordRegex</code>	string	<code>g:ada#WordRegex</code>
	Regular expression to search for Ada words.	
<code>g:ada#DotWordRegex</code>	string	<code>g:ada#DotWordRegex</code>
	Regular expression to search for Ada words separated by dots.	

`g:ada#Comment` string `g:ada#Comment`
Regular expression to search for Ada comments.

`g:ada#Keywords` list of dictionaries `g:ada#Keywords`
List of keywords, attributes etc. pp. in the format used by omni completion. See `complete-items` for details.

`g:ada#Ctags_Kinds` dictionary of lists `g:ada#Ctags_Kinds`
Dictionary of the various kinds of items which the Ada support for Ctags generates.

5.5 Functions

`ft-ada-functions`

`ada#Word([line], [col])` `ada#Word()`
Return full name of Ada entity under the cursor (or at given line/column), stripping white space/newlines as necessary.

`ada#List_Tag([line], [col])` `ada#Listtags()`
List all occurrences of the Ada entity under the cursor (or at given line/column) inside the quick-fix window.

`ada#Jump_Tag (ident, mode)` `ada#Jump_Tag()`
List all occurrences of the Ada entity under the cursor (or at given line/column) in the tag jump list. Mode can either be `'tjump'` or `'stjump'`.

`ada#Create_Tags (option)` `ada#Create_Tags()`
Creates tag file using Ctags. The option can either be `'file'` for the current file, `'dir'` for the directory of the current file or a file name.

`gnat#Insert_Tags_Header()` `gnat#Insert_Tags_Header()`
Adds the tag file header (!TAG_) information to the current file which are missing from the GNAT XREF output.

`ada#Switch_Syntax_Option (option)` `ada#Switch_Syntax_Option()`
Toggles highlighting options on or off. Used for the Ada menu.

`gnat#New ()` `gnat#New()`
Create a new gnat object. See `g:gnat` for details.

6. Extra Plugins

`ada-extra-plugins`

You can optionally install the following extra plug-ins. They work well with Ada and enhance the ability of the Ada mode:

backup.vim

http://www.vim.org/scripts/script.php?script_id=1537

Keeps as many backups as you like so you don't have to.

rainbow_parenthesis.vim

http://www.vim.org/scripts/script.php?script_id=1561

Very helpful since Ada uses only '(' and ')'.

nerd_comments.vim

http://www.vim.org/scripts/script.php?script_id=1218

Excellent commenting and uncommenting support for almost any programming language.

matchit.vim

http://www.vim.org/scripts/script.php?script_id=39

'%' jumping for any language. The normal '%' jump only works for '{}' style languages. The Ada mode will set the needed search patterns.

taglist.vim

http://www.vim.org/scripts/script.php?script_id=273

Source code explorer sidebar. There is a patch for Ada available.

The GNU Ada Project distribution (<http://gnuada.sourceforge.net>) of Vim contains all of the above.

=====
vim: textwidth=78 nowrap tabstop=8 shiftwidth=4 softtabstop=4 noexpandtab

`ft_ps1.txt` A Windows PowerShell syntax plugin for Vim

Author: Peter Provost <<https://www.github.com/PProvost>>

License: Apache 2.0

URL: <https://github.com/PProvost/vim-ps1>

INTRODUCTION

[ps1-syntax](#)

This plugin provides Vim syntax, indent and filetype detection for Windows PowerShell scripts, modules, and XML configuration files.

ABOUT

[ps1-about](#)

Grab the latest version or report a bug on GitHub:

<https://github.com/PProvost/vim-ps1>

FOLDING

[ps1-folding](#)

The `ps1` syntax file provides syntax folding (see `:syn-fold`) for script blocks and digital signatures in scripts.

When `'foldmethod'` is set to "syntax" then function script blocks will be folded unless you use the following in your `.vimrc` or before opening a script:

```
:let g:ps1_nofold_blocks = 1
```

Digital signatures in scripts will also be folded unless you use:

```
:let g:ps1_nofold_sig = 1
```

Note: syntax folding might slow down syntax highlighting significantly, especially for large files.

COMPILER

[ps1-compiler](#)

The powershell `:compiler` script configures `:make` to execute the script in PowerShell.

It tries to pick a smart default PowerShell command: ``pwsh`` if available and ``powershell`` otherwise, but you can customize the command:

```
:let g:ps1_makeprg_cmd = '/path/to/pwsh'
```

To configure whether to show the exception type information:

```
:let g:ps1_efm_show_error_categories = 1
```

KEYWORD LOOKUP

[ps1-keyword](#)

To look up keywords using PowerShell's Get-Help, press the `K` key. For more convenient paging, the pager `less` should be installed, which is included in many Linux distributions and in macOS.

Many other distributions are available for Windows like <https://chocolatey.org/packages/less/>. Make sure `less` is in a directory listed in the `PATH` environment variable, which chocolatey above does.

`ft_raku.txt` The Raku programming language filetype

`vim-raku`

Vim-raku provides syntax highlighting, indentation, and other support for editing Raku programs.

1. Using Unicode in your Raku files `raku-unicode`

=====

1. Using Unicode in your Raku files `raku-unicode`

Defining new operators using Unicode symbols is a good way to make your Raku program easy to read. See:

<https://perl6advent.wordpress.com/2012/12/18/day-18-formulas-resistance-is-futile/>

While Raku does define ASCII alternatives for some common operators (see https://docs.raku.org/language/unicode_ascii), using the full range of Unicode operators is highly desirable. Your operating system provides input facilities, but using the features built in to Vim may be preferable.

The natural way to produce these symbols in Vim is to use digraph shortcuts (:help `digraphs-use`). Many of them are defined; type `:digraphs` to get the list. A convenient way to read the list of digraphs is to save them in a file. From the shell:

```
vim +'redir >/tmp/vim-digraphs-listing.txt' +digraphs +'redir END' +q
```

Some of them are available with standard Vim digraphs:

```
<< «      /0 ☒      !< ☒
>> »      Ob ☒      !> ☒
., ...    00 ∞      (C ☒
(U ☒     -: ÷      )C ☒
)U ☒     (_ ☒     >= ≥
?= ☒     )_ ☒     =< ≤
(- ☒     ?= ☒     != ≠
-) ☒     ?- ☒
```

The Greek alphabet is available with '*' followed by a similar Latin symbol:

```
*p π
*t ☒
*X x
```

Numbers, subscripts and superscripts are available with 's' and 'S':

```
0s 0      0S 0
1s 1      1S 1
2s 2      9S 9
```

But some don't come defined by default. Those are digraph definitions you can add in your `~/.vimrc` file.

```
exec 'digraph \\' .. char2nr('☒')
exec 'digraph \<' .. char2nr('☒')
exec 'digraph \>' .. char2nr('☒')
exec 'digraph (L' .. char2nr('☒')
exec 'digraph )L' .. char2nr('☒')
```

```

exec 'digraph (/ ' .. char2nr('⌘')
exec 'digraph )/ ' .. char2nr('⌘')
exec 'digraph )/ ' .. char2nr('⌘')
exec 'digraph U+ ' .. char2nr('⌘')
exec 'digraph 0- ' .. char2nr('⌘')
" Euler's constant
exec 'digraph ne ' .. char2nr('⌘')
" Raku's atomic operations marker
exec 'digraph @@ ' .. char2nr('⌘')

```

Alternatively, you can write Insert mode abbreviations that convert ASCII-based operators into their single-character Unicode equivalent.

```

iabbrev <buffer> !(<) ⌘
iabbrev <buffer> !(<=) ⌘
iabbrev <buffer> !(>) ⌘
iabbrev <buffer> !(>=) ⌘
iabbrev <buffer> !(cont) ⌘
iabbrev <buffer> !(elem) ⌘
iabbrev <buffer> != ≠
iabbrev <buffer> (&) ⌘
iabbrev <buffer> (+) ⌘
iabbrev <buffer> (-) ⌘
iabbrev <buffer> (.) ⌘
iabbrev <buffer> (<) ⌘
iabbrev <buffer> (<+) ⌘
iabbrev <buffer> (<=) ⌘
iabbrev <buffer> (>) ⌘
iabbrev <buffer> (>+) ⌘
iabbrev <buffer> (>=) ⌘
iabbrev <buffer> (\\|) ⌘
iabbrev <buffer> (^) ⌘
iabbrev <buffer> (atomic) ⌘
iabbrev <buffer> (cont) ⌘
iabbrev <buffer> (elem) ⌘
iabbrev <buffer> * ×
iabbrev <buffer> **0 0
iabbrev <buffer> **1 1
iabbrev <buffer> **2 2
iabbrev <buffer> **3 3
iabbrev <buffer> **4 4
iabbrev <buffer> **5 5
iabbrev <buffer> **6 6
iabbrev <buffer> **7 7
iabbrev <buffer> **8 8
iabbrev <buffer> **9 9
iabbrev <buffer> ... …
iabbrev <buffer> / ÷
iabbrev <buffer> << «
iabbrev <buffer> <<[=]<< «=«
iabbrev <buffer> <<[=]>> «=»
iabbrev <buffer> <= ≤
iabbrev <buffer> =~ ⌘
iabbrev <buffer> >= ≥
iabbrev <buffer> >> »

```

```
iabbrev <buffer> >>[=]<< >=<<
iabbrev <buffer> >>[=]>> >=>>
iabbrev <buffer> Inf ∞
iabbrev <buffer> atomic-add-fetch ☒+=
iabbrev <buffer> atomic-assign ☒=
iabbrev <buffer> atomic-fetch ☒
iabbrev <buffer> atomic-dec-fetch --☒
iabbrev <buffer> atomic-fetch-dec ☒--
iabbrev <buffer> atomic-fetch-inc ☒++
iabbrev <buffer> atomic-inc-fetch ++☒
iabbrev <buffer> atomic-sub-fetch ☒-=
iabbrev <buffer> e ☒
iabbrev <buffer> o ☒
iabbrev <buffer> pi π
iabbrev <buffer> set() ☒
iabbrev <buffer> tau ☒
```

CONTENTS

1. Introduction
2. Settings
3. Commands
4. Mappings

rust

rust-intro
 rust-settings
 rust-commands
 rust-mappings

INTRODUCTION

rust-intro

This plugin provides syntax and supporting functionality for the Rust filetype. It requires Vim 8 or higher for full functionality. Some commands will not work on earlier versions.

SETTINGS

rust-settings

This plugin has a few variables you can define in your vimrc that change the behavior of the plugin.

Some variables can be set buffer local (':b' prefix), and the buffer local will take precedence over the global `g:` counterpart.

g:rustc_path

g:rustc_path

Set this option to the path to rustc for use in the `:RustRun` and `:RustExpand` commands. If unset, "rustc" will be located in \$PATH:
`let g:rustc_path = $HOME."/bin/rustc"`

g:rustc_makeprg_no_percent

g:rustc_makeprg_no_percent

Set this option to 1 to have 'makeprg' default to "rustc" instead of "rustc %":
`let g:rustc_makeprg_no_percent = 1`

g:rust_conceal

g:rust_conceal

Set this option to turn on the basic `conceal` support:
`let g:rust_conceal = 1`

g:rust_conceal_mod_path

g:rust_conceal_mod_path

Set this option to turn on `conceal` for the path connecting token "::
`let g:rust_conceal_mod_path = 1`

g:rust_conceal_pub

g:rust_conceal_pub

Set this option to turn on `conceal` for the "pub" token:
`let g:rust_conceal_pub = 1`

g:rust_recommended_style

g:rust_recommended_style

Set this option to enable vim indentation and textwidth settings to conform to style conventions of the rust standard library (i.e. use 4 spaces for indents and sets `'textwidth'` to 99). This option is enabled by default. To disable it:

```
let g:rust_recommended_style = 0
```

g:rust_fold

g:rust_fold

Set this option to turn on `foldings` :
`let g:rust_fold = 1`

Value	Effect
0	No folding
1	Braced blocks are folded. All folds are open by default.
2	Braced blocks are folded. <code>'foldlevel'</code> is left at the global value (all folds are closed by default).

g:rust_bang_comment_leader

g:rust_bang_comment_leader

Set this option to 1 to preserve the leader on multi-line doc comments using the `/*!` syntax:

```
let g:rust_bang_comment_leader = 1
```

g:rust_use_custom_ctags_defs

g:rust_use_custom_ctags_defs

Set this option to 1 if you have customized ctags definitions for Rust and do not wish for those included with rust.vim to be used:

```
let g:rust_use_custom_ctags_defs = 1
```

NOTE: rust.vim's built-in definitions are only used for the Tagbar Vim plugin, if you have it installed, AND if Universal Ctags is not detected. This is because Universal Ctags already has built-in support for Rust when used with Tagbar.

Also, **note** that when using ctags other than Universal Ctags, it is not automatically used when generating `tags` files that Vim can use to navigate to definitions across different source files. Feel free to copy ``rust.vim/ctags/rust.ctags`` into your own ``~/.ctags`` if you wish to generate `tags` files.

g:ftplugin_rust_source_path

g:ftplugin_rust_source_path

Set this option to a path that should be prepended to **'path'** for Rust source files:

```
let g:ftplugin_rust_source_path = $HOME.'/dev/rust'
```

g:rustfmt_command

g:rustfmt_command

Set this option to the name of the **'rustfmt'** executable in your \$PATH. If not specified it defaults to **'rustfmt'** :

```
let g:rustfmt_command = 'rustfmt'
```

g:rustfmt_autosave

g:rustfmt_autosave

Set this option to 1 to run **:RustFmt** automatically when saving a buffer. If not specified it defaults to 0 :

```
let g:rustfmt_autosave = 0
```

There is also a buffer-local **b:rustfmt_autosave** that can be set for the same purpose, and can override the global setting.

g:rustfmt_autosave_if_config_present

g:rustfmt_autosave_if_config_present

Set this option to 1 to have **b:rustfmt_autosave** be set automatically if a **'rustfmt.toml'** file is present in any parent directly leading to the file being edited. If not set, default to 0:

```
let g:rustfmt_autosave_if_config_present = 0
```

This is useful to have **'rustfmt'** only execute on save, on projects that have **'rustfmt.toml'** configuration.

There is also a buffer-local **b:rustfmt_autosave_if_config_present** that can be set for the same purpose, which can override the global setting.

g:rustfmt_fail_silently

g:rustfmt_fail_silently

Set this option to 1 to prevent **'rustfmt'** from populating the **location-list** with errors. If not specified it defaults to 0:

```
let g:rustfmt_fail_silently = 0
```

g:rustfmt_options

g:rustfmt_options

Set this option to a string of options to pass to **'rustfmt'**. The write-mode is already set to **'overwrite'**. If not specified it defaults to '' :

```
let g:rustfmt_options = ''
```

g:rustfmt_emit_files

g:rustfmt_emit_files

If not specified rust.vim tries to detect the right parameter to pass to rustfmt based on its reported version. Otherwise, it determines whether to run rustfmt with **'--emit=files'** (when 1 is provided) instead of **'--write-mode=overwrite'**.

```
let g:rustfmt_emit_files = 0
```

`g:rust_playpen_url`

`g:rust_playpen_url`

Set this option to override the url for the playpen to use:

```
let g:rust_playpen_url = 'https://play.rust-lang.org/'
```

`g:rust_shortener_url`

`g:rust_shortener_url`

Set this option to override the url for the url shortener:

```
let g:rust_shortener_url = 'https://is.gd/'
```

`g:rust_clip_command`

`g:rust_clip_command`

Set this option to the command used in your OS to copy the Rust Play url to the clipboard:

```
let g:rust_clip_command = 'xclip -selection clipboard'
```

`g:cargo_makeprg_params`

`g:cargo_makeprg_params`

Set this option to the string of parameters to pass to cargo. If not specified it defaults to '\$*' :

```
let g:cargo_makeprg_params = 'build'
```

`g:cargo_shell_command_runner`

`g:cargo_shell_command_runner`

Set this option to change how to run shell commands for cargo commands

`:Cargo` , `:Cbuild` , `:Crun` , ...

By default, `:terminal` is used to run shell command in terminal window asynchronously. But if you prefer `:!` for running the commands, it can be specified:

```
let g:cargo_shell_command_runner = '!'
```

Integration with Syntastic

`rust-syntastic`

This plugin automatically integrates with the Syntastic checker. There are two checkers provided: `'rustc'`, and `'cargo'`. The latter invokes 'Cargo' in order to build code, and the former delivers a single edited '.rs' file as a compilation target directly to the Rust compiler, `'rustc'`.

Because Cargo is almost exclusively being used for building Rust code these days, `'cargo'` is the default checker.

```
let g:syntastic_rust_checkers = ['cargo']
```

If you would like to change it, you can set `'g:syntastic_rust_checkers'` to a different value.

`g:rust_cargo_avoid_whole_workspace`

b:rust_cargo_avoid_whole_workspace

g:rust_cargo_avoid_whole_workspace

When editing a crate that is part of a Cargo workspace, and this option is set to 1 (the default), then '**cargo**' will be executed directly in that crate directory instead of in the workspace directory. Setting 0 prevents this behavior - however be aware that if you are working in large workspace, Cargo commands may take more time, plus the Syntastic error list may include all the crates in the workspace.

```
let g:rust_cargo_avoid_whole_workspace = 0
```

g:rust_cargo_check_all_targets

b:rust_cargo_check_all_targets

g:rust_cargo_check_all_targets

When set to 1, the '**--all-targets**' option will be passed to cargo when Syntastic executes it, allowing the linting of all targets under the package.

The default is 0.

g:rust_cargo_check_all_features

b:rust_cargo_check_all_features

g:rust_cargo_check_all_features

When set to 1, the '**--all-features**' option will be passed to cargo when Syntastic executes it, allowing the linting of all features of the package.

The default is 0.

g:rust_cargo_check_examples

b:rust_cargo_check_examples

g:rust_cargo_check_examples

When set to 1, the '**--examples**' option will be passed to cargo when Syntastic executes it, to prevent the exclusion of examples from linting. The examples are normally under the '**examples/**' directory of the crate.

The default is 0.

g:rust_cargo_check_tests

b:rust_cargo_check_tests

g:rust_cargo_check_tests

When set to 1, the '**--tests**' option will be passed to cargo when Syntastic executes it, to prevent the exclusion of tests from linting. The tests are normally under the '**tests/**' directory of the crate.

The default is 0.

g:rust_cargo_check_benches

b:rust_cargo_check_benches

g:rust_cargo_check_benches

When set to 1, the '**--benches**' option will be passed to cargo when Syntastic executes it. The benches are normally under the '**benches/**' directory of the crate.

The default is 0.

Integration with auto-pairs

rust-auto-pairs

This plugin automatically configures the auto-pairs plugin not to duplicate single quotes, which are used more often for lifetime annotations than for single character literals.

`g:rust_keep_autopairs_default` `g:rust_keep_autopairs_default`

Don't override auto-pairs default for the Rust filetype. The default is 0.

COMMANDS

`rust-commands`

Invoking Cargo

This plug defines very simple shortcuts for invoking Cargo from with Vim.

<code>:Cargo <args></code>	Runs 'cargo' with the provided arguments.	<code>:Cargo</code>
<code>:Cbuild <args></code>	Shortcut for 'cargo build`.	<code>:Cbuild</code>
<code>:Cclean <args></code>	Shortcut for 'cargo clean`.	<code>:Cclean</code>
<code>:Cdoc <args></code>	Shortcut for 'cargo doc`.	<code>:Cdoc</code>
<code>:Cinit <args></code>	Shortcut for 'cargo init`.	<code>:Cinit</code>
<code>:Crun <args></code>	Shortcut for 'cargo run`.	<code>:Crun</code>
<code>:Ctest <args></code>	Shortcut for 'cargo test`.	<code>:Ctest</code>
<code>:Cupdate <args></code>	Shortcut for 'cargo update`.	<code>:Cupdate</code>
<code>:Cbench <args></code>	Shortcut for 'cargo bench`.	<code>:Cbench</code>
<code>:Csearch <args></code>	Shortcut for 'cargo search`.	<code>:Csearch</code>
<code>:Cpublish <args></code>	Shortcut for 'cargo publish`.	<code>:Cpublish</code>
<code>:Cinstall <args></code>	Shortcut for 'cargo install`.	<code>:Cinstall</code>

`:Cruntarget <args>` `:Cruntarget`
Shortcut for 'cargo run --bin' or 'cargo run --example', depending on the currently open buffer.

Formatting

`:RustFmt` `:RustFmt`
Runs `g:rustfmt_command` on the current buffer. If `g:rustfmt_options` is set then those will be passed to the executable.

If `g:rustfmt_fail_silently` is 0 (the default) then it will populate the `location-list` with the errors from `g:rustfmt_command`. If `g:rustfmt_fail_silently` is set to 1 then it will not populate the `location-list`.

`:RustFmtRange` `:RustFmtRange`
Runs `g:rustfmt_command` with selected range. See `:RustFmt` for any other information.

Playpen integration

`:RustPlay` `:RustPlay`
This command will only work if you have web-api.vim installed (available at <https://github.com/matttn/webapi-vim>). It sends the current selection, or if nothing is selected, the entirety of the current buffer to the Rust playpen, and emits a message with the shortened URL to the playpen.

`g:rust_playpen_url` is the base URL to the playpen, by default "<https://play.rust-lang.org/>".

`g:rust_shortener_url` is the base url for the shortener, by default "<https://is.gd/>".

`g:rust_clip_command` is the command to run to copy the playpen url to the clipboard of your system.

Evaluation of a single Rust file

NOTE: These commands are useful only when working with standalone Rust files, which is usually not the case for common Rust development. If you wish to building Rust crates from with Vim can should use Vim's make, Syntastic, or functionality from other plugins.

`:RustRun [args]` `:RustRun`
`:RustRun! [rustc-args] [--] [args]`

Compiles and runs the current file. If it has unsaved changes, it will be saved first using `:update`. If the current file is an unnamed buffer, it will be written to a temporary file first. The compiled binary is always placed in a temporary directory, but is run from the current directory.

The arguments given to `:RustRun` will be passed to the compiled binary.

If `!` is specified, the arguments are passed to rustc instead. A `--` argument will separate the rustc arguments from the arguments passed to the binary.

If `g:rustc_path` is defined, it is used as the path to rustc. Otherwise it is assumed rustc can be found in \$PATH.

`:RustExpand` [args] `:RustExpand`
`:RustExpand!` [TYPE] [args]

Expands the current file using `--pretty` and displays the results in a new split. If the current file has unsaved changes, it will be saved first using `:update`. If the current file is an unnamed buffer, it will be written to a temporary file first.

The arguments given to `:RustExpand` will be passed to rustc. This is largely intended for specifying various `--cfg` configurations.

If `!` is specified, the first argument is the expansion type to pass to rustc `--pretty`. Otherwise it will default to "expanded".

If `g:rustc_path` is defined, it is used as the path to rustc. Otherwise it is assumed rustc can be found in \$PATH.

`:RustEmitIr` [args] `:RustEmitIr`

Compiles the current file to LLVM IR and displays the results in a new split. If the current file has unsaved changes, it will be saved first using `:update`. If the current file is an unnamed buffer, it will be written to a temporary file first.

The arguments given to `:RustEmitIr` will be passed to rustc.

If `g:rustc_path` is defined, it is used as the path to rustc. Otherwise it is assumed rustc can be found in \$PATH.

`:RustEmitAsm` [args] `:RustEmitAsm`

Compiles the current file to assembly and displays the results in a new split. If the current file has unsaved changes, it will be saved first using `:update`. If the current file is an unnamed buffer, it will be written to a temporary file first.

The arguments given to `:RustEmitAsm` will be passed to rustc.

If `g:rustc_path` is defined, it is used as the path to rustc. Otherwise it is assumed rustc can be found in \$PATH.

Running test(s)

`:[N]RustTest[!]` `[options]` `:RustTest`
Runs a test under the cursor when the current buffer is in a cargo project with "cargo test" command. If the command did not find any test function under the cursor, it stops with an error message.

When N is given, adjust the size of the new window to N lines or columns.

When ! is given, runs all tests regardless of current cursor position.

When `[options]` is given, it is passed to "cargo" command arguments.

When the current buffer is outside cargo project, the command runs "rustc --test" command instead of "cargo test" as fallback. All tests are run regardless of adding ! since there is no way to run specific test function with rustc. `[options]` is passed to "rustc" command arguments in the case.

Takes optional modifiers (see `<mods>`):

- `:tab RustTest`
- `:belowright 16RustTest`
- `:leftabove vert 80RustTest`

rust.vim Debugging

`:RustInfo` `:RustInfo`
Emits debugging info of the Vim Rust plugin.

`:RustInfoToClipboard` `:RustInfoClipboard`
Saves debugging info of the Vim Rust plugin to the default register.

`:RustInfoToFile` `[filename]` `:RustInfoToFile`
Saves debugging info of the Vim Rust plugin to the given file, overwriting it.

MAPPINGS

`rust-mappings`

This plugin defines mappings for `[[` and `]]` to support hanging indents.

by David Fishburn

This is a filetype plugin to work with SQL files.

The Structured Query Language (SQL) is a standard which specifies statements that allow a user to interact with a relational database. Vim includes features for navigation, indentation and syntax highlighting.

1. Navigation	sql-navigation
1.1 Matchit	sql-matchit
1.2 Text Object Motions	sql-object-motions
1.3 Predefined Object Motions	sql-predefined-objects
1.4 Macros	sql-macros
2. SQL Dialects	sql-dialects
2.1 SQLSetType	SQLSetType
2.2 SQLGetType	SQLGetType
2.3 SQL Dialect Default	sql-type-default
3. Adding new SQL Dialects	sql-adding-dialects
4. OMNI SQL Completion	sql-completion
4.1 Static mode	sql-completion-static
4.2 Dynamic mode	sql-completion-dynamic
4.3 Tutorial	sql-completion-tutorial
4.3.1 Complete Tables	sql-completion-tables
4.3.2 Complete Columns	sql-completion-columns
4.3.3 Complete Procedures	sql-completion-procedures
4.3.4 Complete Views	sql-completion-views
4.4 Completion Customization	sql-completion-customization
4.5 SQL Maps	sql-completion-maps
4.6 Using with other filetypes	sql-completion-filetypes

1. Navigation

sql-navigation

The SQL ftplugin provides a number of options to assist with file navigation.

1.1 Matchit

sql-matchit

The matchit plugin (http://www.vim.org/scripts/script.php?script_id=39) provides many additional features and can be customized for different languages. The matchit plugin is configured by defining a local buffer variable, b:match_words. Pressing the % key while on various keywords will move the cursor to its match. For example, if the cursor is on an "if", pressing % will cycle between the "else", "elseif" and "end if" keywords.

The following keywords are supported:

```
if
elseif | elsif
else [if]
end if
```

```

[while condition] loop
    leave
    break
    continue
    exit
end loop

for
    leave
    break
    continue
    exit
end loop

do
    statements
doend

case
when
when
default
end case

merge
when not matched
when matched

create[ or replace] procedure|function|event
returns

```

1.2 Text Object Motions

[sql-object-motions](#)

Vim has a number of predefined keys for working with text [object-motions](#) . This filetype plugin attempts to translate these keys to maps which make sense for the SQL language.

The following [Normal](#) mode and [Visual](#) mode maps exist (when you edit a SQL file):

```

]]          move forward to the next 'begin'
[[          move backwards to the previous 'begin'
]]          move forward to the next 'end'
[[          move backwards to the previous 'end'

```

1.3 Predefined Object Motions

[sql-predefined-objects](#)

Most relational databases support various standard features, tables, indices, triggers and stored procedures. Each vendor also has a variety of proprietary objects. The next set of maps have been created to help move between these objects. Depends on which database vendor you are using, the list of objects must be configurable. The filetype plugin attempts to define many of the

standard objects, plus many additional ones. In order to make this as flexible as possible, you can override the list of objects from within your `vimrc` with the following:

```
let g:ftplugin_sql_objects = 'function,procedure,event,table,trigger' ..
    \ ',schema,service,publication,database,datatype,domain' ..
    \ ',index,subscription,synchronization,view,variable'
```

The following `Normal` mode and `Visual` mode maps have been created which use the above list:

```
]]          move forward to the next 'create <object name>'
[{}        move backward to the previous 'create <object name>'
```

Repeatedly pressing `]]` will cycle through each of these create statements:

```
create table t1 (
    ...
);

create procedure p1
begin
    ...
end;

create index i1 on t1 (c1);
```

The default setting for `g:ftplugin_sql_objects` is:

```
let g:ftplugin_sql_objects = 'function,procedure,event,' ..
    \ '\\(existing\\|\\global\\s\\+temporary\\s\\+\\)\\|{,1}' ..
    \ 'table,trigger' ..
    \ ',schema,service,publication,database,datatype,domain' ..
    \ ',index,subscription,synchronization,view,variable'
```

The above will also handle these cases:

```
create table t1 (
    ...
);
create existing table t2 (
    ...
);
create global temporary table t3 (
    ...
);
```

By default, the `ftplugin` only searches for `CREATE` statements. You can also override this via your `vimrc` with the following:

```
let g:ftplugin_sql_statements = 'create,alter'
```

The filetype plugin defines three types of comments:

1. `--`
2. `//`
3. `/*`
 `*`
 `*/`

The following `Normal` mode and `Visual` mode maps have been created to work

with comments:

```
]\"          move forward to the beginning of a comment  
[\"          move forward to the end of a comment
```

1.4 Macros

sql-macros

Vim's feature to find macro definitions, 'define', is supported using this regular expression:

```
\\c\\<\\(VARIABLE\\|DECLARE\\|IN\\|OUT\\|INOUT\\)\\>
```

This addresses the following code:

```
CREATE VARIABLE myVar1 INTEGER;  
  
CREATE PROCEDURE sp_test(  
    IN myVar2 INTEGER,  
    OUT myVar3 CHAR(30),  
    INOUT myVar4 NUMERIC(20,0)  
)  
BEGIN  
    DECLARE myVar5 INTEGER;  
  
    SELECT c1, c2, c3  
        INTO myVar2, myVar3, myVar4  
        FROM T1  
        WHERE c4 = myVar1;  
END;
```

Place your cursor on "myVar1" on this line:

```
WHERE c4 = myVar1;  
          ^
```

Press any of the following keys:

```
[d  
[D  
[CTRL-D
```

2. SQL Dialects

```
sql-dialects  sql-types  
sybase  TSQL  Transact-SQL  
sqlanywhere  
oracle  plsql  sqlj  
sqlserver  
mysql  postgresql  psql  
informix
```

All relational databases support SQL. There is a portion of SQL that is portable across vendors (ex. CREATE TABLE, CREATE INDEX), but there is a great deal of vendor specific extensions to SQL. Oracle supports the "CREATE OR REPLACE" syntax, column defaults specified in the CREATE TABLE statement and the procedural language (for stored procedures and triggers).

The default Vim distribution ships with syntax highlighting based on Oracle's PL/SQL. The default SQL indent script works for Oracle and SQL Anywhere. The default filetype plugin works for all vendors and should remain vendor neutral, but extendable.

Vim currently has support for a variety of different vendors, currently this is via syntax scripts. Unfortunately, to flip between different syntax rules you must either create:

1. New filetypes
2. Custom autocmds
3. Manual steps / commands

The majority of people work with only one vendor's database product, it would be nice to specify a default in your `vimrc`.

2.1 SQLSetType sqlsettype SQLSetType

For the people that work with many different databases, it is nice to be able to flip between the various vendors rules (indent, syntax) on a per buffer basis, at any time. The `ftplugin/sql.vim` file defines this function:

`SQLSetType`

Executing this function without any parameters will set the indent and syntax scripts back to their defaults, see `sql-type-default`. If you have turned off Vi's compatibility mode, `'compatible'`, you can use the `<Tab>` key to complete the optional parameter.

After typing the function name and a space, you can use the completion to supply a parameter. The function takes the name of the Vim script you want to source. Using the `cmdline-completion` feature, the `SQLSetType` function will search the `'runtimepath'` for all Vim scripts with a name containing `'sql'`. This takes the guess work out of the spelling of the names. The following are examples:

```
:SQLSetType
:SQLSetType sqloracle
:SQLSetType sqlanywhere
:SQLSetType sqlinformix
:SQLSetType mysql
```

The easiest approach is to use `<Tab>` character which will first complete the command name (`SQLSetType`), after a space and another `<Tab>`, display a list of available Vim script names:

```
:SQL<Tab><space><Tab>
```

2.2 SQLGetType sqlgettext SQLGetType

At anytime you can determine which SQL dialect you are using by calling the `SQLGetType` command. The `ftplugin/sql.vim` file defines this function:

`SQLGetType`

This will echo:

```
Current SQL dialect in use:sqlanywhere
```

2.3 SQL Dialect Default

sql-type-default

As mentioned earlier, the default syntax rules for Vim is based on Oracle (PL/SQL). You can override this default by placing one of the following in your `vimrc` :

```
let g:sql_type_default = 'sqlanywhere'  
let g:sql_type_default = 'sqlinformix'  
let g:sql_type_default = 'mysql'
```

If you added the following to your `vimrc` :

```
let g:sql_type_default = 'sqlinformix'
```

The next time edit a SQL file the following scripts will be automatically loaded by Vim:

```
ftplugin/sql.vim  
syntax/sqlinformix.vim  
indent/sql.vim
```

Notice `indent/sqlinformix.sql` was not loaded. There is no indent file for Informix, Vim loads the default files if the specified files does not exist.

3. Adding new SQL Dialects

sql-adding-dialects

If you begin working with a SQL dialect which does not have any customizations available with the default Vim distribution you can check <http://www.vim.org> to see if any customization currently exist. If not, you can begin by cloning an existing script. Read `filetype-plugins` for more details.

To help identify these scripts, try to create the files with a "sql" prefix. If you decide you wish to create customizations for the SQLite database, you can create any of the following:

```
Unix  
~/vim/syntax/sqlite.vim  
~/vim/indent/sqlite.vim  
Windows  
$VIM/vimfiles/syntax/sqlite.vim  
$VIM/vimfiles/indent/sqlite.vim
```

No changes are necessary to the `SQLSetType` function. It will automatically pick up the new SQL files and load them when you issue the `SQLSetType` command.

4. OMNI SQL Completion

sql-completion
omni-sql-completion

Vim 7 includes a code completion interface and functions which allows plugin developers to build in code completion for any language. Vim 7 includes code completion for the SQL language.

There are two modes to the SQL completion plugin, static and dynamic. The static mode populates the popups with the data generated from current syntax highlight rules. The dynamic mode populates the popups with data retrieved directly from a database. This includes, table lists, column lists, procedures names and more.

4.1 Static Mode

sql-completion-static

The static popups created contain items defined by the active syntax rules while editing a file with a filetype of SQL. The plugin defines (by default) various maps to help the user refine the list of items to be displayed.

The defaults static maps are:

```
imap <buffer> <C-C>a <C-\><C-O>:call sqlcomplete#Map('syntax')<CR><C-X><C-O>
imap <buffer> <C-C>k <C-\><C-O>:call sqlcomplete#Map('sqlKeyword')<CR><C-X><C-O>
imap <buffer> <C-C>f <C-\><C-O>:call sqlcomplete#Map('sqlFunction')<CR><C-X><C-O>
imap <buffer> <C-C>o <C-\><C-O>:call sqlcomplete#Map('sqlOption')<CR><C-X><C-O>
imap <buffer> <C-C>T <C-\><C-O>:call sqlcomplete#Map('sqlType')<CR><C-X><C-O>
imap <buffer> <C-C>s <C-\><C-O>:call sqlcomplete#Map('sqlStatement')<CR><C-X><C-O>
```

The use of "<C-C>" can be user chosen by using the following in your `.vimrc` as it may not work properly on all platforms:

```
let g:ftplugin_sql_omni_key = '<C-C>'
```

The static maps (which are based on the syntax highlight groups) follow this format:

```
imap <buffer> <C-C>k <C-\><C-O>:call sqlcomplete#Map('sqlKeyword')<CR><C-X><C-O>
imap <buffer> <C-C>k <C-\><C-O>:call sqlcomplete#Map('sqlKeyword\w*')<CR><C-X><C-O>
```

This command breaks down as:

```
imap          - Create an insert map
<buffer>     - Only for this buffer
<C-C>k       - Your choice of key map
<C-\><C-O>   - Execute one command, return to Insert mode
:call sqlcomplete#Map( - Allows the SQL completion plugin to perform some
                        housekeeping functions to allow it to be used in
                        conjunction with other completion plugins.
                        Indicate which item you want the SQL completion
                        plugin to complete.
                        In this case we are asking the plugin to display
                        items from the syntax highlight group
                        'sqlKeyword'.
                        You can view a list of highlight group names to
                        choose from by executing the
                            :syntax list
                        command while editing a SQL file.
'sqlKeyword' - Display the items for the sqlKeyword highlight
group
'sqlKeyword\w*' - A second option available with Vim 7.4 which
uses a regular expression to determine which
syntax groups to use
)<CR>        - Execute the :let command
<C-X><C-O>   - Trigger the standard omni completion key stroke.
Passing in 'sqlKeyword' instructs the SQL
```

completion plugin to populate the popup with items from the sqlKeyword highlight group. The plugin will also cache this result until Vim is restarted. The syntax list is retrieved using the syntaxcomplete plugin.

Using the **'syntax'** keyword is a special case. This instructs the syntaxcomplete plugin to retrieve all syntax items. So this will effectively work for any of Vim's SQL syntax files. At the time of writing this includes 10 different syntax files for the different dialects of SQL (see section 3 above, [sql-dialects](#)).

Here are some examples of the entries which are pulled from the syntax files:

- All
 - Contains the contents of all syntax highlight groups
- Statements
 - Select, Insert, Update, Delete, Create, Alter, ...
- Functions
 - Min, Max, Trim, Round, Date, ...
- Keywords
 - Index, Database, Having, Group, With
- Options
 - Isolation_level, On_error, Qualify_owners, Fire_triggers, ...
- Types
 - Integer, Char, Varchar, Date, DateTime, Timestamp, ...

4.2 Dynamic Mode

[sql-completion-dynamic](#)

Dynamic mode populates the popups with data directly from a database. In order for the dynamic feature to be enabled you must have the dbext.vim plugin installed, (http://vim.sourceforge.net/script.php?script_id=356).

Dynamic mode is used by several features of the SQL completion plugin. After installing the dbext plugin see the dbext-tutorial for additional configuration and usage. The dbext plugin allows the SQL completion plugin to display a list of tables, procedures, views and columns.

- Table List
 - All tables for all schema owners
- Procedure List
 - All stored procedures for all schema owners
- View List
 - All stored procedures for all schema owners
- Column List
 - For the selected table, the columns that are part of the table

To enable the popup, while in INSERT mode, use the following key combinations for each group (where <C-C> means hold the CTRL key down while pressing the space bar):

- Table List
 - <C-C>t
 - <C-X><C-O> (the default map assumes tables)
- Stored Procedure List - <C-C>p
- View List - <C-C>v
- Column List - <C-C>c

Drilling In / Out

- When viewing a popup window displaying the list of tables, you can press `<Right>`, this will replace the table currently highlighted with the column list for that table.
- When viewing a popup window displaying the list of columns, you can press `<Left>`, this will replace the column list with the list of tables.
- This allows you to quickly drill down into a table to view its columns and back again.
- `<Right>` and `<Left>` can also be chosen via your `.vimrc`

```
let g:ftplugin_sql_omni_key_right = '<Right>'
let g:ftplugin_sql_omni_key_left  = '<Left>'
```

The SQL completion plugin caches various lists that are displayed in the popup window. This makes the re-displaying of these lists very fast. If new tables or columns are added to the database it may become necessary to clear the plugins cache. The default map for this is:

```
imap <buffer> <C-C>R <C-\><C-O>:call sqlcomplete#Map('ResetCache')<CR><C-X><C-O>
```

4.3 SQL Tutorial

[sql-completion-tutorial](#)

This tutorial is designed to take you through the common features of the SQL completion plugin so that:

- a) You gain familiarity with the plugin
- b) You are introduced to some of the more common features
- c) Show how to customize it to your preferences
- d) Demonstrate "Best of Use" of the plugin (easiest way to configure).

First, create a new buffer:

```
:e tutorial.sql
```

Static features

To take you through the various lists, simply enter insert mode, hit:

```
<C-C>s (show SQL statements)
```

At this point, you can page down through the list until you find "select". If you are familiar with the item you are looking for, for example you know the statement begins with the letter "s". You can type ahead (without the quotes) "se" then press:

```
<C-Space>t
```

Assuming "select" is highlighted in the popup list press `<Enter>` to choose the entry. Now type:

```
* fr<C-C>a (show all syntax items)
```

choose "from" from the popup list.

When writing stored procedures using the "type" list is useful. It contains a list of all the database supported types. This may or may not be true depending on the syntax file you are using. The SQL Anywhere syntax file (sqlanywhere.vim) has support for this:

```
BEGIN
```

```
  DECLARE customer_id <C-C>T <-- Choose a type from the list
```

Dynamic features

To take advantage of the dynamic features you must first install the dbext.vim plugin (http://vim.sourceforge.net/script.php?script_id=356). It also comes with a tutorial. From the SQL completion plugin's perspective, the main feature dbext provides is a connection to a database. dbext connection profiles are the most efficient mechanism to define connection information. Once connections have been setup, the SQL completion plugin uses the features of dbext in the background to populate the popups.

What follows assumes dbext.vim has been correctly configured, a simple test is to run the command, :DBListTable. If a list of tables is shown, you know dbext.vim is working as expected. If not, please consult the dbext.txt documentation.

Assuming you have followed the dbext-tutorial you can press <C-C>t to display a list of tables. There is a delay while dbext is creating the table list. After the list is displayed press <C-W>. This will remove both the popup window and the table name already chosen when the list became active.

4.3.1 Table Completion:

[sql-completion-tables](#)

Press <C-C>t to display a list of tables from within the database you have connected via the dbext plugin.

NOTE: All of the SQL completion popups support typing a prefix before pressing the key map. This will limit the contents of the popup window to just items beginning with those characters.

4.3.2 Column Completion:

[sql-completion-columns](#)

The SQL completion plugin can also display a list of columns for particular tables. The column completion is triggered via <C-C>c.

NOTE: The following example uses <Right> to trigger a column list while the popup window is active.

Example of using column completion:

- Press <C-C>t again to display the list of tables.
- When the list is displayed in the completion window, press <Right>, this will replace the list of tables, with a list of columns for the table highlighted (after the same short delay).
- If you press <Left>, this will again replace the column list with the list of tables. This allows you to drill into tables and column lists very quickly.
- Press <Right> again while the same table is highlighted. You will notice there is no delay since the column list has been cached. If you change the schema of a cached table you can press <C-C>R, which clears the SQL completion cache.
- **NOTE:** <Right> and <Left> have been designed to work while the completion window is active. If the completion popup window is

not active, a normal `<Right>` or `<Left>` will be executed.

Let's look at how we can build a SQL statement dynamically. A select statement requires a list of columns. There are two ways to build a column list using the SQL completion plugin.

One column at a time:

1. After typing SELECT press `<C-C>t` to display a list of tables.
2. Choose a table from the list.
3. Press `<Right>` to display a list of columns.
4. Choose the column from the list and press enter.
5. Enter a "," and press `<C-C>c`. Generating a column list generally requires having the cursor on a table name. The plugin uses this name to determine what table to retrieve the column list. In this step, since we are pressing `<C-C>c` without the cursor on a table name the column list displayed will be for the previous table. Choose a different column and move on.
6. Repeat step 5 as often as necessary.

All columns for a table:

1. After typing SELECT press `<C-C>t` to display a list of tables.
2. Highlight the table you need the column list for.
3. Press `<Enter>` to choose the table from the list.
4. Press `<C-C>l` to request a comma-separated list of all columns for this table.
5. Based on the table name chosen in step 3, the plugin attempts to decide on a reasonable table alias. You are then prompted to either accept or change the alias. Press OK.
6. The table name is replaced with the column list of the table is replaced with the comma separate list of columns with the alias prepended to each of the columns.
7. Step 3 and 4 can be replaced by pressing `<C-C>L`, which has a `<C-Y>` embedded in the map to choose the currently highlighted table in the list.

There is a special provision when writing select statements. Consider the following statement:

```
select *
  from customer c,
       contact cn,
       department as dp,
       employee e,
       site_options so
 where c.
```

In INSERT mode after typing the final "c." which is an alias for the "customer" table, you can press either `<C-C>c` or `<C-X><C-O>`. This will popup a list of columns for the customer table. It does this by looking back to the beginning of the select statement and finding a list of the tables specified in the FROM clause. In this case it notes that in the string "customer c", "c" is an alias for the customer table. The optional "AS" keyword is also supported, "customer AS c".

4.3.3 Procedure Completion:

[sql-completion-procedures](#)

Similar to the table list, `<C-C>p`, will display a list of stored procedures stored within the database.

4.3.4 View Completion:

[sql-completion-views](#)

Similar to the table list, `<C-C>v`, will display a list of views in the database.

4.4 Completion Customization

[sql-completion-customization](#)

The SQL completion plugin can be customized through various options set in your `vimrc` :

`omni_sql_no_default_maps`

- Default: This variable is not defined
- If this variable is defined, no maps are created for OMNI completion. See [sql-completion-maps](#) for further discussion.

`omni_sql_use_tbl_alias`

- Default: a
- This setting is only used when generating a comma-separated column list. By default the map is `<C-C>l`. When generating a column list, an alias can be prepended to the beginning of each column, for example: `e.emp_id, e.emp_name`. This option has three settings:
 - n - do not use an alias
 - d - use the default (calculated) alias
 - a - ask to confirm the alias name

An alias is determined following a few rules:

1. If the table name has an `'_'`, then use it as a separator:
`MY_TABLE_NAME --> MTN`
`my_table_name --> mtn`
`My_table_NAME --> MtN`
2. If the table name does NOT contain an `'_'`, but DOES use mixed case then the case is used as a separator:
`MyTableName --> MTN`
3. If the table name does NOT contain an `'_'`, and does NOT use mixed case then the first letter of the table is used:
`mytablename --> m`
`MYTABLENAME --> M`

`omni_sql_ignorecase`

- Default: Current setting for `'ignorecase'`
- Valid settings are 0 or 1.
- When entering a few letters before initiating completion, the list will be filtered to display only the entries which begin with the list of characters. When this option is set to 0, the list will be filtered using case sensitivity.

`omni_sql_include_owner`

- Default: 0, unless `dbext.vim 3.00` has been installed
- Valid settings are 0 or 1.

- When completing tables, procedure or views and using dbext.vim 3.00 or higher the list of objects will also include the owner name. When completing these objects and omni_sql_include_owner is enabled the owner name will be replaced.

omni_sql_precache_syntax_groups

- Default: ['syntax','sqlKeyword','sqlFunction','sqlOption','sqlType','sqlStatement']
- sqlcomplete can be used in conjunction with other completion plugins. This is outlined at [sql-completion-filetypes](#). When the filetype is changed temporarily to SQL, the sqlcompletion plugin will cache the syntax groups listed in the List specified in this option.

4.5 SQL Maps

sql-completion-maps

The default SQL maps have been described in other sections of this document in greater detail. Here is a list of the maps with a brief description of each.

Static Maps

These are maps which use populate the completion list using Vim's syntax highlighting rules.

- <C-C>a
 - Displays all SQL syntax items.
- <C-C>k
 - Displays all SQL syntax items defined as 'sqlKeyword'.
- <C-C>f
 - Displays all SQL syntax items defined as 'sqlFunction'.
- <C-C>o
 - Displays all SQL syntax items defined as 'sqlOption'.
- <C-C>T
 - Displays all SQL syntax items defined as 'sqlType'.
- <C-C>s
 - Displays all SQL syntax items defined as 'sqlStatement'.

Dynamic Maps

These are maps which use populate the completion list using the dbext.vim plugin.

- <C-C>t
 - Displays a list of tables.
- <C-C>p
 - Displays a list of procedures.
- <C-C>v
 - Displays a list of views.
- <C-C>c
 - Displays a list of columns for a specific table.
- <C-C>l
 - Displays a comma-separated list of columns for a specific table.
- <C-C>L
 - Displays a comma-separated list of columns for a specific table.

This should only be used when the completion window is active.

<Right>

- Displays a list of columns for the table currently highlighted in the completion window. <Right> is not recognized on most Unix systems, so this maps is only created on the Windows platform. If you would like the same feature on Unix, choose a different key and make the same map in your vimrc.

<Left>

- Displays the list of tables. <Left> is not recognized on most Unix systems, so this maps is only created on the Windows platform. If you would like the same feature on Unix, choose a different key and make the same map in your vimrc.

<C-C>R

- This maps removes all cached items and forces the SQL completion to regenerate the list of items.

Customizing Maps

You can create as many additional key maps as you like. Generally, the maps will be specifying different syntax highlight groups.

If you do not wish the default maps created or the key choices do not work on your platform (often a case on *nix) you define the following variable in your vimrc :

```
let g:omni_sql_no_default_maps = 1
```

Do not edit ftplugin/sql.vim directly! If you change this file your changes will be over written on future updates. Vim has a special directory structure which allows you to make customizations without changing the files that are included with the Vim distribution. If you wish to customize the maps create an after/ftplugin/sql.vim (see [after-directory](#)) and place the same maps from the ftplugin/sql.vim in it using your own key strokes. <C-C> was chosen since it will work on both Windows and *nix platforms. On the windows platform you can also use <C-Space> or ALT keys.

4.6 Using with other filetypes

sql-completion-filetypes

Many times SQL can be used with different filetypes. For example Perl, Java, PHP, Javascript can all interact with a database. Often you need both the SQL completion and the completion capabilities for the current language you are editing.

This can be enabled easily with the following steps (assuming a Perl file):

1. `:e test.pl`
2. `:set filetype=sql`
3. `:set ft=perl`

Step 1

Begins by editing a Perl file. Vim automatically sets the filetype to "perl". By default, Vim runs the appropriate filetype file

ftplugin/perl.vim. If you are using the syntax completion plugin by following the directions at [ft-syntax-omni](#) then the `'omnifunc'` option has been set to `"syntax#Complete"`. Pressing `<C-X><C-O>` will display the omni popup containing the syntax items for Perl.

Step 2

Manually setting the filetype to `'sql'` will also fire the appropriate filetype files `ftplugin/sql.vim`. This file will define a number of buffer specific maps for SQL completion, see [sql-completion-maps](#). Now these maps have been created and the SQL completion plugin has been initialized. All SQL syntax items have been cached in preparation. The SQL filetype script detects we are attempting to use two different completion plugins. Since the SQL maps begin with `<C-C>`, the maps will toggle the `'omnifunc'` when in use. So you can use `<C-X><C-O>` to continue using the completion for Perl (using the syntax completion plugin) and `<C-C>` to use the SQL completion features.

Step 3

Setting the filetype back to Perl sets all the usual "perl" related items back as they were.

Digraphs digraph digraphs Digraphs

Digraphs are used to enter characters that normally cannot be entered by an ordinary keyboard. These are mostly printable non-ASCII characters. The digraphs are easier to remember than the decimal number that can be entered with CTRL-V (see i_CTRL-V).

There is a brief introduction on digraphs in the user manual: 24.9
An alternative is using the 'keymap' option.

- 1. Defining digraphs digraphs-define
- 2. Using digraphs digraphs-use
- 3. Default digraphs digraphs-default



1. Defining digraphs digraphs-define

:dig digraphs

:dig[raps][!] Show currently defined digraphs.
With [!] headers are used to make it a bit easier to find a specific character.

E104 E39

:dig[raps] {char1}{char2} {number} ...
Add digraph {char1}{char2} to the list. {number} is the decimal representation of the character. Normally it is the Unicode character, see digraph-encoding .
Example:

```
:digr e: 235 a: 228
```

You can use `:exe` to enter a hex number:

```
:exe 'digr += ' .. 0x2A72
```

Avoid defining a digraph with '_' (underscore) as the first character, it has a special meaning in the future.

NOTE: This command cannot add a digraph that starts with a white space. If you want to add such digraph, you can use digraph_set() instead.

Vim is normally compiled with the +digraphs feature. If the feature is disabled, the ":digraph" command will display an error message.

Example of the output of ":digraphs":

```
TH Þ 222 ss ß 223 a! à 224 a' á 225 a> â 226 a? ã 227 a: ä 228
```

The first two characters in each column are the characters you have to type to enter the digraph.

In the middle of each column is the resulting character. This may be mangled

if you look at it on a system that does not support digraphs or if you print this file.

digraph-encoding

The decimal number normally is the Unicode number of the character. **Note** that the meaning doesn't change when **'encoding'** changes. The character will be converted from Unicode to **'encoding'** when needed. This does require the conversion to be available, it might fail. For the NUL character you will see "10". That's because NUL characters are internally represented with a NL character. When you write the file it will become a NUL character.

Example:

```
digraph oe 339
```

This defines the "oe" digraph for a character that is number 339 in Unicode.

2. Using digraphs

digraphs-use

There are two methods to enter digraphs:

```
CTRL-K {char1} {char2}      or  
{char1} <BS> {char2}
```

i_digraph

The first is always available; the second only when the **'digraph'** option is set.

If a digraph with `{char1}{char2}` does not exist, Vim searches for a digraph `{char2}{char1}`. This helps when you don't remember which character comes first.

Note that when you enter `CTRL-K {char1}`, where `{char1}` is a special key, Vim enters the code for that special key. This is not a digraph.

Once you have entered the digraph, Vim treats the character like a normal character that occupies only one character in the file and on the screen.

Example:

```
'B' <BS> 'B'   will enter the broken '|' character (166)  
'a' <BS> '>'   will enter an 'a' with a circumflex (226)  
CTRL-K '-' '-' will enter a soft hyphen (173)
```

The current digraphs are listed with the `":digraphs"` command. Some of the default ones are listed below [digraph-table](#) .

For `CTRL-K`, there is one general digraph: `CTRL-K <Space> {char}` will enter `{char}` with the highest bit set. You can use this to enter meta-characters.

The `<Esc>` character cannot be part of a digraph. When hitting `<Esc>`, Vim stops digraph entry and ends Insert mode or Command-line mode, just like hitting an `<Esc>` out of digraph context. Use `CTRL-V 155` to enter meta-ESC (CSI).

If you accidentally typed an 'a' that should be an 'e', you will type 'a' `<BS>` 'e'. But that is a digraph, so you will not get what you want. To correct this, you will have to type `<BS>` e again. To avoid this don't set the **'digraph'** option and use `CTRL-K` to enter digraphs.

You may have problems using Vim with characters which have a value above 128. For example: You insert ue (u-umlaut) and the editor echoes \334 in Insert mode. After leaving the Insert mode everything is fine. On some Unix systems this means you have to define the environment-variable LC_CTYPE. If you are using csh, then put the following line in your .cshrc:

```
setenv LC_CTYPE en_US.utf8
```

(or similar for a different language or country). The value must be a valid locale on your system, i.e. on Unix-like systems it must be present in the output of

```
locale -a
```

3. Default digraphs

digraphs-default

Vim comes with a set of default digraphs. Check the output of ":digraphs" to see them.

On most systems Vim uses the same digraphs. They work for the Unicode and ISO-8859-1 character sets. These default digraphs are taken from the RFC1345 mnemonics. To make it easy to remember the mnemonic, the second character has a standard meaning:

char name	char	meaning
Exclamation mark	!	Grave
Apostrophe	'	Acute accent
Greater-Than sign	>	Circumflex accent
Question mark	?	Tilde
Hyphen-Minus	-	Macron
Left parenthesis	(Breve
Full stop	.	Dot above
Colon	:	Diaeresis
Comma	,	Cedilla
Underline	_	Underline
Solidus	/	Stroke
Quotation mark	"	Double acute accent
Semicolon	;	Ogonek
Less-Than sign	<	Caron
Zero	0	Ring above
Two	2	Hook
Nine	9	Horn
Equals	=	Cyrillic (= used as second char)
Asterisk	*	Greek
Percent sign	%	Greek/Cyrillic special
Plus	+	smalls: Arabic, capitals: Hebrew
Three	3	some Latin/Greek/Cyrillic letters
Four	4	Bopomofo
Five	5	Hiragana
Six	6	Katakana

Example: a: is ä and o: is ö

These are the RFC1345 digraphs for the one-byte characters. See the output of ":digraphs" for the others.

EURO

euro euro-digraph

Exception: RFC1345 doesn't specify the euro sign. In Vim the digraph =e was added for this. **Note** the difference between latin1, where the digraph Cu is used for the currency sign, and latin9 (iso-8859-15), where the digraph =e is used for the euro sign, while both of them are the character 164, 0xa4. For compatibility with zsh Eu can also be used for the euro sign.

ROUBLE

The rouble sign was added in 2014 as 0x20bd. Vim supports the digraphs =R and =P for this. **Note** that R= and P= are other characters.

QUADRUPLE PRIME

The quadruple prime using the digraph 4' was added in 2023. Although it is not part of RFC 1345, it supplements the existing digraph implementation as there already exist digraphs for PRIME, DOUBLE PRIME and TRIPLE PRIME using the 1', 2' and 3' digraphs.

digraph-table

char	digraph	hex	dec	official name
^@	NU	0x00	0	NULL (NUL)
^A	SH	0x01	1	START OF HEADING (SOH)
^B	SX	0x02	2	START OF TEXT (STX)
^C	EX	0x03	3	END OF TEXT (ETX)
^D	ET	0x04	4	END OF TRANSMISSION (EOT)
^E	EQ	0x05	5	ENQUIRY (ENQ)
^F	AK	0x06	6	ACKNOWLEDGE (ACK)
^G	BL	0x07	7	BELL (BEL)
^H	BS	0x08	8	BACKSPACE (BS)
^I	HT	0x09	9	CHARACTER TABULATION (HT)
^@	LF	0x0a	10	LINE FEED (LF)
^K	VT	0x0b	11	LINE TABULATION (VT)
^L	FF	0x0c	12	FORM FEED (FF)
^M	CR	0x0d	13	CARRIAGE RETURN (CR)
^N	SO	0x0e	14	SHIFT OUT (SO)
^O	SI	0x0f	15	SHIFT IN (SI)
^P	DL	0x10	16	DATALINK ESCAPE (DLE)
^Q	D1	0x11	17	DEVICE CONTROL ONE (DC1)
^R	D2	0x12	18	DEVICE CONTROL TWO (DC2)
^S	D3	0x13	19	DEVICE CONTROL THREE (DC3)
^T	D4	0x14	20	DEVICE CONTROL FOUR (DC4)
^U	NK	0x15	21	NEGATIVE ACKNOWLEDGE (NAK)
^V	SY	0x16	22	SYNCHRONOUS IDLE (SYN)
^W	EB	0x17	23	END OF TRANSMISSION BLOCK (ETB)
^X	CN	0x18	24	CANCEL (CAN)
^Y	EM	0x19	25	END OF MEDIUM (EM)
^Z	SB	0x1a	26	SUBSTITUTE (SUB)
^[EC	0x1b	27	ESCAPE (ESC)
^\	FS	0x1c	28	FILE SEPARATOR (IS4)
^]	GS	0x1d	29	GROUP SEPARATOR (IS3)
^^	RS	0x1e	30	RECORD SEPARATOR (IS2)

^_	US	0x1f	31	UNIT SEPARATOR (IS1)
	SP	0x20	32	SPACE
#	Nb	0x23	35	NUMBER SIGN
\$	DO	0x24	36	DOLLAR SIGN
@	At	0x40	64	COMMERCIAL AT
[<(0x5b	91	LEFT SQUARE BRACKET
\	//	0x5c	92	REVERSE SOLIDUS
])>	0x5d	93	RIGHT SQUARE BRACKET
^	'>	0x5e	94	CIRCUMFLEX ACCENT
`	'!	0x60	96	GRAVE ACCENT
{	(!	0x7b	123	LEFT CURLY BRACKET
	!!	0x7c	124	VERTICAL LINE
}	!)	0x7d	125	RIGHT CURLY BRACKET
~	'?	0x7e	126	TILDE
^?	DT	0x7f	127	DELETE (DEL)
~@	PA	0x80	128	PADDING CHARACTER (PAD)
~A	HO	0x81	129	HIGH OCTET PRESET (HOP)
~B	BH	0x82	130	BREAK PERMITTED HERE (BPH)
~C	NH	0x83	131	NO BREAK HERE (NBH)
~D	IN	0x84	132	INDEX (IND)
~E	NL	0x85	133	NEXT LINE (NEL)
~F	SA	0x86	134	START OF SELECTED AREA (SSA)
~G	ES	0x87	135	END OF SELECTED AREA (ESA)
~H	HS	0x88	136	CHARACTER TABULATION SET (HTS)
~I	HJ	0x89	137	CHARACTER TABULATION WITH JUSTIFICATION (HTJ)
~J	VS	0x8a	138	LINE TABULATION SET (VTS)
~K	PD	0x8b	139	PARTIAL LINE FORWARD (PLD)
~L	PU	0x8c	140	PARTIAL LINE BACKWARD (PLU)
~M	RI	0x8d	141	REVERSE LINE FEED (RI)
~N	S2	0x8e	142	SINGLE-SHIFT TWO (SS2)
~O	S3	0x8f	143	SINGLE-SHIFT THREE (SS3)
~P	DC	0x90	144	DEVICE CONTROL STRING (DCS)
~Q	P1	0x91	145	PRIVATE USE ONE (PU1)
~R	P2	0x92	146	PRIVATE USE TWO (PU2)
~S	TS	0x93	147	SET TRANSMIT STATE (STS)
~T	CC	0x94	148	CANCEL CHARACTER (CCH)
~U	MW	0x95	149	MESSAGE WAITING (MW)
~V	SG	0x96	150	START OF GUARDED AREA (SPA)
~W	EG	0x97	151	END OF GUARDED AREA (EPA)
~X	SS	0x98	152	START OF STRING (SOS)
~Y	GC	0x99	153	SINGLE GRAPHIC CHARACTER INTRODUCER (SGCI)
~Z	SC	0x9a	154	SINGLE CHARACTER INTRODUCER (SCI)
~[CI	0x9b	155	CONTROL SEQUENCE INTRODUCER (CSI)
~\	ST	0x9c	156	STRING TERMINATOR (ST)
~]	OC	0x9d	157	OPERATING SYSTEM COMMAND (OSC)
~^	PM	0x9e	158	PRIVACY MESSAGE (PM)
~_	AC	0x9f	159	APPLICATION PROGRAM COMMAND (APC)
	NS	0xa0	160	NO-BREAK SPACE
¡	!I	0xa1	161	INVERTED EXCLAMATION MARK
¢	Ct	0xa2	162	CENT SIGN
£	Pd	0xa3	163	POUND SIGN
¤	Cu	0xa4	164	CURRENCY SIGN
¥	Ye	0xa5	165	YEN SIGN
	BB	0xa6	166	BROKEN BAR

§	SE	0xa7	167	SECTION SIGN
¨	':	0xa8	168	DIAERESIS
©	Co	0xa9	169	COPYRIGHT SIGN
ª	-a	0xaa	170	FEMININE ORDINAL INDICATOR
«	<<	0xab	171	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
¬	NO	0xac	172	NOT SIGN
--	0xad		173	SOFT HYPHEN
®	Rg	0xae	174	REGISTERED SIGN
-	'm	0xaf	175	MACRON
°	DG	0xb0	176	DEGREE SIGN
±	+−	0xb1	177	PLUS-MINUS SIGN
²	2S	0xb2	178	SUPERSCRIPIT TWO
³	3S	0xb3	179	SUPERSCRIPIT THREE
´	''	0xb4	180	ACUTE ACCENT
µ	My	0xb5	181	MICRO SIGN
¶	PI	0xb6	182	PILCROW SIGN
·	.M	0xb7	183	MIDDLE DOT
¸	' ,	0xb8	184	CEDILLA
¹	1S	0xb9	185	SUPERSCRIPIT ONE
º	-o	0xba	186	MASCULINE ORDINAL INDICATOR
»	>>	0xbb	187	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
¼	14	0xbc	188	VULGAR FRACTION ONE QUARTER
½	12	0xbd	189	VULGAR FRACTION ONE HALF
¾	34	0xbe	190	VULGAR FRACTION THREE QUARTERS
¿	?I	0xbf	191	INVERTED QUESTION MARK
À	A!	0xc0	192	LATIN CAPITAL LETTER A WITH GRAVE
Á	A'	0xc1	193	LATIN CAPITAL LETTER A WITH ACUTE
Â	A>	0xc2	194	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
Ã	A?	0xc3	195	LATIN CAPITAL LETTER A WITH TILDE
Ä	A:	0xc4	196	LATIN CAPITAL LETTER A WITH DIAERESIS
Å	AA	0xc5	197	LATIN CAPITAL LETTER A WITH RING ABOVE
Æ	AE	0xc6	198	LATIN CAPITAL LETTER AE
Ç	C,	0xc7	199	LATIN CAPITAL LETTER C WITH CEDILLA
È	E!	0xc8	200	LATIN CAPITAL LETTER E WITH GRAVE
É	E'	0xc9	201	LATIN CAPITAL LETTER E WITH ACUTE
Ê	E>	0xca	202	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
Ë	E:	0xcb	203	LATIN CAPITAL LETTER E WITH DIAERESIS
Ì	I!	0xcc	204	LATIN CAPITAL LETTER I WITH GRAVE
Í	I'	0xcd	205	LATIN CAPITAL LETTER I WITH ACUTE
Î	I>	0xce	206	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
Ï	I:	0xcf	207	LATIN CAPITAL LETTER I WITH DIAERESIS
Ð	D-	0xd0	208	LATIN CAPITAL LETTER ETH (Icelandic)
Ñ	N?	0xd1	209	LATIN CAPITAL LETTER N WITH TILDE
Ò	O!	0xd2	210	LATIN CAPITAL LETTER O WITH GRAVE
Ó	O'	0xd3	211	LATIN CAPITAL LETTER O WITH ACUTE
Ô	O>	0xd4	212	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
Õ	O?	0xd5	213	LATIN CAPITAL LETTER O WITH TILDE
Ö	O:	0xd6	214	LATIN CAPITAL LETTER O WITH DIAERESIS
×	*X	0xd7	215	MULTIPLICATION SIGN
Ø	O/	0xd8	216	LATIN CAPITAL LETTER O WITH STROKE
Ù	U!	0xd9	217	LATIN CAPITAL LETTER U WITH GRAVE
Ú	U'	0xda	218	LATIN CAPITAL LETTER U WITH ACUTE
Û	U>	0xdb	219	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
Ü	U:	0xdc	220	LATIN CAPITAL LETTER U WITH DIAERESIS

Ÿ	Y'	0xdd	221	LATIN CAPITAL LETTER Y WITH ACUTE
þ	TH	0xde	222	LATIN CAPITAL LETTER THORN (Icelandic)
ß	ss	0xdf	223	LATIN SMALL LETTER SHARP S (German)
à	a!	0xe0	224	LATIN SMALL LETTER A WITH GRAVE
á	a'	0xe1	225	LATIN SMALL LETTER A WITH ACUTE
â	a>	0xe2	226	LATIN SMALL LETTER A WITH CIRCUMFLEX
ã	a?	0xe3	227	LATIN SMALL LETTER A WITH TILDE
ä	a:	0xe4	228	LATIN SMALL LETTER A WITH DIAERESIS
å	aa	0xe5	229	LATIN SMALL LETTER A WITH RING ABOVE
æ	ae	0xe6	230	LATIN SMALL LETTER AE
ç	c,	0xe7	231	LATIN SMALL LETTER C WITH CEDILLA
è	e!	0xe8	232	LATIN SMALL LETTER E WITH GRAVE
é	e'	0xe9	233	LATIN SMALL LETTER E WITH ACUTE
ê	e>	0xea	234	LATIN SMALL LETTER E WITH CIRCUMFLEX
ë	e:	0xeb	235	LATIN SMALL LETTER E WITH DIAERESIS
ì	i!	0xec	236	LATIN SMALL LETTER I WITH GRAVE
í	i'	0xed	237	LATIN SMALL LETTER I WITH ACUTE
î	i>	0xee	238	LATIN SMALL LETTER I WITH CIRCUMFLEX
ï	i:	0xef	239	LATIN SMALL LETTER I WITH DIAERESIS
ð	d-	0xf0	240	LATIN SMALL LETTER ETH (Icelandic)
ñ	n?	0xf1	241	LATIN SMALL LETTER N WITH TILDE
ò	o!	0xf2	242	LATIN SMALL LETTER O WITH GRAVE
ó	o'	0xf3	243	LATIN SMALL LETTER O WITH ACUTE
ô	o>	0xf4	244	LATIN SMALL LETTER O WITH CIRCUMFLEX
õ	o?	0xf5	245	LATIN SMALL LETTER O WITH TILDE
ö	o:	0xf6	246	LATIN SMALL LETTER O WITH DIAERESIS
÷	-:	0xf7	247	DIVISION SIGN
ø	o/	0xf8	248	LATIN SMALL LETTER O WITH STROKE
ù	u!	0xf9	249	LATIN SMALL LETTER U WITH GRAVE
ú	u'	0xfa	250	LATIN SMALL LETTER U WITH ACUTE
û	u>	0xfb	251	LATIN SMALL LETTER U WITH CIRCUMFLEX
ü	u:	0xfc	252	LATIN SMALL LETTER U WITH DIAERESIS
ý	y'	0xfd	253	LATIN SMALL LETTER Y WITH ACUTE
þ	th	0xfe	254	LATIN SMALL LETTER THORN (Icelandic)
ÿ	y:	0xff	255	LATIN SMALL LETTER Y WITH DIAERESIS

If you are using a **multibyte 'encoding'**, Vim provides this enhanced set of additional digraphs:

char	digraph	hex	dec	official name	digraph-table-mbyte
Ā	A-	0100	0256	LATIN CAPITAL LETTER A WITH MACRON	
ā	a-	0101	0257	LATIN SMALL LETTER A WITH MACRON	
Ă	A(0102	0258	LATIN CAPITAL LETTER A WITH BREVE	
ă	a(0103	0259	LATIN SMALL LETTER A WITH BREVE	
Ą	A;	0104	0260	LATIN CAPITAL LETTER A WITH OGONEK	
ą	a;	0105	0261	LATIN SMALL LETTER A WITH OGONEK	
Ć	C'	0106	0262	LATIN CAPITAL LETTER C WITH ACUTE	
ć	c'	0107	0263	LATIN SMALL LETTER C WITH ACUTE	
Ĉ	C>	0108	0264	LATIN CAPITAL LETTER C WITH CIRCUMFLEX	
ĉ	c>	0109	0265	LATIN SMALL LETTER C WITH CIRCUMFLEX	
Ċ	C.	010A	0266	LATIN CAPITAL LETTER C WITH DOT ABOVE	
ċ	c.	010B	0267	LATIN SMALL LETTER C WITH DOT ABOVE	
Č	C<	010C	0268	LATIN CAPITAL LETTER C WITH CARON	

č	c<	010D	0269	LATIN SMALL LETTER C WITH CARON
Ď	D<	010E	0270	LATIN CAPITAL LETTER D WITH CARON
ď	d<	010F	0271	LATIN SMALL LETTER D WITH CARON
Đ	D/	0110	0272	LATIN CAPITAL LETTER D WITH STROKE
đ	d/	0111	0273	LATIN SMALL LETTER D WITH STROKE
Ē	E-	0112	0274	LATIN CAPITAL LETTER E WITH MACRON
ē	e-	0113	0275	LATIN SMALL LETTER E WITH MACRON
Ĕ	E(0114	0276	LATIN CAPITAL LETTER E WITH BREVE
ĕ	e(0115	0277	LATIN SMALL LETTER E WITH BREVE
Ė	E.	0116	0278	LATIN CAPITAL LETTER E WITH DOT ABOVE
ė	e.	0117	0279	LATIN SMALL LETTER E WITH DOT ABOVE
Ę	E;	0118	0280	LATIN CAPITAL LETTER E WITH OGONEK
ę	e;	0119	0281	LATIN SMALL LETTER E WITH OGONEK
Ę	E<	011A	0282	LATIN CAPITAL LETTER E WITH CARON
ę	e<	011B	0283	LATIN SMALL LETTER E WITH CARON
Ĝ	G>	011C	0284	LATIN CAPITAL LETTER G WITH CIRCUMFLEX
ĝ	g>	011D	0285	LATIN SMALL LETTER G WITH CIRCUMFLEX
Ġ	G(011E	0286	LATIN CAPITAL LETTER G WITH BREVE
ġ	g(011F	0287	LATIN SMALL LETTER G WITH BREVE
Ģ	G.	0120	0288	LATIN CAPITAL LETTER G WITH DOT ABOVE
ģ	g.	0121	0289	LATIN SMALL LETTER G WITH DOT ABOVE
Ģ	G,	0122	0290	LATIN CAPITAL LETTER G WITH CEDILLA
ģ	g,	0123	0291	LATIN SMALL LETTER G WITH CEDILLA
Ĥ	H>	0124	0292	LATIN CAPITAL LETTER H WITH CIRCUMFLEX
ĥ	h>	0125	0293	LATIN SMALL LETTER H WITH CIRCUMFLEX
Ħ	H/	0126	0294	LATIN CAPITAL LETTER H WITH STROKE
ħ	h/	0127	0295	LATIN SMALL LETTER H WITH STROKE
İ	I?	0128	0296	LATIN CAPITAL LETTER I WITH TILDE
ı	i?	0129	0297	LATIN SMALL LETTER I WITH TILDE
Ī	I-	012A	0298	LATIN CAPITAL LETTER I WITH MACRON
ī	i-	012B	0299	LATIN SMALL LETTER I WITH MACRON
Ĭ	I(012C	0300	LATIN CAPITAL LETTER I WITH BREVE
ĭ	i(012D	0301	LATIN SMALL LETTER I WITH BREVE
Į	I;	012E	0302	LATIN CAPITAL LETTER I WITH OGONEK
į	i;	012F	0303	LATIN SMALL LETTER I WITH OGONEK
Ī	I.	0130	0304	LATIN CAPITAL LETTER I WITH DOT ABOVE
ı	i.	0131	0305	LATIN SMALL LETTER DOTLESS I
Ĳ	IJ	0132	0306	LATIN CAPITAL LIGATURE IJ
ĳ	ij	0133	0307	LATIN SMALL LIGATURE IJ
Ĵ	J>	0134	0308	LATIN CAPITAL LETTER J WITH CIRCUMFLEX
ĵ	j>	0135	0309	LATIN SMALL LETTER J WITH CIRCUMFLEX
Ķ	K,	0136	0310	LATIN CAPITAL LETTER K WITH CEDILLA
ķ	k,	0137	0311	LATIN SMALL LETTER K WITH CEDILLA
κ	kk	0138	0312	LATIN SMALL LETTER KRA
Ĺ	L'	0139	0313	LATIN CAPITAL LETTER L WITH ACUTE
ĺ	l'	013A	0314	LATIN SMALL LETTER L WITH ACUTE
Ļ	L,	013B	0315	LATIN CAPITAL LETTER L WITH CEDILLA
ļ	l,	013C	0316	LATIN SMALL LETTER L WITH CEDILLA
Ľ	L<	013D	0317	LATIN CAPITAL LETTER L WITH CARON
ľ	l<	013E	0318	LATIN SMALL LETTER L WITH CARON
Ł	L.	013F	0319	LATIN CAPITAL LETTER L WITH MIDDLE DOT
ł	l.	0140	0320	LATIN SMALL LETTER L WITH MIDDLE DOT
Ł	L/	0141	0321	LATIN CAPITAL LETTER L WITH STROKE
ł	l/	0142	0322	LATIN SMALL LETTER L WITH STROKE

Ñ	N'	0143	0323	LATIN CAPITAL LETTER N WITH ACUTE
ñ	n'	0144	0324	LATIN SMALL LETTER N WITH ACUTE
Ñ	N,	0145	0325	LATIN CAPITAL LETTER N WITH CEDILLA
ñ	n,	0146	0326	LATIN SMALL LETTER N WITH CEDILLA
Ñ	N<	0147	0327	LATIN CAPITAL LETTER N WITH CARON
ň	n<	0148	0328	LATIN SMALL LETTER N WITH CARON
ṅ	'n	0149	0329	LATIN SMALL LETTER N PRECEDED BY APOSTROPHE
Ŋ	NG	014A	0330	LATIN CAPITAL LETTER ENG
ŋ	ng	014B	0331	LATIN SMALL LETTER ENG
Ō	O-	014C	0332	LATIN CAPITAL LETTER O WITH MACRON
ō	o-	014D	0333	LATIN SMALL LETTER O WITH MACRON
Ö	O(014E	0334	LATIN CAPITAL LETTER O WITH BREVE
ö	o(014F	0335	LATIN SMALL LETTER O WITH BREVE
Ő	O"	0150	0336	LATIN CAPITAL LETTER O WITH DOUBLE ACUTE
ő	o"	0151	0337	LATIN SMALL LETTER O WITH DOUBLE ACUTE
Œ	OE	0152	0338	LATIN CAPITAL LIGATURE OE
œ	oe	0153	0339	LATIN SMALL LIGATURE OE
Ŕ	R'	0154	0340	LATIN CAPITAL LETTER R WITH ACUTE
ř	r'	0155	0341	LATIN SMALL LETTER R WITH ACUTE
Ŗ	R,	0156	0342	LATIN CAPITAL LETTER R WITH CEDILLA
ŗ	r,	0157	0343	LATIN SMALL LETTER R WITH CEDILLA
Ř	R<	0158	0344	LATIN CAPITAL LETTER R WITH CARON
ř	r<	0159	0345	LATIN SMALL LETTER R WITH CARON
Ś	S'	015A	0346	LATIN CAPITAL LETTER S WITH ACUTE
ś	s'	015B	0347	LATIN SMALL LETTER S WITH ACUTE
Ŝ	S>	015C	0348	LATIN CAPITAL LETTER S WITH CIRCUMFLEX
ŝ	s>	015D	0349	LATIN SMALL LETTER S WITH CIRCUMFLEX
Ş	S,	015E	0350	LATIN CAPITAL LETTER S WITH CEDILLA
ş	s,	015F	0351	LATIN SMALL LETTER S WITH CEDILLA
Ș	S<	0160	0352	LATIN CAPITAL LETTER S WITH CARON
ș	s<	0161	0353	LATIN SMALL LETTER S WITH CARON
Ț	T,	0162	0354	LATIN CAPITAL LETTER T WITH CEDILLA
ț	t,	0163	0355	LATIN SMALL LETTER T WITH CEDILLA
Ț	T<	0164	0356	LATIN CAPITAL LETTER T WITH CARON
ț	t<	0165	0357	LATIN SMALL LETTER T WITH CARON
Ț	T/	0166	0358	LATIN CAPITAL LETTER T WITH STROKE
ț	t/	0167	0359	LATIN SMALL LETTER T WITH STROKE
Ŭ	U?	0168	0360	LATIN CAPITAL LETTER U WITH TILDE
ũ	u?	0169	0361	LATIN SMALL LETTER U WITH TILDE
Ū	U-	016A	0362	LATIN CAPITAL LETTER U WITH MACRON
ū	u-	016B	0363	LATIN SMALL LETTER U WITH MACRON
Ů	U(016C	0364	LATIN CAPITAL LETTER U WITH BREVE
ů	u(016D	0365	LATIN SMALL LETTER U WITH BREVE
Ů	U0	016E	0366	LATIN CAPITAL LETTER U WITH RING ABOVE
ů	u0	016F	0367	LATIN SMALL LETTER U WITH RING ABOVE
Ű	U"	0170	0368	LATIN CAPITAL LETTER U WITH DOUBLE ACUTE
ű	u"	0171	0369	LATIN SMALL LETTER U WITH DOUBLE ACUTE
Ū	U;	0172	0370	LATIN CAPITAL LETTER U WITH OGONEK
ū	u;	0173	0371	LATIN SMALL LETTER U WITH OGONEK
Ŵ	W>	0174	0372	LATIN CAPITAL LETTER W WITH CIRCUMFLEX
ŵ	w>	0175	0373	LATIN SMALL LETTER W WITH CIRCUMFLEX
Ŷ	Y>	0176	0374	LATIN CAPITAL LETTER Y WITH CIRCUMFLEX
ŷ	y>	0177	0375	LATIN SMALL LETTER Y WITH CIRCUMFLEX
ÿ	Y:	0178	0376	LATIN CAPITAL LETTER Y WITH DIAERESIS

Ž	Z'	0179	0377	LATIN CAPITAL LETTER Z WITH ACUTE
ž	z'	017A	0378	LATIN SMALL LETTER Z WITH ACUTE
Ž	Z.	017B	0379	LATIN CAPITAL LETTER Z WITH DOT ABOVE
ž	z.	017C	0380	LATIN SMALL LETTER Z WITH DOT ABOVE
Ž	Z<	017D	0381	LATIN CAPITAL LETTER Z WITH CARON
ž	z<	017E	0382	LATIN SMALL LETTER Z WITH CARON
Ō	O9	01A0	0416	LATIN CAPITAL LETTER O WITH HORN
o	o9	01A1	0417	LATIN SMALL LETTER O WITH HORN
Ō	OI	01A2	0418	LATIN CAPITAL LETTER OI
o	oi	01A3	0419	LATIN SMALL LETTER OI
Ÿ	yr	01A6	0422	LATIN LETTER YR
Ū	U9	01AF	0431	LATIN CAPITAL LETTER U WITH HORN
ū	u9	01B0	0432	LATIN SMALL LETTER U WITH HORN
Ź	Z/	01B5	0437	LATIN CAPITAL LETTER Z WITH STROKE
z	z/	01B6	0438	LATIN SMALL LETTER Z WITH STROKE
Ė	ED	01B7	0439	LATIN CAPITAL LETTER EZH
Ą	A<	01CD	0461	LATIN CAPITAL LETTER A WITH CARON
ą	a<	01CE	0462	LATIN SMALL LETTER A WITH CARON
Į	I<	01CF	0463	LATIN CAPITAL LETTER I WITH CARON
į	i<	01D0	0464	LATIN SMALL LETTER I WITH CARON
Ō	O<	01D1	0465	LATIN CAPITAL LETTER O WITH CARON
ō	o<	01D2	0466	LATIN SMALL LETTER O WITH CARON
Ū	U<	01D3	0467	LATIN CAPITAL LETTER U WITH CARON
ū	u<	01D4	0468	LATIN SMALL LETTER U WITH CARON
Ä	A1	01DE	0478	LATIN CAPITAL LETTER A WITH DIAERESIS AND MACRON
ä	a1	01DF	0479	LATIN SMALL LETTER A WITH DIAERESIS AND MACRON
Ā	A7	01E0	0480	LATIN CAPITAL LETTER A WITH DOT ABOVE AND MACRON
ā	a7	01E1	0481	LATIN SMALL LETTER A WITH DOT ABOVE AND MACRON
Æ	A3	01E2	0482	LATIN CAPITAL LETTER AE WITH MACRON
æ	a3	01E3	0483	LATIN SMALL LETTER AE WITH MACRON
Ɔ	G/	01E4	0484	LATIN CAPITAL LETTER G WITH STROKE
g	g/	01E5	0485	LATIN SMALL LETTER G WITH STROKE
Ĝ	G<	01E6	0486	LATIN CAPITAL LETTER G WITH CARON
ĝ	g<	01E7	0487	LATIN SMALL LETTER G WITH CARON
Ķ	K<	01E8	0488	LATIN CAPITAL LETTER K WITH CARON
ķ	k<	01E9	0489	LATIN SMALL LETTER K WITH CARON
Ȫ	O;	01EA	0490	LATIN CAPITAL LETTER O WITH OGONEK
ȫ	o;	01EB	0491	LATIN SMALL LETTER O WITH OGONEK
Ȫ	O1	01EC	0492	LATIN CAPITAL LETTER O WITH OGONEK AND MACRON
ȫ	o1	01ED	0493	LATIN SMALL LETTER O WITH OGONEK AND MACRON
Ė	EZ	01EE	0494	LATIN CAPITAL LETTER EZH WITH CARON
ė	ez	01EF	0495	LATIN SMALL LETTER EZH WITH CARON
Ĵ	j<	01F0	0496	LATIN SMALL LETTER J WITH CARON
Ģ	G'	01F4	0500	LATIN CAPITAL LETTER G WITH ACUTE
ģ	g'	01F5	0501	LATIN SMALL LETTER G WITH ACUTE
ˆ	;S	02BF	0703	MODIFIER LETTER LEFT HALF RING
˘	'<	02C7	0711	CARON
˘	'(02D8	0728	BREVE
˙	'.	02D9	0729	DOT ABOVE
˚	'0	02DA	0730	RING ABOVE
˛	';	02DB	0731	OGONEK
˝	'"	02DD	0733	DOUBLE ACUTE ACCENT
Α	A%	0386	0902	GREEK CAPITAL LETTER ALPHA WITH TONOS
Ε	E%	0388	0904	GREEK CAPITAL LETTER EPSILON WITH TONOS

☒	Y%	0389	0905	GREEK CAPITAL LETTER ETA WITH TONOS
☒	I%	038A	0906	GREEK CAPITAL LETTER IOTA WITH TONOS
☒	O%	038C	0908	GREEK CAPITAL LETTER OMICRON WITH TONOS
☒	U%	038E	0910	GREEK CAPITAL LETTER UPSILON WITH TONOS
☒	W%	038F	0911	GREEK CAPITAL LETTER OMEGA WITH TONOS
☒	i3	0390	0912	GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS
☒	A*	0391	0913	GREEK CAPITAL LETTER ALPHA
☒	B*	0392	0914	GREEK CAPITAL LETTER BETA
☒	G*	0393	0915	GREEK CAPITAL LETTER GAMMA
☒	D*	0394	0916	GREEK CAPITAL LETTER DELTA
☒	E*	0395	0917	GREEK CAPITAL LETTER EPSILON
☒	Z*	0396	0918	GREEK CAPITAL LETTER ZETA
☒	Y*	0397	0919	GREEK CAPITAL LETTER ETA
☒	H*	0398	0920	GREEK CAPITAL LETTER THETA
☒	I*	0399	0921	GREEK CAPITAL LETTER IOTA
☒	K*	039A	0922	GREEK CAPITAL LETTER KAPPA
☒	L*	039B	0923	GREEK CAPITAL LETTER LAMDA (aka LAMBDA)
☒	M*	039C	0924	GREEK CAPITAL LETTER MU
☒	N*	039D	0925	GREEK CAPITAL LETTER NU
☒	C*	039E	0926	GREEK CAPITAL LETTER XI
☒	O*	039F	0927	GREEK CAPITAL LETTER OMICRON
☒	P*	03A0	0928	GREEK CAPITAL LETTER PI
☒	R*	03A1	0929	GREEK CAPITAL LETTER RHO
☒	S*	03A3	0931	GREEK CAPITAL LETTER SIGMA
☒	T*	03A4	0932	GREEK CAPITAL LETTER TAU
☒	U*	03A5	0933	GREEK CAPITAL LETTER UPSILON
☒	F*	03A6	0934	GREEK CAPITAL LETTER PHI
☒	X*	03A7	0935	GREEK CAPITAL LETTER CHI
☒	Q*	03A8	0936	GREEK CAPITAL LETTER PSI
☒	W*	03A9	0937	GREEK CAPITAL LETTER OMEGA
☒	J*	03AA	0938	GREEK CAPITAL LETTER IOTA WITH DIALYTIKA
☒	V*	03AB	0939	GREEK CAPITAL LETTER UPSILON WITH DIALYTIKA
☒	a%	03AC	0940	GREEK SMALL LETTER ALPHA WITH TONOS
☒	e%	03AD	0941	GREEK SMALL LETTER EPSILON WITH TONOS
☒	y%	03AE	0942	GREEK SMALL LETTER ETA WITH TONOS
☒	i%	03AF	0943	GREEK SMALL LETTER IOTA WITH TONOS
☒	u3	03B0	0944	GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS
☒	a*	03B1	0945	GREEK SMALL LETTER ALPHA
☒	b*	03B2	0946	GREEK SMALL LETTER BETA
☒	g*	03B3	0947	GREEK SMALL LETTER GAMMA
☒	d*	03B4	0948	GREEK SMALL LETTER DELTA
☒	e*	03B5	0949	GREEK SMALL LETTER EPSILON
☒	z*	03B6	0950	GREEK SMALL LETTER ZETA
☒	y*	03B7	0951	GREEK SMALL LETTER ETA
☒	h*	03B8	0952	GREEK SMALL LETTER THETA
☒	i*	03B9	0953	GREEK SMALL LETTER IOTA
☒	k*	03BA	0954	GREEK SMALL LETTER KAPPA
☒	l*	03BB	0955	GREEK SMALL LETTER LAMDA (aka LAMBDA)
☒	m*	03BC	0956	GREEK SMALL LETTER MU
☒	n*	03BD	0957	GREEK SMALL LETTER NU
☒	c*	03BE	0958	GREEK SMALL LETTER XI
☒	o*	03BF	0959	GREEK SMALL LETTER OMICRON
π	p*	03C0	0960	GREEK SMALL LETTER PI
☒	r*	03C1	0961	GREEK SMALL LETTER RHO

☒	*s	03C2	0962	GREEK SMALL LETTER FINAL SIGMA
☒	s*	03C3	0963	GREEK SMALL LETTER SIGMA
☒	t*	03C4	0964	GREEK SMALL LETTER TAU
☒	u*	03C5	0965	GREEK SMALL LETTER UPSILON
☒	f*	03C6	0966	GREEK SMALL LETTER PHI
☒	x*	03C7	0967	GREEK SMALL LETTER CHI
☒	q*	03C8	0968	GREEK SMALL LETTER PSI
☒	w*	03C9	0969	GREEK SMALL LETTER OMEGA
☒	j*	03CA	0970	GREEK SMALL LETTER IOTA WITH DIALYTIKA
☒	v*	03CB	0971	GREEK SMALL LETTER UPSILON WITH DIALYTIKA
☒	o%	03CC	0972	GREEK SMALL LETTER OMICRON WITH TONOS
☒	u%	03CD	0973	GREEK SMALL LETTER UPSILON WITH TONOS
☒	w%	03CE	0974	GREEK SMALL LETTER OMEGA WITH TONOS
☒	'G	03D8	0984	GREEK LETTER ARCHAIC KOPPA
☒	,G	03D9	0985	GREEK SMALL LETTER ARCHAIC KOPPA
☒	T3	03DA	0986	GREEK LETTER STIGMA
☒	t3	03DB	0987	GREEK SMALL LETTER STIGMA
☒	M3	03DC	0988	GREEK LETTER DIGAMMA
☒	m3	03DD	0989	GREEK SMALL LETTER DIGAMMA
☒	K3	03DE	0990	GREEK LETTER KOPPA
☒	k3	03DF	0991	GREEK SMALL LETTER KOPPA
☒	P3	03E0	0992	GREEK LETTER SAMPI
☒	p3	03E1	0993	GREEK SMALL LETTER SAMPI
☒	'%	03F4	1012	GREEK CAPITAL THETA SYMBOL
☒	j3	03F5	1013	GREEK LUNATE EPSILON SYMBOL
☒	IO	0401	1025	CYRILLIC CAPITAL LETTER IO
☒	D%	0402	1026	CYRILLIC CAPITAL LETTER DJE
☒	G%	0403	1027	CYRILLIC CAPITAL LETTER GJE
☒	IE	0404	1028	CYRILLIC CAPITAL LETTER UKRAINIAN IE
☒	DS	0405	1029	CYRILLIC CAPITAL LETTER DZE
☒	II	0406	1030	CYRILLIC CAPITAL LETTER BYELORUSSIAN-UKRAINIAN I
☒	YI	0407	1031	CYRILLIC CAPITAL LETTER YI
☒	J%	0408	1032	CYRILLIC CAPITAL LETTER JE
☒	LJ	0409	1033	CYRILLIC CAPITAL LETTER LJE
☒	NJ	040A	1034	CYRILLIC CAPITAL LETTER NJE
☒	Ts	040B	1035	CYRILLIC CAPITAL LETTER TSHE
☒	KJ	040C	1036	CYRILLIC CAPITAL LETTER KJE
☒	V%	040E	1038	CYRILLIC CAPITAL LETTER SHORT U
☒	DZ	040F	1039	CYRILLIC CAPITAL LETTER DZHE
☒	A=	0410	1040	CYRILLIC CAPITAL LETTER A
☒	B=	0411	1041	CYRILLIC CAPITAL LETTER BE
☒	V=	0412	1042	CYRILLIC CAPITAL LETTER VE
☒	G=	0413	1043	CYRILLIC CAPITAL LETTER GHE
☒	D=	0414	1044	CYRILLIC CAPITAL LETTER DE
☒	E=	0415	1045	CYRILLIC CAPITAL LETTER IE
☒	Z%	0416	1046	CYRILLIC CAPITAL LETTER ZHE
☒	Z=	0417	1047	CYRILLIC CAPITAL LETTER ZE
☒	I=	0418	1048	CYRILLIC CAPITAL LETTER I
☒	J=	0419	1049	CYRILLIC CAPITAL LETTER SHORT I
☒	K=	041A	1050	CYRILLIC CAPITAL LETTER KA
☒	L=	041B	1051	CYRILLIC CAPITAL LETTER EL
☒	M=	041C	1052	CYRILLIC CAPITAL LETTER EM
☒	N=	041D	1053	CYRILLIC CAPITAL LETTER EN
☒	O=	041E	1054	CYRILLIC CAPITAL LETTER O

☒	P=	041F	1055	CYRILLIC CAPITAL LETTER PE
☒	R=	0420	1056	CYRILLIC CAPITAL LETTER ER
☒	S=	0421	1057	CYRILLIC CAPITAL LETTER ES
☒	T=	0422	1058	CYRILLIC CAPITAL LETTER TE
☒	U=	0423	1059	CYRILLIC CAPITAL LETTER U
☒	F=	0424	1060	CYRILLIC CAPITAL LETTER EF
☒	H=	0425	1061	CYRILLIC CAPITAL LETTER HA
☒	C=	0426	1062	CYRILLIC CAPITAL LETTER TSE
☒	C%	0427	1063	CYRILLIC CAPITAL LETTER CHE
☒	S%	0428	1064	CYRILLIC CAPITAL LETTER SHA
☒	Sc	0429	1065	CYRILLIC CAPITAL LETTER SHCHA
☒	="	042A	1066	CYRILLIC CAPITAL LETTER HARD SIGN
☒	Y=	042B	1067	CYRILLIC CAPITAL LETTER YERU
☒	%"	042C	1068	CYRILLIC CAPITAL LETTER SOFT SIGN
☒	JE	042D	1069	CYRILLIC CAPITAL LETTER E
☒	JU	042E	1070	CYRILLIC CAPITAL LETTER YU
☒	JA	042F	1071	CYRILLIC CAPITAL LETTER YA
☒	a=	0430	1072	CYRILLIC SMALL LETTER A
☒	b=	0431	1073	CYRILLIC SMALL LETTER BE
☒	v=	0432	1074	CYRILLIC SMALL LETTER VE
☒	g=	0433	1075	CYRILLIC SMALL LETTER GHE
☒	d=	0434	1076	CYRILLIC SMALL LETTER DE
☒	e=	0435	1077	CYRILLIC SMALL LETTER IE
☒	z%	0436	1078	CYRILLIC SMALL LETTER ZHE
☒	z=	0437	1079	CYRILLIC SMALL LETTER ZE
☒	i=	0438	1080	CYRILLIC SMALL LETTER I
☒	j=	0439	1081	CYRILLIC SMALL LETTER SHORT I
☒	k=	043A	1082	CYRILLIC SMALL LETTER KA
☒	l=	043B	1083	CYRILLIC SMALL LETTER EL
☒	m=	043C	1084	CYRILLIC SMALL LETTER EM
☒	n=	043D	1085	CYRILLIC SMALL LETTER EN
☒	o=	043E	1086	CYRILLIC SMALL LETTER O
☒	p=	043F	1087	CYRILLIC SMALL LETTER PE
☒	r=	0440	1088	CYRILLIC SMALL LETTER ER
☒	s=	0441	1089	CYRILLIC SMALL LETTER ES
☒	t=	0442	1090	CYRILLIC SMALL LETTER TE
☒	u=	0443	1091	CYRILLIC SMALL LETTER U
☒	f=	0444	1092	CYRILLIC SMALL LETTER EF
☒	h=	0445	1093	CYRILLIC SMALL LETTER HA
☒	c=	0446	1094	CYRILLIC SMALL LETTER TSE
☒	c%	0447	1095	CYRILLIC SMALL LETTER CHE
☒	s%	0448	1096	CYRILLIC SMALL LETTER SHA
☒	sc	0449	1097	CYRILLIC SMALL LETTER SHCHA
☒	='	044A	1098	CYRILLIC SMALL LETTER HARD SIGN
☒	y=	044B	1099	CYRILLIC SMALL LETTER YERU
☒	%'	044C	1100	CYRILLIC SMALL LETTER SOFT SIGN
☒	je	044D	1101	CYRILLIC SMALL LETTER E
☒	ju	044E	1102	CYRILLIC SMALL LETTER YU
☒	ja	044F	1103	CYRILLIC SMALL LETTER YA
☒	io	0451	1105	CYRILLIC SMALL LETTER IO
☒	d%	0452	1106	CYRILLIC SMALL LETTER DJE
☒	g%	0453	1107	CYRILLIC SMALL LETTER GJE
☒	ie	0454	1108	CYRILLIC SMALL LETTER UKRAINIAN IE
☒	ds	0455	1109	CYRILLIC SMALL LETTER DZE

☒	ii	0456	1110	CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I
☒	yi	0457	1111	CYRILLIC SMALL LETTER YI
☒	j%	0458	1112	CYRILLIC SMALL LETTER JE
☒	lj	0459	1113	CYRILLIC SMALL LETTER LJE
☒	nj	045A	1114	CYRILLIC SMALL LETTER NJE
☒	ts	045B	1115	CYRILLIC SMALL LETTER TSHE
☒	kj	045C	1116	CYRILLIC SMALL LETTER KJE
☒	v%	045E	1118	CYRILLIC SMALL LETTER SHORT U
☒	dz	045F	1119	CYRILLIC SMALL LETTER DZHE
☒	Y3	0462	1122	CYRILLIC CAPITAL LETTER YAT
☒	y3	0463	1123	CYRILLIC SMALL LETTER YAT
☒	O3	046A	1130	CYRILLIC CAPITAL LETTER BIG YUS
☒	o3	046B	1131	CYRILLIC SMALL LETTER BIG YUS
☒	F3	0472	1138	CYRILLIC CAPITAL LETTER FITA
☒	f3	0473	1139	CYRILLIC SMALL LETTER FITA
☒	V3	0474	1140	CYRILLIC CAPITAL LETTER IZHITSA
☒	v3	0475	1141	CYRILLIC SMALL LETTER IZHITSA
☒	C3	0480	1152	CYRILLIC CAPITAL LETTER KOPPA
☒	c3	0481	1153	CYRILLIC SMALL LETTER KOPPA
☒	G3	0490	1168	CYRILLIC CAPITAL LETTER GHE WITH UPTURN
☒	g3	0491	1169	CYRILLIC SMALL LETTER GHE WITH UPTURN
☒	A+	05D0	1488	HEBREW LETTER ALEF
☒	B+	05D1	1489	HEBREW LETTER BET
☒	G+	05D2	1490	HEBREW LETTER GIMEL
☒	D+	05D3	1491	HEBREW LETTER DALET
☒	H+	05D4	1492	HEBREW LETTER HE
☒	W+	05D5	1493	HEBREW LETTER VAV
☒	Z+	05D6	1494	HEBREW LETTER ZAYIN
☒	X+	05D7	1495	HEBREW LETTER HET
☒	Tj	05D8	1496	HEBREW LETTER TET
☒	J+	05D9	1497	HEBREW LETTER YOD
☒	K%	05DA	1498	HEBREW LETTER FINAL KAF
☒	K+	05DB	1499	HEBREW LETTER KAF
☒	L+	05DC	1500	HEBREW LETTER LAMED
☒	M%	05DD	1501	HEBREW LETTER FINAL MEM
☒	M+	05DE	1502	HEBREW LETTER MEM
☒	N%	05DF	1503	HEBREW LETTER FINAL NUN
☒	N+	05E0	1504	HEBREW LETTER NUN
☒	S+	05E1	1505	HEBREW LETTER SAMEKH
☒	E+	05E2	1506	HEBREW LETTER AYIN
☒	P%	05E3	1507	HEBREW LETTER FINAL PE
☒	P+	05E4	1508	HEBREW LETTER PE
☒	Zj	05E5	1509	HEBREW LETTER FINAL TSADI
☒	ZJ	05E6	1510	HEBREW LETTER TSADI
☒	Q+	05E7	1511	HEBREW LETTER QOF
☒	R+	05E8	1512	HEBREW LETTER RESH
☒	Sh	05E9	1513	HEBREW LETTER SHIN
☒	T+	05EA	1514	HEBREW LETTER TAV
☒	,+	060C	1548	ARABIC COMMA
☒	;+	061B	1563	ARABIC SEMICOLON
☒	?+	061F	1567	ARABIC QUESTION MARK
☒	H'	0621	1569	ARABIC LETTER HAMZA
☒	aM	0622	1570	ARABIC LETTER ALEF WITH MADDA ABOVE
☒	aH	0623	1571	ARABIC LETTER ALEF WITH HAMZA ABOVE

☒	wH	0624	1572	ARABIC LETTER WAW WITH HAMZA ABOVE
☒	ah	0625	1573	ARABIC LETTER ALEF WITH HAMZA BELOW
☒	yH	0626	1574	ARABIC LETTER YEH WITH HAMZA ABOVE
☒	a+	0627	1575	ARABIC LETTER ALEF
☒	b+	0628	1576	ARABIC LETTER BEH
☒	tm	0629	1577	ARABIC LETTER TEH MARBUTA
☒	t+	062A	1578	ARABIC LETTER TEH
☒	tk	062B	1579	ARABIC LETTER THEH
☒	g+	062C	1580	ARABIC LETTER JEEM
☒	hk	062D	1581	ARABIC LETTER HAH
☒	x+	062E	1582	ARABIC LETTER KHAH
☒	d+	062F	1583	ARABIC LETTER DAL
☒	dk	0630	1584	ARABIC LETTER THAL
☒	r+	0631	1585	ARABIC LETTER REH
☒	z+	0632	1586	ARABIC LETTER ZAIN
☒	s+	0633	1587	ARABIC LETTER SEEN
☒	sn	0634	1588	ARABIC LETTER SHEEN
☒	c+	0635	1589	ARABIC LETTER SAD
☒	dd	0636	1590	ARABIC LETTER DAD
☒	tj	0637	1591	ARABIC LETTER TAH
☒	zH	0638	1592	ARABIC LETTER ZAH
☒	e+	0639	1593	ARABIC LETTER AIN
☒	i+	063A	1594	ARABIC LETTER GHAIN
☒	++	0640	1600	ARABIC TATWEEL
☒	f+	0641	1601	ARABIC LETTER FEH
☒	q+	0642	1602	ARABIC LETTER QAF
☒	k+	0643	1603	ARABIC LETTER KAF
☒	l+	0644	1604	ARABIC LETTER LAM
☒	m+	0645	1605	ARABIC LETTER MEEM
☒	n+	0646	1606	ARABIC LETTER NOON
☒	h+	0647	1607	ARABIC LETTER HEH
☒	w+	0648	1608	ARABIC LETTER WAW
☒	j+	0649	1609	ARABIC LETTER ALEF MAKSURA
☒	y+	064A	1610	ARABIC LETTER YEH
☒	:+	064B	1611	ARABIC FATHATAN
☒	"+	064C	1612	ARABIC DAMMATAN
☒	=+	064D	1613	ARABIC KASRATAN
☒	/+	064E	1614	ARABIC FATHA
☒	'+	064F	1615	ARABIC DAMMA
☒	1+	0650	1616	ARABIC KASRA
☒	3+	0651	1617	ARABIC SHADDA
☒	0+	0652	1618	ARABIC SUKUN
☒	aS	0670	1648	ARABIC LETTER SUPERScript ALEF
☒	p+	067E	1662	ARABIC LETTER PEH
☒	v+	06A4	1700	ARABIC LETTER VEH
☒	gf	06AF	1711	ARABIC LETTER GAF
☒	0a	06F0	1776	EXTENDED ARABIC-INDIC DIGIT ZERO
☒	1a	06F1	1777	EXTENDED ARABIC-INDIC DIGIT ONE
☒	2a	06F2	1778	EXTENDED ARABIC-INDIC DIGIT TWO
☒	3a	06F3	1779	EXTENDED ARABIC-INDIC DIGIT THREE
☒	4a	06F4	1780	EXTENDED ARABIC-INDIC DIGIT FOUR
☒	5a	06F5	1781	EXTENDED ARABIC-INDIC DIGIT FIVE
☒	6a	06F6	1782	EXTENDED ARABIC-INDIC DIGIT SIX
☒	7a	06F7	1783	EXTENDED ARABIC-INDIC DIGIT SEVEN

Ø	8a	06F8	1784	EXTENDED ARABIC-INDIC DIGIT EIGHT
Ø	9a	06F9	1785	EXTENDED ARABIC-INDIC DIGIT NINE
Ḃ	B.	1E02	7682	LATIN CAPITAL LETTER B WITH DOT ABOVE
ḃ	b.	1E03	7683	LATIN SMALL LETTER B WITH DOT ABOVE
Ḅ	B_	1E06	7686	LATIN CAPITAL LETTER B WITH LINE BELOW
ḅ	b_	1E07	7687	LATIN SMALL LETTER B WITH LINE BELOW
Ḍ	D.	1E0A	7690	LATIN CAPITAL LETTER D WITH DOT ABOVE
ḍ	d.	1E0B	7691	LATIN SMALL LETTER D WITH DOT ABOVE
Ḏ	D_	1E0E	7694	LATIN CAPITAL LETTER D WITH LINE BELOW
ḏ	d_	1E0F	7695	LATIN SMALL LETTER D WITH LINE BELOW
Ḑ	D,	1E10	7696	LATIN CAPITAL LETTER D WITH CEDILLA
ḑ	d,	1E11	7697	LATIN SMALL LETTER D WITH CEDILLA
Ḓ	F.	1E1E	7710	LATIN CAPITAL LETTER F WITH DOT ABOVE
ḓ	f.	1E1F	7711	LATIN SMALL LETTER F WITH DOT ABOVE
Ḕ	G-	1E20	7712	LATIN CAPITAL LETTER G WITH MACRON
ḕ	g-	1E21	7713	LATIN SMALL LETTER G WITH MACRON
Ḗ	H.	1E22	7714	LATIN CAPITAL LETTER H WITH DOT ABOVE
ḗ	h.	1E23	7715	LATIN SMALL LETTER H WITH DOT ABOVE
Ḙ	H:	1E26	7718	LATIN CAPITAL LETTER H WITH DIAERESIS
ḙ	h:	1E27	7719	LATIN SMALL LETTER H WITH DIAERESIS
Ḛ	H,	1E28	7720	LATIN CAPITAL LETTER H WITH CEDILLA
ḛ	h,	1E29	7721	LATIN SMALL LETTER H WITH CEDILLA
Ḝ	K'	1E30	7728	LATIN CAPITAL LETTER K WITH ACUTE
ḝ	k'	1E31	7729	LATIN SMALL LETTER K WITH ACUTE
Ḟ	K_	1E34	7732	LATIN CAPITAL LETTER K WITH LINE BELOW
ḟ	k_	1E35	7733	LATIN SMALL LETTER K WITH LINE BELOW
Ḡ	L_	1E3A	7738	LATIN CAPITAL LETTER L WITH LINE BELOW
ḡ	l_	1E3B	7739	LATIN SMALL LETTER L WITH LINE BELOW
Ḣ	M'	1E3E	7742	LATIN CAPITAL LETTER M WITH ACUTE
ḣ	m'	1E3F	7743	LATIN SMALL LETTER M WITH ACUTE
Ḥ	M.	1E40	7744	LATIN CAPITAL LETTER M WITH DOT ABOVE
ḥ	m.	1E41	7745	LATIN SMALL LETTER M WITH DOT ABOVE
Ḧ	N.	1E44	7748	LATIN CAPITAL LETTER N WITH DOT ABOVE
ḧ	n.	1E45	7749	LATIN SMALL LETTER N WITH DOT ABOVE
Ḩ	N_	1E48	7752	LATIN CAPITAL LETTER N WITH LINE BELOW
ḩ	n_	1E49	7753	LATIN SMALL LETTER N WITH LINE BELOW
Ḑ	P'	1E54	7764	LATIN CAPITAL LETTER P WITH ACUTE
ḑ	p'	1E55	7765	LATIN SMALL LETTER P WITH ACUTE
Ḓ	P.	1E56	7766	LATIN CAPITAL LETTER P WITH DOT ABOVE
ḓ	p.	1E57	7767	LATIN SMALL LETTER P WITH DOT ABOVE
Ḕ	R.	1E58	7768	LATIN CAPITAL LETTER R WITH DOT ABOVE
ḕ	r.	1E59	7769	LATIN SMALL LETTER R WITH DOT ABOVE
Ḗ	R_	1E5E	7774	LATIN CAPITAL LETTER R WITH LINE BELOW
ḗ	r_	1E5F	7775	LATIN SMALL LETTER R WITH LINE BELOW
Ḙ	S.	1E60	7776	LATIN CAPITAL LETTER S WITH DOT ABOVE
ḙ	s.	1E61	7777	LATIN SMALL LETTER S WITH DOT ABOVE
Ḛ	T.	1E6A	7786	LATIN CAPITAL LETTER T WITH DOT ABOVE
ḛ	t.	1E6B	7787	LATIN SMALL LETTER T WITH DOT ABOVE
Ḝ	T_	1E6E	7790	LATIN CAPITAL LETTER T WITH LINE BELOW
ḝ	t_	1E6F	7791	LATIN SMALL LETTER T WITH LINE BELOW
Ḟ	V?	1E7C	7804	LATIN CAPITAL LETTER V WITH TILDE
ḟ	v?	1E7D	7805	LATIN SMALL LETTER V WITH TILDE
Ḡ	W!	1E80	7808	LATIN CAPITAL LETTER W WITH GRAVE
ḡ	w!	1E81	7809	LATIN SMALL LETTER W WITH GRAVE

Ŵ	W'	1E82	7810	LATIN CAPITAL LETTER W WITH ACUTE
ŵ	w'	1E83	7811	LATIN SMALL LETTER W WITH ACUTE
Ẁ	W:	1E84	7812	LATIN CAPITAL LETTER W WITH DIAERESIS
ẁ	w:	1E85	7813	LATIN SMALL LETTER W WITH DIAERESIS
Ẃ	W.	1E86	7814	LATIN CAPITAL LETTER W WITH DOT ABOVE
ẃ	w.	1E87	7815	LATIN SMALL LETTER W WITH DOT ABOVE
Ẅ	X.	1E8A	7818	LATIN CAPITAL LETTER X WITH DOT ABOVE
ẅ	x.	1E8B	7819	LATIN SMALL LETTER X WITH DOT ABOVE
Ẇ	X:	1E8C	7820	LATIN CAPITAL LETTER X WITH DIAERESIS
ẇ	x:	1E8D	7821	LATIN SMALL LETTER X WITH DIAERESIS
Ỳ	Y.	1E8E	7822	LATIN CAPITAL LETTER Y WITH DOT ABOVE
ỳ	y.	1E8F	7823	LATIN SMALL LETTER Y WITH DOT ABOVE
Ẑ	Z>	1E90	7824	LATIN CAPITAL LETTER Z WITH CIRCUMFLEX
ẑ	z>	1E91	7825	LATIN SMALL LETTER Z WITH CIRCUMFLEX
Ẓ	Z_	1E94	7828	LATIN CAPITAL LETTER Z WITH LINE BELOW
ẓ	z_	1E95	7829	LATIN SMALL LETTER Z WITH LINE BELOW
Ẕ	h_	1E96	7830	LATIN SMALL LETTER H WITH LINE BELOW
ẕ	t:	1E97	7831	LATIN SMALL LETTER T WITH DIAERESIS
ẖ	w0	1E98	7832	LATIN SMALL LETTER W WITH RING ABOVE
ẗ	y0	1E99	7833	LATIN SMALL LETTER Y WITH RING ABOVE
Ȧ	A2	1EA2	7842	LATIN CAPITAL LETTER A WITH HOOK ABOVE
ȧ	a2	1EA3	7843	LATIN SMALL LETTER A WITH HOOK ABOVE
Ĕ	E2	1EBA	7866	LATIN CAPITAL LETTER E WITH HOOK ABOVE
ĕ	e2	1EBB	7867	LATIN SMALL LETTER E WITH HOOK ABOVE
Ě	E?	1EBC	7868	LATIN CAPITAL LETTER E WITH TILDE
ě	e?	1EBD	7869	LATIN SMALL LETTER E WITH TILDE
İ	I2	1EC8	7880	LATIN CAPITAL LETTER I WITH HOOK ABOVE
ı	i2	1EC9	7881	LATIN SMALL LETTER I WITH HOOK ABOVE
Ȫ	O2	1ECE	7886	LATIN CAPITAL LETTER O WITH HOOK ABOVE
ȫ	o2	1ECF	7887	LATIN SMALL LETTER O WITH HOOK ABOVE
Ȭ	U2	1EE6	7910	LATIN CAPITAL LETTER U WITH HOOK ABOVE
ȭ	u2	1EE7	7911	LATIN SMALL LETTER U WITH HOOK ABOVE
Ỳ	Y!	1EF2	7922	LATIN CAPITAL LETTER Y WITH GRAVE
ỳ	y!	1EF3	7923	LATIN SMALL LETTER Y WITH GRAVE
Ỳ	Y2	1EF6	7926	LATIN CAPITAL LETTER Y WITH HOOK ABOVE
ỳ	y2	1EF7	7927	LATIN SMALL LETTER Y WITH HOOK ABOVE
Ỳ	Y?	1EF8	7928	LATIN CAPITAL LETTER Y WITH TILDE
ỳ	y?	1EF9	7929	LATIN SMALL LETTER Y WITH TILDE
Ɑ	;'	1F00	7936	GREEK SMALL LETTER ALPHA WITH PSILI
Ɱ	,'	1F01	7937	GREEK SMALL LETTER ALPHA WITH DASIA
Ɐ	;!'	1F02	7938	GREEK SMALL LETTER ALPHA WITH PSILI AND VARIA
Ɒ	,!'	1F03	7939	GREEK SMALL LETTER ALPHA WITH DASIA AND VARIA
ⱱ	?;	1F04	7940	GREEK SMALL LETTER ALPHA WITH PSILI AND OXIA
Ⱳ	?,	1F05	7941	GREEK SMALL LETTER ALPHA WITH DASIA AND OXIA
ⱳ	!:	1F06	7942	GREEK SMALL LETTER ALPHA WITH PSILI AND PERISPOMENI
ⱴ	?:	1F07	7943	GREEK SMALL LETTER ALPHA WITH DASIA AND PERISPOMENI
	1N	2002	8194	EN SPACE
	1M	2003	8195	EM SPACE
	3M	2004	8196	THREE-PER-EM SPACE
	4M	2005	8197	FOUR-PER-EM SPACE
	6M	2006	8198	SIX-PER-EM SPACE
	1T	2009	8201	THIN SPACE
	1H	200A	8202	HAIR SPACE
␣	-1	2010	8208	HYPHEN

-	-N	2013	8211	EN DASH
-	-M	2014	8212	EM DASH
-	-3	2015	8213	HORIZONTAL BAR
☒	!2	2016	8214	DOUBLE VERTICAL LINE
☒	=2	2017	8215	DOUBLE LOW LINE
'	'6	2018	8216	LEFT SINGLE QUOTATION MARK
'	'9	2019	8217	RIGHT SINGLE QUOTATION MARK
,	.9	201A	8218	SINGLE LOW-9 QUOTATION MARK
☒	9'	201B	8219	SINGLE HIGH-REVERSED-9 QUOTATION MARK
“	"6	201C	8220	LEFT DOUBLE QUOTATION MARK
”	"9	201D	8221	RIGHT DOUBLE QUOTATION MARK
„	:9	201E	8222	DOUBLE LOW-9 QUOTATION MARK
☒	9"	201F	8223	DOUBLE HIGH-REVERSED-9 QUOTATION MARK
†	/-	2020	8224	DAGGER
‡	/=	2021	8225	DOUBLE DAGGER
•	oo	2022	8226	BULLET
☒	..	2025	8229	TWO DOT LEADER
...	,.	2026	8230	HORIZONTAL ELLIPSIS
‰	%0	2030	8240	PER MILLE SIGN
'	1'	2032	8242	PRIME
"	2'	2033	8243	DOUBLE PRIME
☒	3'	2034	8244	TRIPLE PRIME
☒	4'	2057	8279	QUADRUPLE PRIME
☒	1"	2035	8245	REVERSED PRIME
☒	2"	2036	8246	REVERSED DOUBLE PRIME
☒	3"	2037	8247	REVERSED TRIPLE PRIME
☒	Ca	2038	8248	CARET
<	<1	2039	8249	SINGLE LEFT-POINTING ANGLE QUOTATION MARK
>	>1	203A	8250	SINGLE RIGHT-POINTING ANGLE QUOTATION MARK
☒	:X	203B	8251	REFERENCE MARK
☒	'-	203E	8254	OVERLINE
/	/f	2044	8260	FRACTION SLASH
°	0S	2070	8304	SUPERSCRIPIT ZERO
⁴	4S	2074	8308	SUPERSCRIPIT FOUR
⁵	5S	2075	8309	SUPERSCRIPIT FIVE
⁶	6S	2076	8310	SUPERSCRIPIT SIX
⁷	7S	2077	8311	SUPERSCRIPIT SEVEN
⁸	8S	2078	8312	SUPERSCRIPIT EIGHT
⁹	9S	2079	8313	SUPERSCRIPIT NINE
☒	+S	207A	8314	SUPERSCRIPIT PLUS SIGN
☒	-S	207B	8315	SUPERSCRIPIT MINUS
☒	=S	207C	8316	SUPERSCRIPIT EQUALS SIGN
((S	207D	8317	SUPERSCRIPIT LEFT PARENTHESIS
))S	207E	8318	SUPERSCRIPIT RIGHT PARENTHESIS
ⁿ	nS	207F	8319	SUPERSCRIPIT LATIN SMALL LETTER N
₀	0s	2080	8320	SUBSCRIPT ZERO
₁	1s	2081	8321	SUBSCRIPT ONE
₂	2s	2082	8322	SUBSCRIPT TWO
₃	3s	2083	8323	SUBSCRIPT THREE
₄	4s	2084	8324	SUBSCRIPT FOUR
₅	5s	2085	8325	SUBSCRIPT FIVE
₆	6s	2086	8326	SUBSCRIPT SIX
₇	7s	2087	8327	SUBSCRIPT SEVEN
₈	8s	2088	8328	SUBSCRIPT EIGHT

₉	9s	2089	8329	SUBSCRIPT NINE
⊕	+s	208A	8330	SUBSCRIPT PLUS SIGN
⊖	-s	208B	8331	SUBSCRIPT MINUS
⊗	=s	208C	8332	SUBSCRIPT EQUALS SIGN
((s	208D	8333	SUBSCRIPT LEFT PARENTHESIS
))s	208E	8334	SUBSCRIPT RIGHT PARENTHESIS
₣	Li	20A4	8356	LIRA SIGN
₧	Pt	20A7	8359	PESETA SIGN
₩	W=	20A9	8361	WON SIGN
€	Eu	20AC	8364	EURO SIGN
₹	=R	20BD	8381	ROUBLE SIGN
₺	=P	20BD	8381	ROUBLE SIGN
°C	oC	2103	8451	DEGREE CELSIUS
co	co	2105	8453	CARE OF
oF	oF	2109	8457	DEGREE FAHRENHEIT
№	N0	2116	8470	NUMERO SIGN
©	PO	2117	8471	SOUND RECORDING COPYRIGHT
℞	Rx	211E	8478	PRESCRIPTION TAKE
SM	SM	2120	8480	SERVICE MARK
™	TM	2122	8482	TRADE MARK SIGN
Ω	Om	2126	8486	OHM SIGN
Å	A0	212B	8491	ANGSTROM SIGN
⅓	13	2153	8531	VULGAR FRACTION ONE THIRD
⅔	23	2154	8532	VULGAR FRACTION TWO THIRDS
⅕	15	2155	8533	VULGAR FRACTION ONE FIFTH
⅖	25	2156	8534	VULGAR FRACTION TWO FIFTHS
⅗	35	2157	8535	VULGAR FRACTION THREE FIFTHS
⅘	45	2158	8536	VULGAR FRACTION FOUR FIFTHS
⅙	16	2159	8537	VULGAR FRACTION ONE SIXTH
⅚	56	215A	8538	VULGAR FRACTION FIVE SIXTHS
⅛	18	215B	8539	VULGAR FRACTION ONE EIGHTH
⅜	38	215C	8540	VULGAR FRACTION THREE EIGHTHS
⅝	58	215D	8541	VULGAR FRACTION FIVE EIGHTHS
⅞	78	215E	8542	VULGAR FRACTION SEVEN EIGHTHS
Ⅰ	1R	2160	8544	ROMAN NUMERAL ONE
Ⅱ	2R	2161	8545	ROMAN NUMERAL TWO
Ⅲ	3R	2162	8546	ROMAN NUMERAL THREE
Ⅳ	4R	2163	8547	ROMAN NUMERAL FOUR
Ⅴ	5R	2164	8548	ROMAN NUMERAL FIVE
Ⅵ	6R	2165	8549	ROMAN NUMERAL SIX
Ⅶ	7R	2166	8550	ROMAN NUMERAL SEVEN
Ⅷ	8R	2167	8551	ROMAN NUMERAL EIGHT
Ⅸ	9R	2168	8552	ROMAN NUMERAL NINE
Ⅹ	aR	2169	8553	ROMAN NUMERAL TEN
Ⅺ	bR	216A	8554	ROMAN NUMERAL ELEVEN
Ⅻ	cR	216B	8555	ROMAN NUMERAL TWELVE
ⅰ	1r	2170	8560	SMALL ROMAN NUMERAL ONE
ⅱ	2r	2171	8561	SMALL ROMAN NUMERAL TWO
ⅲ	3r	2172	8562	SMALL ROMAN NUMERAL THREE
ⅳ	4r	2173	8563	SMALL ROMAN NUMERAL FOUR
ⅴ	5r	2174	8564	SMALL ROMAN NUMERAL FIVE
ⅵ	6r	2175	8565	SMALL ROMAN NUMERAL SIX
ⅶ	7r	2176	8566	SMALL ROMAN NUMERAL SEVEN
ⅷ	8r	2177	8567	SMALL ROMAN NUMERAL EIGHT

Ⅸ	9r	2178	8568	SMALL ROMAN NUMERAL NINE
Ⅹ	ar	2179	8569	SMALL ROMAN NUMERAL TEN
Ⅺ	br	217A	8570	SMALL ROMAN NUMERAL ELEVEN
Ⅻ	cr	217B	8571	SMALL ROMAN NUMERAL TWELVE
←	<-	2190	8592	LEFTWARDS ARROW
↑	-!	2191	8593	UPWARDS ARROW
→	->	2192	8594	RIGHTWARDS ARROW
↓	-v	2193	8595	DOWNWARDS ARROW
↔	<>	2194	8596	LEFT RIGHT ARROW
↕	UD	2195	8597	UP DOWN ARROW
⇐	<=	21D0	8656	LEFTWARDS DOUBLE ARROW
⇒	=>	21D2	8658	RIGHTWARDS DOUBLE ARROW
⇔	==	21D4	8660	LEFT RIGHT DOUBLE ARROW
∀	FA	2200	8704	FOR ALL
∂	dP	2202	8706	PARTIAL DIFFERENTIAL
∃	TE	2203	8707	THERE EXISTS
∅	/0	2205	8709	EMPTY SET
Δ	DE	2206	8710	INCREMENT
∇	NB	2207	8711	NABLA
∈	(-	2208	8712	ELEMENT OF
∋	-)	220B	8715	CONTAINS AS MEMBER
∏	*P	220F	8719	N-ARY PRODUCT
∑	+Z	2211	8721	N-ARY SUMMATION
-	-2	2212	8722	MINUS SIGN
±	-+	2213	8723	MINUS-OR-PLUS SIGN
*	*-	2217	8727	ASTERISK OPERATOR
⊙	Ob	2218	8728	RING OPERATOR
⋅	Sb	2219	8729	BULLET OPERATOR
√	RT	221A	8730	SQUARE ROOT
∝	0(221D	8733	PROPORTIONAL TO
∞	00	221E	8734	INFINITY
∟	-L	221F	8735	RIGHT ANGLE
∠	-V	2220	8736	ANGLE
∥	PP	2225	8741	PARALLEL TO
∧	AN	2227	8743	LOGICAL AND
∨	OR	2228	8744	LOGICAL OR
∩	(U	2229	8745	INTERSECTION
∪)U	222A	8746	UNION
∫	In	222B	8747	INTEGRAL
∬	DI	222C	8748	DOUBLE INTEGRAL
∮	Io	222E	8750	CONTOUR INTEGRAL
∴	∴	2234	8756	THEREFORE
∵	∵	2235	8757	BECAUSE
∶	:R	2236	8758	RATIO
∝	∴	2237	8759	PROPORTION
˜	?1	223C	8764	TILDE OPERATOR
≍	CG	223E	8766	INVERTED LAZY S
≈	?-	2243	8771	ASYMPTOTICALLY EQUAL TO
≈	?=	2245	8773	APPROXIMATELY EQUAL TO
≈	?2	2248	8776	ALMOST EQUAL TO
≡	=?	224C	8780	ALL EQUAL TO
≍	HI	2253	8787	IMAGE OF OR APPROXIMATELY EQUAL TO
≠	!=	2260	8800	NOT EQUAL TO
≡	=3	2261	8801	IDENTICAL TO

≤	=<	2264	8804	LESS-THAN OR EQUAL TO
≥	>=	2265	8805	GREATER-THAN OR EQUAL TO
⊠	<*	226A	8810	MUCH LESS-THAN
⊠	*>	226B	8811	MUCH GREATER-THAN
⊠	!<	226E	8814	NOT LESS-THAN
⊠	!>	226F	8815	NOT GREATER-THAN
⊠	(C	2282	8834	SUBSET OF
⊠)C	2283	8835	SUPERSET OF
⊠	(_	2286	8838	SUBSET OF OR EQUAL TO
⊠)_	2287	8839	SUPERSET OF OR EQUAL TO
⊠	0.	2299	8857	CIRCLED DOT OPERATOR
⊠	02	229A	8858	CIRCLED RING OPERATOR
⊠	-T	22A5	8869	UP TACK
⊠	.P	22C5	8901	DOT OPERATOR
⊠	:3	22EE	8942	VERTICAL ELLIPSIS
⊠	.3	22EF	8943	MIDLINE HORIZONTAL ELLIPSIS
⊠	Eh	2302	8962	HOUSE
⊠	<7	2308	8968	LEFT CEILING
⊠	>7	2309	8969	RIGHT CEILING
⊠	7<	230A	8970	LEFT FLOOR
⊠	7>	230B	8971	RIGHT FLOOR
⊠	NI	2310	8976	REVERSED NOT SIGN
⊠	(A	2312	8978	ARC
⊠	TR	2315	8981	TELEPHONE RECORDER
⊠	Iu	2320	8992	TOP HALF INTEGRAL
⊠	Il	2321	8993	BOTTOM HALF INTEGRAL
⊠	</	2329	9001	LEFT-POINTING ANGLE BRACKET
⊠	/>	232A	9002	RIGHT-POINTING ANGLE BRACKET
⊠	Vs	2423	9251	OPEN BOX
⊠	1h	2440	9280	OCR HOOK
⊠	3h	2441	9281	OCR CHAIR
⊠	2h	2442	9282	OCR FORK
⊠	4h	2443	9283	OCR INVERTED FORK
⊠	1j	2446	9286	OCR BRANCH BANK IDENTIFICATION
⊠	2j	2447	9287	OCR AMOUNT OF CHECK
⊠	3j	2448	9288	OCR DASH
⊠	4j	2449	9289	OCR CUSTOMER ACCOUNT NUMBER
⊠	1.	2488	9352	DIGIT ONE FULL STOP
⊠	2.	2489	9353	DIGIT TWO FULL STOP
⊠	3.	248A	9354	DIGIT THREE FULL STOP
⊠	4.	248B	9355	DIGIT FOUR FULL STOP
⊠	5.	248C	9356	DIGIT FIVE FULL STOP
⊠	6.	248D	9357	DIGIT SIX FULL STOP
⊠	7.	248E	9358	DIGIT SEVEN FULL STOP
⊠	8.	248F	9359	DIGIT EIGHT FULL STOP
⊠	9.	2490	9360	DIGIT NINE FULL STOP
—	hh	2500	9472	BOX DRAWINGS LIGHT HORIZONTAL
—	HH	2501	9473	BOX DRAWINGS HEAVY HORIZONTAL
	vv	2502	9474	BOX DRAWINGS LIGHT VERTICAL
	VV	2503	9475	BOX DRAWINGS HEAVY VERTICAL
---	3-	2504	9476	BOX DRAWINGS LIGHT TRIPLE DASH HORIZONTAL
---	3_	2505	9477	BOX DRAWINGS HEAVY TRIPLE DASH HORIZONTAL
⋮	3!	2506	9478	BOX DRAWINGS LIGHT TRIPLE DASH VERTICAL
⋮	3/	2507	9479	BOX DRAWINGS HEAVY TRIPLE DASH VERTICAL

4-	2508	9480	BOX DRAWINGS LIGHT QUADRUPLE DASH HORIZONTAL
4_	2509	9481	BOX DRAWINGS HEAVY QUADRUPLE DASH HORIZONTAL
4!	250A	9482	BOX DRAWINGS LIGHT QUADRUPLE DASH VERTICAL
4/	250B	9483	BOX DRAWINGS HEAVY QUADRUPLE DASH VERTICAL
dr	250C	9484	BOX DRAWINGS LIGHT DOWN AND RIGHT
dR	250D	9485	BOX DRAWINGS DOWN LIGHT AND RIGHT HEAVY
Dr	250E	9486	BOX DRAWINGS DOWN HEAVY AND RIGHT LIGHT
DR	250F	9487	BOX DRAWINGS HEAVY DOWN AND RIGHT
dł	2510	9488	BOX DRAWINGS LIGHT DOWN AND LEFT
dL	2511	9489	BOX DRAWINGS DOWN LIGHT AND LEFT HEAVY
Dł	2512	9490	BOX DRAWINGS DOWN HEAVY AND LEFT LIGHT
LD	2513	9491	BOX DRAWINGS HEAVY DOWN AND LEFT
ur	2514	9492	BOX DRAWINGS LIGHT UP AND RIGHT
uR	2515	9493	BOX DRAWINGS UP LIGHT AND RIGHT HEAVY
Ur	2516	9494	BOX DRAWINGS UP HEAVY AND RIGHT LIGHT
UR	2517	9495	BOX DRAWINGS HEAVY UP AND RIGHT
uł	2518	9496	BOX DRAWINGS LIGHT UP AND LEFT
uL	2519	9497	BOX DRAWINGS UP LIGHT AND LEFT HEAVY
Uł	251A	9498	BOX DRAWINGS UP HEAVY AND LEFT LIGHT
UL	251B	9499	BOX DRAWINGS HEAVY UP AND LEFT
vr	251C	9500	BOX DRAWINGS LIGHT VERTICAL AND RIGHT
vR	251D	9501	BOX DRAWINGS VERTICAL LIGHT AND RIGHT HEAVY
Vr	2520	9504	BOX DRAWINGS VERTICAL HEAVY AND RIGHT LIGHT
VR	2523	9507	BOX DRAWINGS HEAVY VERTICAL AND RIGHT
vł	2524	9508	BOX DRAWINGS LIGHT VERTICAL AND LEFT
vL	2525	9509	BOX DRAWINGS VERTICAL LIGHT AND LEFT HEAVY
Vł	2528	9512	BOX DRAWINGS VERTICAL HEAVY AND LEFT LIGHT
VL	252B	9515	BOX DRAWINGS HEAVY VERTICAL AND LEFT
dh	252C	9516	BOX DRAWINGS LIGHT DOWN AND HORIZONTAL
dH	252F	9519	BOX DRAWINGS DOWN LIGHT AND HORIZONTAL HEAVY
Dh	2530	9520	BOX DRAWINGS DOWN HEAVY AND HORIZONTAL LIGHT
DH	2533	9523	BOX DRAWINGS HEAVY DOWN AND HORIZONTAL
uh	2534	9524	BOX DRAWINGS LIGHT UP AND HORIZONTAL
uH	2537	9527	BOX DRAWINGS UP LIGHT AND HORIZONTAL HEAVY
Uh	2538	9528	BOX DRAWINGS UP HEAVY AND HORIZONTAL LIGHT
UH	253B	9531	BOX DRAWINGS HEAVY UP AND HORIZONTAL
vh	253C	9532	BOX DRAWINGS LIGHT VERTICAL AND HORIZONTAL
vH	253F	9535	BOX DRAWINGS VERTICAL LIGHT AND HORIZONTAL HEAVY
Vh	2542	9538	BOX DRAWINGS VERTICAL HEAVY AND HORIZONTAL LIGHT
VH	254B	9547	BOX DRAWINGS HEAVY VERTICAL AND HORIZONTAL
FD	2571	9585	BOX DRAWINGS LIGHT DIAGONAL UPPER RIGHT TO LOWER LEFT
BD	2572	9586	BOX DRAWINGS LIGHT DIAGONAL UPPER LEFT TO LOWER RIGHT
TB	2580	9600	UPPER HALF BLOCK
LB	2584	9604	LOWER HALF BLOCK
FB	2588	9608	FULL BLOCK
łB	258C	9612	LEFT HALF BLOCK
RB	2590	9616	RIGHT HALF BLOCK
.S	2591	9617	LIGHT SHADE
:S	2592	9618	MEDIUM SHADE
?S	2593	9619	DARK SHADE
.	25A0	9632	BLACK SQUARE

☒	OS	25A1	9633	WHITE SQUARE
☒	RO	25A2	9634	WHITE SQUARE WITH ROUNDED CORNERS
☒	Rr	25A3	9635	WHITE SQUARE CONTAINING BLACK SMALL SQUARE
☒	RF	25A4	9636	SQUARE WITH HORIZONTAL FILL
☒	RY	25A5	9637	SQUARE WITH VERTICAL FILL
☒	RH	25A6	9638	SQUARE WITH ORTHOGONAL CROSSHATCH FILL
☒	RZ	25A7	9639	SQUARE WITH UPPER LEFT TO LOWER RIGHT FILL
☒	RK	25A8	9640	SQUARE WITH UPPER RIGHT TO LOWER LEFT FILL
☒	RX	25A9	9641	SQUARE WITH DIAGONAL CROSSHATCH FILL
☒	sB	25AA	9642	BLACK SMALL SQUARE
☒	SR	25AC	9644	BLACK RECTANGLE
☒	Or	25AD	9645	WHITE RECTANGLE
▲	UT	25B2	9650	BLACK UP-POINTING TRIANGLE
△	uT	25B3	9651	WHITE UP-POINTING TRIANGLE
►	PR	25B6	9654	BLACK RIGHT-POINTING TRIANGLE
▷	Tr	25B7	9655	WHITE RIGHT-POINTING TRIANGLE
▼	Dt	25BC	9660	BLACK DOWN-POINTING TRIANGLE
▽	dT	25BD	9661	WHITE DOWN-POINTING TRIANGLE
◄	PL	25C0	9664	BLACK LEFT-POINTING TRIANGLE
◁	Tl	25C1	9665	WHITE LEFT-POINTING TRIANGLE
•	Db	25C6	9670	BLACK DIAMOND
☒	Dw	25C7	9671	WHITE DIAMOND
◇	LZ	25CA	9674	LOZENGE
☒	0m	25CB	9675	WHITE CIRCLE
☒	0o	25CE	9678	BULLSEYE
☒	0M	25CF	9679	BLACK CIRCLE
☒	0L	25D0	9680	CIRCLE WITH LEFT HALF BLACK
☒	0R	25D1	9681	CIRCLE WITH RIGHT HALF BLACK
☒	Sn	25D8	9688	INVERSE BULLET
☒	Ic	25D9	9689	INVERSE WHITE CIRCLE
☒	Fd	25E2	9698	BLACK LOWER RIGHT TRIANGLE
☒	Bd	25E3	9699	BLACK LOWER LEFT TRIANGLE
☒	*2	2605	9733	BLACK STAR
☒	*1	2606	9734	WHITE STAR
☒	<H	261C	9756	WHITE LEFT POINTING INDEX
☒	>H	261E	9758	WHITE RIGHT POINTING INDEX
☒	0u	263A	9786	WHITE SMILING FACE
☒	0U	263B	9787	BLACK SMILING FACE
☒	SU	263C	9788	WHITE SUN WITH RAYS
☒	Fm	2640	9792	FEMALE SIGN
☒	Ml	2642	9794	MALE SIGN
☒	cS	2660	9824	BLACK SPADE SUIT
☒	cH	2661	9825	WHITE HEART SUIT
☒	cD	2662	9826	WHITE DIAMOND SUIT
☒	cC	2663	9827	BLACK CLUB SUIT
☒	Md	2669	9833	QUARTER NOTE
♪	M8	266A	9834	EIGHTH NOTE
☒	M2	266B	9835	BEAMED EIGHTH NOTES
☒	Mb	266D	9837	MUSIC FLAT SIGN
☒	Mx	266E	9838	MUSIC NATURAL SIGN
☒	MX	266F	9839	MUSIC SHARP SIGN
✓	OK	2713	10003	CHECK MARK
☒	XX	2717	10007	BALLOT X
☒	-X	2720	10016	MALTESE CROSS

	IS	3000	12288	IDEOGRAPHIC SPACE
☒	,_	3001	12289	IDEOGRAPHIC COMMA
☒	._	3002	12290	IDEOGRAPHIC FULL STOP
☒	+"	3003	12291	DITTO MARK
☒	+_	3004	12292	JAPANESE INDUSTRIAL STANDARD SYMBOL
☒	*_	3005	12293	IDEOGRAPHIC ITERATION MARK
☒	;-	3006	12294	IDEOGRAPHIC CLOSING MARK
☒	0_	3007	12295	IDEOGRAPHIC NUMBER ZERO
☒	<+	300A	12298	LEFT DOUBLE ANGLE BRACKET
☒	>+	300B	12299	RIGHT DOUBLE ANGLE BRACKET
☒	<'	300C	12300	LEFT CORNER BRACKET
☒	>'	300D	12301	RIGHT CORNER BRACKET
☒	<"	300E	12302	LEFT WHITE CORNER BRACKET
☒	>"	300F	12303	RIGHT WHITE CORNER BRACKET
☒	("	3010	12304	LEFT BLACK LENTICULAR BRACKET
☒)"	3011	12305	RIGHT BLACK LENTICULAR BRACKET
☒	=T	3012	12306	POSTAL MARK
☒	=_	3013	12307	GETA MARK
☒	('	3014	12308	LEFT TORTOISE SHELL BRACKET
☒)'	3015	12309	RIGHT TORTOISE SHELL BRACKET
☒	(I	3016	12310	LEFT WHITE LENTICULAR BRACKET
☒)I	3017	12311	RIGHT WHITE LENTICULAR BRACKET
☒	-?	301C	12316	WAVE DASH
☒	A5	3041	12353	HIRAGANA LETTER SMALL A
☒	a5	3042	12354	HIRAGANA LETTER A
☒	I5	3043	12355	HIRAGANA LETTER SMALL I
☒	i5	3044	12356	HIRAGANA LETTER I
☒	U5	3045	12357	HIRAGANA LETTER SMALL U
☒	u5	3046	12358	HIRAGANA LETTER U
☒	E5	3047	12359	HIRAGANA LETTER SMALL E
☒	e5	3048	12360	HIRAGANA LETTER E
☒	O5	3049	12361	HIRAGANA LETTER SMALL O
☒	o5	304A	12362	HIRAGANA LETTER O
☒	ka	304B	12363	HIRAGANA LETTER KA
☒	ga	304C	12364	HIRAGANA LETTER GA
☒	ki	304D	12365	HIRAGANA LETTER KI
☒	gi	304E	12366	HIRAGANA LETTER GI
☒	ku	304F	12367	HIRAGANA LETTER KU
☒	gu	3050	12368	HIRAGANA LETTER GU
☒	ke	3051	12369	HIRAGANA LETTER KE
☒	ge	3052	12370	HIRAGANA LETTER GE
☒	ko	3053	12371	HIRAGANA LETTER KO
☒	go	3054	12372	HIRAGANA LETTER GO
☒	sa	3055	12373	HIRAGANA LETTER SA
☒	za	3056	12374	HIRAGANA LETTER ZA
☒	si	3057	12375	HIRAGANA LETTER SI
☒	zi	3058	12376	HIRAGANA LETTER ZI
☒	su	3059	12377	HIRAGANA LETTER SU
☒	zu	305A	12378	HIRAGANA LETTER ZU
☒	se	305B	12379	HIRAGANA LETTER SE
☒	ze	305C	12380	HIRAGANA LETTER ZE
☒	so	305D	12381	HIRAGANA LETTER SO
☒	zo	305E	12382	HIRAGANA LETTER ZO
☒	ta	305F	12383	HIRAGANA LETTER TA

☒	da	3060	12384	HIRAGANA LETTER DA	
☒	ti	3061	12385	HIRAGANA LETTER TI	
☒	di	3062	12386	HIRAGANA LETTER DI	
☒	tU	3063	12387	HIRAGANA LETTER SMALL TU	
☒	tu	3064	12388	HIRAGANA LETTER TU	
☒	du	3065	12389	HIRAGANA LETTER DU	
☒	te	3066	12390	HIRAGANA LETTER TE	
☒	de	3067	12391	HIRAGANA LETTER DE	
☒	to	3068	12392	HIRAGANA LETTER TO	
☒	do	3069	12393	HIRAGANA LETTER DO	
☒	na	306A	12394	HIRAGANA LETTER NA	
☒	ni	306B	12395	HIRAGANA LETTER NI	
☒	nu	306C	12396	HIRAGANA LETTER NU	
☒	ne	306D	12397	HIRAGANA LETTER NE	
☒	no	306E	12398	HIRAGANA LETTER NO	
☒	ha	306F	12399	HIRAGANA LETTER HA	
☒	ba	3070	12400	HIRAGANA LETTER BA	
☒	pa	3071	12401	HIRAGANA LETTER PA	
☒	hi	3072	12402	HIRAGANA LETTER HI	
☒	bi	3073	12403	HIRAGANA LETTER BI	
☒	pi	3074	12404	HIRAGANA LETTER PI	
☒	hu	3075	12405	HIRAGANA LETTER HU	
☒	bu	3076	12406	HIRAGANA LETTER BU	
☒	pu	3077	12407	HIRAGANA LETTER PU	
☒	he	3078	12408	HIRAGANA LETTER HE	
☒	be	3079	12409	HIRAGANA LETTER BE	
☒	pe	307A	12410	HIRAGANA LETTER PE	
☒	ho	307B	12411	HIRAGANA LETTER HO	
☒	bo	307C	12412	HIRAGANA LETTER BO	
☒	po	307D	12413	HIRAGANA LETTER PO	
☒	ma	307E	12414	HIRAGANA LETTER MA	
☒	mi	307F	12415	HIRAGANA LETTER MI	
☒	mu	3080	12416	HIRAGANA LETTER MU	
☒	me	3081	12417	HIRAGANA LETTER ME	
☒	mo	3082	12418	HIRAGANA LETTER MO	
☒	yA	3083	12419	HIRAGANA LETTER SMALL YA	
☒	ya	3084	12420	HIRAGANA LETTER YA	
☒	yU	3085	12421	HIRAGANA LETTER SMALL YU	
☒	yu	3086	12422	HIRAGANA LETTER YU	
☒	yO	3087	12423	HIRAGANA LETTER SMALL YO	
☒	yo	3088	12424	HIRAGANA LETTER YO	
☒	ra	3089	12425	HIRAGANA LETTER RA	
☒	ri	308A	12426	HIRAGANA LETTER RI	
☒	ru	308B	12427	HIRAGANA LETTER RU	
☒	re	308C	12428	HIRAGANA LETTER RE	
☒	ro	308D	12429	HIRAGANA LETTER RO	
☒	wA	308E	12430	HIRAGANA LETTER SMALL WA	
☒	wa	308F	12431	HIRAGANA LETTER WA	
☒	wi	3090	12432	HIRAGANA LETTER WI	
☒	we	3091	12433	HIRAGANA LETTER WE	
☒	wo	3092	12434	HIRAGANA LETTER WO	
☒	n5	3093	12435	HIRAGANA LETTER N	
☒	vu	3094	12436	HIRAGANA LETTER VU	
☒	"5	309B	12443	KATAKANA-HIRAGANA VOICED SOUND MARK	

☒	05	309C	12444	KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK
☒	*5	309D	12445	HIRAGANA ITERATION MARK
☒	+5	309E	12446	HIRAGANA VOICED ITERATION MARK
☒	a6	30A1	12449	KATAKANA LETTER SMALL A
☒	A6	30A2	12450	KATAKANA LETTER A
☒	i6	30A3	12451	KATAKANA LETTER SMALL I
☒	I6	30A4	12452	KATAKANA LETTER I
☒	u6	30A5	12453	KATAKANA LETTER SMALL U
☒	U6	30A6	12454	KATAKANA LETTER U
☒	e6	30A7	12455	KATAKANA LETTER SMALL E
☒	E6	30A8	12456	KATAKANA LETTER E
☒	o6	30A9	12457	KATAKANA LETTER SMALL O
☒	O6	30AA	12458	KATAKANA LETTER O
☒	Ka	30AB	12459	KATAKANA LETTER KA
☒	Ga	30AC	12460	KATAKANA LETTER GA
☒	Ki	30AD	12461	KATAKANA LETTER KI
☒	Gi	30AE	12462	KATAKANA LETTER GI
☒	Ku	30AF	12463	KATAKANA LETTER KU
☒	Gu	30B0	12464	KATAKANA LETTER GU
☒	Ke	30B1	12465	KATAKANA LETTER KE
☒	Ge	30B2	12466	KATAKANA LETTER GE
☒	Ko	30B3	12467	KATAKANA LETTER KO
☒	Go	30B4	12468	KATAKANA LETTER GO
☒	Sa	30B5	12469	KATAKANA LETTER SA
☒	Za	30B6	12470	KATAKANA LETTER ZA
☒	Si	30B7	12471	KATAKANA LETTER SI
☒	Zi	30B8	12472	KATAKANA LETTER ZI
☒	Su	30B9	12473	KATAKANA LETTER SU
☒	Zu	30BA	12474	KATAKANA LETTER ZU
☒	Se	30BB	12475	KATAKANA LETTER SE
☒	Ze	30BC	12476	KATAKANA LETTER ZE
☒	So	30BD	12477	KATAKANA LETTER SO
☒	Zo	30BE	12478	KATAKANA LETTER ZO
☒	Ta	30BF	12479	KATAKANA LETTER TA
☒	Da	30C0	12480	KATAKANA LETTER DA
☒	Ti	30C1	12481	KATAKANA LETTER TI
☒	Di	30C2	12482	KATAKANA LETTER DI
☒	TU	30C3	12483	KATAKANA LETTER SMALL TU
☒	Tu	30C4	12484	KATAKANA LETTER TU
☒	Du	30C5	12485	KATAKANA LETTER DU
☒	Te	30C6	12486	KATAKANA LETTER TE
☒	De	30C7	12487	KATAKANA LETTER DE
☒	To	30C8	12488	KATAKANA LETTER TO
☒	Do	30C9	12489	KATAKANA LETTER DO
☒	Na	30CA	12490	KATAKANA LETTER NA
☒	Ni	30CB	12491	KATAKANA LETTER NI
☒	Nu	30CC	12492	KATAKANA LETTER NU
☒	Ne	30CD	12493	KATAKANA LETTER NE
☒	No	30CE	12494	KATAKANA LETTER NO
☒	Ha	30CF	12495	KATAKANA LETTER HA
☒	Ba	30D0	12496	KATAKANA LETTER BA
☒	Pa	30D1	12497	KATAKANA LETTER PA
☒	Hi	30D2	12498	KATAKANA LETTER HI
☒	Bi	30D3	12499	KATAKANA LETTER BI

☒	Pi	30D4	12500	KATAKANA LETTER PI
☒	Hu	30D5	12501	KATAKANA LETTER HU
☒	Bu	30D6	12502	KATAKANA LETTER BU
☒	Pu	30D7	12503	KATAKANA LETTER PU
☒	He	30D8	12504	KATAKANA LETTER HE
☒	Be	30D9	12505	KATAKANA LETTER BE
☒	Pe	30DA	12506	KATAKANA LETTER PE
☒	Ho	30DB	12507	KATAKANA LETTER HO
☒	Bo	30DC	12508	KATAKANA LETTER BO
☒	Po	30DD	12509	KATAKANA LETTER PO
☒	Ma	30DE	12510	KATAKANA LETTER MA
☒	Mi	30DF	12511	KATAKANA LETTER MI
☒	Mu	30E0	12512	KATAKANA LETTER MU
☒	Me	30E1	12513	KATAKANA LETTER ME
☒	Mo	30E2	12514	KATAKANA LETTER MO
☒	YA	30E3	12515	KATAKANA LETTER SMALL YA
☒	Ya	30E4	12516	KATAKANA LETTER YA
☒	YU	30E5	12517	KATAKANA LETTER SMALL YU
☒	Yu	30E6	12518	KATAKANA LETTER YU
☒	YO	30E7	12519	KATAKANA LETTER SMALL YO
☒	Yo	30E8	12520	KATAKANA LETTER YO
☒	Ra	30E9	12521	KATAKANA LETTER RA
☒	Ri	30EA	12522	KATAKANA LETTER RI
☒	Ru	30EB	12523	KATAKANA LETTER RU
☒	Re	30EC	12524	KATAKANA LETTER RE
☒	Ro	30ED	12525	KATAKANA LETTER RO
☒	WA	30EE	12526	KATAKANA LETTER SMALL WA
☒	Wa	30EF	12527	KATAKANA LETTER WA
☒	Wi	30F0	12528	KATAKANA LETTER WI
☒	We	30F1	12529	KATAKANA LETTER WE
☒	Wo	30F2	12530	KATAKANA LETTER WO
☒	N6	30F3	12531	KATAKANA LETTER N
☒	Vu	30F4	12532	KATAKANA LETTER VU
☒	KA	30F5	12533	KATAKANA LETTER SMALL KA
☒	KE	30F6	12534	KATAKANA LETTER SMALL KE
☒	Va	30F7	12535	KATAKANA LETTER VA
☒	Vi	30F8	12536	KATAKANA LETTER VI
☒	Ve	30F9	12537	KATAKANA LETTER VE
☒	Vo	30FA	12538	KATAKANA LETTER VO
☒	.6	30FB	12539	KATAKANA MIDDLE DOT
☒	-6	30FC	12540	KATAKANA-HIRAGANA PROLONGED SOUND MARK
☒	*6	30FD	12541	KATAKANA ITERATION MARK
☒	+6	30FE	12542	KATAKANA VOICED ITERATION MARK
☒	b4	3105	12549	BOPOMOFO LETTER B
☒	p4	3106	12550	BOPOMOFO LETTER P
☒	m4	3107	12551	BOPOMOFO LETTER M
☒	f4	3108	12552	BOPOMOFO LETTER F
☒	d4	3109	12553	BOPOMOFO LETTER D
☒	t4	310A	12554	BOPOMOFO LETTER T
☒	n4	310B	12555	BOPOMOFO LETTER N
☒	l4	310C	12556	BOPOMOFO LETTER L
☒	g4	310D	12557	BOPOMOFO LETTER G
☒	k4	310E	12558	BOPOMOFO LETTER K
☒	h4	310F	12559	BOPOMOFO LETTER H

☒	j4	3110	12560	BOPOMOFO LETTER J
☒	q4	3111	12561	BOPOMOFO LETTER Q
☒	x4	3112	12562	BOPOMOFO LETTER X
☒	zh	3113	12563	BOPOMOFO LETTER ZH
☒	ch	3114	12564	BOPOMOFO LETTER CH
☒	sh	3115	12565	BOPOMOFO LETTER SH
☒	r4	3116	12566	BOPOMOFO LETTER R
☒	z4	3117	12567	BOPOMOFO LETTER Z
☒	c4	3118	12568	BOPOMOFO LETTER C
☒	s4	3119	12569	BOPOMOFO LETTER S
☒	a4	311A	12570	BOPOMOFO LETTER A
☒	o4	311B	12571	BOPOMOFO LETTER O
☒	e4	311C	12572	BOPOMOFO LETTER E
☒	ai	311E	12574	BOPOMOFO LETTER AI
☒	ei	311F	12575	BOPOMOFO LETTER EI
☒	au	3120	12576	BOPOMOFO LETTER AU
☒	ou	3121	12577	BOPOMOFO LETTER OU
☒	an	3122	12578	BOPOMOFO LETTER AN
☒	en	3123	12579	BOPOMOFO LETTER EN
☒	aN	3124	12580	BOPOMOFO LETTER ANG
☒	eN	3125	12581	BOPOMOFO LETTER ENG
☒	er	3126	12582	BOPOMOFO LETTER ER
☒	i4	3127	12583	BOPOMOFO LETTER I
☒	u4	3128	12584	BOPOMOFO LETTER U
☒	iu	3129	12585	BOPOMOFO LETTER IU
☒	v4	312A	12586	BOPOMOFO LETTER V
☒	nG	312B	12587	BOPOMOFO LETTER NG
☒	gn	312C	12588	BOPOMOFO LETTER GN
☒	1c	3220	12832	PARENTHESESIZED IDEOGRAPH ONE
☒	2c	3221	12833	PARENTHESESIZED IDEOGRAPH TWO
☒	3c	3222	12834	PARENTHESESIZED IDEOGRAPH THREE
☒	4c	3223	12835	PARENTHESESIZED IDEOGRAPH FOUR
☒	5c	3224	12836	PARENTHESESIZED IDEOGRAPH FIVE
☒	6c	3225	12837	PARENTHESESIZED IDEOGRAPH SIX
☒	7c	3226	12838	PARENTHESESIZED IDEOGRAPH SEVEN
☒	8c	3227	12839	PARENTHESESIZED IDEOGRAPH EIGHT
☒	9c	3228	12840	PARENTHESESIZED IDEOGRAPH NINE
☒	ff	FB00	64256	LATIN SMALL LIGATURE FF
fi	fi	FB01	64257	LATIN SMALL LIGATURE FI
fl	fl	FB02	64258	LATIN SMALL LIGATURE FL
☒	ft	FB05	64261	LATIN SMALL LIGATURE LONG S T
☒	st	FB06	64262	LATIN SMALL LIGATURE ST

Multi-byte support

multibyte multi-byte
Chinese Japanese Korean

This is about editing text in languages which have many characters that can not be represented using one byte (one octet). Examples are Chinese, Japanese and Korean. Unicode is also covered here.

For an introduction to the most common features, see [usr_45.txt](#) in the user manual.

For changing the language of messages and menus see [mlang.txt](#) .

- | | |
|--|-----------------------------------|
| 1. Getting started | mbyte-first |
| 2. Locale | mbyte-locale |
| 3. Encoding | mbyte-encoding |
| 4. Using a terminal | mbyte-terminal |
| 5. Fonts on X11 | mbyte-fonts-X11 |
| 6. Fonts on MS-Windows | mbyte-fonts-MSwin |
| 7. Input on X11 | mbyte-XIM |
| 8. Input on MS-Windows | mbyte-IME |
| 9. Input with a keymap | mbyte-keymap |
| 10. Input with <code>imactivatefunc()</code> | mbyte-func |
| 11. Using UTF-8 | mbyte-utf8 |
| 12. Overview of options | mbyte-options |

NOTE: This file contains UTF-8 characters. These may show up as strange characters or boxes when using another encoding.

=====

1. Getting started

[mbyte-first](#)

This is a summary of the multibyte features in Vim. If you are lucky it works as described and you can start using Vim without much trouble. If something doesn't work you will have to read the rest. Don't be surprised if it takes quite a bit of work and experimenting to make Vim use all the multibyte features. Unfortunately, every system has its own way to deal with multibyte languages and it is quite complicated.

LOCALE

First of all, you must make sure your current locale is set correctly. If your system has been installed to use the language, it probably works right away. If not, you can often make it work by setting the `$LANG` environment variable in your shell:

```
setenv LANG ja_JP.EUC
```

Unfortunately, the name of the locale depends on your system. Japanese might also be called "ja_JP.EUCjp" or just "ja". To see what is currently used:

```
:language
```

To change the locale inside Vim use:

```
:language ja_JP.EUC
```

Vim will give an error message if this doesn't work. This is a good way to experiment and find the locale name you want to use. But it's always better to set the locale in the shell, so that it is used right from the start.

See [mbyte-locale](#) for details.

ENCODING

If your locale works properly, Vim will try to set the **'encoding'** option accordingly. If this doesn't work you can overrule its value:

```
:set encoding=utf-8
```

See [encoding-values](#) for a list of acceptable values.

The result is that all the text that is used inside Vim will be in this encoding. Not only the text in the buffers, but also in registers, variables, etc. This also means that changing the value of **'encoding'** makes the existing text invalid! The text doesn't change, but it will be displayed wrong.

You can edit files in another encoding than what **'encoding'** is set to. Vim will convert the file when you read it and convert it back when you write it. See **'fileencoding'**, **'fileencodings'** and `++enc` .

DISPLAY AND FONTS

If you are working in a terminal (emulator) you must make sure it accepts the same encoding as which Vim is working with. If this is not the case, you can use the **'termencoding'** option to make Vim convert text automatically.

For the GUI you must select fonts that work with the current **'encoding'**. This is the difficult part. It depends on the system you are using, the locale and a few other things. See the chapters on fonts: [mbyte-fonts-X11](#) for X-Windows and [mbyte-fonts-MSwin](#) for MS-Windows.

For GTK+ 2, you can skip most of this section. The option **'guifontset'** does no longer exist. You only need to set **'guifont'** and everything should "just work". If your system comes with Xft2 and fontconfig and the current font does not contain a certain glyph, a different font will be used automatically if available. The **'guifontwide'** option is still supported but usually you do not need to set it. It is only necessary if the automatic font selection does not suit your needs.

For X11 you can set the **'guifontset'** option to a list of fonts that together cover the characters that are used. Example for Korean:


```
:set guifontset=k12,r12
```

Alternatively, you can set **'guifont'** and **'guifontwide'**. **'guifont'** is used for the single-width characters, **'guifontwide'** for the double-width characters. Thus the **'guifontwide'** font must be exactly twice as wide as **'guifont'**.

Example for UTF-8:

```
:set guifont=-misc-fixed-medium-r-normal-*-18-120-100-100-c-90-iso10646-1
:set guifontwide=-misc-fixed-medium-r-normal-*-18-120-100-100-c-180-iso10646-1
```

You can also set **'guifont'** alone, Vim will try to find a matching **'guifontwide'** for you.

INPUT

There are several ways to enter multibyte characters:

- For X11 XIM can be used. See [XIM](#) .
- For MS-Windows IME can be used. See [IME](#) .
- For all systems keymaps can be used. See [mbyte-keymap](#) .

The options **'iminsert'**, **'imsearch'** and **'imcmdline'** can be used to choose the different input methods or disable them temporarily.

2. Locale

[mbyte-locale](#)

The easiest setup is when your whole system uses the locale you want to work in. But it's also possible to set the locale for one shell you are working in, or just use a certain locale inside Vim.

WHAT IS A LOCALE?

[locale](#)

There are many languages in the world. And there are different cultures and environments at least as many as the number of languages. A linguistic environment corresponding to an area is called "locale". This includes information about the used language, the charset, collating order for sorting, date format, currency format and so on. For Vim only the language and charset really matter.

You can only use a locale if your system has support for it. Some systems have only a few locales, especially in the USA. The language which you want to use may not be on your system. In that case you might be able to install it as an extra package. Check your system documentation for how to do that.

The location in which the locales are installed varies from system to system. For example, `"/usr/share/locale"` or `"/usr/lib/locale"`. See your system's `setlocale()` man page.

Looking in these directories will show you the exact name of each locale. Mostly upper/lowercase matters, thus `"ja_JP.EUC"` and `"ja_jp.euc"` are different. Some systems have a `locale.alias` file, which allows translation

from a short name like "nl" to the full name "nl_NL.ISO_8859-1".

Note that X-windows has its own locale stuff. And unfortunately uses locale names different from what is used elsewhere. This is confusing! For Vim it matters what the `setlocale()` function uses, which is generally NOT the X-windows stuff. You might have to do some experiments to find out what really works.

The (simplified) format of `locale` name is: locale-name

```
language
or language_territory
or language_territory.codeset
```

Territory means the country (or part of it), codeset means the `charset`. For example, the locale name "ja_JP.eucJP" means:

```
ja      the language is Japanese
JP      the country is Japan
eucJP   the codeset is EUC-JP
```

But it also could be "ja", "ja_JP.EUC", "ja_JP.ujis", etc. And unfortunately, the locale name for a specific language, territory and codeset is not unified and depends on your system.

Examples of locale name:

charset	language	locale name
GB2312	Chinese (simplified)	zh_CN.EUC, zh_CN.GB2312
Big5	Chinese (traditional)	zh_TW.BIG5, zh_TW.Big5
CNS-11643	Chinese (traditional)	zh_TW
EUC-JP	Japanese	ja, ja_JP.EUC, ja_JP.ujis, ja_JP.eucJP
Shift_JIS	Japanese	ja_JP.SJIS, ja_JP.Shift_JIS
EUC-KR	Korean	ko, ko_KR.EUC

USING A LOCALE

To start using a locale for the whole system, see the documentation of your system. Mostly you need to set it in a configuration file in "/etc".

To use a locale in a shell, set the `$LANG` environment value. When you want to use Korean and the `locale` name is "ko", do this:

```
sh:   export LANG=ko
csh:  setenv LANG ko
```

You can put this in your `~/.profile` or `~/.cshrc` file to always use it.

To use a locale in Vim only, use the `:language` command:

```
:language ko
```

Put this in your `~/.vimrc` file to use it always.

Or specify `$LANG` when starting Vim:

```
sh: LANG=ko vim {vim-arguments}
csh: env LANG=ko vim {vim-arguments}
```

You could make a small shell script for this.

3. Encoding mbyte-encoding

Vim uses the **'encoding'** option to specify how characters are identified and encoded when they are used inside Vim. This applies to all the places where text is used, including buffers (files loaded into memory), registers and variables.

charset codeset

Charset is another name for encoding. There are subtle differences, but these don't matter when using Vim. "codeset" is another similar name.

Each character is encoded as one or more bytes. When all characters are encoded with one byte, we call this a single-byte encoding. The most often used one is called "latin1". This limits the number of characters to 256. Some of these are control characters, thus even fewer can be used for text.

When some characters use two or more bytes, we call this a multibyte encoding. This allows using much more than 256 characters, which is required for most East Asian languages.

Most multibyte encodings use one byte for the first 127 characters. These are equal to ASCII, which makes it easy to exchange plain-ASCII text, no matter what language is used. Thus you might see the right text even when the encoding was set wrong.

encoding-names

Vim can use many different character encodings. There are three major groups:

- | | | |
|---|---------|--|
| 1 | 8bit | Single-byte encodings, 256 different characters. Mostly used in USA and Europe. Example: ISO-8859-1 (Latin1). All characters occupy one screen cell only. |
| 2 | 2byte | Double-byte encodings, over 10000 different characters. Mostly used in Asian countries. Example: euc-kr (Korean) The number of screen cells is equal to the number of bytes (except for euc-jp when the first byte is 0x8e). |
| u | Unicode | Universal encoding, can replace all others. ISO 10646. Millions of different characters. Example: UTF-8. The relation between bytes and screen cells is complex. |

Other encodings cannot be used by Vim internally. But files in other encodings can be edited by using conversion, see **'fileencoding'**.
Note that all encodings must use ASCII for the characters up to 128 (except when compiled for EBCDIC).

Supported **'encoding'** values are: encoding-values

```

1 latin1      8-bit characters (ISO 8859-1, also used for cp1252)
1 iso-8859-n  ISO_8859 variant (n = 2 to 15)
1 koi8-r      Russian
1 koi8-u      Ukrainian
1 macroman    MacRoman (Macintosh encoding)
1 8bit-{name} any 8-bit encoding (Vim specific name)
1 cp437       similar to iso-8859-1
1 cp737       similar to iso-8859-7
1 cp775       Baltic
1 cp850       similar to iso-8859-4
1 cp852       similar to iso-8859-1
1 cp855       similar to iso-8859-2
1 cp857       similar to iso-8859-5
1 cp860       similar to iso-8859-9
1 cp861       similar to iso-8859-1
1 cp862       similar to iso-8859-1
1 cp863       similar to iso-8859-8
1 cp865       similar to iso-8859-1
1 cp866       similar to iso-8859-5
1 cp869       similar to iso-8859-7
1 cp874       Thai
1 cp1250      Czech, Polish, etc.
1 cp1251      Cyrillic
1 cp1253      Greek
1 cp1254      Turkish
1 cp1255      Hebrew
1 cp1256      Arabic
1 cp1257      Baltic
1 cp1258      Vietnamese
1 cp{number}  MS-Windows: any installed single-byte codepage
2 cp932       Japanese (Windows only)
2 euc-jp      Japanese (Unix only)
2 sjis        Japanese (Unix only)
2 cp949       Korean (Unix and Windows)
2 euc-kr      Korean (Unix only)
2 cp936       simplified Chinese (Windows only)
2 euc-cn      simplified Chinese (Unix only)
2 cp950       traditional Chinese (on Unix alias for big5)
2 big5        traditional Chinese (on Windows alias for cp950)
2 euc-tw      traditional Chinese (Unix only)
2 2byte-{name} Unix: any double-byte encoding (Vim specific name)
2 cp{number}  MS-Windows: any installed double-byte codepage
u utf-8       32 bit UTF-8 encoded Unicode (ISO/IEC 10646-1)
u ucs-2       16 bit UCS-2 encoded Unicode (ISO/IEC 10646-1)
u ucs-2le     like ucs-2, little endian
u utf-16      ucs-2 extended with double-words for more characters
u utf-16le    like utf-16, little endian
u ucs-4       32 bit UCS-4 encoded Unicode (ISO/IEC 10646-1)
u ucs-4le     like ucs-4, little endian

```

The `{name}` can be any encoding name that your system supports. It is passed to `iconv()` to convert between the encoding of the file and the current locale. For MS-Windows "cp{number}" means using codepage `{number}`.

Examples:

```
:set encoding=8bit-cp1252
:set encoding=2byte-cp932
```

The MS-Windows codepage 1252 is very similar to latin1. For practical reasons the same encoding is used and it's called latin1. **'isprint'** can be used to display the characters 0x80 - 0xA0 or not.

Several aliases can be used, they are translated to one of the names above. An incomplete list:

```
1  ansi          same as latin1 (obsolete, for backward compatibility)
2  japan        Japanese: on Unix "euc-jp", on MS-Windows cp932
2  korea        Korean: on Unix "euc-kr", on MS-Windows cp949
2  prc          simplified Chinese: on Unix "euc-cn", on MS-Windows cp936
2  chinese      same as "prc"
2  taiwan       traditional Chinese: on Unix "euc-tw", on MS-Windows cp950
u  utf8         same as utf-8
u  unicode      same as ucs-2
u  ucs2be       same as ucs-2 (big endian)
u  ucs-2be      same as ucs-2 (big endian)
u  ucs-4be      same as ucs-4 (big endian)
u  utf-32       same as ucs-4
u  utf-32le     same as ucs-4le
   default      stands for the default value of 'encoding', depends on the
                 environment
```

For the UCS codes the byte order matters. This is tricky, use UTF-8 whenever you can. The default is to use big-endian (most significant byte comes first):

name	bytes	char
ucs-2	11 22	1122
ucs-2le	22 11	1122
ucs-4	11 22 33 44	11223344
ucs-4le	44 33 22 11	11223344

On MS-Windows systems you often want to use "ucs-2le", because it uses little endian UCS-2.

There are a few encodings which are similar, but not exactly the same. Vim treats them as if they were different encodings, so that conversion will be done when needed. You might want to use the similar name to avoid conversion or when conversion is not possible:

```
cp932, shift-jis, sjis
cp936, euc-cn
```

encoding-table

Normally **'encoding'** is equal to your current locale and **'termencoding'** is empty. This means that your keyboard and display work with characters encoded in your current locale, and Vim uses the same characters internally.

You can make Vim use characters in a different encoding by setting the **'encoding'** option to a different value. Since the keyboard and display still use the current locale, conversion needs to be done. The **'termencoding'** then

takes over the value of the current locale, so Vim converts between **'encoding'** and **'termencoding'**. Example:

```
:let &termencoding = &encoding
:set encoding=utf-8
```

However, not all combinations of values are possible. The table below tells you how each of the nine combinations works. This is further restricted by not all conversions being possible, `iconv()` being present, etc. Since this depends on the system used, no detailed list can be given.

('tenc' is the short name for **'termencoding'** and **'enc'** short for **'encoding'**)

'tenc'	'enc'	remark
8bit	8bit	Works. When 'termencoding' is different from 'encoding' typing and displaying may be wrong for some characters, Vim does NOT perform conversion (set 'encoding' to "utf-8" to get this).
8bit	2byte	MS-Windows: works for all codepages installed on your system; you can only type 8bit characters; Other systems: does NOT work.
8bit	Unicode	Works, but only 8bit characters can be typed directly (others through digraphs, keymaps, etc.); in a terminal you can only see 8bit characters; the GUI can show all characters that the 'guifont' supports.
2byte	8bit	Works, but typing non-ASCII characters might be a problem.
2byte	2byte	MS-Windows: works for all codepages installed on your system; typing characters might be a problem when locale is different from 'encoding' . Other systems: Only works when 'termencoding' is equal to 'encoding' , you might as well leave it empty.
2byte	Unicode	works, Vim will translate typed characters.
Unicode	8bit	works (unusual)
Unicode	2byte	does NOT work
Unicode	Unicode	works very well (leaving 'termencoding' empty works the same way, because all Unicode is handled internally as UTF-8)

CONVERSION

charset-conversion

Vim will automatically convert from one to another encoding in several places:

- When reading a file and **'fileencoding'** is different from **'encoding'**
- When writing a file and **'fileencoding'** is different from **'encoding'**
- When displaying characters and **'termencoding'** is different from **'encoding'**
- When reading input and **'termencoding'** is different from **'encoding'**
- When displaying messages and the encoding used for LC_MESSAGES differs from **'encoding'** (requires a gettext version that supports this).
- When reading a Vim script where **:scriptencoding** is different from **'encoding'**.
- When reading or writing a `viminfo` file.

Most of these require the `+iconv` feature. Conversion for reading and

writing files may also be specified with the **'charconvert'** option.

Useful utilities for converting the charset:

All: iconv

GNU iconv can convert most encodings. Unicode is used as the intermediate encoding, which allows conversion from and to all other encodings. See <http://www.gnu.org/directory/libiconv.html>.

Japanese: nkf

Nkf is "Network Kanji code conversion Filter". One of the most unique facility of nkf is the guess of the input Kanji code. So, you don't need to know what the inputting file's **charset** is. When convert to EUC-JP from ISO-2022-JP or Shift_JIS, simply do the following command in Vim:

```
:%!nkf -e
```

Nkf can be found at:

<http://www.sfc.wide.ad.jp/~max/FreeBSD/ports/distfiles/nkf-1.62.tar.gz>

Chinese: hc

Hc is "Hanzi Converter". Hc convert a GB file to a Big5 file, or Big5 file to GB file. Hc can be found at:

<ftp://ftp.cuhk.hk/pub/chinese/ifcss/software/unix/convert/hc-30.tar.gz>

Korean: hmconv

Hmconv is Korean code conversion utility especially for E-mail. It can convert between EUC-KR and ISO-2022-KR. Hmconv can be found at:

<ftp://ftp.kaist.ac.kr/pub/hangul/code/hmconv/>

Multilingual: lv

Lv is a Powerful Multilingual File Viewer. And it can be worked as **charset** converter. Supported **charset** : ISO-2022-CN, ISO-2022-JP, ISO-2022-KR, EUC-CN, EUC-JP, EUC-KR, EUC-TW, UTF-7, UTF-8, ISO-8859 series, Shift_JIS, Big5 and HZ. Lv can be found at:

<http://www.ff.iij4u.or.jp/~nrt/lv/index.html>

mbyte-conversion

When reading and writing files in an encoding different from **'encoding'**, conversion needs to be done. These conversions are supported:

- All conversions between Latin-1 (ISO-8859-1), UTF-8, UCS-2 and UCS-4 are handled internally.
- For MS-Windows, when **'encoding'** is a Unicode encoding, conversion from and to any codepage should work.
- Conversion specified with **'charconvert'**
- Conversion with the iconv library, if it is available.

Old versions of GNU iconv() may cause the conversion to fail (they request a very large buffer, more than Vim is willing to provide). Try getting another iconv() implementation.

iconv-dynamic

On MS-Windows Vim can be compiled with the **+iconv/dyn** feature. This means Vim will search for the "iconv.dll" and "libiconv.dll" libraries. When neither of them can be found Vim will still work but some conversions won't be possible.

4. Using a terminal

mbyte-terminal

The GUI fully supports multibyte characters. It is also possible in a terminal, if the terminal supports the same encoding that Vim uses. Thus this is less flexible.

For example, you can run Vim in a xterm with added multibyte support and/or `XIM`. Examples are `kterm` (Kanji term) and `hanterm` (for Korean), `Eterm` (Enlightened terminal) and `rxvt`.

If your terminal does not support the right encoding, you can set the `'termencoding'` option. Vim will then convert the typed characters from `'termencoding'` to `'encoding'`. And displayed text will be converted from `'encoding'` to `'termencoding'`. If the encoding supported by the terminal doesn't include all the characters that Vim uses, this leads to lost characters. This may mess up the display. If you use a terminal that supports Unicode, such as the xterm mentioned below, it should work just fine, since nearly every character set can be converted to Unicode without loss of information.

UTF-8 IN XFREE86 XTERM

UTF8-xterm

This is a short explanation of how to use UTF-8 character encoding in the xterm that comes with XFree86 by Thomas Dickey (text by Markus Kuhn).

Get the latest xterm version which has now UTF-8 support:

<http://invisible-island.net/xterm/xterm.html>

Compile it with `./configure --enable-wide-chars ; make`

Also get the ISO 10646-1 version of various fonts, which is available on

<http://www.cl.cam.ac.uk/~mgk25/download/ucs-fonts.tar.gz>

and install the font as described in the README file.

Now start xterm with

```
xterm -u8 -fn -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1  
or, for bigger character:
```

```
xterm -u8 -fn -misc-fixed-medium-r-normal--15-140-75-75-c-90-iso10646-1
```

and you will have a working UTF-8 terminal emulator. Try both

```
cat utf-8-demo.txt  
vim utf-8-demo.txt
```

with the demo text that comes with `ucs-fonts.tar.gz` in order to see whether there are any problems with UTF-8 in your xterm.

For Vim you may need to set **'encoding'** to "utf-8".

5. Fonts on X11

mbyte-fonts-X11

Unfortunately, using fonts in X11 is complicated. The name of a single-byte font is a long string. For multibyte fonts we need several of these...

Note: Most of this is no longer relevant for GTK+ 2. Selecting a font via its XLFD is not supported; see **'guifont'** for an example of how to set the font. Do yourself a favor and ignore the **XLFD** and **xfontset** sections below.

First of all, Vim only accepts fixed-width fonts for displaying text. You cannot use proportionally spaced fonts. This excludes many of the available (and nicer looking) fonts. However, for menus and tooltips any font can be used.

Note that Display and Input are independent. It is possible to see your language even though you have no input method for it.

You should get a default font for menus and tooltips that works, but it might be ugly. Read the following to find out how to select a better font.

X LOGICAL FONT DESCRIPTION (XLFD)

XLFD

XLFD is the X font name and contains the information about the font size, charset, etc. The name is in this format:

FOUNDRY-FAMILY-WEIGHT-SLANT-WIDTH-STYLE-PIXEL-POINT-X-Y-SPACE-AVE-CR-CE

Each field means:

- FOUNDRY: FOUNDRY field. The company that created the font.
- FAMILY: FAMILY_NAME field. Basic font family name. (helvetica, gothic, times, etc)
- WEIGHT: WEIGHT_NAME field. How thick the letters are. (light, medium, bold, etc)
- SLANT: SLANT field.
 - r: Roman (no slant)
 - i: Italic
 - o: Oblique
 - ri: Reverse Italic
 - ro: Reverse Oblique
 - ot: Other
 - number: Scaled font
- WIDTH: SETWIDTH_NAME field. Width of characters. (normal, condensed, narrow, double wide)
- STYLE: ADD_STYLE_NAME field. Extra info to describe font. (Serif, Sans Serif, Informal, Decorated, etc)
- PIXEL: PIXEL_SIZE field. Height, in pixels, of characters.
- POINT: POINT_SIZE field. Ten times height of characters in points.
- X: RESOLUTION_X field. X resolution (dots per inch).

- Y: RESOLUTION_Y field. Y resolution (dots per inch).
- SPACE: SPACING field.
 - p: Proportional
 - m: Monospaced
 - c: CharCell
- AVE: AVERAGE_WIDTH field. Ten times average width in pixels.
- CR: CHARSET_REGISTRY field. The name of the charset group.
- CE: CHARSET_ENCODING field. The rest of the charset name. For some charsets, such as JIS X 0208, if this field is 0, code points has the same value as GL, and GR if 1.

For example, in case of a 16 dots font corresponding to JIS X 0208, it is written like:

```
-misc-fixed-medium-r-normal--16-110-100-100-c-160-jisx0208.1990-0
```

X FONTSET

fontset xfontset

A single-byte charset is typically associated with one font. For multibyte charsets a combination of fonts is often used. This means that one group of characters are used from one font and another group from another font (which might be double wide). This collection of fonts is called a fontset.

Which fonts are required in a fontset depends on the current locale. X windows maintains a table of which groups of characters are required for a locale. You have to specify all the fonts that a locale requires in the **'guifontset'** option.

Setting the **'guifontset'** option also means that all font names will be handled as a fontset name. Also the ones used for the "font" argument of the **:highlight** command.

Note the difference between **'guifont'** and **'guifontset'**: In **'guifont'** the comma-separated names are alternative names, one of which will be used. In **'guifontset'** the whole string is one fontset name, including the commas. It is not possible to specify alternative fontset names.

This example works on many X11 systems:

```
:set guifontset=-*-medium-r-normal--16-*-*-c-*-*
```

The fonts must match with the current locale. If fonts for the character sets that the current locale uses are not included, setting **'guifontset'** will fail.

NOTE: The fontset always uses the current locale, even though **'encoding'** may be set to use a different charset. In that situation you might want to use **'guifont'** and **'guifontwide'** instead of **'guifontset'**.

Example:

 charset 	language	"groups of characters"
GB2312	Chinese (simplified)	ISO-8859-1 and GB 2312
Big5	Chinese (traditional)	ISO-8859-1 and Big5
CNS-11643	Chinese (traditional)	ISO-8859-1, CNS 11643-1 and CNS 11643-2
EUC-JP	Japanese	JIS X 0201 and JIS X 0208
EUC-KR	Korean	ISO-8859-1 and KS C 5601 (KS X 1001)

You can search for fonts using the `xlsfonts` command. For example, when you're searching for a font for KS C 5601:

```
xlsfonts | grep ksc5601
```

This is complicated and confusing. You might want to consult the X-Windows documentation if there is something you don't understand.

base_font_name_list

When you have found the names of the fonts you want to use, you need to set the `'guifontset'` option. You specify the list by concatenating the font names and putting a comma in between them.

For example, when you use the `ja_JP.eucJP` locale, this requires JIS X 0201 and JIS X 0208. You could supply a list of fonts that explicitly specifies the charsets, like:

```
:set guifontset=-misc-fixed-medium-r-normal--14-130-75-75-c-140-jisx0208.1983-0,  
 \-misc-fixed-medium-r-normal--14-130-75-75-c-70-jisx0201.1976-0
```

Alternatively, you can supply a base font name list that omits the charset name, letting X-Windows select font characters required for the locale. For example:

```
:set guifontset=-misc-fixed-medium-r-normal--14-130-75-75-c-140,  
 \-misc-fixed-medium-r-normal--14-130-75-75-c-70
```

Alternatively, you can supply a single base font name that allows X-Windows to select from all available fonts. For example:

```
:set guifontset=-misc-fixed-medium-r-normal--14-*
```

Alternatively, you can specify alias names. See the `fonts.alias` file in the fonts directory (e.g., `/usr/X11R6/lib/X11/fonts/`). For example:

```
:set guifontset=k14,r14
```

E253

Note that in East Asian fonts, the standard character cell is square. When mixing a Latin font and an East Asian font, the East Asian font width should be twice the Latin font width.

If `'guifontset'` is not empty, the "font" argument of the `:highlight` command is also interpreted as a fontset. For example, you should use for highlighting:

```
:hi Comment font=english_font,your_font
```

If you use a wrong "font" argument you will get an error message.

Also make sure that you set `'guifontset'` before setting fonts for highlight groups.

USING RESOURCE FILES

Instead of specifying `'guifontset'`, you can set X11 resources and Vim will

pick them up. This is only for people who know how X resource files work.

For Motif insert these three lines in your \$HOME/.Xdefaults file:

```
Vim.font: base_font_name_list
Vim*fontSet: base_font_name_list
Vim*fontList: your_language_font
```

Note: Vim.font is for text area.
Vim*fontSet is for menu.
Vim*fontList is for menu (for Motif GUI)

For example, when you are using Japanese and a 14 dots font,

```
Vim.font: -misc-fixed-medium-r-normal--14-*
Vim*fontSet: -misc-fixed-medium-r-normal--14-*
Vim*fontList: -misc-fixed-medium-r-normal--14-*
```

or:

```
Vim*font: k14,r14
Vim*fontSet: k14,r14
Vim*fontList: k14,r14
```

To have them take effect immediately you will have to do

```
xrdb -merge ~/.Xdefaults
```

Otherwise you will have to stop and restart the X server before the changes take effect.

The GTK+ version of GUI Vim does not use .Xdefaults, use ~/.gtkrc instead. The default mostly works OK. But for the menus you might have to change it. Example:

```
style "default"
{
    fontset="-*-*-medium-r-normal--14-*-*-*c-*-*-*"
}
widget_class "*" style "default"
```

6. Fonts on MS-Windows

mbyte-fonts-MSwin

The simplest is to use the font dialog to select fonts and try them out. You can find this at the "Edit/Select Font..." menu. Once you find a font name that works well you can use this command to see its name:

```
:set guifont
```

Then add a command to your `gvimrc` file to set **'guifont'**:

```
:set guifont=courier_new:h12
```

7. Input on X11

mbyte-XIM

X INPUT METHOD (XIM) BACKGROUND

XIM xim x-input-method

XIM is an international input module for X. There are two kinds of structures, Xlib unit type and `IM-server` (Input-Method server) type. `IM-server` type is suitable for complex input, such as CJK.

- IM-server

`IM-server`

In `IM-server` type input structures, the input event is handled by either of the two ways: FrontEnd system and BackEnd system. In the FrontEnd system, input events are snatched by the `IM-server` first, then `IM-server` give the application the result of input. On the other hand, the BackEnd system works reverse order. MS-Windows adopt BackEnd system. In X, most of `IM-server` s adopt FrontEnd system. The demerit of BackEnd system is the large overhead in communication, but it provides safe synchronization with no restrictions on applications.

For example, there are `xwnmo` and `kinput2` Japanese `IM-server` , both are FrontEnd system. `xwnmo` is distributed with `Wnn` (see below), `kinput2` can be found at: <ftp://ftp.sra.co.jp/pub/x11/kinput2/>

For Chinese, there's a great XIM server named "xcin", you can input both Traditional and Simplified Chinese characters. And it can accept other locale if you make a correct input table. `Xcin` can be found at:

<http://cle.linux.org.tw/xcin/>

Others are `scim`: <http://scim.freedesktop.org/> and `fcitx`:

<http://www.fcitx.org/>

- Conversion Server

`conversion-server`

Some system needs additional server: conversion server. Most of Japanese `IM-server` s need it, Kana-Kanji conversion server. For Chinese inputting, it depends on the method of inputting, in some methods, PinYin or ZhuYin to HanZi conversion server is needed. For Korean inputting, if you want to input Hanja, Hangul-Hanja conversion server is needed.

For example, the Japanese inputting process is divided into 2 steps. First we pre-input Hira-gana, second Kana-Kanji conversion. There are so many Kanji characters (6349 Kanji characters are defined in JIS X 0208) and the number of Hira-gana characters are 76. So, first, we pre-input text as pronounced in Hira-gana, second, we convert Hira-gana to Kanji or Kata-Kana, if needed. There are some Kana-Kanji conversion server: `jserver`

(distributed with `Wnn`, see below) and `canna`. `Canna` can be found at:

<http://canna.sourceforge.jp/>

There is a good input system: `Wnn4.2`. `Wnn 4.2` contains,

`xwnmo` (`IM-server`)

`jserver` (Japanese Kana-Kanji conversion server)

`cserver` (Chinese PinYin or ZhuYin to simplified HanZi conversion server)

`tserver` (Chinese PinYin or ZhuYin to traditional HanZi conversion server)

kserver (Hangul-Hanja conversion server)

Wnn 4.2 for several systems can be found at various places on the internet. Use the RPM or port for your system.

- Input Style

`xim-input-style`

When inputting CJK, there are four areas:

1. The area to display of the input while it is being composed
2. The area to display the currently active input mode.
3. The area to display the next candidate for the selection.
4. The area to display other tools.

The third area is needed when converting. For example, in Japanese inputting, multiple Kanji characters could have the same pronunciation, so a sequence of Hira-gana characters could map to a distinct sequence of Kanji characters.

The first and second areas are defined in international input of X with the names of "Preedit Area", "Status Area" respectively. The third and fourth areas are not defined and are left to be managed by the `IM-server`. In the international input, four input styles have been defined using combinations of Preedit Area and Status Area: `OnTheSpot`, `OffTheSpot`, `OverTheSpot` and `Root`.

Currently, GUI Vim supports three styles, `OverTheSpot`, `OffTheSpot` and `Root`.

When compiled with `+GUI_GTK` feature, GUI Vim supports two styles, `OnTheSpot` and `OverTheSpot`. You can select the style with the `'imstyle'` option.

- *. `on-the-spot` `OnTheSpot`
Preedit Area and Status Area are performed by the client application in the area of application. The client application is directed by the `IM-server` to display all pre-edit data at the location of text insertion. The client registers callbacks invoked by the input method during pre-editing.
- *. `over-the-spot` `OverTheSpot`
Status Area is created in a fixed position within the area of application, in case of Vim, the position is the additional status line. Preedit Area is made at present input position of application. The input method displays pre-edit data in a window which it brings up directly over the text insertion position.
- *. `off-the-spot` `OffTheSpot`
Preedit Area and Status Area are performed in the area of application, in case of Vim, the area is additional status line. The client application provides display windows for the pre-edit data to the input method which displays into them directly.
- *. `root-window` `Root`
Preedit Area and Status Area are outside of the application. The input method displays all pre-edit data in a separate area of the screen in a window specific to the input method.

USING XIM

multibyte-input E284 E285 E286 E287
E288 E289

Note that Display and Input are independent. It is possible to see your language even though you have no input method for it. But when your Display method doesn't match your Input method, the text will be displayed wrong.

Note: You can not use IM unless you specify **'guifontset'**.
Therefore, Latin users, you have to also use **'guifontset'**
if you use IM.

To input your language you should run the `IM-server` which supports your language and `conversion-server` if needed.

The next 3 lines should be put in your `~/.Xdefaults` file. They are common for all X applications which uses `XIM`. If you already use `XIM`, you can skip this.

```
*international: True
*.inputMethod: your_input_server_name
*.preeditType: your_input_style
```

`input_server_name` is your `IM-server` name (check your `IM-server` manual).
`your_input_style` is one of `OverTheSpot`, `OffTheSpot`, `Root`. See also `xim-input-style`.

`*international` may not be necessary if you use X11R6.
`*.inputMethod` and `*.preeditType` are optional if you use X11R6.

For example, when you are using `kinput2` as `IM-server`,

```
*international: True
*.inputMethod: kinput2
*.preeditType: OverTheSpot
```

When using `OverTheSpot`, GUI Vim always connects to the IM Server even in Normal mode, so you can input your language with commands like "f" and "r". But when using one of the other two methods, GUI Vim connects to the IM Server only if it is not in Normal mode.

If your IM Server does not support `OverTheSpot`, and if you want to use your language with some Normal mode command like "f" or "r", then you should use a localized xterm or an xterm which supports `XIM`

If needed, you can set the `XMODIFIERS` environment variable:

```
sh: export XMODIFIERS="@im=input_server_name"
csh: setenv XMODIFIERS "@im=input_server_name"
```

For example, when you are using `kinput2` as `IM-server` and sh,

```
export XMODIFIERS="@im=kinput2"
```

FULLY CONTROLLED XIM

You can fully control XIM, like with IME of MS-Windows (see [multibyte-ime](#)). This is currently only available for the GTK GUI.

Before using fully controlled XIM, one setting is required. Set the **'imactivatekey'** option to the key that is used for the activation of the input method. For example, when you are using kinput2 + canna as IM Server, the activation key is probably Shift+Space:

```
:set imactivatekey=S-space
```

See **'imactivatekey'** for the format.

8. Input on MS-Windows

mbyte-IME

(Windows IME support)

multibyte-ime IME

{only works Windows GUI and compiled with the |+multi_byte_ime| feature}

To input multibyte characters on Windows, you can use an Input Method Editor (IME). In process of your editing text, you must switch status (on/off) of IME many many many times. Because IME with status on is hooking all of your key inputs, you cannot input 'j', 'k', or almost all of keys to Vim directly.

The [+multi_byte_ime](#) feature helps for this. It reduces the number of times the IME status has to be switched manually. In Normal mode, there is almost no need to use IME, even when editing multibyte text. So when exiting Insert mode, Vim memorizes the last status of IME and turns off IME. When re-entering Insert mode, Vim sets the IME status to that memorized status automatically.

This works on not only insert-normal mode, but also search-command input and replace mode.

The options **'iminsert'**, **'imsearch'** and **'imcmdline'** can be used to choose the different input methods or disable them temporarily.

On Windows 9x and Windows NT 4.0 there was [global-ime](#) , but this is no longer supported. You can still find documentation for Active Input Method Manager (Global IME) here:

[http://msdn.microsoft.com/en-us/library/aa741221\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa741221(v=VS.85).aspx)

NOTE: For IME to work you must make sure the input locales of your language are added to your system. The exact location of this depends on the version of Windows you use. For example, on my Windows 2000 box:

1. Control Panel
2. Regional Options
3. Input Locales Tab
4. Add Installed input locales -> Chinese(PRC)
The default is still English (United Stated)

Cursor color when IME or XIM is on

CursorIM

There is a little cute feature for IME. Cursor can indicate status of IME by changing its color. Usually status of IME was indicated by little icon at a corner of desktop (or taskbar). It is not easy to verify status of IME. But this feature help this.

This works in the same way when using XIM.

You can select cursor color when status is on by using highlight group CursorIM. For example, add these lines to your `gvimrc` :

```
if has('multi_byte_ime')
    highlight Cursor guifg=NONE guibg=Green
    highlight CursorIM guifg=NONE guibg=Purple
endif
```

Cursor color with off IME is green. And purple cursor indicates that status is on.

9. Input with a keymap

mbyte-keymap

When the keyboard doesn't produce the characters you want to enter in your text, you can use the **'keymap'** option. This will translate one or more (English) characters to another (non-English) character. This only happens when typing text, not when typing Vim commands. This avoids having to switch between two keyboard settings.

{only available when compiled with the |+keymap| feature}

The value of the **'keymap'** option specifies a keymap file to use. The name of this file is one of these two:

```
keymap/{keymap}_{encoding}.vim
keymap/{keymap}.vim
```

Here {keymap} is the value of the **'keymap'** option and {encoding} of the **'encoding'** option. The file name with the {encoding} included is tried first.

'runtimepath' is used to find these files. To see an overview of all available keymap files, use this:

```
:echo globpath(&rtp, "keymap/*.vim")
```

In Insert and Command-line mode you can use **CTRL-^** to toggle between using the keyboard map or not. `i_CTRL-^` `c_CTRL-^`

This flag is remembered for Insert mode with the **'iminsert'** option. When leaving and entering Insert mode the previous value is used. The same value is also used for commands that take a single character argument, like `f` and `r`.

For Command-line mode the flag is NOT remembered. You are expected to type an Ex command first, which is ASCII.

For typing search patterns the **'imsearch'** option is used. It can be set to use the same value as for **'iminsert'**.

lCursor

It is possible to give the GUI cursor another color when the language mappings are being used. This is disabled by default, to avoid that the cursor becomes

invisible when you use a non-standard background color. Here is an example to use a brightly colored cursor:

```
:highlight Cursor guifg=NONE guibg=Green
:highlight lCursor guifg=NONE guibg=Cyan
```

```
keymap-file-format :loadk :loadkeymap E105 E791
```

The keymap file looks something like this:

```
" Maintainer:  name <email@address>
" Last Changed: 2001 Jan 1

let b:keymap_name = "short"

loadkeymap
a      A
b      B      comment
```

The lines starting with a " are comments and will be ignored. Blank lines are also ignored. The lines with the mappings may have a comment after the useful text.

The "b:keymap_name" can be set to a short name, which will be shown in the status line. The idea is that this takes less room than the value of **'keymap'**, which might be long to distinguish between different languages, keyboards and encodings.

The actual mappings are in the lines below "loadkeymap". In the example "a" is mapped to "A" and "b" to "B". Thus the first item is mapped to the second item. This is done for each line, until the end of the file.

These items are exactly the same as what can be used in a `:lnoremap` command, using "<buffer>" to make the mappings local to the buffer.

You can check the result with this command:

```
:lmap
```

The two items must be separated by white space. You cannot include white space inside an item, use the special names "<Tab>" and "<Space>" instead. The length of the two items together must not exceed 200 bytes.

It's possible to have more than one character in the first column. This works like a dead key. Example:

```
'a      á
```

Since Vim doesn't know if the next character after a quote is really an "a", it will wait for the next character. To be able to insert a single quote, also add this line:

```
''      '
```

Since the mapping is defined with `:lnoremap` the resulting quote will not be used for the start of another character.

The "accents" keymap uses this.

[keymap-accents](#)

The first column can also be in `<>` form:

```
<C-c>      Ctrl-C
<A-c>      Alt-c
<A-C>      Alt-C
```

Note that the Alt mappings may not work, depending on your keyboard and terminal.

Although it's possible to have more than one character in the second column, this is unusual. But you can use various ways to specify the character:

```
A      a      literal character
A      <char-97>  decimal value
A      <char-0x61> hexadecimal value
A      <char-0141> octal value
x      <Space>   special key name
```

The characters are assumed to be encoded for the current value of **'encoding'**. It's possible to use `":scriptencoding"` when all characters are given literally. That doesn't work when using the `<char->` construct, because the conversion is done on the keymap file, not on the resulting character.

The lines after `"loadkeymap"` are interpreted with **'cptions'** set to `"C"`. This means that continuation lines are not used and a backslash has a special meaning in the mappings. Examples:

```
" a comment line
\"      x      maps " to x
\\     y      maps \ to y
```

If you write a keymap file that will be useful for others, consider submitting it to the Vim maintainer for inclusion in the distribution:

[<maintainer@vim.org>](mailto:maintainer@vim.org)

HEBREW KEYMAP

[keymap-hebrew](#)

This file explains what characters are available in UTF-8 and CP1255 encodings, and what the keymaps are to get those characters:

glyph	encoding		keymap		
Char	UTF-8	cp1255	hebrew	hebrewp	name
⌘	0x5d0	0xe0	t	a	'alef
⌘	0x5d1	0xe1	c	b	bet
⌘	0x5d2	0xe2	d	g	gimel
⌘	0x5d3	0xe3	s	d	dalet
⌘	0x5d4	0xe4	v	h	he
⌘	0x5d5	0xe5	u	v	vav
⌘	0x5d6	0xe6	z	z	zayin
⌘	0x5d7	0xe7	j	j	het
⌘	0x5d8	0xe8	y	T	tet
⌘	0x5d9	0xe9	h	y	yod
⌘	0x5da	0xea	l	K	kaf sofit
⌘	0x5db	0xeb	f	k	kaf
⌘	0x5dc	0xec	k	l	lamed
⌘	0x5dd	0xed	o	M	mem sofit
⌘	0x5de	0xee	n	m	mem
⌘	0x5df	0xef	i	N	nun sofit
⌘	0x5e0	0xf0	b	n	nun
⌘	0x5e1	0xf1	x	s	samech
⌘	0x5e2	0xf2	g	u	`ayin
⌘	0x5e3	0xf3	;	P	pe sofit

☒	0x5e4	0xf4	p	p	pe
☒	0x5e5	0xf5	.	X	tsadi sofit
☒	0x5e6	0xf6	m	x	tsadi
☒	0x5e7	0xf7	e	q	qof
☒	0x5e8	0xf8	r	r	resh
☒	0x5e9	0xf9	a	w	shin
☒	0x5ea	0xfa	,	t	tav

Vowel marks and special punctuation:

☒☒	0x5b0	0xc0	A:	A:	sheva
☒☒	0x5b1	0xc1	HE	HE	hataf segol
☒☒	0x5b2	0xc2	HA	HA	hataf patah
☒☒	0x5b3	0xc3	HO	HO	hataf qamats
☒☒	0x5b4	0xc4	I	I	hiriq
☒☒	0x5b5	0xc5	AY	AY	tsere
☒☒	0x5b6	0xc6	E	E	segol
☒☒	0x5b7	0xc7	AA	AA	patah
☒☒	0x5b8	0xc8	AO	AO	qamats
☒☒	0x5b9	0xc9	O	O	holam
☒☒	0x5bb	0xcb	U	U	qubuts
☒☒	0x5bc	0xcc	D	D	dagesh
☒☒	0x5bd	0xcd]T]T	meteg
☒☒	0x5be	0xce]Q]Q	maqaf
☒☒	0x5bf	0xcf]R]R	rafe
☒☒	0x5c0	0xd0]p]p	paseq
☒☒	0x5c1	0xd1	SR	SR	shin-dot
☒☒	0x5c2	0xd2	SL	SL	sin-dot
☒	0x5c3	0xd3]P]P	sof-pasuq
☒	0x5f0	0xd4	VV	VV	double-vav
☒	0x5f1	0xd5	VY	VY	vav-yod
☒	0x5f2	0xd6	YY	YY	yod-yod

The following are only available in UTF-8

Cantillation marks:

glyph

Char UTF-8 hebrew name

☒☒	0x591	C:	etnahta
☒☒	0x592	Cs	segol
☒☒	0x593	CS	shalsholet
☒☒	0x594	Cz	zaqef qatan
☒☒	0x595	CZ	zaqef gadol
☒☒	0x596	Ct	tipeha
☒☒	0x597	Cr	revia
☒☒	0x598	Cq	zarqa
☒☒	0x599	Cp	pashta
☒☒	0x59a	C!	yetiv
☒☒	0x59b	Cv	tevir
☒☒	0x59c	Cg	geresh
☒☒	0x59d	C*	geresh qadim
☒☒	0x59e	CG	gershayim
☒☒	0x59f	CP	qarnei-parah
☒☒	0x5aa	Cy	yerach-ben-yomo
☒☒	0x5ab	Co	ole

☒	0x5ac	Ci	iluy
☒	0x5ad	Cd	dehi
☒	0x5ae	Cn	zinor
☒	0x5af	CC	masora circle

Combining forms:

☒	0xfb20	X`	Alternative `ayin
☒	0xfb21	X'	Alternative 'alef
☒	0xfb22	X-d	Alternative dalet
☒	0xfb23	X-h	Alternative he
☒	0xfb24	X-k	Alternative kaf
☒	0xfb25	X-l	Alternative lamed
☒	0xfb26	X-m	Alternative mem-sofit
☒	0xfb27	X-r	Alternative resh
☒	0xfb28	X-t	Alternative tav
☒	0xfb29	X-+	Alternative plus
☒	0xfb2a	XW	shin+shin-dot
☒	0xfb2b	Xw	shin+sin-dot
☒	0xfb2c	X..W	shin+shin-dot+dagesh
☒	0xfb2d	X..w	shin+sin-dot+dagesh
☒	0xfb2e	XA	alef+patah
☒	0xfb2f	X0	alef+qamats
☒	0xfb30	XI	alef+hiriq (mapiq)
☒	0xfb31	X.b	bet+dagesh
☒	0xfb32	X.g	gimel+dagesh
☒	0xfb33	X.d	dalet+dagesh
☒	0xfb34	X.h	he+dagesh
☒	0xfb35	Xu	vav+dagesh
☒	0xfb36	X.z	zayin+dagesh
☒	0xfb38	X.T	tet+dagesh
☒	0xfb39	X.y	yud+dagesh
☒	0xfb3a	X.K	kaf sofit+dagesh
☒	0xfb3b	X.k	kaf+dagesh
☒	0xfb3c	X.l	lamed+dagesh
☒	0xfb3e	X.m	mem+dagesh
☒	0xfb40	X.n	nun+dagesh
☒	0xfb41	X.s	samech+dagesh
☒	0xfb43	X.P	pe sofit+dagesh
☒	0xfb44	X.p	pe+dagesh
☒	0xfb46	X.x	tsadi+dagesh
☒	0xfb47	X.q	qof+dagesh
☒	0xfb48	X.r	resh+dagesh
☒	0xfb49	X.w	shin+dagesh
☒	0xfb4a	X.t	tav+dagesh
☒	0xfb4b	Xo	vav+holam
☒	0xfb4c	XRb	bet+rafe
☒	0xfb4d	XRk	kaf+rafe
☒	0xfb4e	XRp	pe+rafe
☒	0xfb4f	Xal	alef-lamed

10. Input with `imactivatefunc()`

`mbyte-func`

Vim has the `'imactivatefunc'` and `'imstatusfunc'` options. These are useful to

activate/deactivate the input method from Vim in any way, also with an external command. For example, fcitx provide fcitx-remote command:

```
set iminsert=2
set imsearch=2
set imcmdline

set imactivatefunc=ImActivate
function! ImActivate(active)
  if a:active
    call system('fcitx-remote -o')
  else
    call system('fcitx-remote -c')
  endif
endfunction

set imstatusfunc=ImStatus
function! ImStatus()
  return system('fcitx-remote')[0] is# '2'
endfunction
```

Using this script, you can activate/deactivate XIM via Vim even when it is not compiled with `+xim` .

11. Using UTF-8 mbyte-utf8 UTF-8 utf-8 utf8 Unicode unicode

The Unicode character set was designed to include all characters from other character sets. Therefore it is possible to write text in any language using Unicode (with a few rarely used languages excluded). And it's mostly possible to mix these languages in one file, which is impossible with other encodings.

Unicode can be encoded in several ways. The most popular one is UTF-8, which uses one or more bytes for each character and is backwards compatible with ASCII. On MS-Windows UTF-16 is also used (previously UCS-2), which uses 16-bit words. Vim can support all of these encodings, but always uses UTF-8 internally.

Vim has comprehensive UTF-8 support. It works well in:

- xterm with UTF-8 support enabled
- Motif and GTK GUI
- MS-Windows GUI
- several other platforms

Double-width characters are supported. This works best with `'guifontwide'` or `'guifontset'`. When using only `'guifont'` the wide characters are drawn in the normal width and a space to fill the gap. **Note** that the `'guifontset'` option is no longer relevant in the GTK+ 2 GUI.

bom-bytes

When reading a file a BOM (Byte Order Mark) can be used to recognize the Unicode encoding:

```
EF BB BF UTF-8
FE FF UTF-16 big endian
```

```
FF FE      UTF-16 little endian
00 00 FE FF UTF-32 big endian
FF FE 00 00 UTF-32 little endian
```

UTF-8 is the recommended encoding. **Note** that it's difficult to tell utf-16 and utf-32 apart. Utf-16 is often used on MS-Windows, utf-32 is not widespread as file format.

mbyte-combining mbyte-composing

A composing or combining character is used to change the meaning of the character before it. The combining characters are drawn on top of the preceding character.

Up to two combining characters can be used by default. This can be changed with the **'maxcombine'** option.

When editing text a composing character is mostly considered part of the preceding character. For example "x" will delete a character and its following composing characters by default.

If the **'delcombine'** option is on, then pressing 'x' will delete the combining characters, one at a time, then the base character. But when inserting, you type the first character and the following composing characters separately, after which they will be joined. The "r" command will not allow you to type a combining character, because it doesn't know one is coming. Use "R" instead.

Bytes which are not part of a valid UTF-8 byte sequence are handled like a single character and displayed as <xx>, where "xx" is the hex value of the byte.

Overlong sequences are not handled specially and displayed like a valid character. However, search patterns may not match on an overlong sequence. (an overlong sequence is where more bytes are used than required for the character.) An exception is NUL (zero) which is displayed as "<00>".

In the file and buffer the full range of Unicode characters can be used (31 bits). However, displaying only works for the characters present in the selected font.

Useful commands:

- "ga" shows the decimal, hexadecimal and octal value of the character under the cursor. If there are composing characters these are shown too. (If the message is truncated, use ":messages").
- "g8" shows the bytes used in a UTF-8 character, also the composing characters, as hex numbers.
- ":set encoding=utf-8 fileencodings=" forces using UTF-8 for all files. The default is to use the current locale for **'encoding'** and set **'fileencodings'** to automatically detect the encoding of a file.

STARTING VIM

If your current locale is in an UTF-8 encoding, Vim will automatically start in UTF-8 mode.

If you are using another locale:

```
set encoding=utf-8
```

You might also want to select the font used for the menus. Unfortunately this doesn't always work. See the system specific remarks below, and ['langmenu'](#).

USING UTF-8 IN X-Windows

[utf-8-in-xwindows](#)

Note: This section does not apply to the GTK+ 2 GUI.

You need to specify a font to be used. For double-wide characters another font is required, which is exactly twice as wide. There are three ways to do this:

1. Set ['guifont'](#) and let Vim find a matching ['guifontwide'](#)
2. Set ['guifont'](#) and ['guifontwide'](#)
3. Set ['guifontset'](#)

See the documentation for each option for details. Example:

```
:set guifont=-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso10646-1
```

You might also want to set the font used for the menus. This only works for Motif. Use the `":hi Menu font={fontname}"` command for this. [:highlight](#)

TYPING UTF-8

[utf-8-typing](#)

If you are using X-Windows, you should find an input method that supports UTF-8.

If your system does not provide support for typing UTF-8, you can use the ['keymap'](#) feature. This allows writing a keymap file, which defines a UTF-8 character as a sequence of ASCII characters. See [mbyte-keymap](#).

Another method is to set the current locale to the language you want to use and for which you have a XIM available. Then set ['termencoding'](#) to that language and Vim will convert the typed characters to ['encoding'](#) for you.

If everything else fails, you can type any character as four hex bytes:

```
CTRL-V u 1234
```

"1234" is interpreted as a hex number. You must type four characters, prepend a zero if necessary.

COMMAND ARGUMENTS

[utf-8-char-arg](#)

Commands like `f`, `F`, `t` and `r` take an argument of one character. For UTF-8 this argument may include one or two composing characters. These need to be produced together with the base character, Vim doesn't wait for the next character to be typed to find out if it is a composing character or not.

Using **'keymap'** or `:lmap` is a nice way to type these characters.

The commands that search for a character in a line handle composing characters as follows. When searching for a character without a composing character, this will find matches in the text with or without composing characters. When searching for a character with a composing character, this will only find matches with that composing character. It was implemented this way, because not everybody is able to type a composing character.

12. Overview of options

mbyte-options

These options are relevant for editing multibyte files. Check the help in options.txt for detailed information.

- 'encoding'** Encoding used for the keyboard and display. It is also the default encoding for files.
- 'fileencoding'** Encoding of a file. When it's different from **'encoding'** conversion is done when reading or writing the file.
- 'fileencodings'** List of possible encodings of a file. When opening a file these will be tried and the first one that doesn't cause an error is used for **'fileencoding'**.
- 'charconvert'** Expression used to convert files from one encoding to another.
- 'formatoptions'** The 'm' flag can be included to have formatting break a line at a multibyte character of 256 or higher. Thus is useful for languages where a sequence of characters can be broken anywhere.
- 'guifontset'** The list of font names used for a multibyte encoding. When this option is not empty, it replaces **'guifont'**.
- 'keymap'** Specify the name of a keyboard mapping.

Contributions specifically for the multibyte features by:

Chi-Deok Hwang <hwang@mizi.co.kr>
SungHyun Nam <goweol@gmail.com>
K.Nagano <nagano@atese.advantest.co.jp>
Taro Muraoka <koron@tka.att.ne.jp>
Yasuhiro Matsumoto <mattn@mail.goo.ne.jp>

Multi-language features

[multilang](#) [multi-lang](#)

This is about using messages and menus in various languages. For editing multibyte text see [multibyte](#) .

The basics are explained in the user manual: [usr_45.txt](#) .

- | | |
|-------------|------------------------------------|
| 1. Messages | multilang-messages |
| 2. Menus | multilang-menus |
| 3. Scripts | multilang-scripts |

Also see [help-translated](#) for multi-language help.

{not available when compiled without the |+multi_lang| feature}

=====

1. Messages

[multilang-messages](#)

Vim picks up the locale from the environment. In most cases this means Vim will use the language that you prefer, unless it's not available.

To see a list of supported locale names on your system, look in one of these directories (for Unix):

[/usr/lib/locale](#)
[/usr/share/locale](#)

Unfortunately, upper/lowercase differences matter. Also watch out for the use of "-" and "_".

[:lan](#) [:lang](#) [:language](#) E197

```
:lan[guage]
:lan[guage] mes[sages]
:lan[guage] cty[pe]
:lan[guage] tim[e]
:lan[guage] col[late]
```

Print the current language (aka locale).
 With the "messages" argument the language used for messages is printed. Technical: LC_MESSAGES.
 With the "ctype" argument the language used for character encoding is printed. Technical: LC_CTYPE.
 With the "time" argument the language used for strftime() is printed. Technical: LC_TIME.
 With the "collate" argument the language used for collation order is printed. Technical: LC_COLLATE.
 Without argument all parts of the locale are printed (this is system dependent).
 The current language can also be obtained with the [v:lang](#) , [v:ctype](#) , [v:collate](#) and [v:lc_time](#) variables.

```
:lan[guage] {name}
:lan[guage] mes[sages] {name}
:lan[guage] cty[pe] {name}
:lan[guage] tim[e] {name}
:lan[guage] col[late] {name}
```

Set the current language (aka locale) to {name}. The locale {name} must be a valid locale on your system. Some systems accept aliases like "en" or "en_US", but some only accept the full specification like "en_US.ISO_8859-1". On Unix systems you can use this command to see what locales are supported:

```
:!locale -a
```

With the "messages" argument the language used for messages is set. This can be different when you want, for example, English messages while editing Japanese text. This sets \$LC_MESSAGES.

With the "ctype" argument the language used for character encoding is set. This affects the libraries that Vim was linked with. It's unusual to set this to a different value from 'encoding' or "C". This sets \$LC_CTYPE.

With the "time" argument the language used for time and date messages is set. This affects strftime(). This sets \$LC_TIME.

With the "collate" argument the language used for the collation order is set. This affects sorting of characters. This sets \$LC_COLLATE.

Without an argument all are set, and additionally \$LANG is set.

If available the LC_NUMERIC value will always be set to "C", so that floating point numbers use '.' as the decimal point.

This will make a difference for items that depend on the language (some messages, time and date format). Not fully supported on all systems

If this fails there will be an error message. If it succeeds there is no message. Example:

```
:language
Current language: C
:language de_DE.ISO_8859-1
:language mes
Current messages language: de_DE.ISO_8859-1
:lang mes en
```

MS-WINDOWS MESSAGE TRANSLATIONS

win32-gettext

If you used the self-installing .exe file, message translations should work already. Otherwise get the libintl.dll file if you don't have it yet:

<http://sourceforge.net/projects/gettext>

Or:

<https://mlocati.github.io/gettext-conv-windows/>

This also contains tools `xgettext`, `msgformat` and others.

`libintl.dll` should be placed in same directory as `(g)vim.exe`, or one of the directories listed in the `PATH` environment value. Vim also looks for the alternate names `"libintl-8.dll"` and `"intl.dll"`.

Message files (`vim.mo`) have to be placed in `"$VIMRUNTIME/lang/xx/LC_MESSAGES"`, where `"xx"` is the abbreviation of the language (mostly two letters).

If you write your own translations you need to generate the `.po` file and convert it to a `.mo` file. You need to get the source distribution and read the file `"src/po/README.txt"`.

To overrule the automatic choice of the language, set the `$LANG` variable to the language of your choice. use `"en"` to disable translations.

```
:let $LANG = 'ja'
```

(text for Windows by Muraoka Taro)

2. Menus

`multilang-menus`

See [45.2](#) for the basics, esp. using `'langmenu'`.

Note that if changes have been made to the menus after the translation was done, some of the menus may be shown in English. Please try contacting the maintainer of the translation and ask him to update it. You can find the name and e-mail address of the translator in `"$VIMRUNTIME/lang/menu_<lang>.vim"`.

To set the font (or fontset) to use for the menus, use the `:highlight` command. Example:

```
:highlight Menu font=k12,r12
```

ALIAS LOCALE NAMES

Unfortunately, the locale names are different on various systems, even though they are for the same language and encoding. If you do not get the menu translations you expected, check the output of this command:

```
echo v:lang
```

Now check the `"$VIMRUNTIME/lang"` directory for menu translation files that use a similar language. A difference in a `"-"` being a `"_"` already causes a file not to be found! Another common difference to watch out for is `"iso8859-1"` versus `"iso_8859-1"`. Fortunately Vim makes all names lowercase, thus you don't have to worry about case differences. Spaces are changed to underscores, to avoid having to escape them.

If you find a menu translation file for your language with a different name,

create a file in your own runtime directory to load that one. The name of that file could be:

```
~/.vim/lang/menu_<v:lang>.vim
```

Check the **'runtimepath'** option for directories which are searched. In that file put a command to load the menu file with the other name:

```
runtime lang/menu_<other_lang>.vim
```

TRANSLATING MENUS

If you want to do your own translations, you can use the `:menutrans` command, explained below. It is recommended to put the translations for one language in a Vim script. For a language that has no translation yet, please consider becoming the maintainer and make your translations available to all Vim users. Send an e-mail to the Vim maintainer maintainer@vim.org.

```
                                :menut   :menutrans   :menutranslate
:menut[ranslate] clear                Clear all menu translations.

:menut[ranslate] {english} {mylang}
                                Translate menu name {english} to {mylang}. All
                                special characters like "&" and "<Tab>" need to be
                                included. Spaces and dots need to be escaped with a
                                backslash, just like in other :menu commands.
                                Case in {english} is ignored.
```

See the `$VIMRUNTIME/lang` directory for examples.

To try out your translations you first have to remove all menus. This is how you can do it without restarting Vim:

```
:source $VIMRUNTIME/delmenu.vim
:source <your-new-menu-file>
:source $VIMRUNTIME/menu.vim
```

Each part of a menu path is translated separately. The result is that when "Help" is translated to "Hilfe" and "Overview" to "Überblick" then "Help.Overview" will be translated to "Hilfe.Überblick".

3. Scripts

multilang-scripts

In Vim scripts you can use the `v:lang` variable to get the current language (locale). The default value is "C" or comes from the `$LANG` environment variable.

The following example shows how this variable is used in a simple way, to make a message adapt to language preferences of the user,

```
:if v:lang =~ "de_DE"
:  echo "Guten Morgen"
```

```
:else  
: echo "Good morning"  
:endif
```


Right to Left display mode for Vim rileft

These functions were originally created by Avner Lottem:

E-mail: alottem@iil.intel.com

Phone: +972-4-8307322

E26

{only available when compiled with the |+rightleft| feature}

Introduction

Some languages such as Arabic, Farsi, Hebrew (among others) require the ability to display their text from right-to-left. Files in those languages are stored conventionally and the right-to-left requirement is only a function of the display engine (per the Unicode specification). In right-to-left oriented files the characters appear on the screen from right to left.

Bidirectionality (or bidi for short) is what Unicode offers as a full solution to these languages. Bidi offers the user the ability to view both right-to-left as well as left-to-right text properly at the same time within the same window. Vim currently, due to simplicity, does not offer bidi and is merely opting to present a functional means to display/enter/use right-to-left languages. An older hybrid solution in which direction is encoded for every character (or group of characters) are not supported either as this kind of support is out of the scope of a simple addition to an existing editor (and it's not sanctioned by Unicode either).

As many people working on the code do not use the right-to-left mode, this feature may not work in some situations. If you can describe what is wrong and how it would work when fixed, please create an issue on github, see [bug-reports](#) .

Highlights

- o Editing left-to-right files as in the original Vim, no change.
- o Viewing and editing files in right-to-left windows. File orientation is per window, so it is possible to view the same file in right-to-left and left-to-right modes, simultaneously. (Useful for editing mixed files in which both right-to-left and left-to-right text exist).
- o Compatibility to the original Vim. Almost all features work in right-to-left mode (see Bugs below).

- o Backing from reverse insert mode to the correct place in the file (if possible).
- o No special terminal with right-to-left capabilities is required. The right-to-left changes are completely hardware independent.
- o Many languages use and require right-to-left support. These languages can quite easily be supported given the inclusion of their required keyboard mappings and some possible minor code change. Some of the current supported languages include - `arabic.txt` , `farsi.txt` and `hebrew.txt` .

Of Interest...

o Invocations

- + `'rightleft'` ('rl') sets window orientation to right-to-left.
- + `'delcombine'` ('deco'), boolean, if editing UTF-8 encoded languages, allows one to remove a composing character which gets superimposed on those that preceded them (some languages require this).
- + `'rightleftcmd'` ('rlc') sets the command-line within certain modes (such as search) to be utilized in right-to-left orientation as well.

o Typing backwards

`ins-reverse`

In lieu of using the full-fledged `'rightleft'` option, one can opt for reverse insertion. When the `'revins'` (reverse insert) option is set, inserting happens backwards. This can be used to type right-to-left text. When inserting characters the cursor is not moved and the text moves rightwards. A `<BS>` deletes the character under the cursor. `CTRL-W` and `CTRL-U` also work in the opposite direction. `<BS>`, `CTRL-W` and `CTRL-U` do not stop at the start of insert or end of line, no matter how the `'backspace'` option is set.

There is no reverse replace mode (yet).

If the `'showmode'` option is set, "-- REVERSE INSERT --" will be shown in the status line when reverse Insert mode is active.

o Pasting when in a rightleft window

When cutting text with the mouse and pasting it in a rightleft window the text will be reversed, because the characters come from the cut buffer from the left to the right, while inserted in the file from the right to the left. In order to avoid it, toggle `'revins'` before pasting.

Bugs

- o Does not handle `CTRL-A` and `CTRL-X` commands (add and subtract) correctly when in rightleft window.

- o Does not support reverse insert and rightleft modes on the command-line. However, functionality of the editor is not reduced, because it is possible to enter mappings, abbreviations and searches typed from the left to the right on the command-line.
- o Somewhat slower in right-to-left mode, because right-to-left motion is emulated inside Vim, not by the controlling terminal.
- o When both **'rightleft'** and **'revins'** are on: **'textwidth'** does not work. Lines do not wrap at all; you just get a single, long line.
- o There is no full bidirectionality (bidi) support.

Arabic Language support (options & mappings) for Vim

Arabic

E800

In order to use right-to-left and Arabic mapping support, it is necessary to compile Vim with the `+arabic` feature.

These functions have been created by Nadim Shaikli <nadim-at-arabeyes.org>

It is best to view this file with these settings within Vim's GUI:

```
:set encoding=utf-8
:set arabicshape
```

Introduction

Arabic is a rather demanding language in which a number of special features are required. Characters are right-to-left oriented and ought to appear as such on the screen (i.e. from right to left). Arabic also requires shaping of its characters, meaning the same character has a different visual form based on its relative location within a word (initial, medial, final or stand-alone). Arabic also requires two different forms of combining and the ability, in certain instances, to either superimpose up to two characters on top of another (composing) or the actual substitution of two characters into one (combining). Lastly, to display Arabic properly one will require not only ISO-8859-6 (U+0600-U+06FF) fonts, but will also require Presentation Form-B (U+FE70-U+FEFF) fonts both of which are subsets within a so-called ISO-10646-1 font.

The commands, prompts and help files are not in Arabic, therefore the user interface remains the standard Vi interface.

Highlights

- o Editing left-to-right files as in the original Vim hasn't changed.
- o Viewing and editing files in right-to-left windows. File orientation is per window, so it is possible to view the same file in right-to-left and left-to-right modes, simultaneously.
- o No special terminal with right-to-left capabilities is required. The right-to-left changes are completely hardware independent. Only Arabic fonts are necessary.
- o Compatible with the original Vim. Almost all features work in

right-to-left mode (there are liable to be bugs).

- o Changing keyboard mapping and reverse insert modes using a single command.
- o Toggling complete Arabic support via a single command.
- o While in Arabic mode, numbers are entered from left to right. Upon entering a none number character, that character will be inserted just into the left of the last number.
- o Arabic keymapping on the command line in reverse insert mode.
- o Proper Bidirectional functionality is possible given Vim is started within a Bidi capable terminal emulator.

Arabic Fonts

arabicfonts

Vim requires monospaced fonts of which there are many out there. Arabic requires ISO-8859-6 as well as Presentation Form-B fonts (without Form-B, Arabic will NOT be usable). It is highly recommended that users search for so-called 'ISO-10646-1' fonts. Do an Internet search or check www.arabeyes.org for further info on where to obtain the necessary Arabic fonts.

Font Installation

- o Installation of fonts for X Window systems (Unix/Linux)

Depending on your system, copy your_ARABIC_FONT file into a directory of your choice. Change to the directory containing the Arabic fonts and execute the following commands:

```
% mkfontdir
% xset +fp path_name_of_arabic_fonts_directory
```

Usage

Prior to the actual usage of Arabic within Vim, a number of settings need to be accounted for and invoked.

- o Setting the Arabic fonts

- + For Vim GUI set the '**guifont**' to your_ARABIC_FONT. This is done by entering the following command in the Vim window.

```
:set guifont=your_ARABIC_FONT
```

NOTE: the string 'your_ARABIC_FONT' is used to denote a complete

font name akin to that used in Linux/Unix systems.
(e.g. -misc-fixed-medium-r-normal--20-200-75-75-c-100-iso10646-1)

You can append the **'guifont'** set command to your .vimrc file in order to get the same above noted results. In other words, you can include `':set guifont=your_ARABIC_FONT'` to your .vimrc file.

+ Under the X Window environment, you can also start Vim with `'-fn your_ARABIC_FONT'` option.

- o Setting the appropriate character Encoding
To enable the correct Arabic encoding the following command needs to be appended,

```
:set encoding=utf-8
```

to your .vimrc file (entering the command manually into your Vim window is highly discouraged). In short, include `':set encoding=utf-8'` to your .vimrc file.

Attempts to use Arabic without UTF-8 will result the following warning message,

W17

```
Arabic requires UTF-8, do ':set encoding=utf-8'
```

- o Enable Arabic settings [short-cut]

In order to simplify and streamline things, you can either invoke Vim with the command-line option,

```
% vim -A my_utf8_arabic_file ...
```

or enable **'arabic'** via the following command within Vim

```
:set arabic
```

The two above noted possible invocations are the preferred manner in which users are instructed to proceed. Barring an enabled **'termbidi'** setting, both command options:

1. set the appropriate keymap
2. enable the deletion of a single combined pair character
3. enable `rightleft` mode
4. enable `rightleftcmd` mode (affecting the command-line)
5. enable `arabicshape` mode (do visual character alterations)

You may also append the command to your .vimrc file and simply include `':set arabic'` to it.

You are also capable of disabling Arabic support via

```
:set noarabic
```

which resets everything that the command had enabled without touching the global settings as they could affect other possible open buffers. In short the **'noarabic'** command,

1. resets to the alternate keymap
2. disables the deletion of a single combined pair character
3. disables rightleft mode

NOTE: the **'arabic'** command takes into consideration **'termbidi'** for possible external bi-directional (bidi) support from the terminal ("mlterm" for instance offers such support). **'termbidi'**, if available, is superior to rightleft support and its support is preferred due to its level of offerings. **'arabic'** when **'termbidi'** is enabled only sets the keymap.

For vertical window isolation while setting **'termbidi'** an LTR vertical separator like "l" or "⌘" may be used. It may also be hidden by changing its color to the foreground color:

```
:set fillchars=vert:l  
:hi VertSplit ctermfg=White
```

Note that this is a workaround, not a proper solution.

If, on the other hand, you'd like to be verbose and explicit and are opting not to use the **'arabic'** short-cut command, here's what is needed (i.e. if you use **':set arabic'** you can skip this section) -

+ Arabic Keymapping Activation

To activate the Arabic keymap (i.e. to remap your English/Latin keyboard to look-n-feel like a standard Arabic one), set the **'keymap'** command to "arabic". This is done by entering

```
:set keymap=arabic
```

in your Vim window. You can also append the **'keymap'** set command to your .vimrc file. In other words, you can include **':set keymap=arabic'** to your .vimrc file.

To turn toggle (or switch) your keymapping between Arabic and the default mapping (English), it is advised that users use the **'CTRL-^'** key press while in insert (or add/replace) mode. The command-line will display your current mapping by displaying an "Arabic" string next to your insertion mode (e.g. -- INSERT Arabic --) indicating your current keymap.

+ Arabic deletion of a combined pair character

By default Vim has the **'delcombine'** option disabled. This option allows the deletion of ALEF in a LAM_ALEF (LAA) combined character and still retain the LAM (i.e. it reverts to treating the combined character as its natural two characters form -- this also pertains to harakat and their combined forms). You can enable this option by entering


```
:set delcombine
```

in our Vim window. You can also append the **'delcombine'** set command to your `.vimrc` file. In other words, you can include `':set delcombine'` to your `.vimrc` file.

+ Arabic right-to-left Mode

By default Vim starts in Left-to-right mode. **'rightleft'** is the command that allows one to alter a window's orientation - that can be accomplished via,

- Toggling between left-to-right and right-to-left modes is accomplished through `':set rightleft'` and `':set norightleft'`.
- While in Left-to-right mode, enter `':set rl'` in the command line (`'rl'` is the abbreviation for `rightleft`).
- Put the `':set rl'` line in your `.vimrc` file to start Vim in right-to-left mode permanently.

+ Arabic right-to-left command-line Mode

For certain commands the editing can be done in right-to-left mode. Currently this is only applicable to search commands.

This is controlled with the **'rightleftcmd'** option. The default is "search", which means that windows in which **'rightleft'** is set will edit search commands in right-left mode. To disable this behavior,

```
:set rightleftcmd=
```

To enable right-left editing of search commands again,

```
:set rightleftcmd&
```

+ Arabic Shaping Mode

To activate the required visual characters alterations (shaping, composing, combining) which the Arabic language requires, enable the **'arabicshape'** command. This is done by entering

```
:set arabicshape
```

in our Vim window. You can also append the **'arabicshape'** set command to your `.vimrc` file. In other words, you can include `':set arabicshape'` to your `.vimrc` file.

Keymap/Keyboard

arabickeymap

The character/letter encoding used in Vim is the standard UTF-8.

It is widely discouraged that any other encoding be used or even attempted.

Note: UTF-8 is an all encompassing encoding and as such is the only supported (and encouraged) encoding with regard to Arabic (all other proprietary encodings should be discouraged and frowned upon).

o Keyboard

- + **CTRL-^** in insert/replace mode toggles between Arabic/Latin mode
- + Keyboard mapping is based on the Microsoft's Arabic keymap (the de facto standard in the Arab world):

```
+-----+
|!  |@  |#  |$  |%  |^  |&  |*  |(  |)  |_  |+  ||  |~  | |
|1  |2  |3  |4  |5  |6  |7  |8  |9  |0  |-  |=  |\  |`  | |
+-----+
|Q  |W  |E  |R  |T  |Y  |U  |I  |O  |P  |{  |<  |}  |>  |
|q  |w  |e  |r  |t  |y  |u  |i  |o  |p  |[  |]  | |
+-----+
|A  |S  |D  |[  |F  ]|G  |H  |J  |K  |L  | /  | :  | "  |
|a  |s  |d  |[  |f  ]|g  |h  |j  |k  |l  | /  | ;  | '  |
+-----+
|Z  |~  |X  |C  |{  |V  }|B  |N  |M  |'  |<  | ,  |>  | .  | ?  | |
|z  |x  |c  |v  |b  |n  |m  |'  | ,  | .  | /  | |
+-----+
```

Restrictions

- o Vim in its GUI form does not currently support Bi-directionality (i.e. the ability to see both Arabic and Latin intermixed within the same line).

Known Bugs

There is one known minor bug,

1. If you insert a haraka (e.g. Fatha (U+064E)) after a LAM (U+0644) and then insert an ALEF (U+0627), the appropriate combining will not happen due to the sandwiched haraka resulting in something that will NOT be displayed correctly.

WORK-AROUND: Don't include harakats between LAM and ALEF combos. In general, don't anticipate to see correct visual representation with regard to harakats and LAM+ALEF combined characters (even those entered after both characters). The problem noted is strictly a visual one, meaning saving such a file will contain all the appropriate info/encodings - nothing is lost.

No other bugs are known to exist.

farsi.txt

For Vim version 9.1. Last change: 2019 May 05

VIM REFERENCE MANUAL by Mortaza Ghassab Shiran

Right to Left and Farsi Mapping for Vim

farsi Farsi

E27

Farsi support has been removed in patch 8.1.0932. At that time it was outdated and unused.

If you would like to bring Farsi support back, please have a look at the old Farsi code, as it was present at the 8.1 release. It should be merged with Arabic support using the Unicode character set.

Hebrew Language support (options & mapping) for Vim hebrew

The supporting **'rightleft'** functionality was originally created by Avner Lottem. <alottem at gmail dot com> Ron Aaron <ron at ronware dot org> is currently helping support these features.

{only available when the |+rightleft| feature was enabled at compile time}

Introduction

Hebrew-specific options are **'hkmap'**, **'hkmapp'** **'keymap'**=hebrew and **'aleph'**. Hebrew-useful options are **'delcombine'**, **'allowrevins'**, **'revins'**, **'rightleft'** and **'rightleftcmd'**.

The **'rightleft'** mode reverses the display order, so characters are displayed from right to left instead of the usual left to right. This is useful primarily when editing Hebrew or other Middle-Eastern languages. See [rileft.txt](#) for further details.

Details

+ Options:

- + **'rightleft'** ('rl') sets window orientation to right-to-left. This means that the logical text 'ABC' will be displayed as 'CBA', and will start drawing at the right edge of the window, not the left edge.
- + **'hkmap'** ('hk') sets keyboard mapping to Hebrew, in insert/replace modes.
- + **'aleph'** ('al'), numeric, holds the decimal code of Aleph, for keyboard mapping.
- + **'hkmapp'** ('hkp') sets keyboard mapping to 'phonetic hebrew'

NOTE: these three ('hkmap', **'hkmapp'** and **'aleph'**) are obsolete. You should use ":set keymap=hebrewp" instead.

- + **'delcombine'** ('deco'), boolean, if editing UTF-8 encoded Hebrew, allows one to remove the niqud or te'amim by pressing 'x' on a character (with associated niqud).
- + **'rightleftcmd'** ('rlc') makes the command-prompt for searches show up on the right side. It only takes effect if the window is **'rightleft'**.

+ Encoding:

- + Under Unix, ISO 8859-8 encoding (Hebrew letters codes: 224-250).
- + Under MS DOS, PC encoding (Hebrew letters codes: 128-154). These are defaults, that can be overridden using the **'aleph'** option.
- + You should prefer using UTF8, as it supports the combining-characters ('deco' does nothing if UTF8 encoding is not active).

- + Vim arguments:
 - + 'vim -H file' starts editing a Hebrew file, i.e. **'rightleft'** and **'hkmap'** are set.
 - + Keyboard:
 - + The **'allowrevins'** option enables the **CTRL-_** command in Insert mode and in Command-line mode.
 - + **CTRL-_** in insert/replace modes toggles **'revins'** and **'hkmap'** as follows:

When in rightleft window, **'revins'** and **'nohkmap'** are toggled, since English will likely be inserted in this case.

When in norightleft window, **'revins'** **'hkmap'** are toggled, since Hebrew will likely be inserted in this case.

CTRL-_ moves the cursor to the end of the typed text.
 - + **CTRL-_** in command mode only toggles keyboard mapping (see Bugs below). This setting is independent of **'hkmap'** option, which only applies to insert/replace mode.
- Note:** On some keyboards, **CTRL-_** is mapped to **CTRL-?**.
- + Keyboard mapping while **'hkmap'** is set (standard Israeli keyboard):

```

q w e r t y u i o p
/ '  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ

a s d f g h j k l ; '
ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ,

z x c v b n m , . /
ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  ⓧ  .

```

This is also the keymap when 'keymap=hebrew' is set. The advantage of **'keymap'** is that it works properly when using UTF8, e.g. it inserts the correct characters; **'hkmap'** does not. The **'keymap'** keyboard can also insert niqud and te'amim. To see what those mappings are, look at the keymap file 'hebrew.vim' etc.

Typing backwards

If the **'revins'** (reverse insert) option is set, inserting happens backwards. This can be used to type Hebrew. When inserting characters the cursor is not moved and the text moves rightwards. A **<BS>** deletes the character under the cursor. **CTRL-W** and **CTRL-U** also work in the opposite direction. **<BS>**, **CTRL-W** and **CTRL-U** do not stop at the start of insert or end of line, no matter how the **'backspace'** option is set.

There is no reverse replace mode (yet).

If the **'showmode'** option is set, "-- REVERSE INSERT --" will be shown in the

status line when reverse Insert mode is active.

When the **'allowrevins'** option is set, reverse Insert mode can be also entered via **CTRL-_**, which has some extra functionality: First, keyboard mapping is changed according to the window orientation -- if in a left-to-right window, **'revins'** is used to enter Hebrew text, so the keyboard changes to Hebrew ('hkmap' is set); if in a right-to-left window, **'revins'** is used to enter English text, so the keyboard changes to English ('hkmap' is reset). Second, when exiting **'revins'** via **CTRL-_**, the cursor moves to the end of the typed text (if possible).

Pasting when in a rightleft window

When cutting text with the mouse and pasting it in a rightleft window the text will be reversed, because the characters come from the cut buffer from the left to the right, while inserted in the file from the right to the left. In order to avoid it, toggle **'revins'** (by typing **CTRL-?** or **CTRL-_**) before pasting.

Hebrew characters and the **'isprint'** variable

Sometimes Hebrew character codes are in the non-printable range defined by the **'isprint'** variable. For example in the Linux console, the Hebrew font encoding starts from 128, while the default **'isprint'** variable is @,161-255. The result is that all Hebrew characters are displayed as ~x. To solve this problem, set `isprint=@,128-255`.

VIM REFERENCE MANUAL by Vassily Ragosin

Russian language localization and support in Vim russian Russian

- 1. Introduction russian-intro
- 2. Russian keymaps russian-keymap
- 3. Localization russian-l18n
- 4. Known issues russian-issues

=====
1. Introduction russian-intro

Russian language is supported perfectly well in Vim. You can type and view Russian text just as any other, without the need to tweak the settings.

=====
2. Russian keymaps russian-keymap

To switch between languages you can use your system native keyboard switcher, or use one of the Russian keymaps, included in the Vim distribution. For example,

```
:set keymap=russian-jcukenwin
```

In the latter case, you can switch between languages even if you do not have system Russian keyboard or independently from a system-wide keyboard settings. See 'keymap'. You can also map a key to switch between keyboards, if you choose the latter option. See :map .

For your convenience, to avoid switching between keyboards, when you need to enter Normal mode command, you can also set 'langmap' option:

```
:set langmap=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX;ABCDEFGHIJKLMNQRSTUUVWXYZ,  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX;abcdefghijklmnopqrstuvwxyz
```

This is in utf-8, you cannot read this if your 'encoding' is not utf-8. You have to type this command in one line, it is wrapped for the sake of readability.

=====
3. Localization russian-l18n

If you wish to use messages, help files, menus and other items translated to Russian, you will need to install the RuVim Language Pack, available in different codepages from

```
http://www.sourceforge.net/projects/ruvim/
```

Make sure that your Vim is at least 6.2.506 and use ruvim 0.5 or later for automatic installs. Vim also needs to be compiled with +gettext feature for

user interface items translations to work.

After downloading an archive from RuVim project, unpack it into your \$VIMRUNTIME directory. We recommend using UTF-8 archive.

In order to use the Russian documentation, make sure you have set the **'helplang'** option to "ru".

4. Known issues

[russian-issues](#)

- If you are using Russian message translations in Win32 console, then you may see the output produced by "vim --help", "vim --version" commands and Win32 console window title appearing in a wrong codepage. This problem is related to a bug in GNU gettext library and may be fixed in the future releases of gettext.
-

hangulin.txt For Vim version 9.1. Last change: 2019 Nov 21

VIM REFERENCE MANUAL by Chi-Deok Hwang and Sung-Hyun Nam

hangul

Vim had built-in support for hangul, the Korean language, for users without XIM (X Input Method). Since it didn't work well and was not maintained it was removed in Vim 8.1.2327.

If you want this hangul input method you can go back to Vim 8.1.2326 or earlier. If you think this code is still useful and want to maintain it, make a patch to add it back. However, making it work with UTF-8 encoding would be best.

Vim's Graphical User Interface

gui GUI

- | | |
|--------------------------|--------------------------------|
| 1. Starting the GUI | gui-start |
| 2. Scrollbars | gui-scrollbars |
| 3. Mouse Control | gui-mouse |
| 4. Making GUI Selections | gui-selections |
| 5. Menus | menus |
| 6. Font | gui-font |
| 7. Extras | gui-extras |
| 8. Shell Commands | gui-shell |

Other GUI documentation:

- [gui_x11.txt](#) For specific items of the X11 GUI.
[gui_w32.txt](#) For specific items of the Win32 GUI.

=====

1. Starting the GUI [gui-start](#) [E229](#) [E233](#)

First you must make sure you actually have a version of Vim with the GUI code included. You can check this with the `":version"` command, it says "with xxx GUI", where "xxx" is X11-Motif, Photon, GTK2, GTK3, etc., or "MS-Windows 32 bit GUI version".

How to start the GUI depends on the system used. Mostly you can run the GUI version of Vim with:

```
gvim [options] [files...]
```

The X11 version of Vim can run both in GUI and in non-GUI mode. See [gui-x11-start](#) .

[gui-init](#) [gvimrc](#) [.gvimrc](#) [_gvimrc](#) [\\$MYGVIMRC](#)

The `gvimrc` file is where GUI-specific startup commands should be placed. It is always sourced after the `vimrc` file. If you have one then the `$MYGVIMRC` environment variable has its name.

When the GUI starts up initializations are carried out, in this order:

- The `'term'` option is set to `"builtin_gui"` and terminal options are reset to their default value for the GUI [terminal-options](#) .
- If the system menu file exists, it is sourced. The name of this file is normally `"$VIMRUNTIME/menu.vim"`. You can check this with `":version"`. Also see [\\$VIMRUNTIME](#) . To skip loading the system menu include `'M'` in `'guioptions'`. [buffers-menu](#) [no_buffers_menu](#)

The system menu file includes a "Buffers" menu. If you don't want this, set the `"no_buffers_menu"` variable in your `.vimrc` (not `.gvimrc`):

```
:let no_buffers_menu = 1
```

NOTE: Switching on syntax highlighting also loads the menu file, thus disabling the Buffers menu must be done before `":syntax on"`.

The path names are truncated to 35 characters. You can truncate them at a different length, for example 50, like this:

```
:let bmenu_max_pathlen = 50
```

- If the "-U {gvimrc}" command-line option has been used when starting Vim, the {gvimrc} file will be read for initializations. The following initializations are skipped. When {gvimrc} is "NONE" no file will be read for initializations.
 - For Unix and MS-Windows, if the system gvimrc exists, it is sourced. The name of this file is normally "\$VIM/gvimrc". You can check this with ":version". Also see \$VIM .
 - The following are tried, and only the first one that exists is used:
 - If the GVIMINIT environment variable exists and is not empty, it is executed as an Ex command.
 - If the user gvimrc file exists, it is sourced. The name of this file is normally "\$HOME/.gvimrc". You can check this with ":version".
 - For Win32, \$HOME is set by Vim if needed, see \$HOME-windows .
 - When a "_gvimrc" file is not found, ".gvimrc" is tried too. And vice versa.
- The name of the first file found is stored in \$MYGVIMRC, unless it was already set.
- If the 'exrc' option is set (which is NOT the default) the file ./gvimrc is sourced, if it exists and isn't the same file as the system or user gvimrc file. If this file is not owned by you, some security restrictions apply. When ".gvimrc" is not found, "_gvimrc" is tried too. For Macintosh and DOS/Win32 "_gvimrc" is tried first.

NOTE: All but the first one are not carried out if Vim was started with "-u NONE" or "-u DEFAULTS" and no "-U" argument was given, or when started with "-U NONE".

All this happens AFTER the normal Vim initializations, like reading your .vimrc file. See [initialization](#) .

But the GUI window is only opened after all the initializations have been carried out. If you want some commands to be executed just after opening the GUI window, use the GUIEnter autocommand event. Example:

```
:autocmd GUIEnter * winpos 100 50
```

You can use the gvimrc files to set up your own customized menus (see [:menu](#)) and initialize other things that you may want to set up differently from the terminal version.

Recommended place for your personal GUI initializations:

Unix	\$HOME/.gvimrc or \$HOME/.vim/gvimrc
Win32	\$HOME/_gvimrc, \$HOME/vimfiles/gvimrc or \$VIM/_gvimrc
Amiga	s:./gvimrc, home:./gvimrc, home:vimfiles:gvimrc or \$VIM/.gvimrc
Haiku	\$HOME/config/settings/vim/gvimrc

The personal initialization files are searched in the order specified above and only the first one that is found is read.

There are a number of options which only have meaning in the GUI version of Vim. These are 'guicursor', 'guifont', 'guipty' and 'guioptions'. They are

documented in [options.txt](#) with all the other options.

If using the Motif version of the GUI (but not for the GTK+ or Win32 version), a number of X resources are available. See [gui-resources](#) .

Another way to set the colors for different occasions is with highlight groups. The "Normal" group is used to set the background and foreground colors. Example (which looks nice):

```
:highlight Normal guibg=grey90
```

The "guibg" and "guifg" settings override the normal background and foreground settings. The other settings for the Normal highlight group are not used. Use the '[guifont](#)' option to set the font.

Also check out the '[guicursor](#)' option, to set the colors for the cursor in various modes.

Vim tries to make the window fit on the screen when it starts up. This avoids that you can't see part of it. On the X Window System this requires a bit of guesswork. You can change the height that is used for the window title and a task bar with the '[guiheadroom](#)' option.

[:winp](#) [:winpos](#) E188

[:winp\[os\]](#)

Display current position of the top left corner of the GUI vim window in pixels. Does not work in all versions. Also see [getwinpos\(\)](#) , [getwinposx\(\)](#) and [getwinposy\(\)](#) .

[:winp\[os\]](#) {X} {Y}

E466

Put the GUI vim window at the given {X} and {Y} coordinates. The coordinates should specify the position in pixels of the top left corner of the window. Does not work in all versions. Does work in an (new) xterm [xterm-color](#) . When the GUI window has not been opened yet, the values are remembered until the window is opened. The position is adjusted to make the window fit on the screen (if possible).

[:wi](#) [:win](#) [:winsize](#) E465

[:win\[size\]](#) {width} {height}

Set the window height to {width} by {height} characters. It is recommended to use ``:set lines=11 columns=22`` instead, since it's easy to see what the numbers mean. If you get less lines than expected, check the '[guiheadroom](#)' option.

If you are running the X Window System, you can get information about the window Vim is running in with these commands:

```
:!xwininfo -id $WINDOWID
:!xprop -id $WINDOWID
:execute '!xwininfo -id ' .. v:windowid
:execute '!xprop -id ' .. v:windowid
```

[gui-IME](#) [iBus](#)

current window and when, taking the middle of the current window and drawing a vertical line, this line goes through the window. When there are scrollbars on both sides, and the middle of the current window is on the left half, the right scrollbar column will contain scrollbars for the rightmost windows. The same happens on the other side.

HORIZONTAL SCROLLBARS

[gui-horiz-scroll](#)

The horizontal scrollbar (at the bottom of the Vim GUI) may be used to scroll text sideways when the **'wrap'** option is turned off. The scrollbar-thumb size is such that the text of the longest visible line may be scrolled as far as possible left and right. The cursor is moved when necessary, it must remain on a visible character (unless **'virtualedit'** is set).

Computing the length of the longest visible line takes quite a bit of computation, and it has to be done every time something changes. If this takes too much time or you don't like the cursor jumping to another line, include the 'h' flag in **'guioptions'**. Then the scrolling is limited by the text of the current cursor line.

[motif-intellimouse](#)

If you have an Intellimouse and an X server that supports using the wheel, then you can use the wheel to scroll the text up and down in gvim. This works with XFree86 4.0 and later, and with some older versions when you add patches. See [scroll-mouse-wheel](#).

For older versions of XFree86 you must patch your X server. The following page has a bit of information about using the Intellimouse on Linux as well as links to the patches and X server binaries (may not have the one you need though):

<http://www.inria.fr/koala/colas/mouse-wheel-scroll/>

3. Mouse Control

[gui-mouse](#)

The mouse only works if the appropriate flag in the **'mouse'** option is set. When the GUI is switched on, and **'mouse'** wasn't set yet, the **'mouse'** option is automatically set to "a", enabling it for all modes except for the [hit-enter](#) prompt. If you don't want this, a good place to change the **'mouse'** option is the "gvimrc" file.

Other options that are relevant:

'mousefocus' window focus follows mouse pointer [gui-mouse-focus](#)
'mousemodel' what mouse button does which action
'mousehide' hide mouse pointer while typing text
'mousemoveevent' enable mouse move events so that `<MouseMove>` can be mapped
'selectmode' whether to start Select mode or Visual mode

A quick way to set these is with the `":behave"` command.

[:behave](#) [:be](#)

`:be[have] {model}` Set behavior for mouse and selection. Valid arguments are:

mswin	MS-Windows behavior
xterm	Xterm behavior

Using `":behave"` changes these options:

option	mswin	xterm
'selectmode'	"mouse,key"	""
'mousemodel'	"popup"	"extend"
'keymodel'	"startsel,stopse"	""
'selection'	"exclusive"	"inclusive"

In the `$VIMRUNTIME` directory, there is a script called `mswin.vim`, which will also map a few keys to the MS-Windows cut/copy/paste commands. This is NOT compatible, since it uses the **CTRL-V**, **CTRL-X** and **CTRL-C** keys. If you don't mind, use this command:

```
:so $VIMRUNTIME/mswin.vim
```

For scrolling with a wheel on a mouse, see [scroll-mouse-wheel](#).

3.1 Moving Cursor with Mouse

[gui-mouse-move](#)

Click the left mouse button somewhere in a text buffer where you want the cursor to go, and it does!

This works in when 'mouse' contains

Normal mode	'n' or 'a'
Visual mode	'v' or 'a'
Insert mode	'i' or 'a'

Select mode is handled like Visual mode.

You may use this with an operator such as 'd' to delete text from the current cursor position to the position you point to with the mouse. That is, you hit 'd' and then click the mouse somewhere.

[gui-mouse-focus](#)

The `'mousefocus'` option can be set to make the keyboard focus follow the mouse pointer. This means that the window where the mouse pointer is, is the active window. Warning: this doesn't work very well when using a menu, because the menu command will always be applied to the top window.

If you are on the ':' line (or '/' or '?'), then clicking the left or right mouse button will position the cursor on the ':' line (if `'mouse'` contains 'c' or 'a').

In any situation the middle mouse button may be clicked to paste the current selection.

3.2 Selection with Mouse

[gui-mouse-select](#)

The mouse can be used to start a selection. How depends on the `'mousemodel'` option:

`'mousemodel'` is "extend": use the right mouse button

`'mousemodel'` is "popup": use the left mouse button, while keeping the Shift

key pressed.

If there was no selection yet, this starts a selection from the old cursor position to the position pointed to with the mouse. If there already is a selection then the closest end will be extended.

If `'selectmode'` contains "mouse", then the selection will be in Select mode. This means that typing normal text will replace the selection. See `Select-mode`. Otherwise, the selection will be in Visual mode.

Double clicking may be done to make the selection word-wise, triple clicking makes it line-wise, and quadruple clicking makes it rectangular block-wise.

See `gui-selections` on how the selection is used.

3.3 Other Text Selection with Mouse

`gui-mouse-modeless`
`modeless-selection`

A different kind of selection is used when:

- in Command-line mode
- in the Command-line window and pointing in another window
- at the `hit-enter` prompt
- whenever the current mode is not in the `'mouse'` option
- when holding the CTRL and SHIFT keys in the GUI

Since Vim continues like the selection isn't there, and there is no mode associated with the selection, this is called modeless selection. Any text in the Vim window can be selected. Select the text by pressing the left mouse button at the start, drag to the end and release. To extend the selection, use the right mouse button when `'mousemodel'` is "extend", or the left mouse button with the shift key pressed when `'mousemodel'` is "popup". The selection is removed when the selected text is scrolled or changed.

On the command line `CTRL-Y` can be used to copy the selection into the clipboard. To do this from Insert mode, use `CTRL-O : CTRL-Y <CR>`. When `'guioptions'` contains a or A (default on X11), the selection is automatically copied to the "*" register.

The middle mouse button can then paste the text. On non-X11 systems, you can use `CTRL-R +`.

3.4 Using Mouse on Status Lines

`gui-mouse-status`

Clicking the left or right mouse button on the status line below a Vim window makes that window the current window. This actually happens on button release (to be able to distinguish a click from a drag action).

With the left mouse button a status line can be dragged up and down, thus resizing the windows above and below it. This does not change window focus.

The same can be used on the vertical separator: click to give the window left of it focus, drag left and right to make windows wider and narrower.

3.5 Various Mouse Clicks

gui-mouse-various

<code><S-LeftMouse></code>	Search forward for the word under the mouse click. When <code>'mousemodel'</code> is "popup" this starts or extends a selection.
<code><S-RightMouse></code>	Search backward for the word under the mouse click.
<code><C-LeftMouse></code>	Jump to the tag name under the mouse click.
<code><C-RightMouse></code>	Jump back to position before the previous tag jump (same as <code>"CTRL-T"</code>)

3.6 Mouse Mappings

gui-mouse-mapping

The mouse events, complete with modifiers, may be mapped. Eg:

```
:map <S-LeftMouse>    <RightMouse>
:map <S-LeftDrag>     <RightDrag>
:map <S-LeftRelease>  <RightRelease>
:map <2-S-LeftMouse>  <2-RightMouse>
:map <2-S-LeftDrag>   <2-RightDrag>
:map <2-S-LeftRelease> <2-RightRelease>
:map <3-S-LeftMouse>  <3-RightMouse>
:map <3-S-LeftDrag>   <3-RightDrag>
:map <3-S-LeftRelease> <3-RightRelease>
:map <4-S-LeftMouse>  <4-RightMouse>
:map <4-S-LeftDrag>   <4-RightDrag>
:map <4-S-LeftRelease> <4-RightRelease>
```

These mappings make selection work the way it probably should in a Motif application, with shift-left mouse allowing for extending the visual area rather than the right mouse button.

`<MouseMove>` may be mapped, but `'mousemoveevent'` must be enabled to use the mapping.

Mouse mapping with modifiers does not work for modeless selection.

3.7 Drag and drop

drag-n-drop

You can drag and drop one or more files into the Vim window, where they will be opened as if a `:drop` command was used. You can check if this is supported with the `drop_file` feature: ``has('drop_file')``.

If you hold down Shift while doing this, Vim changes to the first dropped file's directory. If you hold Ctrl Vim will always split a new window for the file. Otherwise it's only done if the current buffer has been changed.

You can also drop a directory on Vim. This starts the explorer plugin for that directory (assuming it was enabled, otherwise you'll get an error message). Keep Shift pressed to change to the directory instead.

If Vim happens to be editing a command line, the names of the dropped files and directories will be inserted at the cursor. This allows you to use these names with any Ex command. Special characters (space, tab, double quote and

'|'; backslash on non-MS-Windows systems) will be escaped.

4. Making GUI Selections

[gui-selections](#)

[quotestar](#)

You may make selections with the mouse (see [gui-mouse-select](#)), or by using Vim's Visual mode (see [v](#)). If 'a' is present in '[guioptions](#)', then whenever a selection is started (Visual or Select mode), or when the selection is changed, Vim becomes the owner of the windowing system's primary selection (on MS-Windows the [gui-clipboard](#) is used; under X11, the [x11-selection](#) is used - you should read whichever of these is appropriate now).

[clipboard](#)

There is a special register for storing this selection, it is the "*" register. Nothing is put in here unless the information about what text is selected is about to change (e.g. with a left mouse click somewhere), or when another application wants to paste the selected text. Then the text is put in the "*" register. For example, to cut a line and make it the current selection/put it on the clipboard:

```
"*dd
```

Similarly, when you want to paste a selection from another application, e.g., by clicking the middle mouse button, the selection is put in the "*" register first, and then '[put](#)' like any other register. For example, to put the selection (contents of the clipboard):

```
"*p
```

When using this register under X11, also see [x11-selection](#). This also explains the related "+" register.

Note that when pasting text from one Vim into another separate Vim, the type of selection (character, line, or block) will also be copied. For other applications the type is always character. However, if the text gets transferred via the [x11-cut-buffer](#), the selection type is ALWAYS lost.

When the "unnamed" string is included in the '[clipboard](#)' option, the unnamed register is the same as the "*" register. Thus you can yank to and paste the selection without prepending "*" to commands.

5. Menus

[menus](#)

For an introduction see [usr_42.txt](#) in the user manual.

5.1 Using Menus

[using-menus](#)

Basically, menus can be used just like mappings. You can define your own menus, as many as you like. Long-time Vim users won't use menus much. But the power is in adding your own menus and menu items. They are most useful for things that you can't remember

what the key sequence was.

For creating menus in a different language, see `:menutrans` .
If you don't want to use menus at all, see `'go-M'` .

menu.vim

The default menus are read from the file "\$VIMRUNTIME/menu.vim". See `$VIMRUNTIME` for where the path comes from. You can set up your own menus. Starting off with the default set is a good idea. You can add more items, or, if you don't like the defaults at all, start with removing all menus `:unmenu-all` . You can also avoid the default menus being loaded by adding this line to your .vimrc file (NOT your .gvimrc file!):

```
:let did_install_default_menus = 1
```

If you also want to avoid the Syntax menu:

```
:let did_install_syntax_menu = 1
```

The first item in the Syntax menu can be used to show all available filetypes in the menu (which can take a bit of time to load). If you want to have all filetypes already present at startup, add:

```
:let do_syntax_sel_menu = 1
```

The following menuitems show all available color schemes, keymaps and compiler settings:

```
Edit > Color Scheme
```

```
Edit > Keymap
```

```
Tools > Set Compiler
```

However, they can also take a bit of time to load, because they search all related files from the directories in `'runtimepath'`. Therefore they are loaded lazily (by the `CursorHold` event), or you can also load them manually. If you want to have all these items already present at startup, add:

```
:let do_no_lazyload_menus = 1
```

Note that the menu.vim is sourced when `:syntax on` or `:filetype on` is executed or after your .vimrc file is sourced. This means that the `'encoding'` option and the language of messages (`:language messages`) must be set before that (if you want to change them).

console-menus

Although this documentation is in the GUI section, you can actually use menus in console mode too. You will have to load `menu.vim` explicitly then, it is not done by default. You can use the `:emenu` command and command-line completion with `'wildmenu'` to access the menu entries almost like a real menu system. To do this, put these commands in your .vimrc file:

```
:source $VIMRUNTIME/menu.vim
```

```
:set wildmenu
```

```
:set cpo-=<
```

```
:set wcm=<C-Z>
```

```
:map <F4> :emenu <C-Z>
```

Pressing `<F4>` will start the menu. You can now use the cursor keys to select a menu entry. Hit `<Enter>` to execute it. Hit `<Esc>` if you want to cancel. This does require the `+menu` feature enabled at compile time.

tear-off-menus

GTK+ 2 and Motif support Tear-off menus. These are sort of sticky menus or pop-up menus that are present all the time. If the resizing does not work

correctly, this may be caused by using something like "Vim*geometry" in the defaults. Use "Vim.geometry" instead.

As to GTK+ 3, tear-off menus have been deprecated since GTK+ 3.4. Accordingly, they are disabled if gvim is linked against GTK+ 3.4 or later.

The Win32 GUI version emulates Motif's tear-off menus. Actually, a Motif user will spot the differences easily, but hopefully they're just as useful. You can also use the `:tearoff` command together with `hidden-menus` to create floating menus that do not appear on the main menu bar.

5.2 Creating New Menus

creating-menus

```
:me      :menu      :noreme     :noremenu
E330     E327     E331      E336     E333
E328     E329     E337     E792
```

To create a new menu item, use the `:menu` commands. They are mostly like the `:map` set of commands (see [map-modes](#)), but the first argument is a menu item name, given as a path of menus and submenus with a `.'` between them, e.g.:

```
:menu File.Save :w<CR>
:inoremenu File.Save <C-O>:w<CR>
:menu Edit.Big\ Changes.Delete\ All\ Spaces :%s/[ ^I]//g<CR>
```

This last one will create a new item in the menu bar called "Edit", holding the mouse button down on this will pop up a menu containing the item "Big Changes", which is a sub-menu containing the item "Delete All Spaces", which when selected, performs the operation.

To create a menu for terminal mode, use `:tmenu` instead of `:menu` unlike key mapping (`:tmap`). This is because `:tmenu` is already used for defining tooltips for menus. See [terminal-typing](#).

Special characters in a menu name:

- `&` The next character is the shortcut key. Make sure each shortcut key is only used once in a (sub)menu. If you want to insert a literal "&" in the menu name use "&&".
menu-shortcut
- `<Tab>` Separates the menu name from right-aligned text. This can be used to show the equivalent typed command. The text "`<Tab>`" can be used here for convenience. If you are using a real tab, don't forget to put a backslash before it!
menu-text

Example:

```
:amenu &File.&Open<Tab>:e :browse e<CR>
```

[typed literally]

With the shortcut "F" (while keeping the `<Alt>` key pressed), and then "O", this menu can be used. The second part is shown as "Open :e". The `:e` is right aligned, and the "O" is underlined, to indicate it is the shortcut.

`:am` `:amenu` `:an` `:anoremenu`

The `":amenu"` command can be used to define menu entries for all modes at once, except for Terminal mode. To make the command work correctly, a character is automatically inserted for some modes:

mode	inserted	appended
Normal	nothing	nothing
Visual	<C-C>	<C-\><C-G>
Insert	<C-\><C-O>	
Cmdline	<C-C>	<C-\><C-G>
Op-pending	<C-C>	<C-\><C-G>

Appending `CTRL-\ CTRL-G` is for going back to insert mode when `'insertmode'` is set. `CTRL-_CTRL-G`

Example:

```
:amenu File.Next      :next^M
```

is equal to:

```
:nmenu File.Next      :next^M
:vmenu File.Next      ^C:next^M^_\^G
:imenu File.Next      ^\^O:next^M
:cmenu File.Next      ^C:next^M^_\^G
:omenu File.Next      ^C:next^M^_\^G
```

Careful: In Insert mode this only works for a SINGLE Normal mode command, because of the `CTRL-O`. If you have two or more commands, you will need to use the `":imenu"` command. For inserting text in any mode, you can use the expression register:

```
:amenu Insert.foobar  "'=foobar'<CR>P
```

The special text `<Cmd>` begins a "command menu", it executes the command directly without changing modes. Where you might use `":...<CR>"` you can instead use `"<Cmd>...<CR>"`. See `<Cmd>` for more info. Example:

```
anoremenu File.Next <Cmd>next<CR>
```

Note that the `'<'` and `'k'` flags in `'cptions'` also apply here (when included they make the `<>` form and raw key codes not being recognized).

Note that `<Esc>` in Cmdline mode executes the command, like in a mapping. This is Vi compatible. Use `CTRL-C` to quit Cmdline mode.

`:nme` `:nmenu` `:nnoreme` `:nnoremenu` `:nunme` `:nunmenu`

Menu commands starting with "n" work in Normal mode. `mapmode-n`

`:ome` `:omenu` `:onoreme` `:onoremenu` `:ounme` `:ounmenu`

Menu commands starting with "o" work in Operator-pending mode. `mapmode-o`

`:vme` `:vmenu` `:vnoreme` `:vnoremenu` `:vunme` `:vunmenu`

Menu commands starting with "v" work in Visual mode. `mapmode-v`

`:xme` `:xmenu` `:xnoreme` `:xnoremenu` `:xunme` `:xunmenu`
Menu commands starting with "x" work in Visual and Select mode. `mapmode-x`

`:sme` `:smenu` `:snoreme` `:snoremenu` `:sunme` `:sunmenu`
Menu commands starting with "s" work in Select mode. `mapmode-s`

`:ime` `:imenu` `:inoreme` `:inoremenu` `:iunme` `:iunmenu`
Menu commands starting with "i" work in Insert mode. `mapmode-i`

`:cme` `:cmenu` `:cnoreme` `:cnoremenu` `:cunme` `:cunmenu`
Menu commands starting with "c" work in Cmdline mode. `mapmode-c`

`:tlm` `:tlmenu` `:tln` `:tlnoremenu` `:tlu` `:tlunmenu`
Menu commands starting with "tl" work in Terminal mode. `mapmode-t`

`:menu-<silent>` `:menu-silent`
To define a menu which will not be echoed on the command line, add "`<silent>`" as the first argument. Example:

```
:menu <silent> Settings.Ignore\ case :set ic<CR>
```

The `:set ic` will not be echoed when using this menu. Messages from the executed command are still given though. To shut them up too, add a `:silent` in the executed command:

```
:menu <silent> Search.Header :exe ":silent normal /Header\r"<CR>
```

`<silent>` may also appear just after "`<special>`" or "`<script>`".

`:menu-<special>` `:menu-special`
Define a menu with `<>` notation for special keys, even though the "`<`" flag may appear in '`coptions`'. This is useful if the side effect of setting '`coptions`' is not desired. Example:

```
:menu <special> Search.Header /Header<CR>
```

`<special>` must appear as the very first argument to the `:menu` command or just after "`<silent>`" or "`<script>`".

`:menu-<script>` `:menu-script`
The "to" part of the menu will be inspected for mappings. If you don't want this, use the `:noremenu` command (or the similar one for a specific mode). If you do want to use script-local mappings, add "`<script>`" as the very first argument to the `:menu` command or just after "`<silent>`" or "`<special>`".

`menu-priority`
You can give a priority to a menu. Menus with a higher priority go more to the right. The priority is given as a number before the `:menu` command. Example:

```
:80menu Buffer.next :bn<CR>
```

The default menus have these priorities:

File	10
Edit	20
Tools	40
Syntax	50
Buffers	60
Window	70
Help	9999

When no or zero priority is given, 500 is used.
The priority for the PopUp menu is not used.

The Help menu will be placed on the far right side of the menu bar on systems which support this (Motif and GTK+). For GTK+ 2 and 3, this is not done anymore because right-aligning the Help menu is now discouraged UI design.

You can use a priority higher than 9999, to make it go after the Help menu, but that is non-standard and is discouraged. The highest possible priority is about 32000. The lowest is 1.

sub-menu-priority

The same mechanism can be used to position a sub-menu. The priority is then given as a dot-separated list of priorities, before the menu name:

```
:menu 80.500 Buffer.next :bn<CR>
```

Giving the sub-menu priority is only needed when the item is not to be put in a normal position. For example, to put a sub-menu before the other items:

```
:menu 80.100 Buffer.first :brew<CR>
```

Or to put a sub-menu after the other items, and further items with default priority will be put before it:

```
:menu 80.900 Buffer.last :blast<CR>
```

When a number is missing, the default value 500 will be used:

```
:menu .900 myMenu.test :echo "text"<CR>
```

The menu priority is only used when creating a new menu. When it already existed, e.g., in another mode, the priority will not change. Thus, the priority only needs to be given the first time a menu is used.

An exception is the PopUp menu. There is a separate menu for each mode (Normal, Op-pending, Visual, Insert, Cmdline). The order in each of these menus can be different. This is different from menu-bar menus, which have the same order for all modes.

NOTE: sub-menu priorities currently don't work for all versions of the GUI.

menu-separator E332

Menu items can be separated by a special item that inserts some space between items. Depending on the system this is displayed as a line or a dotted line. These items must start with a '-' and end in a '-'. The part in between is used to give it a unique name. Priorities can be used as with normal items. Example:

```
:menu Example.item1      :do something
:menu Example.-Sep-      :
:menu Example.item2      :do something different
```

Note that the separator also requires a rhs. It doesn't matter what it is, because the item will never be selected. Use a single colon to keep it simple.

gui-toolbar

The toolbar is currently available in the Win32, Motif, GTK+ (X11), and Photon GUI. It should turn up in other GUIs in due course. The default toolbar is setup in menu.vim.

The display of the toolbar is controlled by the **'guioptions'** letter 'T'. You can thus have menu & toolbar together, or either on its own, or neither.

The appearance is controlled by the **'toolbar'** option. You can choose between an image, text or both.

toolbar-icon

The toolbar is defined as a special menu called ToolBar, which only has one level. Vim interprets the items in this menu as follows:

- 1) If an "icon=" argument was specified, the file with this name is used. The file can either be specified with the full path or with the base name. In the last case it is searched for in the "bitmaps" directory in **'runtimepath'**, like in point 3. Examples:

```
:amenu icon=/usr/local/pixmaps/foo_icon.xpm ToolBar.Foo :echo "Foo"<CR>
:amenu icon=FooIcon ToolBar.Foo :echo "Foo"<CR>
```

Note that in the first case the extension is included, while in the second case it is omitted.

If the file cannot be opened the next points are tried.

A space in the file name must be escaped with a backslash.

A menu priority must come **_after_** the icon argument:

```
:amenu icon=foo 1.42 ToolBar.Foo :echo "42!"<CR>
```

- 2) An item called 'Builtin##', where ## is a number, is taken as number ## of the built-in bitmaps available in Vim. Currently there are 31 numbered from 0 to 30 which cover most common editing operations **builtin-tools**.

```
:amenu ToolBar.Builtin22 :call SearchNext("back")<CR>
```

- 3) An item with another name is first searched for in the directory "bitmaps" in **'runtimepath'**. If found, the bitmap file is used as the toolbar button image. **Note** that the exact filename is OS-specific: For example, under Win32 the command

```
:amenu ToolBar.Hello :echo "hello"<CR>
```

would find the file 'hello.bmp'. Under GTK+/X11 it is 'Hello.xpm'. With GTK+ 2 the files 'Hello.png', 'Hello.xpm' and 'Hello.bmp' are checked for existence, and the first one found would be used.

For MS-Windows and GTK+ 2 the bitmap is scaled to fit the button. For MS-Windows a size of 18 by 18 pixels works best.

For MS-Windows the bitmap should have 16 colors with the standard palette. The light grey pixels will be changed to the Window frame color and the dark grey pixels to the window shadow color. More colors might also work, depending on your system.

- 4) If the bitmap is still not found, Vim checks for a match against its list of built-in names. Each built-in button image has a name. So the command

```
:amenu ToolBar.Open :e
```

will show the built-in "open a file" button image if no open.bmp exists.

All the built-in names can be seen used in menu.vim.

- 5) If all else fails, a blank, but functioning, button is displayed.

builtin-tools

nr	Name	Normal action
00	New	open new window
01	Open	browse for file to open in current window
02	Save	write buffer to file
03	Undo	undo last change
04	Redo	redo last undone change
05	Cut	delete selected text to clipboard
06	Copy	copy selected text to clipboard
07	Paste	paste text from clipboard
08	Print	print current buffer
09	Help	open a buffer on Vim's builtin help
10	Find	start a search command

11	SaveAll	write all modified buffers to file
12	SaveSesn	write session file for current situation
13	NewSesn	write new session file
14	LoadSesn	load session file
15	RunScript	browse for file to run as a Vim script
16	Replace	prompt for substitute command
17	WinClose	close current window
18	WinMax	make current window use many lines
19	WinMin	make current window use few lines
20	WinSplit	split current window
21	Shell	start a shell
22	FindPrev	search again, backward
23	FindNext	search again, forward
24	FindHelp	prompt for word to search help for
25	Make	run make and jump to first error
26	TagJump	jump to tag under the cursor
27	RunCtags	build tags for files in current directory
28	WinVSplit	split current window vertically
29	WinMaxWidth	make current window use many columns
30	WinMinWidth	make current window use few columns

[hidden-menus](#) [win32-hidden-menus](#)

In the Win32 and GTK+ GUI, starting a menu name with ']' excludes that menu from the main menu bar. You must then use the [:popup](#) or [:tearoff](#) command to display it.

[window-toolbar](#) [WinBar](#)

Each window can have a local toolbar. This uses the first line of the window, thus reduces the space for the text by one line. The items in the toolbar must start with "WinBar".

Only text can be used. When using Unicode, special characters can be used to make the items look like icons.

If the items do not fit then the last ones cannot be used. The toolbar does not wrap.

Note that Vim may be in any mode when executing these commands. The menu should be defined for Normal mode and will be executed without changing the current mode. Thus if the current window is in Visual mode and the menu command does not intentionally change the mode, Vim will remain in Visual mode. Best is to use ``:nnoremenu`` to avoid side effects.

Example for debugger tools:

```
nnoremenu 1.10 WinBar.Step :Step<CR>
nnoremenu 1.20 WinBar.Next :Next<CR>
nnoremenu 1.30 WinBar.Finish :Finish<CR>
nnoremenu 1.40 WinBar.Cont :Continue<CR>
```

[hl-ToolbarLine](#) [hl-ToolbarButton](#)

The window toolbar uses the ToolbarLine and ToolbarButton highlight groups.

When splitting the window the window toolbar is not copied to the new window.

[popup-menu](#)

In the Win32, GTK+, Motif and Photon GUI, you can define the special menu "PopUp". This is the menu that is displayed when the right mouse button is pressed, if `'mousemodel'` is set to `popup` or `popup_setpos`.

Example:

```
nnoremenu 1.40 PopUp.&Paste "+gP
menu PopUp
```

5.3 Showing What Menus Are Mapped To

[showing-menus](#)

To see what an existing menu is mapped to, use just one argument after the menu commands (just like you would with the `":map"` commands). If the menu specified is a submenu, then all menus under that hierarchy will be shown. If no argument is given after `:menu` at all, then ALL menu items are shown for the appropriate mode (e.g., Command-line mode for `:cmenu`).

Special characters in the list, just before the rhs:

- * The menu was defined with "nore" to disallow remapping.
- & The menu was defined with "`<script>`" to allow remapping script-local mappings only.
- s The menu was defined with "`<silent>`" to avoid showing what it is mapped to when triggered.
- The menu was disabled.

Note that hitting `<Tab>` while entering a menu name after a menu command may be used to complete the name of the menu item.

It is not allowed to change menus while listing them. [E1310](#)

This doesn't normally happen, only when, for example, you would have a timer callback define a menu and the user lists menus in a way it shows

[more-prompt](#) .

5.4 Executing Menus

[execute-menus](#)

```
:[range]em[enu] {menu} :em :emenu E334 E335
Execute {menu} from the command line.
The default is to execute the Normal mode
menu. If a range is specified, it executes
the Visual mode menu.
If used from <c-o>, it executes the
insert-mode menu Eg:
```

```
:emenu File.Exit
```

```
:[range]em[enu] {mode} {menu} Like above, but execute the menu for {mode}:
'n': :nmenu Normal mode
'v': :vmenu Visual mode
's': :smenu Select mode
'o': :omenu Operator-pending mode
't': :tmenu Terminal mode
'i': :imenu Insert mode
'c': :cmenu Cmdline mode
```

If the console-mode vim has been compiled with WANT_MENU defined, you can use `:emenu` to access useful menu items you may have got used to from GUI mode. See **'wildmenu'** for an option that works well with this. See `console-menus` for an example.

When using a range, if the lines match with '<,>', then the menu is executed using the last visual selection.

5.5 Deleting Menus

delete-menus

```
:unme      :unmenu
:aun       :aunmenu
```

To delete a menu item or a whole submenu, use the `unmenu` commands, which are analogous to the `unmap` commands. Eg:

```
:unmenu! Edit.Paste
```

This will remove the `Paste` item from the `Edit` menu for `Insert` and `Command-line` modes.

Note that hitting `<Tab>` while entering a menu name after an `umenu` command may be used to complete the name of the menu item for the appropriate mode.

To remove all menus use:

```
:unmenu-all
```

```
:unmenu *      " remove all menus in Normal and visual mode
:unmenu! *     " remove all menus in Insert and Command-line mode
:aunmenu *     " remove all menus in all modes, except for Terminal
               " mode
:tlunmenu *    " remove all menus in Terminal mode
```

If you want to get rid of the menu bar:

```
:set guioptions-=m
```

5.6 Disabling Menus

disable-menus

```
:menu-disable  :menu-enable
```

If you do not want to remove a menu, but disable it for a moment, this can be done by adding the `"enable"` or `"disable"` keyword to a `":menu"` command.

Examples:

```
:menu disable &File.&Open\.\.\.
:aunmenu enable *
:aunmenu disable &Tools.*
```

The command applies to the modes as used with all menu commands. **Note** that characters like `"&"` need to be included for translated names to be found. When the argument is `"*"`, all menus are affected. Otherwise the given menu name and all existing submenus below it are affected.

5.7 Examples for Menus

menu-examples

Here is an example on how to add menu items with `menu's!` You can add a menu item for the keyword under the cursor. The register `"z"` is used.


```

:nmenu Words.Add\ Var      wb"zye:menu! Words.<C-R>z <C-R>z<CR>
:nmenu Words.Remove\ Var  wb"zye:unmenu! Words.<C-R>z<CR>
:vmenu Words.Add\ Var     "zy:menu! Words.<C-R>z <C-R>z <CR>
:vmenu Words.Remove\ Var  "zy:unmenu! Words.<C-R>z<CR>
:imenu Words.Add\ Var     <Esc>wb"zye:menu! Words.<C-R>z <C-R>z<CR>a
:imenu Words.Remove\ Var  <Esc>wb"zye:unmenu! Words.<C-R>z<CR>a

```

(the rhs is in <> notation, you can copy/paste this text to try out the mappings, or put these lines in your gvimrc; "<C-R>" is **CTRL-R**, "<CR>" is the <CR> key. <>)

tooltips menu-tips

5.8 Tooltips & Menu tips

See section 42.4 in the user manual.

```

:tm[enu] {menupath} {rhs}      :tmenu :tm      Define a tip for a menu or tool. {only in
                                X11 and Win32 GUI}

:tm[enu] [menupath]           :tmenu           List menu tips. {only in X11 and Win32 GUI}

:tu[nmenu] {menupath}        :tunmenu :tu     Remove a tip for a menu or tool.
                                {only in X11 and Win32 GUI}

```

Note: To create menus for terminal mode, use `:tmenu` instead.

When a tip is defined for a menu item, it appears in the command-line area when the mouse is over that item, much like a standard Windows menu hint in the status bar. (Except when Vim is in Command-line mode, when of course nothing is displayed.)

When a tip is defined for a ToolBar item, it appears as a tooltip when the mouse pauses over that button, in the usual fashion. Use the `hl-Tooltip` highlight group to change its colors.

A "tip" can be defined for each menu item. For example, when defining a menu item like this:

```
:amenu MyMenu.Hello :echo "Hello"<CR>
```

The tip is defined like this:

```
:tmenu MyMenu.Hello Displays a greeting.
```

And delete it with:

```
:tunmenu MyMenu.Hello
```

Tooltips are currently only supported for the X11 and Win32 GUI. However, they should appear for the other gui platforms in the not too distant future.

The `:tmenu` command works just like other menu commands, it uses the same arguments. `:tunmenu` deletes an existing menu tip, in the same way as the other unmenu commands.

If a menu item becomes invalid (i.e. its actions in all modes are deleted) Vim deletes the menu tip (and the item) for you. This means that `:aunmenu` deletes

a menu item - you don't need to do a `:tunmenu` as well.

5.9 Popup Menus

In the Win32 and GTK+ GUI, you can cause a menu to popup at the cursor. This behaves similarly to the `PopUp` menu except that any menu tree can be popped up.

This command is for backwards compatibility, using it is discouraged, because it behaves in a strange way.

<code>:popu[p] {name}</code>	<code>:popup</code> <code>:popu</code> Popup the menu <code>{name}</code> . The menu named must have at least one subentry, but need not appear on the menu-bar (see hidden-menus). {only available for Win32 and GTK GUI or in the terminal}
<code>:popu[p]! {name}</code>	Like above, but use the position of the mouse pointer instead of the cursor. In the terminal this is the last known position, which is usually at the last click or release (mouse movement is irrelevant).

Example:

```
:popup File  
will make the "File" menu (if there is one) appear at the text cursor (mouse pointer if ! was used).
```

```
:amenu ]Toolbar.Make    :make<CR>  
:popup ]Toolbar
```

This creates a popup menu that doesn't exist on the main menu-bar.

Note that in the GUI the `:popup` command will return immediately, before a selection has been made. In the terminal the commands waits for the user to make a selection.

Note that a menu that starts with `']'` will not be displayed.

6. Font

This section describes font related options.

GUIFONT

`gui-font`

`'guifont'` is the option that tells Vim what font to use. In its simplest form the value is just one font name. It can also be a list of font names separated with commas. The first valid font is used. When no valid font can be found you will get an error message.

On systems where `'guifontset'` is supported (X11) and `'guifontset'` is not empty, then `'guifont'` is not used. See [xfontset](#).

Note: As to the GTK GUIs, no error is given against any invalid names, and the first element of the list is always picked up and made use of. This is because, instead of identifying a given name with a font, the GTK GUIs use it to construct a pattern and try to look up a font which best matches the pattern among available fonts, and this way, the matching never fails. An invalid name doesn't matter because a number of font properties other than name will do to get the matching done.

Spaces after a comma are ignored. To include a comma in a font name precede it with a backslash. Setting an option requires an extra backslash before a space and a backslash. See also [option-backslash](#) . For example:

```
:set guifont=Screen15,\ 7x13,font\,with\,commas
```

will make Vim try to use the font "Screen15" first, and if it fails it will try to use "7x13" and then "font,with,commas" instead.

If none of the fonts can be loaded, Vim will keep the current setting. If an empty font list is given, Vim will try using other resource settings (for X, it will use the Vim.font resource), and finally it will try some builtin default which should always be there ("7x13" in the case of X). The font names given should be "normal" fonts. Vim will try to find the related bold and italic fonts.

For Win32, GTK, Motif, Mac OS and Photon:

```
:set guifont=*
```

will bring up a font requester, where you can pick the font you want.

The font name depends on the GUI used. See [setting-guifont](#) for a way to set '**guifont**' for various systems.

For the GTK+ 2 and 3 GUIs, the font name looks like this:

```
:set guifont=Andale\ Mono\ 11
```

That's all. XLFDS are not used. For Chinese this is reported to work well:

```
if has("gui_gtk2")
  set guifont=Bitstream\ Vera\ Sans\ Mono\ 12,Fixed\ 12
  set guifontwide=Microsoft\ Yahei\ 12,WenQuanYi\ Zen\ Hei\ 12
endif
```

(Replace gui_gtk2 with gui_gtk3 for the GTK+ 3 GUI)

For Mac OSX you can use something like this:

```
:set guifont=Monaco:h10
```

Mono-spaced fonts

[E236](#)

Note that the fonts must be mono-spaced (all characters have the same width). An exception is GTK: all fonts are accepted, but mono-spaced fonts look best.

To preview a font on X11, you might be able to use the "xfontsel" program. The "xlsfonts" program gives a list of all available fonts.

For the Win32 GUI

[E244](#) [E245](#)

- Takes these options in the font name (use a ':' to separate the options):
hXX - height is XX (points, can be floating-point)

- wXX - width is XX (points, can be floating-point)
 - WXX - weight is XX (see [Note](#) on Weights below)
 - b - bold. This is equivalent to setting the weight to 700.
 - i - italic
 - u - underline
 - s - strikethrough
 - cXX - character set XX. Valid charsets are: ANSI, ARABIC, BALTIC, CHINESEBIG5, DEFAULT, EASTEUROPE, GB2312, GREEK, HANGEUL, HEBREW, JOHAB, MAC, OEM, RUSSIAN, SHIFTJIS, SYMBOL, THAI, TURKISH and VIETNAMESE. Normally you would use "cDEFAULT".
 - qXX - quality XX. Valid quality names are: PROOF, DRAFT, ANTIALIASED, NONANTIALIASED, CLEARTEXT and DEFAULT. Normally you would use "qDEFAULT".
- Some quality values are not supported in legacy OSs.
- A '_' can be used in the place of a space, so you don't need to use backslashes to escape the spaces.

Examples:

```
:set guifont=courier_new:h12:w5:b:cRUSSIAN
:set guifont=Andale_Mono:h7.5:w4.5
```

See also [font-sizes](#) .

Note on Weights: Fonts often come with a variety of weights. "Normal" weights in Windows have a value of 400 and, left unspecified, this is the value that will be used when attempting to find fonts. Windows will often match fonts based on their weight with higher priority than the font name which means a Book or Medium variant of a font might be used despite specifying a Light or ExtraLight variant. If you are experiencing heavier weight substitution, then explicitly setting a lower weight value may mitigate against this unwanted substitution.

GUIFONTWIDE

[gui-fontwide](#)

When not empty, **'guifontwide'** specifies a comma-separated list of fonts to be used for double-width characters. The first font that can be loaded is used.

Note: The size of these fonts must be exactly twice as wide as the one specified with **'guifont'** and the same height. If there is a mismatch then the text will not be drawn correctly.

All GUI versions but GTK+:

'guifontwide' is only used when **'encoding'** is set to "utf-8" and **'guifontset'** is empty or invalid.

When **'guifont'** is set and a valid font is found in it and **'guifontwide'** is empty Vim will attempt to find a matching double-width font and set **'guifontwide'** to it.

GTK+ GUI only:

[guifontwide_gtk](#)

If set and valid, **'guifontwide'** is always used for double width characters, even if **'encoding'** is not set to "utf-8".

Vim does not attempt to find an appropriate value for **'guifontwide'**

automatically. If **'guifontwide'** is empty Pango/Xft will choose the font for characters not available in **'guifont'**. Thus you do not need to set **'guifontwide'** at all unless you want to override the choice made by Pango/Xft.

Windows +multibyte only: **guifontwide_win_mbyte**

If set and valid, **'guifontwide'** is used for IME instead of **'guifont'**.

7. Extras **gui-extras**

This section describes other features which are related to the GUI.

- With the GUI, there is no wait for one second after hitting escape, because the key codes don't start with `<Esc>`.
- Typing `^V` followed by a special key in the GUI will insert "`<Key>`", since the internal string used is meaningless. Modifiers may also be held down to get "`<Modifiers-Key>`".
- In the GUI, the modifiers SHIFT, CTRL, and ALT (or META) may be used within mappings of special keys and mouse events.
E.g.: `:map <M-LeftDrag> <LeftDrag>`
- In the GUI, several normal keys may have modifiers in mappings etc, these are `<Space>`, `<Tab>`, `<NL>`, `<CR>`, `<Esc>`.
- To check in a Vim script if the GUI is being used, you can use something like this:

```
if has("gui_running")
  echo "yes, we have a GUI"
else
  echo "Boring old console"
endif
```

setting-guifont

- When you use the same vimrc file on various systems, you can use something like this to set options specifically for each type of GUI:

```
if has("gui_running")
  if has("gui_gtk")
    :set guifont=Luxi\ Mono\ 12
  elseif has("x11")
    :set guifont=*-lucidatypewriter-medium-r-normal-*-*-180-*-*-m-*-*
  elseif has("gui_win32")
    :set guifont=Luxi_Mono:h12:cANSI
  endif
endif
```

A recommended Japanese font is MS Mincho. You can find info here:

<http://www.lexikan.com/mincho.htm>

8. Shell Commands **gui-shell**

For the X11 GUI the external commands are executed inside the gvim window.
See [gui-pty](#) .

WARNING: Executing an external command from the X11 GUI will not always work. "normal" commands like "ls", "grep" and "make" mostly work fine. Commands that require an intelligent terminal like "less" and "ispell" won't work. Some may even hang and need to be killed from another terminal. So be careful!

For the Win32 GUI the external commands are executed in a separate window.
See [gui-shell-win32](#) .

Vim's Win32 Graphical User Interface

gui-w32 win32-gui

- | | |
|-------------------------------|--------------------|
| 1. Starting the GUI | gui-w32-start |
| 2. Vim as default editor | vim-default-editor |
| 3. Using the clipboard | gui-clipboard |
| 4. Shell Commands | gui-shell-win32 |
| 5. Special colors | win32-colors |
| 6. Windows dialogs & browsers | gui-w32-dialogs |
| 7. Command line arguments | gui-w32-cmdargs |
| 8. Various | gui-w32-various |

Other relevant documentation:

- gui.txt For generic items of the GUI.
os_win32.txt For Win32 specific items.

1. Starting the GUI

gui-w32-start

The Win32 GUI version of Vim will always start the GUI, no matter how you start it or what it's called.

The GUI will always run in the Windows subsystem. Mostly shells automatically return with a command prompt after starting gvim. If not, you should use the "start" command:

```
start gvim [options] file ..
```

E988

The console version with the `-g` option may also start the GUI by executing gvim.exe:

```
vim -g [options] file ..
```

To make this work, gvim.exe must exist in the same directory as the vim.exe, and this feature must be enabled at compile time.

One may also use `:gui` from the console version. However, this is an experimental feature and this feature must be enabled at compile time. It uses a session file to recreate the current state of the console Vim in the GUI Vim.

Note: All fonts (bold, italic) must be of the same size!!! If you don't do this, text will disappear or mess up the display. Vim does not check the font sizes. It's the size in screen pixels that must be the same. **Note** that some fonts that have the same point size don't have the same pixel size! Additionally, the positioning of the fonts must be the same (ascent and descent).

The Win32 GUI has an extra menu item: "Edit/Select Font". It brings up the standard Windows font selector.

Setting the menu height doesn't work for the Win32 GUI.

`gui-win32-maximized`

If you want Vim to start with a maximized window, add this command to your vimrc or gvimrc file:

```
au GUIEnter * simalt ~x
```

Using Vim as a plugin

`gui-w32-windowid`

When gvim starts up normally, it creates its own top level window. If you pass Vim the command-line option `--windowid` with a decimal or hexadecimal value, Vim will create a window that is a child of the window with the given ID. This enables Vim to act as a plugin in another application. This really is a programmer's interface, and is of no use without a supporting application to spawn Vim correctly.

=====

2. Vim as default editor

`vim-default-editor`

To set Vim as the default editor for a file type:

1. Start a Windows Explorer
2. Choose View/Options -> File Types
3. Select the path to gvim for every file type that you want to use it for. (you can also use three spaces in the file type field, for files without an extension).

In the "open" action, use:

```
gvim "%1"
```

The quotes are required for using file names with embedded spaces.

You can also use this:

```
gvim "%L"
```

This should avoid short (8.3 character) file names in some situations. But I'm not sure if this works everywhere.

When you open a file in Vim by double clicking it, Vim changes to that file's directory.

If you want Vim to start full-screen, use this for the Open action:

```
gvim -c "simalt ~x" "%1"
```

Another method, which also works when you put Vim in another directory (e.g., when you have got a new version):

1. select a file you want to use Vim with
2. <Shift-F10>
3. select "Open With..." menu entry
4. click "Other..."
5. browse to the (new) location of Vim and click "Open"
6. make "Always Use this program..." checked
7. <OK>

`send-to-menu` `sendto`

You can also install Vim in the "Send To" menu:

1. Start a Windows Explorer
2. Navigate to your sendto directory:

- C:\Users\%user%\AppData\Roaming\Microsoft\Windows\SendTo .
3. Right-click in the file pane and select New->Shortcut
 4. Follow the shortcut wizard, using the full path to VIM/GVIM.

When you 'send a file to Vim', Vim changes to that file's directory. Note, however, that any long directory names will appear in their short (MS-DOS) form on some Windows versions. This is a limitation of the Windows "Send To" mechanism.

notepad

You could replace notepad.exe with gvim.exe, but that has a few side effects. Some programs rely on notepad arguments, which are not recognized by Vim. For example "notepad -p" is used by some applications to print a file. It's better to leave notepad where it is and use another way to start Vim.

win32-popup-menu

A more drastic approach is to install an "Edit with Vim" entry in the popup menu for the right mouse button. With this you can edit any file with Vim.

This can co-exist with the file associations mentioned above. The difference is that the file associations will make starting Vim the default action. With the "Edit with Vim" menu entry you can keep the existing file association for double clicking on the file, and edit the file with Vim when you want. For example, you can associate "*.mak" with your make program. You can execute the makefile by double clicking it and use the "Edit with Vim" entry to edit the makefile.

You can select any files and right-click to see a menu option called "Edit with gvim". Choosing this menu option will invoke gvim with the file you have selected. If you select multiple files, you will find two gvim-related menu options:

```
"Edit with multiple gvims" -- one gvim for each file in the selection
"Edit with single gvim"   -- one gvim for all the files in the selection
And if there already is a gvim running:
"Edit with existing gvim" -- edit the file with the running gvim
```

The "edit with existing Vim" entries can be disabled by adding an entry in the registry under HKLM\Software\Vim\Gvim, named DisableEditWithExisting, and with any value.

install-registry

You can add the "Edit with Vim" menu entry in an easy way by using the "install.exe" program. It will add several registry entries for you.

You can also do this by hand. This is complicated! Use the install.exe if you can.

1. Start the registry editor with "regedit".
2. Add these keys:

key	value name	value
HKEY_CLASSES_ROOT\CLSID\{51EEE242-AD87-11d3-9C1E-0090278BBD99}	{default}	Vim Shell Extension
HKEY_CLASSES_ROOT\CLSID\{51EEE242-AD87-11d3-9C1E-0090278BBD99}\InProcServer32	{default}	{path}\gvimext.dll
ThreadingModel		Apartment

```

HKEY_CLASSES_ROOT\*\shellex\ContextMenuHandlers\gvim
    {default}                                {51EEE242-AD87-11d3-9C1E-0090278BBD99}
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Shell Extensions\Approved
    {51EEE242-AD87-11d3-9C1E-0090278BBD99}
    Vim Shell Extension
HKEY_LOCAL_MACHINE\Software\Vim\Gvim
    path                                     {path}\gvim.exe
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\vim 8.2
    DisplayName                             Vim 8.2: Edit with Vim popup menu entry
    UninstallString                          {path}\uninstall.exe

```

Replace {path} with the path that leads to the executable.
 Don't type {default}, this is the value for the key itself.

To remove "Edit with Vim" from the popup menu, just remove the registry entries mentioned above. The "uninstall.exe" program can do this for you. You can also use the entry in the Windows standard "Add/Remove Programs" list.

If you notice that this entry overrules other file type associations, set those associations again by hand (using Windows Explorer, see above). This only seems to happen on some Windows NT versions (Windows bug?). Procedure:

1. Find the name of the file type. This can be done by starting the registry editor, and searching for the extension in \\HKEY_CLASSES_ROOT
2. In a Windows Explorer, use View/Options/File Types. Search for the file type in the list and click "Edit". In the actions list, you can select on to be used as the default (normally the "open" action) and click on the "Set Default" button.

Vim in the "Open With..." context menu win32-open-with-menu

If you use the Vim install program you have the choice to add Vim to the "Open With..." menu. This means you can use Vim to edit many files. Not every file (for unclear reasons...), thus the "Edit with Vim" menu entry is still useful.

One reason to add this is to be able to edit HTML files directly from Internet Explorer. To enable this use the "Tools" menu, "Internet Options..." entry. In the dialog select the "Programs" tab and select Vim in the "HTML editor" choice. If it's not there then installing didn't work properly.

Doing this manually can be done with this script:

```

-----
REGEDIT4

[HKEY_CLASSES_ROOT\Applications\gvim.exe]

[HKEY_CLASSES_ROOT\Applications\gvim.exe\shell]

[HKEY_CLASSES_ROOT\Applications\gvim.exe\shell\edit]

[HKEY_CLASSES_ROOT\Applications\gvim.exe\shell\edit\command]
@"c:\\vim\\vim82\\gvim.exe \"%1\""
```

```
[HKEY_CLASSES_ROOT\.htm\OpenWithList\gvim.exe]
```

```
[HKEY_CLASSES_ROOT\*\OpenWithList\gvim.exe]
```

Change the "c:\\vim\\vim82" bit to where gvim.exe is actually located.

To uninstall this run the Vim uninstall program or manually delete the registry entries with "regedit".

=====

3. Using the clipboard

[gui-clipboard](#)

Windows has a clipboard, where you can copy text to, and paste text from. Vim supports this in several ways. For other systems see [gui-selections](#).

The "*" register reflects the contents of the clipboard. [quotestar](#)

When the "unnamed" string is included in the '[clipboard](#)' option, the unnamed register is the same. Thus you can yank to and paste from the clipboard without prepending "*" to commands. If this doesn't work use the "unnamedplus" string in the '[clipboard](#)' option.

The 'a' flag in '[guioptions](#)' is not included by default. This means that text is only put on the clipboard when an operation is performed on it. Just Visually selecting text doesn't put it on the clipboard. When the 'a' flag is included, the text is copied to the clipboard even when it is not operated upon.

[mswin.vim](#)

To use the standard MS-Windows way of **CTRL-X**, **CTRL-C** and **CTRL-V**, use the \$VIMRUNTIME/mswin.vim script. You could add this line to your _vimrc file:
`source $VIMRUNTIME/mswin.vim`

Since **CTRL-C** is used to copy the text to the clipboard, it can't be used to cancel an operation. Use **CTRL-Break** for that.

CTRL-Z is used for undo. This means you can't suspend Vim with this key, use `:suspend` instead (if it's supported at all).

[CTRL-V-alternative](#) [CTRL-Q](#)

Since **CTRL-V** is used to paste, you can't use it to start a blockwise Visual selection. You can use **CTRL-Q** instead. You can also use **CTRL-Q** in Insert mode and Command-line mode to get the old meaning of **CTRL-V**. But **CTRL-Q** doesn't work for terminals when it's used for control flow.

NOTE: The clipboard support still has a number of bugs. See [todo](#).

=====

4. Shell Commands

[gui-shell-win32](#)

Vim uses another window for external commands, to make it possible to run any command. The external command gets its own environment for running, just like

it was started from a DOS prompt.

win32-vimrun

Executing an external command is done indirectly by the "vimrun" command. The "vimrun.exe" must be in the path for this to work. Or it must be in the same directory as the Vim executable. If "vimrun" cannot be found, the command is executed directly, but then the DOS window closes immediately after the external command has finished.

WARNING: If you close this window with the "X" button, and confirm the question if you really want to kill the application, Vim may be killed too! (This does not apply to commands run asynchronously with "!:start".)

The window in which the commands are executed will be the default you have set up for "Console" in Control Panel.

win32-!start

Normally, Vim waits for a command to complete before continuing (this makes sense for most shell commands which produce output for Vim to use). If you want Vim to start a program and return immediately, you can use the following syntax:

```
!:start [/min] {command}
```

The optional "/min" causes the window to be minimized.

5. Special colors

win32-colors

On Win32, the normal DOS colors can be used. See [dos-colors](#) .

Additionally the system configured colors can also be used. These are known by the names Sys_XXX, where XXX is the appropriate system color name, from the following list (see the Win32 documentation for full descriptions). Case is ignored.

Sys_3DDKShadow	Sys_3DFace	Sys_BTNFace
Sys_3DHiligh	Sys_3DHighlight	Sys_BTNHiligh
Sys_BTNHighlight	Sys_3DLight	Sys_3DShadow
Sys_BTNShadow	Sys_ActiveBorder	Sys_ActiveCaption
Sys_AppWorkspace	Sys_Background	Sys_Desktop
Sys_BTNTText	Sys_CaptionText	Sys_GrayText
Sys_Highlight	Sys_HighlightText	Sys_InactiveBorder
Sys_InactiveCaption	Sys_InactiveCaptionText	Sys_InfoBK
Sys_InfoText	Sys_Menu	Sys_MenuText
Sys_ScrollBar	Sys_Window	Sys_WindowFrame
Sys_WindowText		

Probably the most useful values are

Sys_Window	Normal window background
Sys_WindowText	Normal window text
Sys_Highlight	Highlighted background
Sys_HighlightText	Highlighted text

These extra colors are also available:

Gray, Grey, LightYellow, SeaGreen, Orange, Purple, SlateBlue, Violet,

Additionally, colors defined by a default color list can be used. For more info see `:colorscheme`. These colors used to be defined in `$VIMRUNTIME/rgb.txt`, now they are in `v:colornames` which is initialized from `$VIMRUNTIME/colors/lists/default.vim`.

=====
`gui-w32-dialogs` `dialog`

6. Windows dialogs & browsers

The Win32 GUI can use familiar Windows components for some operations, as well as the traditional interface shared with the console version.

6.1 Dialogs

The dialogs displayed by the "confirm" family (i.e. the `'confirm'` option, `:confirm` command and `confirm()` function) are GUI-based rather than the console-based ones used by other versions. The `'c'` flag in `'guioptions'` changes this.

6.2 File Browsers

When prepending `":browse"` before file editing commands, a file requester is used to allow you to select an existing file. See `:browse`.

6.3 Tearoff Menus

The Win32 GUI emulates Motif's tear-off menus. At the top of each menu you will see a small graphic "rip here" sign. Selecting it will cause a floating window to be created with the same menu entries on it. The floating menu can then be accessed just as if it was the original (including sub-menus), but without having to go to the menu bar each time.

This is most useful if you find yourself using a command buried in a sub-menu over and over again.

The tearoff menus can be positioned where you like, and always stay just above the Main Vim window. You can get rid of them by closing them as usual; they also of course close when you exit Vim.

`:te[aroff] {name}` `:tearoff` `:te`
Tear-off the menu `{name}`. The menu named must have at least one subentry, but need not appear on the menu-bar (see `win32-hidden-menus`).

Example:

`:tearoff File`
will make the "File" menu (if there is one) appear as a tearoff menu.

`:amenu]Toolbar.Make :make<CR>`
`:tearoff]Toolbar`

This creates a floating menu that doesn't exist on the main menu-bar.

Note that a menu that starts with ']' will not be displayed.

7. Command line arguments

gui-w32-cmdargs

Command line arguments behave the same way as with the console application, see [win32-cmdargs](#) .

8. Various

gui-w32-various

gui-w32-printing

The "File/Print" menu prints the text with syntax highlighting, see [:hardcopy](#) . If you just want to print the raw text and have a default printer installed this should also work:

```
:w >>prn
```

Vim supports a number of standard MS-Windows features. Some of these are detailed elsewhere: see ['mouse'](#) , [win32-hidden-menus](#) .

drag-n-drop-win32

You can drag and drop one or more files into the Vim window, where they will be opened as normal. See [drag-n-drop](#) .

:simalt :sim

```
:sim[alt] {key}      simulate pressing {key} while holding Alt pressed.  
                    {only for Win32 versions}  
Note: ":si" means ":s" with the "i" flag.
```

Normally, Vim takes control of all Alt-<Key> combinations, to increase the number of possible mappings. This clashes with the standard use of Alt as the key for accessing menus.

The quick way of getting standard behavior is to set the ['winaltkeys'](#) option to "yes". This however prevents you from mapping Alt keys at all.

Another way is to set ['winaltkeys'](#) to "menu". Menu shortcut keys are then handled by windows, other ALT keys can be mapped. This doesn't allow a dependency on the current state though.

To get round this, the [:simalt](#) command allows Vim (when ['winaltkeys'](#) is not "yes") to fake a Windows-style Alt keypress. You can use this to map Alt key combinations (or anything else for that matter) to produce standard Windows actions. Here are some examples:

```
:map <M-f> :simalt f<CR>
```

This makes Alt-F pop down the 'File' menu (with the stock Menu.vim) by simulating the keystrokes Alt, F.

```
:map <M-Space> :simalt ~<CR>
```

This maps Alt-Space to pop down the system menu for the Vim window. Note that ~ is used by simalt to represent the <Space> character.

```
:map <C-n> :simalt ~n<CR>
```

Maps Control-N to produce the keys Alt-Space followed by N. This minimizes the Vim window via the system menu.

Note that the key changes depending on the language you are using.

intellimouse-wheel-problems

When using the Intellimouse mouse wheel causes Vim to stop accepting input, go to:

```
ControlPanel - Mouse - Wheel - UniversalScrolling - Exceptions
```

And add gvim to the list of applications. This problem only appears to happen with the Intellimouse driver 2.2 and when "Universal Scrolling" is turned on.

XPM support

w32-xpm-support

GVim can be built on MS-Windows with support for XPM files. [+xpm_w32](#)
See the Make_mvc.mak file for instructions, search for XPM.

To try out if XPM support works do this:

```
:help  
:let runtime = escape($VIMRUNTIME, ' \')  
:exe 'sign define vimxpm icon=' .. runtime .. '\\vim16x16.xpm'  
:exe 'sign place 1 line=1 name=vimxpm file=' .. expand('%:~')
```

You may need to get the vim16x16.xpm file from github:

<https://github.com/vim/vim/blob/master/runtime/vim16x16.xpm>

Keycode translation strategy

w32-experimental-keycode-trans-strategy

In Patch v8.2.4807 W32 GVIM was changed over to experimental keycode translation method with the aim to be able to use more keyboard shortcuts and especially supporting non-standard keyboard layouts. In order to implement this support Win API TranslateMessage() call was dropped, and instead the recognition of keycode was changed over to ToUnicode() Win API call. This approach uncovered numerous corner cases, which are apparently covered by TranslateMessage() implementation, each of it is necessary to be dealt with on an individual basis. Therefore the decision was taken to declare this functionality experimental for the time being and to recover "classic" keycode translation method as default again.

Discussion about use of "experimental" keycode translation method will probably last some time yet. In the meantime, if you are impacted by this change over back to "classic" keycode translation method in W32 GVIM, you can enable "experimental" translation method again in your vimrc using following snippet:

```
:call test_mswin_event('set_keycode_trans_strategy', {'strategy': 'experimental'})
```

Similarly, in case you need to turn back "classic" keycode translation method (for example for testing purposes), please use:

```
:call test_mswin_event('set_keycode_trans_strategy', {'strategy': 'classic'})
```

Alternatively (this method is especially useful for the TINY GVIM build, where test_mswin_event() cannot be called), an environment variable VIM_KEYCODE_TRANS_STRATEGY can be set to the desired value ("experimental" or "classic"), to override the default, e.g., type in dos prompt:

```
set VIM_KEYCODE_TRANS_STRATEGY=experimental
gvim.exe
```


Vim's Graphical User Interface

gui-x11 GUI-X11
Motif

- 1. Starting the X11 GUI [gui-x11-start](#)
- 2. GUI Resources [gui-resources](#)
- 3. Shell Commands [gui-pty](#)
- 4. Various [gui-x11-various](#)
- 5. GTK version [gui-gtk](#)
- 6. GNOME version [gui-gnome](#)
- 7. KDE version [gui-kde](#)
- 8. Compiling [gui-x11-compiling](#)
- 9. X11 selection mechanism [x11-selection](#)

Other relevant documentation:
[gui.txt](#) For generic items of the GUI.

=====

1. Starting the X11 GUI [gui-x11-start](#) E665

Then you can run the GUI version of Vim in either of these ways:

```
gvim [options] [files...]
vim -g [options] [files...]
```

So if you call the executable "gvim", or make "gvim" a link to the executable, then the GUI version will automatically be used. Additional characters may be added after "gvim", for example "gvim-5".

You may also start up the GUI from within the terminal version by using one of these commands:

```
:gui [++opt] [+cmd] [-f|-b] [files...] :gu :gui
:gvim [++opt] [+cmd] [-f|-b] [files...] :gv :gvim
```

The "-f" option runs Vim in the foreground.
The "-b" option runs Vim in the background (this is the default).
Also see [++opt](#) and [+cmd](#) .

[gui-fork](#)

When the GUI is started, it does a fork() and exits the current process. When gvim was started from a shell this makes the shell accept further commands. If you don't want this (e.g. when using gvim for a mail program that waits for gvim to exit), start gvim with "gvim -f", "vim -gf" or use ":gui -f". Don't use "vim -fg", because "-fg" specifies the foreground color.

When using "vim -f" and then ":gui", Vim will run in the foreground. The "-f" argument will be remembered. To force running Vim in the background use ":gui -b".

"gvim --nofork" does the same as "gvim -f".

When there are running jobs Vim will not fork, because the processes would no longer be child processes.

E851 E852

When starting the GUI fails Vim will try to continue running in the terminal.

If you want the GUI to run in the foreground always, include the 'f' flag in '**guioptions**'. `-f` .

2. GUI Resources

`gui-resources` `.Xdefaults`

If using the Motif version of the GUI (not for the KDE, GTK+ or Win32 version), a number of X resources are available. You should use Vim's class "Vim" when setting these. They are as follows:

Resource name	Meaning
<code>reverseVideo</code>	Boolean: should reverse video be used?
<code>background</code>	Color of background.
<code>foreground</code>	Color of normal text.
<code>scrollBackground</code>	Color of trough portion of scrollbars.
<code>scrollForeground</code>	Color of slider and arrow portions of scrollbars.
<code>menuBackground</code>	Color of menu backgrounds.
<code>menuForeground</code>	Color of menu foregrounds.
<code>tooltipForeground</code>	Color of tooltip and balloon foreground.
<code>tooltipBackground</code>	Color of tooltip and balloon background.
<code>font</code>	Name of font used for normal text.
<code>boldFont</code>	Name of font used for bold text.
<code>italicFont</code>	Name of font used for italic text.
<code>boldItalicFont</code>	Name of font used for bold, italic text.
<code>menuFont</code>	Name of font used for the menus, used when compiled without the <code>+xfontset</code> feature
<code>menuFontSet</code>	Name of fontset used for the menus, used when compiled with the <code>+xfontset</code> feature
<code>tooltipFont</code>	Name of the font used for the tooltip and balloons. When compiled with the <code>+xfontset</code> feature this is a fontset name.
<code>geometry</code>	Initial geometry to use for gvim's window (default is same size as terminal that started it).
<code>scrollbarWidth</code>	Thickness of scrollbars.
<code>borderWidth</code>	Thickness of border around text area.

A special font for italic, bold, and italic-bold text will only be used if the user has specified one via a resource. No attempt is made to guess what fonts should be used for these based on the normal text font.

Note that the colors can also be set with the `":highlight"` command, using the "Normal", "Menu", "Tooltip", and "Scrollbar" groups. Example:

```
:highlight Menu guibg=lightblue
:highlight Tooltip guibg=yellow
:highlight Scrollbar guibg=lightblue guifg=blue
```

```
:highlight Normal guibg=grey90
```

font-sizes

Note: All fonts (except for the menu and tooltip) must be of the same size!!! If you don't do this, text will disappear or mess up the display. Vim does not check the font sizes. It's the size in screen pixels that must be the same. **Note** that some fonts that have the same point size don't have the same pixel size! Additionally, the positioning of the fonts must be the same (ascent and descent). You can check this with "xlsfonts -l {fontname}".

If any of these things are also set with Vim commands, e.g. with ":set guifont=Screen15", then this will override the X resources (currently '**guifont**' is the only option that is supported).

Here is an example of what you might put in your ~/.Xdefaults file:

```
Vim*useSchemes:          all
Vim*sgiMode:             true
Vim*useEnhancedFSB:     true
Vim.foreground:         Black
Vim.background:         Wheat
Vim*fontList:            7x13
```

The first three of these are standard resources on Silicon Graphics machines which make Motif applications look even better, highly recommended!

The "Vim*fontList" is to set the menu font for Motif. Example:

```
Vim*menuBar*fontList:   -*courier-medium-r-*-*10-*-*-*-*-*-*
```

NOTE: A more portable, and indeed more correct, way to specify the menu font in Motif is through the resource:

```
Vim.menuFont:          -*courier-medium-r-*-*10-*-*-*-*-*-*
```

Or, when compiled with the `+xfontset` feature:

```
Vim.menuFontSet:       -*courier-medium-r-*-*10-*-*-*-*-*-*
```

Don't use "Vim*geometry" in the defaults. This will break the menus. Use "Vim.geometry" instead.

If you get an error message "Cannot allocate colormap entry for "gray60", try adding this to your Vim resources (change the colors to your liking):

```
Vim*scrollBackground:  Black
Vim*scrollForeground:  Blue
```

The resources can also be set with arguments to Vim:

argument	meaning	
-display {display}	Run vim on {display}	-gui
-iconic	Start vim iconified	-display
-background {color}	Use {color} for the background	-iconic
-bg {color}	idem	-background
-foreground {color}	Use {color} for normal text	-bg
-fg {color}	idem	-foreground
		-fg

-ul {color}	idem	-ul
-font {font}	Use {font} for normal text	-font
-fn {font}	idem	-fn
-boldfont {font}	Use {font} for bold text	-boldfont
-italicfont {font}	Use {font} for italic text	-italicfont
-menufont {font}	Use {font} for menu items	-menufont
-menufontset {fontset}	Use {fontset} for menu items	-menufontset
-mf {font}	idem	-mf
-geometry {geom}	Use {geom} for initial geometry	-geometry
-geom {geom}	idem, see -geometry-example	-geom
-borderwidth {width}	Use a border width of {width}	-borderwidth
-bw {width}	idem	-bw
		-scrollbarwidth
-scrollbarwidth {width}	Use a scrollbar width of {width}	
-sw {width}	idem	-sw
-menuheight {height}	Use a menu bar height of {height}	-menuheight
-mh {height}	idem	-mh
	NOTE: On Motif the value is ignored, the menu height is computed to fit the menus.	
-reverse	Use reverse video	-reverse
-rv	idem	-rv
+reverse	Don't use reverse video	+reverse
+rv	idem	+rv
-xrm {resource}	Set the specified resource	-xrm

Note about reverse video: Vim checks that the result is actually a light text on a dark background. The reason is that some X11 versions swap the colors, and some don't. These two examples will both give yellow text on a blue background:

```
gvim -fg Yellow -bg Blue -reverse
gvim -bg Yellow -fg Blue -reverse
```

[-geometry-example](#)

An example for the geometry argument:

```
gvim -geometry 80x63+8+100
```

This creates a window with 80 columns and 63 lines at position 8 pixels from the left and 100 pixels from the top of the screen.

3. Shell Commands

[gui-pty](#)

WARNING: Executing an external command from the GUI will not always work. "normal" commands like "ls", "grep" and "make" mostly work fine. Commands that require an intelligent terminal like "less" and "ispell" won't work. Some may even hang and need to be killed from another terminal. So be careful!

There are two ways to do the I/O with a shell command: Pipes and a pseudo-tty. The default is to use a pseudo-tty. This should work best on most systems.

Unfortunately, the implementation of the pseudo-tty is different on every Unix system. And some systems require root permission. To avoid running into problems with a pseudo-tty when you least expect it, test it when not editing a file. Be prepared to "kill" the started command or Vim. Commands like

":r !cat" may hang!

If using a pseudo-tty does not work for you, reset the **'guipty'** option:

```
:set noguipty
```

Using a pipe should work on any Unix system, but there are disadvantages:

- Some shell commands will notice that a pipe is being used and behave differently. E.g., "!:ls" will list the files in one column.
- The ":sh" command won't show a prompt, although it will sort of work.
- When using ":make" it's not possible to interrupt with a **CTRL-C**.

Typeahead while the external command is running is often lost. This happens both with a pipe and a pseudo-tty. This is a known problem, but it seems it can't be fixed (or at least, it's very difficult).

gui-pty-erase

When your erase character is wrong for an external command, you should fix this in your "~/.cshrc" file, or whatever file your shell uses for initializations. For example, when you want to use backspace to delete characters, but hitting backspaces produces "^H" instead, try adding this to your "~/.cshrc":

```
stty erase ^H
```

The ^H is a real **CTRL-H**, type it as **CTRL-V CTRL-H**.

4. Various

gui-x11-various

gui-x11-printing

The "File/Print" menu simply sends the current buffer to "lpr". No options or whatever. If you want something else, you can define your own print command. For example:

```
:10amenu File.Print :w !lpr -Php3  
:10vmenu File.Print :w !lpr -Php3
```

X11-icon

Vim uses a black&white icon by default when compiled with Motif. A colored Vim icon is included as \$VIMRUNTIME/vim32x32.xpm. For GTK+, this is the builtin icon used. Unfortunately, how you should install it depends on your window manager. When you use this, remove the 'i' flag from **'guioptions'**, to remove the black&white icon:

```
:set guioptions-=i
```

If you use one of the fvwm* family of window managers simply add this line to your .fvwm2rc configuration file:

```
Style "vim" Icon vim32x32.xpm
```

Make sure the icon file's location is consistent with the window manager's ImagePath statement. Either modify the ImagePath from within your .fvwm2rc or drop the icon into one the pre-defined directories:

```
ImagePath /usr/X11R6/include/X11/pixmaps:/usr/X11R6/include/X11/bitmaps
```

Note: older versions of fvwm use "IconPath" instead of "ImagePath".

For CDE "dtwm" (a derivative of Motif) add this line in the .Xdefaults:
`Dtwm*Vim*iconImage: /usr/local/share/vim/vim32x32.xpm`

For "mwm" (Motif window manager) the line would be:
`Mwm*Vim*iconImage: /usr/local/share/vim/vim32x32.xpm`

Mouse Pointers Available in X11

X11_mouse_shapes

By using the `'mousethape'` option, the mouse pointer can be automatically changed whenever Vim enters one of its various modes (e.g., Insert or Command). Currently, the available pointers are:

arrow	an arrow pointing northwest
beam	a I-like vertical bar
size	an arrow pointing up and down
busy	a wristwatch
blank	an invisible pointer
crosshair	a thin "+" sign
hand1	a dark hand pointing northeast
hand2	a light hand pointing northwest
pencil	a pencil pointing southeast
question	question_arrow
right_arrow	an arrow pointing northeast
up_arrow	an arrow pointing upwards

Additionally, any of the mouse pointers that are built into X11 may be used by specifying an integer from the X11/cursorfont.h include file.

If a name is used that exists on other systems, but not in X11, the default "arrow" pointer is used.

5. GTK version

`gui-gtk` `GTK+` `GTK` `GTK3`

The GTK version of the GUI works a little bit different.

GTK does not use the traditional X resource settings. Thus items in your `~/.Xdefaults` or `app-defaults` files are not used.

Many of the traditional X command line arguments are not supported. (e.g., stuff like `-bg`, `-fg`, etc). The ones that are supported are:

command line argument	resource name	meaning
<code>-fn</code> or <code>-font</code>	<code>.font</code>	font name for the text
<code>-geom</code> or <code>-geometry</code>	<code>.geometry</code>	size of the gvim window
<code>-rv</code> or <code>-reverse</code>	<code>*reverseVideo</code>	white text on black background
<code>-display</code>		display to be used
<code>-fg</code> <code>-foreground</code> {color}		foreground color
<code>-bg</code> <code>-background</code> {color}		background color

To set the font, see `'guifont'` . For GTK, there's also a menu option that

does this.

Additionally, there are these command line arguments, which are handled by GTK internally. Look in the GTK documentation for how they are used:

```
--sync
--gdk-debug
--gdk-no-debug
--no-xshm      (not in GTK+ 2)
--xim-preedit  (not in GTK+ 2)
--xim-status   (not in GTK+ 2)
--gtk-debug
--gtk-no-debug
--g-fatal-warnings
--gtk-module
--display      (GTK+ counterpart of -display; works the same way.)
--screen       (The screen number; for GTK+ 2.2 multihead support.)
```

These arguments are ignored when the `+netbeans_intg` feature is used:

```
-xrm
-mf
```

As for colors, Vim's color settings (for syntax highlighting) is still done the traditional Vim way. See `:highlight` for more help.

If you want to set the colors of remaining gui components (e.g., the menubar, scrollbar, whatever), those are GTK specific settings and you need to set those up in some sort of `gtkrc` file. You'll have to refer to the GTK documentation, however little there is, on how to do this. See <http://developer.gnome.org/doc/API/2.0/gtk/gtk-Resource-Files.html> for more information.

`gtk3-slow`

If you are using GTK3 and Vim appears to be slow, try setting the environment variable `$GDK_RENDERING` to "image".

Tooltip Colors

`gtk-tooltip-colors`

Example, which sets the tooltip colors to black on light-yellow:

```
style "tooltips"
{
    bg[NORMAL] = "#ffffcc"
    fg[NORMAL] = "#000000"
}

widget "gtk-tooltips*"          style "tooltips"
```

Write this in the file `~/.gtkrc` and it will be used by GTK+. For GTK+ 2 you might have to use the file `~/.gtkrc-2.0` instead, depending on your distribution.

For GTK+ 3, an effect similar to the above can be obtained by adding the following snippet of CSS code to `$XDG_HOME_DIR/gtk-3.0/gtk.css` (see the next section):

For GTK+ 3 < 3.20:

```
.tooltip {
    background-color: #ffffcc;
    color: #000000;
}
```

For GTK+ 3 >= 3.20:

```
tooltip {
    background-color: #ffffcc;
    text-shadow: none;
}

tooltip label {
    color: #2e3436;
}
```

A Quick Look at GTK+ CSS

[gtk-css](#)

The contents of this subsection apply to GTK+ 3.20 or later which provides stable support for GTK+ CSS:

<https://developer.gnome.org/gtk3/stable/theming.html>

GTK+ uses CSS for styling and layout of widgets. In this subsection, we'll have a quick look at GTK+ CSS through simple, illustrative examples.

You can usually edit the config with:

```
vim $HOME/.config/gtk-3.0/gtk.css
```

Example 1. Empty Space Adjustment

By default, the toolbar and the tabline of the GTK+ 3 GUI are somewhat larger than those of the GTK+ 2 GUI. Some people may want to make them look similar to the GTK+ 2 GUI in size.

To do that, we'll try reducing empty space around icons and labels that looks apparently superfluous.

Add the following lines to `$XDG_HOME_DIR/gtk-3.0/gtk.css` (usually, `$HOME/.config/gtk-3.0/gtk.css`):

```
toolbar button {
    margin-top: -2px;
    margin-right: 0px;
    margin-bottom: -2px;
    margin-left: 0px;

    padding-top: 0px;
    padding-right: 0px;
```



```

padding-bottom: 0px;
padding-left: 0px
}

notebook tab {
margin-top: -1px;
margin-right: 3px;
margin-bottom: -1px;
margin-left: 3px;

padding-top: 0px;
padding-right: 0px;
padding-bottom: 0px;
padding-left: 0px
}

```

Since it's a CSS, they can be rewritten using shorthand:

```

toolbar button {
margin: -2px 0px;
padding: 0px;
}

notebook tab {
margin: -1px 3px;
padding: 0px
}

```

Note: You might want to use **'toolbariconsize'** to adjust the icon size, too.

Note: Depending on the icon theme and/or the font in use, some extra tweaks may be needed for a satisfactory result.

Note: In addition to margin and padding, you can use border. For details, refer to the box model of CSS, e.g.,

https://www.w3schools.com/css/css_boxmodel.asp

Example 2. More Than Just Colors

GTK+ CSS supports gradients as well:

```

tooltip {
background-image: -gtk-gradient(linear,
                                0 0, 0 1,
                                color-stop(0, #344752),
                                color-stop(0.5, #546772),
                                color-stop(1, #243742));
}

tooltip label {
color: #f3f3f3;
}

```

Gradients can be used to make a GUI element visually distinguishable from others without relying on high contrast. Accordingly, effective use of them is a useful technique to give a theme a sense of unity in color and luminance.

Note: Theming can be difficult since it must make every application look equally good; making a single application more charming often gets others unexpectedly less attractive or even deteriorates their usability. Keep this in mind always when you try improving a theme.

Example 3. border color

To eliminate borders when maximized:

```
@define-color bg_color #1B2B34;
#vim-main-window {
    background-color: @bg_color;
}
```

Using Vim as a GTK+ plugin

[gui-gtk-socketid](#)

When the GTK+ version of Vim starts up normally, it creates its own top level window (technically, a 'GtkWindow'). GTK+ provides an embedding facility with its GtkSocket and GtkPlug widgets. If one GTK+ application creates a GtkSocket widget in one of its windows, an entirely different GTK+ application may embed itself into the first application by creating a top-level GtkPlug widget using the socket's ID.

If you pass Vim the command-line option '--socketid' with a decimal or hexadecimal value, Vim will create a GtkPlug widget using that value instead of the normal GtkWindow. This enables Vim to act as a GTK+ plugin.

This really is a programmer's interface, and is of no use without a supporting application to spawn the Vim correctly. For more details on GTK+ sockets, see <http://www.gtk.org/api/>

Note that this feature requires the latest GTK version. GTK 1.2.10 still has a small problem. The socket feature has not yet been tested with GTK+ 2 -- feel free to volunteer.

6. GNOME version

[gui-gnome](#) [Gnome](#) [GNOME](#)

The GNOME GUI works just like the GTK+ version. See [GTK+](#) above for how it works. It looks a bit different though, and implements one important feature that's not available in the plain GTK+ GUI: Interaction with the session manager. [gui-gnome-session](#)

These are the different looks:

- Uses GNOME dialogs (GNOME 1 only). The GNOME 2 GUI uses the same nice dialogs as the GTK+ 2 version.
- Uses the GNOME dock, so that the toolbar and menubar can be moved to different locations other than the top (e.g., the toolbar can be placed on

the left, right, top, or bottom). The placement of the menubar and toolbar is only saved in the GNOME 2 version.

- That means the menubar and toolbar handles are back! Yeah! And the resizing grid still works too.

GNOME is compiled with if it was found by configure and the `--enable-gnome-check` argument was used.

Note: Avoid use of `--enable-gnome-check` with GTK+ 3 GUI build. The functionality mentioned above is consolidated in GTK+ 3.

GNOME session support

`gui-gnome-session` `gnome-session`

On logout, Vim shows the well-known exit confirmation dialog if any buffers are modified. Clicking `[Cancel]` will stop the logout process. Otherwise the current session is stored to disk by using the `:mksession` command, and restored the next time you log in.

The GNOME session support should also work with the KDE session manager. If you are experiencing any problems please report them as bugs.

Note: The automatic session save works entirely transparent, in order to avoid conflicts with your own session files, scripts and autocommands. That means in detail:

- The session file is stored to a separate directory (usually `$HOME/.gnome2`).
- `'sessionoptions'` is ignored, and a hardcoded set of appropriate flags is used instead:

`blank,curdir,folds,globals,help,options,tabpages,winsize`

- The internal variable `v:this_session` is not changed when storing the session. Also, it is restored to its old value when logging in again.

The position and size of the GUI window is not saved by Vim since doing so is the window manager's job. But if compiled with GTK+ 2 support, Vim helps the WM to identify the window by restoring the window role (using the `--role` command line argument).

7. KDE version

`gui-kde` `kde` `KDE` `KVim`
`gui-x11-kde`

There is no KDE version of Vim. There has been some work on a port using the Qt toolkit, but it never worked properly and it has been abandoned. Work continues on Yzis: <https://github.com/chrizel/Yzis> but it seems also abandoned.

8. Compiling

`gui-x11-compiling`

If using X11, Vim's configure will by default first try to find the necessary GTK+ files on your system. When both GTK+ 2 and GTK+ 3 are available, GTK+ 2 will be chosen unless `--enable-gui=gtk3` is passed explicitly to configure.

If the GTK+ files cannot be found, then the Motif files will be searched for. If both fail, the GUI will be disabled.

For GTK+, Vim's configuration process uses pkg-config(1) to check if the GTK+ required for a specified build is properly installed and usable. Accordingly, it is a good idea to make sure before running configure that your system has a working pkg-config together with the .pc file of the required GTK+. For that, say, run the following on the command line to see if your pkg-config works with your GTK+ 2:

```
$ pkg-config --modversion gtk+-2.0
```

Replace gtk+-2.0 with gtk+-3.0 for GTK+ 3. If you get the correct version number of your GTK+, you can proceed; if not, you probably need to do some system administration chores to set up pkg-config and GTK+ correctly.

The GTK+ 2 GUI is built by default. Therefore, you usually don't need to pass any options such as --enable-gui=gtk2 to configure and build that.

Optionally, the GTK+ 2 GUI can consolidate the GNOME 2 support. This support is enabled by passing --enable-gnome-check to configure.

If you want to build the GTK+ 3 GUI, you have to pass --enable-gui=gtk3 explicitly to configure, and avoid passing --enable-gnome-check to that, as the functionality of the GNOME 2 support has already been consolidated in GTK+ 3.

Otherwise, if you are using Motif, when you have the Motif files in a directory where configure doesn't look, edit the Makefile to enter the names of the directories. Search for "GUI_INC_LOC" for an example to set the Motif directories.

[gui-x11-gtk](#)

Currently, Vim supports both GTK+ 2 and GTK+ 3.

The GTK+ 2 GUI requires GTK+ 2.2 or later.

Although the GTK+ 3 GUI is written in such a way that the source code can be compiled against all versions of the 3.x series, we recommend GTK+ 3.10 or later because of its substantial implementation changes in redraw done at that version.

[gui-x11-motif](#)

For Motif, you need at least Motif version 1.2 and/or X11R5. Motif 2.0 and X11R6 are OK. Motif 1.1 and X11R4 might work, no guarantee (there may be a few problems, but you might make it compile and run with a bit of work, please send me the patches if you do). The newest releases of LessTif have been reported to work fine too.

[gui-x11-athena](#) [gui-x11-neXtaw](#)

Support for the Athena GUI and neXtaw was removed in patch 8.2.4677.

[gui-x11-misc](#)

In general, do not try to mix files from different GTK+, Motif and X11 versions. This will cause problems. For example, using header files for X11R5 with a library for X11R6 probably doesn't work (although the linking

won't give an error message, Vim will crash later).

gui-wayland

Initial support for the Wayland display server protocol has landed in patch 9.1.0064. To enable it, you need to set the environment variable "\$GVIM_ENABLE_WAYLAND" in your shell.

Note: The Wayland protocol is subject to some restrictions, so the following functions won't work: `getwinpos()`, `getwinposx()`, `getwinposy()` and the `v:windowid` variable won't be available.

9. X11 selection mechanism

x11-selection

If using X11, in either the GUI or an xterm with an X11-aware Vim, then Vim provides varied access to the X11 selection and clipboard. These are accessed by using the two selection registers "*" and "+.

X11 provides two basic types of global store, selections and cut-buffers, which differ in one important aspect: selections are "owned" by an application, and disappear when that application (e.g., Vim) exits, thus losing the data, whereas cut-buffers, are stored within the X-server itself and remain until written over or the X-server exits (e.g., upon logging out).

The contents of selections are held by the originating application (e.g., upon a copy), and only passed on to another application when that other application asks for them (e.g., upon a paste).

The contents of cut-buffers are immediately written to, and are then accessible directly from the X-server, without contacting the originating application.

quoteplus quote+

There are three documented X selections: PRIMARY (which is expected to represent the current visual selection - as in Vim's Visual mode), SECONDARY (which is ill-defined) and CLIPBOARD (which is expected to be used for cut, copy and paste operations).

Of these three, Vim uses PRIMARY when reading and writing the "*" register (hence when the X11 selections are available, Vim sets a default value for '`clipboard`' of "autoselect"), and CLIPBOARD when reading and writing the "+" register. Vim does not access the SECONDARY selection.

This applies both to the GUI and the terminal version. For non-X11 systems the plus and the star register both use the system clipboard.

Examples: (assuming the default option values)

- Select a URL in Visual mode in Vim. Go to your browser and click the middle mouse button in the URL text field. The selected text will be inserted (hopefully!). **Note:** in Firefox you can set the `middlemouse.contentLoadURL` preference to true in `about:config`, then the selected URL will be used when pressing middle mouse button in most places in the window.
- Select some text in your browser by dragging with the mouse. Go to Vim and

press the middle mouse button: The selected text is inserted.

- Select some text in Vim and do "+y. Go to your browser, select some text in a textfield by dragging with the mouse. Now use the right mouse button and select "Paste" from the popup menu. The selected text is overwritten by the text from Vim.

Note that the text in the "+" register remains available when making a Visual selection, which makes other text available in the "*" register. That allows overwriting selected text.

x11-cut-buffer

There are, by default, 8 cut-buffers: CUT_BUFFER0 to CUT_BUFFER7. Vim only uses CUT_BUFFER0, which is the one that xterm uses by default.

Whenever Vim is about to become unavailable (either via exiting or becoming suspended), and thus unable to respond to another application's selection request, it writes the contents of any owned selection to CUT_BUFFER0. If the "+ CLIPBOARD selection is owned by Vim, then this is written in preference, otherwise if the "*" PRIMARY selection is owned by Vim, then that is written.

Similarly, when Vim tries to paste from "*" or "+" (either explicitly, or, in the case of the "*" register, when the middle mouse button is clicked), if the requested X selection is empty or unavailable, Vim reverts to reading the current value of the CUT_BUFFER0.

Note that when text is copied to CUT_BUFFER0 in this way, the type of selection (character, line or block) is always lost, even if it is a Vim which later pastes it.

Xterm, by default, always writes visible selections to both PRIMARY and CUT_BUFFER0. When it pastes, it uses PRIMARY if this is available, or else falls back upon CUT_BUFFER0. For this reason, when cutting and pasting between Vim and an xterm, you should use the "*" register. Xterm doesn't use CLIPBOARD, thus the "+" doesn't work with xterm.

Most newer applications will provide their current selection via PRIMARY ("*") and use CLIPBOARD ("+") for cut/copy/paste operations. You thus have access to both by choosing to use either of the "*" or "+" registers.

This document explains how to use Vim's cscope interface.

Cscope is a tool like ctags, but think of it as ctags on steroids since it does a lot more than what ctags provides. In Vim, jumping to a result from a cscope query is just like jumping to any tag; it is saved on the tag stack so that with the right keyboard mappings, you can jump back and forth between functions as you normally would with tags .

- | | |
|-------------------------------|------------------------------------|
| 1. Cscope introduction | cscope-intro |
| 2. Cscope related commands | cscope-commands |
| 3. Cscope options | cscope-options |
| 4. How to use cscope in Vim | cscope-howtouse |
| 5. Limitations | cscope-limitations |
| 6. Suggested usage | cscope-suggestions |
| 7. Availability & Information | cscope-info |

This is currently for Unix and Win32 only.

-
1. Cscope introduction [cscope-intro](#)

The following text is taken from a version of the cscope man page:

Cscope is an interactive screen-oriented tool that helps you:

Learn how a C program works without endless flipping through a thick listing.

Locate the section of code to change to fix a bug without having to learn the entire program.

Examine the effect of a proposed change such as adding a value to an enum variable.

Verify that a change has been made in all source files such as adding an argument to an existing function.

Rename a global variable in all source files.

Change a constant to a preprocessor symbol in selected lines of files.

It is designed to answer questions like:

Where is this symbol used?

Where is it defined?

Where did this variable get its value?

What is this global symbol's definition?

Where is this function in the source files?
What functions call this function?
What functions are called by this function?
Where does the message "out of space" come from?
Where is this source file in the directory structure?
What files include this header file?

Cscope answers these questions from a symbol database that it builds the first time it is used on the source files. On a subsequent call, cscope rebuilds the database only if a source file has changed or the list of source files is different. When the database is rebuilt the data for the unchanged files is copied from the old database, which makes rebuilding much faster than the initial build.

When cscope is normally invoked, you will get a full-screen selection screen allowing you to make a query for one of the above questions. However, once a match is found to your query and you have entered your text editor to edit the source file containing match, you cannot simply jump from tag to tag as you normally would with vi's Ctrl-] or :tag command.

Vim's cscope interface is done by invoking cscope with its line-oriented interface, and then parsing the output returned from a query. The end result is that cscope query results become just like regular tags, so you can jump to them just like you do with normal tags (Ctrl-] or :tag) and then go back by popping off the tagstack with Ctrl-T. (Please [note](#) however, that you don't actually jump to a cscope tag simply by doing Ctrl-] or :tag without remapping these commands or setting an option. See the remaining sections on how the cscope interface works and for suggested use.)

=====

2. Cscope related commands

[cscope-commands](#)

[:cscope](#) [:cs](#) [:scs](#) [:scscope](#) [E259](#) [E262](#) [E560](#) [E561](#)

All cscope commands are accessed through suboptions to the cscope commands.

[`:cscope`](#) or [`:cs`](#) is the main command
[`:scscope`](#) or [`:scs`](#) does the same and splits the window
[`:lcscope`](#) or [`:lcs`](#) uses the location list, see [:lcscope](#)

The available subcommands are:

[E563](#) [E564](#) [E566](#) [E568](#) [E622](#) [E623](#) [E625](#)
[E626](#) [E609](#)

add : Add a new cscope database/connection.

USAGE :cs add {file|dir} [pre-path] [flags]

[pre-path] is the pathname used with the -P command to cscope.

[flags] are any additional flags you want to pass to cscope.

EXAMPLES

```
:cscope add /usr/local/cdb/cscope.out
:cscope add /projects/vim/cscope.out /usr/local/vim
:cscope add cscope.out /usr/local/vim -C
```

`find` : Query cscope. All cscope query options are available except option #5 ("Change this grep pattern").

USAGE :cs find {querytype} {name}

{querytype} corresponds to the actual cscope line interface numbers as well as default nvi commands:

- 0 or s: Find this C symbol
- 1 or g: Find this definition
- 2 or d: Find functions called by this function
- 3 or c: Find functions calling this function
- 4 or t: Find this text string
- 6 or e: Find this egrep pattern
- 7 or f: Find this file
- 8 or i: Find files #including this file
- 9 or a: Find places where this symbol is assigned a value

For all types, except 4 and 6, leading white space for {name} is removed. For 4 and 6 there is exactly one space between {querytype} and {name}. Further white space is included in {name}.

EXAMPLES

```
:cscope find c vim_free
:cscope find 3 vim_free
```

These two examples perform the same query: functions calling "vim_free".

```
:cscope find t initOnce
:cscope find t initOnce
```

The first one searches for the text "initOnce", the second one for " initOnce".

```
:cscope find 0 DEFAULT_TERM
```

Executing this example on the source code for Vim 5.1 produces the following output:

```
Cscope tag: DEFAULT_TERM
# line filename / context / line
1 1009 vim-5.1-gtk/src/term.c <<GLOBAL>>
#define DEFAULT_TERM (char_u *)"amiga"
2 1013 vim-5.1-gtk/src/term.c <<GLOBAL>>
#define DEFAULT_TERM (char_u *)"win32"
3 1017 vim-5.1-gtk/src/term.c <<GLOBAL>>
```

```

4 1021 #define DEFAULT_TERM (char_u *)"pcterm"
vim-5.1-gtk/src/term.c <<GLOBAL>>
5 1025 #define DEFAULT_TERM (char_u *)"ansi"
vim-5.1-gtk/src/term.c <<GLOBAL>>
6 1029 #define DEFAULT_TERM (char_u *)"vt52"
vim-5.1-gtk/src/term.c <<GLOBAL>>
7 1033 #define DEFAULT_TERM (char_u *)"os2ansi"
vim-5.1-gtk/src/term.c <<GLOBAL>>
8 1037 #define DEFAULT_TERM (char_u *)"ansi"
vim-5.1-gtk/src/term.c <<GLOBAL>>
9 1038 # undef DEFAULT_TERM
vim-5.1-gtk/src/term.c <<GLOBAL>>
10 1042 #define DEFAULT_TERM (char_u *)"beos-ansi"
vim-5.1-gtk/src/term.c <<GLOBAL>>
11 1335 #define DEFAULT_TERM (char_u *)"mac-ansi"
vim-5.1-gtk/src/term.c <<set_termname>>
term = DEFAULT_TERM;
12 1459 vim-5.1-gtk/src/term.c <<set_termname>>
if (STRCMP(term, DEFAULT_TERM))
13 1826 vim-5.1-gtk/src/term.c <<termcapinit>>
term = DEFAULT_TERM;
14 1833 vim-5.1-gtk/src/term.c <<termcapinit>>
term = DEFAULT_TERM;
15 3635 vim-5.1-gtk/src/term.c <<update_tcap>>
p = find_builtin_term(DEFAULT_TERM);
Enter nr of choice (<CR> to abort):

```

The output shows several pieces of information:

1. The tag number (there are 15 in this example).
2. The line number where the tag occurs.
3. The filename where the tag occurs.
4. The context of the tag (e.g., global, or the function name).
5. The line from the file itself.

help : Show a brief synopsis.

USAGE :cs help

E261

kill : Kill a cscope connection (or kill all cscope connections).

USAGE :cs kill {num|partial_name}

To kill a cscope connection, the connection number or a partial name must be specified. The partial name is simply any part of the pathname of the cscope database. Kill a cscope connection using the partial name with caution!

If the specified connection number is -1, then `_ALL_` cscope connections will be killed.

reset : Reinit all cscope connections.

USAGE :cs reset

show : Show cscope connections.

USAGE :cs show

[:lscope](#) [:lcs](#)

This command is same as the ":cscope" command, except when the '[cscopequickfix](#)' option is set, the location list for the current window is used instead of the quickfix list to show the cscope results.

[:cstag](#) [E257](#) [E562](#)

If you use cscope as well as ctags, [:cstag](#) allows you to search one or the other before making a jump. For example, you can choose to first search your cscope database(s) for a match, and if one is not found, then your tags file(s) will be searched. The order in which this happens is determined by the value of [csto](#) . See [cscope-options](#) for more details.

[:cstag](#) performs the equivalent of ":cs find g" on the identifier when searching through the cscope database(s).

[:cstag](#) performs the equivalent of [:tjump](#) on the identifier when searching through your tags file(s).

3. Cscope options

[cscope-options](#)

Use the [:set](#) command to set all cscope options. Ideally, you would do this in one of your startup files (e.g., .vimrc). Some cscope related variables are only valid within .vimrc . Setting them after vim has started will have no effect!

[cscopeprg](#) [csprg](#)

'[cscopeprg](#)' specifies the command to execute cscope. The default is "cscope". For example:

```
:set csprg=/usr/local/bin/cscope
```

[cscopequickfix](#) [csqf](#) [E469](#)

{not available when compiled without the [|+quickfix|](#) feature}

'[cscopequickfix](#)' specifies whether to use quickfix window to show cscope results. This is a list of comma-separated values. Each item consists of [cscope-find](#) command (s, g, d, c, t, e, f, i or a) and flag (+, - or 0). '+' indicates that results must be appended to quickfix window, '-' implies previous results clearance, '0' or command absence - don't use quickfix. Search is performed from start until first command occurrence. The default value is "" (don't use quickfix anyway). The following value seems to be useful:

```
:set cscopequickfix=s-,c-,d-,i-,t-,e-,a-
```

[cscopetag](#) [cst](#)

If '[cscopetag](#)' is set, the commands ":tag" and **CTRL-]** as well as "vim -t" will always use [:cstag](#) instead of the default :tag behavior. Effectively, by setting '[cst](#)', you will always search your cscope databases as well as

your tag files. The default is off. Examples:

```
:set cst
:set nocst
```

cscoperelative **csre**

If '**cscoperelative**' is set, then in absence of a prefix given to cscope (prefix is the argument of -P option of cscope), basename of cscope.out location (usually the project root directory) will be used as the prefix to construct an absolute path. The default is off. **Note:** This option is only effective when cscope (cscopeprg) is initialized without a prefix path (-P). Examples:

```
:set csre
:set nocsre
```

cscopetagorder **csto**

The value of '**csto**' determines the order in which **:cstag** performs a search. If '**csto**' is set to zero, cscope database(s) are searched first, followed by tag file(s) if cscope did not return any matches. If '**csto**' is set to one, tag file(s) are searched before cscope database(s). The default is zero. Examples:

```
:set csto=0
:set csto=1
```

cscopeverbose **csverb**

If '**cscopeverbose**' is not set (the default), messages will not be printed indicating success or failure when adding a cscope database. Ideally, you should reset this option in your **.vimrc** before adding any cscope databases, and after adding them, set it. From then on, when you add more databases within Vim, you will get a (hopefully) useful message should the database fail to be added. Examples:

```
:set csverb
:set nocsverb
```

cscopepathcomp **cspc**

The value of '**cspc**' determines how many components of a file's path to display. With the default value of zero the entire path will be displayed. The value one will display only the filename with no path. Other values display that many components. For example:

```
:set cspc=3
```

will display the last 3 components of the file's path, including the file name itself.

4. How to use cscope in Vim

cscope-howtouse

The first thing you need to do is to build a cscope database for your source files. For the most basic case, simply do "cscope -b". Please refer to the cscope man page for more details.

Assuming you have a cscope database, you need to "add" the database to Vim. This establishes a cscope "connection" and makes it available for Vim to use. You can do this in your **.vimrc** file, or you can do it manually after starting vim. For example, to add the cscope database "cscope.out", you would do:

```
:cs add cscope.out
```

You can double-check the result of this by executing `":cs show"`. This will produce output which looks like this:

```
# pid      database name      prepend path
0 28806    cscope.out         <none>
```

Note:

Because of the Microsoft RTL limitations, Win32 version shows 0 instead of the real pid.

Once a cscope connection is established, you can make queries to cscope and the results will be printed to you. Queries are made using the command `":cs find"`. For example:

```
:cs find g ALIGN_SIZE
```

This can get a little cumbersome since one ends up doing a significant amount of typing. Fortunately, there are ways around this by mapping shortcut keys. See [cscope-suggestions](#) for suggested usage.

If the results return only one match, you will automatically be taken to it. If there is more than one match, you will be given a selection screen to pick the match you want to go to. After you have jumped to the new location, simply hit Ctrl-T to get back to the previous one.

5. Limitations

[cscope-limitations](#)

Cscope support for Vim is only available on systems that support these four system calls: `fork()`, `pipe()`, `execl()`, `waitpid()`. This means it is mostly limited to Unix systems.

Additionally Cscope support works for Win32. For more information and a cscope version for Win32 see:

<http://iamphet.nm.ru/cscope/index.html>

The DJGPP-built version from <http://cscope.sourceforge.net> is known to not work with Vim.

Hard-coded limitation: doing a `:tjump` when `:cstag` searches the tag files is not configurable (e.g., you can't do a `tselect` instead).

6. Suggested usage

[cscope-suggestions](#)

Put these entries in your `.vimrc` (adjust the pathname accordingly to your setup):

```
if has("cscope")
    set csprg=/usr/local/bin/cscope
```

```

        set csto=0
        set cst
        set nocsverb
        " add any database in current directory
        if filereadable("cscope.out")
            cs add cscope.out
        " else add database pointed to by environment
        elseif $CSCOPE_DB != ""
            cs add $CSCOPE_DB
        endif
        set csverb
    endif
endif

```

By setting '**cscopetag**', we have effectively replaced all instances of the `:tag` command with `:cstag`. This includes `:tag`, `Ctrl-]`, and `"vim -t"`. In doing this, the regular tag command not only searches your ctags generated tag files, but your cscope databases as well.

Some users may want to keep the regular tag behavior and have a different shortcut to access `:cstag`. For example, one could map `Ctrl-_` (underscore) to `:cstag` with the following command:

```
map <C-_> :cstag <C-R>=expand("<word>")<CR><CR>
```

A couple of very commonly used cscope queries (using `:cs find`) is to find all functions calling a certain function and to find all occurrences of a particular C symbol. To do this, you can use these mappings as an example:

```
map g<C-]> :cs find 3 <C-R>=expand("<word>")<CR><CR>
map g<C-\> :cs find 0 <C-R>=expand("<word>")<CR><CR>
```

These mappings for `Ctrl-]` (right bracket) and `Ctrl-\` (backslash) allow you to place your cursor over the function name or C symbol and quickly query cscope for any matches.

Or you may use the following scheme, inspired by Vim/Cscope tutorial from Cscope Home Page (<http://cscope.sourceforge.net/>):

```

nmap <C-_>s :cs find s <C-R>=expand("<word>")<CR><CR>
nmap <C-_>g :cs find g <C-R>=expand("<word>")<CR><CR>
nmap <C-_>c :cs find c <C-R>=expand("<word>")<CR><CR>
nmap <C-_>t :cs find t <C-R>=expand("<word>")<CR><CR>
nmap <C-_>e :cs find e <C-R>=expand("<word>")<CR><CR>
nmap <C-_>f :cs find f <C-R>=expand("<file>")<CR><CR>
nmap <C-_>i :cs find i ^<C-R>=expand("<file>")<CR>${<CR>
nmap <C-_>d :cs find d <C-R>=expand("<word>")<CR><CR>
nmap <C-_>a :cs find a <C-R>=expand("<word>")<CR><CR>

```

```

" Using 'CTRL-spacebar' then a search type makes the vim window
" split horizontally, with search result displayed in
" the new window.

```

```
nmap <C-Space>s :scs find s <C-R>=expand("<word>")<CR><CR>
```

```

nmap <C-Space>g :scs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space>c :scs find c <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space>t :scs find t <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space>e :scs find e <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space>f :scs find f <C-R>=expand("<cfile>")<CR><CR>
nmap <C-Space>i :scs find i ^<C-R>=expand("<cfile>")<CR>${<CR>
nmap <C-Space>d :scs find d <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space>a :scs find a <C-R>=expand("<cword>")<CR><CR>

```

" Hitting CTRL-space *twice* before the search type does a vertical
" split instead of a horizontal one

```

nmap <C-Space><C-Space>s
\ :vert scs find s <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>g
\ :vert scs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>c
\ :vert scs find c <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>t
\ :vert scs find t <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>e
\ :vert scs find e <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>i
\ :vert scs find i ^<C-R>=expand("<cfile>")<CR>${<CR>
nmap <C-Space><C-Space>d
\ :vert scs find d <C-R>=expand("<cword>")<CR><CR>
nmap <C-Space><C-Space>a
\ :vert scs find a <C-R>=expand("<cword>")<CR><CR>

```

7. Cscope availability and information

[cscope-info](#)

If you do not already have cscope (it did not come with your compiler license or OS distribution), then you can download it for free from:

<http://cscope.sourceforge.net/>

This is released by SCO under the BSD license.

In Solaris 2.x, if you have the C compiler license, you will also have cscope. Both are usually located under /opt/SUNWspro/bin

There is source to an older version of a cscope clone (called "cs") available on the net. Due to various reasons, this is not supported with Vim.

The cscope interface/support for Vim was originally written by Andy Kahn <ackahn@netapp.com>. The original structure (as well as a tiny bit of code) was adapted from the cscope interface in nvi.

[cscope-win32](#)

For a cscope version for Win32 see (seems abandoned):

<https://code.google.com/archive/p/cscope-win32/>

Win32 support was added by Sergey Khorev <sergey.khorev@gmail.com>. Contact him if you have Win32-specific issues.

VIM REFERENCE MANUAL by Luis Carvalho

The Lua Interface to Vim

lua Lua

1. Commands	lua-commands
2. The vim module	lua-vim
3. List userdata	lua-list
4. Dict userdata	lua-dict
5. Blob userdata	lua-blob
6. Funcref userdata	lua-funcref
7. Buffer userdata	lua-buffer
8. Window userdata	lua-window
9. luaeval() Vim function	lua-luaeval
10. Dynamic loading	lua-dynamic

{only available when Vim was compiled with the |+lua| feature}

1. Commands

lua-commands

:[range]lua {chunk}

:lua

Execute Lua chunk {chunk}.

Examples:

```
:lua print("Hello, Vim!")
:lua local curbuf = vim.buffer() curbuf[7] = "line #7"
```

```
:[range]lua << [trim] [{endmarker}]
{script}
{endmarker}
```

Execute Lua script {script}.

Note: This command doesn't work when the Lua feature wasn't compiled in. To avoid errors, see [script-here](#) .

If [endmarker] is omitted from after the "<<", a dot '.' must be used after {script}, like for the [:append](#) and [:insert](#) commands. Refer to [:let-heredoc](#) for more information.

This form of the [:lua](#) command is mainly useful for including Lua code in Vim scripts.

Example:

```
function! CurrentLineInfo()
lua << EOF
local linenr = vim.window().line
```

```

local curline = vim.buffer()[linenr]
print(string.format("Current line [%d] has %d chars",
    linenr, #curline))
EOF
endfunction

```

To see what version of Lua you have:

```
:lua print(_VERSION)
```

If you use LuaJIT you can also use this:

```
:lua print(jit.version)
```

```

:[:range]luado {body} :luado
Execute Lua function "function (line, linenr) {body}
end" for each line in the [range], with the function
argument being set to the text of each line in turn,
without a trailing <EOL>, and the current line number.
If the value returned by the function is a string it
becomes the text of the line in the current turn. The
default for [range] is the whole file: "1,$".

```

Examples:

```

:luado return string.format("%s\t%d", line:reverse(), #line)

:lua require"lpeg"
:lua -- balanced parenthesis grammar:
:lua bp = lpeg.P{ "(" * ((1 - lpeg.S"()") + lpeg.V(1))^0 * ")" }
:luado if bp:match(line) then return "-->\t" .. line end

```

```

:[:range]luafile {file} :luafile
Execute Lua script in {file}.
The whole argument is used as a single file name.

```

Examples:

```

:luafile script.lua
:luafile %

```

All these commands execute a Lua chunk from either the command line (:lua and :luado) or a file (:luafile) with the given line [range]. Similarly to the Lua interpreter, each chunk has its own scope and so only global variables are shared between command calls. All Lua default libraries are available. In addition, Lua "print" function has its output redirected to the Vim message area, with arguments separated by a white space instead of a tab.

Lua uses the "vim" module (see [lua-vim](#)) to issue commands to Vim and manage buffers ([lua-buffer](#)) and windows ([lua-window](#)). However, procedures that alter buffer content, open new buffers, and change cursor position are restricted when the command is executed in the [sandbox](#).

2. The vim module

lua-vim

Lua interfaces Vim through the "vim" module. The first and last line of the input range are stored in "vim.firstline" and "vim.lastline" respectively. The module also includes routines for buffer, window, and current line queries, Vim evaluation and command execution, and others.

<code>vim.list([arg])</code>	Returns an empty list or, if "arg" is a Lua table with numeric keys 1, ..., n (a "sequence"), returns a list <code>l</code> such that <code>l[i] = arg[i]</code> for <code>i = 1, ..., n</code> (see List). Non-numeric keys are not used to initialize the list. See also lua-eval for conversion rules. Example: <pre>:lua t = {math.pi, false, say = 'hi'} :echo luaeval('vim.list(t)') :" [3.141593, v:false], 'say' is ignored</pre>
<code>vim.dict([arg])</code>	Returns an empty dictionary or, if "arg" is a Lua table, returns a dict <code>d</code> such that <code>d[k] = arg[k]</code> for all string keys <code>k</code> in "arg" (see Dictionary). Number keys are converted to strings. Keys that are not strings are not used to initialize the dictionary. See also lua-eval for conversion rules. Example: <pre>:lua t = {math.pi, false, say = 'hi'} :echo luaeval('vim.dict(t)') :" {'1': 3.141593, '2': v:false, :" 'say': 'hi'}</pre>
<code>vim.blob([arg])</code>	Returns an empty blob or, if "arg" is a Lua string, returns a blob <code>b</code> such that <code>b</code> is equivalent to "arg" as a byte string. Examples: <pre>:lua s = "12ab\x00\x80\xfe\xff" :echo luaeval('vim.blob(s)') :" 0z31326162.0080FEFF</pre>
<code>vim.funcref({name})</code>	Returns a Funcref to function <code>{name}</code> (see Funcref). It is equivalent to Vim's <code>function()</code> .
<code>vim.buffer([arg])</code>	If "arg" is a number, returns buffer with number "arg" in the buffer list or, if "arg" is a string, returns buffer whose full or short name is "arg". In both cases, returns 'nil' (nil value, not string) if the buffer is not found. Otherwise, if <code>"toboolean(arg)"</code> is 'true' returns the first buffer in the buffer list or else the current buffer.

<code>vim.window([arg])</code>	If "arg" is a number, returns window with number "arg" or 'nil' (nil value, not string) if not found. Otherwise, if <code>"toboolean(arg)"</code> is 'true' returns the first window or else the current window.
<code>vim.type({arg})</code>	Returns the type of <code>{arg}</code> . It is equivalent to Lua's "type" function, but returns "list", "dict", "funcref", "buffer", or "window" if <code>{arg}</code> is a list, dictionary, funcref, buffer, or window, respectively. Examples: <pre>:lua l = vim.list() :lua print(type(l), vim.type(l)) :" list</pre>
<code>vim.command({cmds})</code>	Executes one or more lines of Ex-mode commands in <code>{cmds}</code> . Examples: <pre>:lua vim.command"set tw=60" :lua vim.command"normal ddp" lua << trim END vim.command([[new Myfile.js call search('start')]]) END</pre>
<code>vim.eval({expr})</code>	Evaluates expression <code>{expr}</code> (see expression), converts the result to Lua, and returns it. Vim strings and numbers are directly converted to Lua strings and numbers respectively. Vim lists and dictionaries are converted to Lua userdata (see lua-list and lua-dict). Examples: <pre>:lua tw = vim.eval"&tw" :lua print(vim.eval"{'a': 'one'}".a)</pre>
<code>vim.line()</code>	Returns the current line (without the trailing <code><EOL></code>), a Lua string.
<code>vim.beep()</code>	Beeps.
<code>vim.open({fname})</code>	Opens a new buffer for file <code>{fname}</code> and returns it. Note that the buffer is not set as current.
<code>vim.call({name} [, {args}])</code>	Proxy to call Vim function named <code>{name}</code> with arguments <code>{args}</code> . Example: <pre>:lua print(vim.call('has', 'timers'))</pre>
<code>vim.fn</code>	Proxy to call Vim functions. Proxy methods are created on demand. Example: <pre>:lua print(vim.fn.has('timers'))</pre>

<code>vim.lua_version</code>	The Lua version Vim was compiled with, in the form <code>{major}.{minor}.{patch}</code> , e.g. "5.1.4".
<code>vim.version()</code>	Returns a Lua table with the Vim version. The table will have the following keys: <ul style="list-style-type: none"> major - major Vim version. minor - minor Vim version. patch - latest patch included.

lua-vim-variables

The Vim editor global dictionaries `g:` `w:` `b:` `t:` `v:` can be accessed from Lua conveniently and idiomatically by referencing the ``vim.*`` Lua tables described below. In this way you can easily read and modify global Vim script variables from Lua.

Example:

```
vim.g.foo = 5      -- Set the g:foo Vim script variable.
print(vim.g.foo) -- Get and print the g:foo Vim script variable.
vim.g.foo = nil   -- Delete (:unlet) the Vim script variable.
```

`vim.g` `vim.g`
Global (`g:`) editor variables.
Key with no value returns ``nil``.

`vim.b` `vim.b`
Buffer-scoped (`b:`) variables for the current buffer.
Invalid or unset key returns ``nil``.

`vim.w` `vim.w`
Window-scoped (`w:`) variables for the current window.
Invalid or unset key returns ``nil``.

`vim.t` `vim.t`
Tabpage-scoped (`t:`) variables for the current tabpage.
Invalid or unset key returns ``nil``.

`vim.v` `vim.v`
`v:` variables.
Invalid or unset key returns ``nil``.

3. List userdata lua-list

List userdata represent vim lists, and the interface tries to follow closely Vim's syntax for lists. Since lists are objects, changes in list references in Lua are reflected in Vim and vice-versa. A list "l" has the following properties and methods:

NOTE: In patch 8.2.1066 array indexes were changed from zero-based to one-based. You can check with:

```
if has("patch-8.2.1066")
```

Properties

- o "#l" is the number of items in list "l", equivalent to "len(l)" in Vim.
- o "l[k]" returns the k-th item in "l"; "l" is one-indexed, as in Lua. To modify the k-th item, simply do "l[k] = newitem"; in particular, "l[k] = nil" removes the k-th item from "l". Item can be added to the end of the list by "l[#l + 1] = newitem"
- o "l()" returns an iterator for "l".
- o "table.insert(l, newitem)" inserts an item at the end of the list. (only Lua 5.3 and later)
- o "table.insert(l, position, newitem)" inserts an item at the specified position. "position" is one-indexed. (only Lua 5.3 and later)
- o "table.remove(l, position)" removes an item at the specified position. "position" is one-indexed.

Methods

- o "l:add(item)" appends "item" to the end of "l".
- o "l:insert(item[, pos])" inserts "item" at (optional) position "pos" in the list. The default value for "pos" is 0.

Examples:

```
:let l = [1, 'item']
:lua l = vim.eval('l') -- same 'l'
:lua l:add(vim.list())
:lua l[1] = math.pi
:echo l[0] " 3.141593
:lua l[1] = nil -- remove first item
:lua l:insert(true, 1)
:lua print(l, #l, l[1], l[2])
:lua l[#l + 1] = 'value'
:lua table.insert(l, 100)
:lua table.insert(l, 2, 200)
:lua table.remove(l, 1)
:lua for item in l() do print(item) end
```

4. Dict userdata

lua-dict

Similarly to list userdata, dict userdata represent vim dictionaries; since dictionaries are also objects, references are kept between Lua and Vim. A dict "d" has the following properties:

Properties

- o "#d" is the number of items in dict "d", equivalent to "len(d)" in Vim.
- o "d.key" or "d['key']" returns the value at entry "key" in "d". To modify the entry at this key, simply do "d.key = newvalue"; in particular, "d.key = nil" removes the entry from "d".

- o "d()" returns an iterator for "d" and is equivalent to "items(d)" in Vim.

Examples:

```
:let d = {'n':10}
:lua d = vim.eval('d') -- same 'd'
:lua print(d, d.n, #d)
:let d.self = d
:lua for k, v in d() do print(d, k, v) end
:lua d.x = math.pi
:lua d.self = nil -- remove entry
:echo d
```

5. Blob userdata

lua-blob

Blob userdata represent vim blobs. A blob "b" has the following properties:

Properties

- o "#b" is the length of blob "b", equivalent to "len(b)" in Vim.
- o "b[k]" returns the k-th item in "b"; "b" is zero-indexed, as in Vim. To modify the k-th item, simply do "b[k] = number"; in particular, "b[#b] = number" can append a byte to tail.

Methods

- o "b:add(bytes)" appends "bytes" to the end of "b".

Examples:

```
:let b = 0z001122
:lua b = vim.eval('b') -- same 'b'
:lua print(b, b[0], #b)
:lua b[1] = 32
:lua b[#b] = 0x33 -- append a byte to tail
:lua b:add("\x80\x81\xfe\xff")
:echo b
```

6. Funcref userdata

lua-funcref

Funcref userdata represent funcref variables in Vim. Funcrefs that were defined with a "dict" attribute need to be obtained as a dictionary key in order to have "self" properly assigned to the dictionary (see examples below.) A funcref "f" has the following properties:

Properties

- o "#f" is the name of the function referenced by "f"
- o "f(...)" calls the function referenced by "f" (with arguments)

Examples:

```
:function I(x)
:  return a:x
:  endfunction
:let R = function('I')
:lua i1 = vim.funcref('I')
:lua i2 = vim.eval('R')
:lua print(#i1, #i2) -- both 'I'
:lua print(i1, i2, #i2(i1) == #i1(i2))
:function Mylen() dict
:  return len(self.data)
:  endfunction
:let mydict = {'data': [0, 1, 2, 3]}
:lua d = vim.eval('mydict'); d.len = vim.funcref('Mylen')
:echo mydict.len()
:lua l = d.len -- assign d as 'self'
:lua print(l())
```

Lua functions and closures are automatically converted to a Vim `Funcref` and can be accessed in Vim scripts. Example:

```
lua <<EOF
vim.fn.timer_start(1000, function(timer)
    print('timer callback')
end)
EOF
```

7. Buffer userdata

lua-buffer

Buffer userdata represent vim buffers. A buffer userdata "b" has the following properties and methods:

Properties

- o "b()" sets "b" as the current buffer.
- o "#b" is the number of lines in buffer "b".
- o "b[k]" represents line number k: "b[k] = newline" replaces line k with string "newline" and "b[k] = nil" deletes line k.
- o "b.name" contains the short name of buffer "b" (read-only).
- o "b.fname" contains the full name of buffer "b" (read-only).
- o "b.number" contains the position of buffer "b" in the buffer list (read-only).

Methods

- o "b:insert(newline[, pos])" inserts string "newline" at (optional) position "pos" in the buffer. The default value for "pos" is "#b + 1". If "pos == 0" then "newline" becomes the first line in the buffer.
- o "b:next()" returns the buffer next to "b" in the buffer list.
- o "b:previous()" returns the buffer previous to "b" in the buffer list.

- list.
- o "b:isvalid()" returns **'true'** (boolean) if buffer "b" corresponds to a "real" (not freed from memory) Vim buffer.

Examples:

```
:lua b = vim.buffer() -- current buffer
:lua print(b.name, b.number)
:lua b[1] = "first line"
:lua b:insert("FIRST!", 0)
:lua b[1] = nil -- delete top line
:lua for i=1,3 do b:insert(math.random()) end
:3,4lua for i=vim.lastline,vim.firstline,-1 do b[i] = nil end
:lua vim.open"myfile"() -- open buffer and set it as current

function! ListBuffers()
lua << EOF
local b = vim.buffer(true) -- first buffer in list
while b ~= nil do
    print(b.number, b.name, #b)
    b = b:next()
end
vim.beep()
EOF
endfunction
```

8. Window userdata

lua-window

Window objects represent vim windows. A window userdata "w" has the following properties and methods:

Properties

- o "w()" sets "w" as the current window.
- o "w.buffer" contains the buffer of window "w" (read-only).
- o "w.line" represents the cursor line position in window "w".
- o "w.col" represents the cursor column position in window "w".
- o "w.width" represents the width of window "w".
- o "w.height" represents the height of window "w".

Methods

- o "w:next()" returns the window next to "w".
- o "w:previous()" returns the window previous to "w".
- o "w:isvalid()" returns **'true'** (boolean) if window "w" corresponds to a "real" (not freed from memory) Vim window.

Examples:

```
:lua w = vim.window() -- current window
:lua print(w.buffer.name, w.line, w.col)
:lua w.width = w.width + math.random(10)
```

```

:lua w.height = 2 * math.random() * w.height
:lua n,w = 0,vim.window(true) while w~=nil do n,w = n + 1,w:next() end
:lua print("There are " .. n .. " windows")

```

9. luaeval() Vim function

lua-luaeval lua-eval

The (dual) equivalent of "vim.eval" for passing Lua values to Vim is "luaeval". "luaeval" takes an expression string and an optional argument and returns the result of the expression. It is semantically equivalent in Lua to:

```

local chunkheader = "local _A = select(1, ...) return "
function luaeval (expstr, arg)
    local chunk = assert(loadstring(chunkheader .. expstr, "luaeval"))
    return chunk(arg) -- return typval
end

```

Note that "_A" receives the argument to "luaeval". Lua numbers, strings, and list, dict, blob, and funcref userdata are converted to their Vim respective types, while Lua booleans are converted to numbers. An error is thrown if conversion of any of the remaining Lua types, including userdata other than lists, dicts, blobs, and funcrefs, is attempted.

Examples:

```

:echo luaeval('math.pi')
:lua a = vim.list():add('newlist')
:let a = luaeval('a')
:echo a[0] " 'newlist'
:function Rand(x,y) " random uniform between x and y
: return luaeval('(_A.y-_A.x)*math.random()+_A.x', {'x':a:x,'y':a:y})
: endfunction
:echo Rand(1,10)

```

10. Dynamic loading

lua-dynamic

On MS-Windows and Unix the Lua library can be loaded dynamically. The `:version` output then includes `+lua/dyn`.

This means that Vim will search for the Lua DLL or shared library file only when needed. When you don't use the Lua interface you don't need it, thus you can use Vim without this file.

MS-Windows

To use the Lua interface the Lua DLL must be in your search path. In a console window type "path" to see what directories are used. The `'luadll'` option can be also used to specify the Lua DLL. The version of the DLL must match the Lua version Vim was compiled with.

Unix

The `'luadll'` option can be used to specify the Lua shared library file instead of `DYNAMIC_LUA_DLL` file what was specified at compile time. The version of the shared library must match the Lua version Vim was compiled with.

=====

The MzScheme Interface to Vim

mzscheme MzScheme

- 1. Commands mzscheme-commands
- 2. Examples mzscheme-examples
- 3. Threads mzscheme-threads
- 4. Vim access from MzScheme mzscheme-vim
- 5. mzeval() Vim function mzscheme-mzeval
- 6. Using Function references mzscheme-funcref
- 7. Dynamic loading mzscheme-dynamic
- 8. MzScheme setup mzscheme-setup

{only available when Vim was compiled with the |+mzscheme| feature}

Based on the work of Brent Fulgham.
Dynamic loading added by Sergey Khorev

MzScheme and PLT Scheme names have been rebranded as Racket. For more information please check <http://racket-lang.org>

Futures and places of Racket version 5.x up to and including 5.3.1 do not work correctly with processes created by Vim.
The simplest solution is to build Racket on your own with these features disabled:

```
./configure --disable-futures --disable-places --prefix=your-install-prefix
```

To speed up the process, you might also want to use --disable-gracket and --disable-docs



1. Commands mzscheme-commands

:mzscheme :mz

`:[range]mz[scheme] {stmt}`
Execute MzScheme statement {stmt}.

`:[range]mz[scheme] << [trim] [{endmarker}]`
`{script}`
`{endmarker}`
Execute inlined MzScheme script {script}.
Note: This command doesn't work when the MzScheme feature wasn't compiled in. To avoid errors, see [script-here](#) .

If [endmarker] is omitted from after the "<<", a dot '.' must be used after {script}, like for the `:append` and `:insert` commands. Refer to `:let-heredoc` for more information.

`:mzfile` `:mzf`
`:[range]mzf[ile] {file}` Execute the MzScheme script in `{file}`.

All of these commands do essentially the same thing - they execute a piece of MzScheme code, with the "current range" set to the given line range.

In the case of `:mzscheme`, the code to execute is in the command-line.
In the case of `:mzfile`, the code to execute is the contents of the given file.

MzScheme interface defines exception `exn:vim`, derived from `exn`.
It is raised for various Vim errors.

During compilation, the MzScheme interface will remember the current MzScheme collection path. If you want to specify additional paths use the 'current-library-collection-paths' parameter. E.g., to cons the user-local MzScheme collection path:

```
:mz << EOF
(current-library-collection-paths
 (cons
  (build-path (find-system-path 'addon-dir) (version) "collects")
  (current-library-collection-paths)))
EOF
```

All functionality is provided through module `vimext`.

The `exn:vim` is available without explicit import.

To avoid clashes with MzScheme, consider using prefix when requiring module, e.g.:

```
:mzscheme (require (prefix vim- vimext))
```

All the examples below assume this naming scheme.

When executed in the `sandbox`, access to some filesystem and Vim interface procedures is restricted.

2. Examples `mzscheme-examples`

```
:mzscheme (display "Hello")
:mz (display (string-append "Using MzScheme version " (version)))
:mzscheme (require (prefix vim- vimext)) ; for MzScheme < 4.x
:mzscheme (require (prefix-in vim- 'vimext)) ; MzScheme 4.x
:mzscheme (vim-set-buff-line 10 "This is line #10")
```

To see what version of MzScheme you have:

```
:mzscheme (display (version))
```

Inline script usage:

```
function! <SID>SetFirstLine()
```

```

    :mz << EOF
    (display "!!!")
    (require (prefix vim- vimext))
    ; for newer versions (require (prefix-in vim- 'vimext))
    (vim-set-buff-line 1 "This is line #1")
    (vim-beep)
EOF
endfunction

nmap <F9> :call <SID>SetFirstLine() <CR>

```

File execution:

```
:mzfile supascript.scm
```

Vim exception handling:

```

:mz << EOF
(require (prefix vim- vimext))
; for newer versions (require (prefix-in vim- 'vimext))
(with-handlers
  ([exn:vim? (lambda (e) (display (exn-message e))]))
  (vim-eval "nonsense-string"))
EOF

```

Auto-instantiation of vimext module (can be placed in your `vimrc`):

```

function! MzRequire()
  :redir => l:mzversion
  :mz (version)
  :redir END
  if strpart(l:mzversion, 1, 1) < "4"
    " MzScheme versions < 4.x:
    :mz (require (prefix vim- vimext))
  else
    " newer versions:
    :mz (require (prefix-in vim- 'vimext))
  endif
endfunction

if has("mzscheme")
  silent call MzRequire()
endif

```

3. Threads

[mzscheme-threads](#)

The MzScheme interface supports threads. They are independent from OS threads, thus scheduling is required. The option `'mzquantum'` determines how often Vim should poll for available MzScheme threads.

NOTE

Thread scheduling in the console version of Vim is less reliable than in the GUI version.

4. Vim access from MzScheme

[mzscheme-vim](#)

The **'vimext'** module provides access to procedures defined in the MzScheme interface.

Common

(command {command-string})	Perform the vim ":Ex" style command.
(eval {expr-string})	Evaluate the vim expression into respective MzScheme object: Lists are represented as Scheme lists, Dictionaries as hash tables, Funcrefs as functions (see also mzscheme-funcref)
(range-start)	NOTE the name clashes with MzScheme eval, use module qualifiers to overcome this.
(range-end)	Start/End of the range passed with the Scheme command.
(beep)	beep
(get-option {option-name} [buffer-or-window])	Get Vim option value (either local or global, see set-option).
(set-option {string} [buffer-or-window])	Set a Vim option. String must have option setting form (like optname=optval, or optname+=optval, etc.) When called with {buffer} or {window} the local option will be set. The symbol 'global can be passed as {buffer-or-window}. Then :setglobal will be used.

Buffers

(buff? {object})	Is object a buffer?
(buff-valid? {object})	Is object a valid buffer? (i.e. corresponds to the real Vim buffer)
(get-buff-line {linenr} [buffer])	Get line from a buffer.
(set-buff-line {linenr} {string} [buffer])	Set a line in a buffer. If {string} is #f, the line gets deleted. The [buffer] argument is optional. If omitted, the current buffer will be used.
(get-buff-line-list {start} {end} [buffer])	Get a list of lines in a buffer. {Start} and {end} are 1-based and inclusive.
(set-buff-line-list {start} {end} {string-list} [buffer])	Set a list of lines in a buffer. If string-list is #f or null, the lines get deleted. If a list is shorter than {end}-{start} the remaining lines will be deleted.
(get-buff-name [buffer])	Get a buffer's text name.
(get-buff-num [buffer])	Get a buffer's number.
(get-buff-size [buffer])	Get buffer line count.
(insert-buff-line-list {linenr} {string/string-list} [buffer])	

	Insert a list of lines into a buffer after {linenr}. If {linenr} is 0, lines will be inserted at start.
(curr-buff)	Get the current buffer. Use other MzScheme interface procedures to change it.
(buff-count)	Get count of total buffers in the editor.
(get-next-buff [buffer])	Get next buffer.
(get-prev-buff [buffer])	Get previous buffer. Return #f when there are no more buffers.
(open-buff {filename})	Open a new buffer (for file "name")
(get-buff-by-name {buffername})	Get a buffer by its filename or #f if there is no such buffer.
(get-buff-by-num {buffernum})	Get a buffer by its number (return #f if there is no buffer with this number).

Windows

mzscheme-window

(win? {object})	Is object a window?
(win-valid? {object})	Is object a valid window (i.e. corresponds to the real Vim window)?
(curr-win)	Get the current window.
(win-count)	Get count of windows.
(get-win-num [window])	Get window number.
(get-win-by-num {windownum})	Get window by its number.
(get-win-buffer [window])	Get the buffer for a given window.
(get-win-height [window])	
(set-win-height {height} [window])	Get/Set height of window.
(get-win-width [window])	
(set-win-width {width} [window])	Get/Set width of window.
(get-win-list [buffer])	Get list of windows for a buffer.
(get-cursor [window])	Get cursor position in a window as a pair (linenr . column).
(set-cursor (line . col) [window])	Set cursor position.

5. mzeval() Vim function

mzscheme-mzeval

To facilitate bi-directional interface, you can use `mzeval()` function to evaluate MzScheme expressions and pass their values to Vim script.

6. Using Function references

mzscheme-funcref

MzScheme interface allows use of `Funcrefs` so you can call Vim functions directly from Scheme. For instance:

```
function! MyAdd2(arg)
  return a:arg + 2
endfunction
mz (define f2 (vim-eval "function(\"MyAdd2\")"))
mz (f2 7)
```

or :

```
:mz (define indent (vim-eval "function('indent')"))
" return Vim indent for line 12
:mz (indent 12)
```

7. Dynamic loading

mzscheme-dynamic E815

On MS-Windows the MzScheme libraries can be loaded dynamically. The `:version` output then includes `+mzscheme/dyn` .

This means that Vim will search for the MzScheme DLL files only when needed. When you don't use the MzScheme interface you don't need them, thus you can use Vim without these DLL files.

NOTE: Newer version of MzScheme (Racket) require earlier (trampolined) initialisation via `scheme_main_setup`. So Vim always loads the MzScheme DLL at startup if possible. This may make Vim startup slower.

To use the MzScheme interface the MzScheme DLLs must be in your search path. In a console window type "path" to see what directories are used.

On MS-Windows the options `'mzschemedll'` and `'mzschemegcdll'` are used for the name of the library to load. The initial value is specified at build time.

The version of the DLL must match the MzScheme version Vim was compiled with. For MzScheme version 209 they will be "libmzsch209_000.dll" and "libmzgc209_000.dll". To know for sure look at the output of the `":version"` command, look for `-DDYNAMIC_MZSCH_DLL="something"` and `-DDYNAMIC_MZGC_DLL="something"` in the "Compilation" info.

For example, if MzScheme (Racket) is installed at `C:\Racket63`, you may need to set the environment variable as the following:

```
PATH=%PATH%;C:\Racket63\lib
PLTCOLLECTS=C:\Racket63\collects
PLTCONFIGDIR=C:\Racket63\etc
```

8. MzScheme setup

mzscheme-setup E895

Vim requires "racket/base" module for `if_mzsch` core (fallback to "scheme/base" if it doesn't exist), "r5rs" module for `test` and "raco ctool" command for building Vim. If MzScheme did not have them, you can install them with MzScheme's `raco` command:

```
raco pkg install scheme-lib      # scheme/base module
raco pkg install r5rs-lib        # r5rs module
raco pkg install cext-lib        # raco ctool command
```

Perl and Vim `perl` `Perl`

- 1. Editing Perl files `perl-editing`
- 2. Compiling Vim with Perl interface `perl-compiling`
- 3. Using the Perl interface `perl-using`
- 4. Dynamic loading `perl-dynamic`

{only available when Vim was compiled with the |+perl| feature}

1. Editing Perl files `perl-editing`

Vim syntax highlighting supports Perl and POD files. Vim assumes a file is Perl code if the filename has a .pl or .pm suffix. Vim also examines the first line of a file, regardless of the filename suffix, to check if a file is a Perl script (see `scripts.vim` in Vim's syntax directory). Vim assumes a file is POD text if the filename has a .POD suffix.

To use tags with Perl, you need Universal/Exuberant Ctags. Look here:

Universal Ctags (preferred): <https://ctags.io>

Exuberant Ctags: <http://ctags.sourceforge.net>

Alternatively, you can use the Perl script `pltags.pl`, which is shipped with Vim in the `$VIMRUNTIME/tools` directory. This script has currently more features than Exuberant ctags' Perl support.

2. Compiling Vim with Perl interface `perl-compiling`

To compile Vim with Perl interface, you need Perl 5.004 (or later). Perl must be installed before you compile Vim. Vim's Perl interface does NOT work with the 5.003 version that has been officially released! It will probably work with Perl 5.003_05 and later.

The Perl patches for Vim were made by:

Sven Verdoolaege <skimo@breughel.ufsia.ac.be>

Matt Gerassimof

Perl for MS-Windows (and other platforms) can be found at:

<http://www.perl.org/>

The ActiveState one should work, Strawberry Perl is a good alternative.

3. Using the Perl interface `perl-using`

`:pe[rl] {cmd}` `:perl` `:pe`
Execute Perl command `{cmd}`. The current package is "main". Simple example to test if `:perl` is

```
working:
    :perl VIM::Msg("Hello")
```

```
:pe[rl] << [trim] [{endmarker}]
{script}
{endmarker}
```

Execute Perl script `{script}`.
The `{endmarker}` after `{script}` must NOT be preceded by any white space.

If `[endmarker]` is omitted, it defaults to a dot `'.'` like for the `:append` and `:insert` commands. Using `'.'` helps when inside a function, because `"$i;"` looks like the start of an `:insert` command to Vim.

This form of the `:perl` command is mainly useful for including perl code in vim scripts.

Note: This command doesn't work when the Perl feature wasn't compiled in. To avoid errors, see [script-here](#) .

Example vim script:

```
function! WhitePearl()
perl << EOF
    VIM::Msg("pearls are nice for necklaces");
    VIM::Msg("rubys for rings");
    VIM::Msg("pythons for bags");
    VIM::Msg("tcls????");
EOF
endfunction
```

To see what version of Perl you have:

```
:perl print $^V
```

```
:[range]perld[o] {cmd} Execute Perl command {cmd} for each line in the
                        :perldo  :perld
[range], with $_ being set to the text of each line in
turn, without a trailing <EOL>. Setting $_ will change
the text, but note that it is not possible to add or
delete lines using this command.
The default for [range] is the whole file: "1,$".
```

Here are some things you can try:

```
:perl $a=1
:perldo $_ = reverse($_);1
:perl VIM::Msg("hello")
:perl $line = $curbuf->Get(42)
```

E299

Executing Perl commands in the `sandbox` is limited. `":perldo"` will not be possible at all. `":perl"` will be evaluated in the Safe environment, if

possible.

perl-overview

Here is an overview of the functions that are available to Perl:

```
:perl VIM::Msg("Text") # displays a message
:perl VIM::Msg("Wrong!", "ErrorMsg") # displays an error message
:perl VIM::Msg("remark", "Comment") # displays a highlighted message
:perl VIM::SetOption("ai") # sets a vim option
:perl $nbuf = VIM::Buffers() # returns the number of buffers
:perl @buflist = VIM::Buffers() # returns array of all buffers
:perl $mybuf = (VIM::Buffers('qq.c'))[0] # returns buffer object for 'qq.c'
:perl @winlist = VIM::Windows() # returns array of all windows
:perl $nwin = VIM::Windows() # returns the number of windows
:perl ($success, $v) = VIM::Eval('&path') # $v: option 'path', $success: 1
:perl ($success, $v) = VIM::Eval('&xyz') # $v: ' ' and $success: 0
:perl $v = VIM::Eval('expand("<cfile>")') # expands <cfile>
:perl $curwin->SetHeight(10) # sets the window height
:perl @pos = $curwin->Cursor() # returns (row, col) array
:perl @pos = (10, 10)
:perl $curwin->Cursor(@pos) # sets cursor to @pos
:perl $curwin->Cursor(10,10) # sets cursor to row 10 col 10
:perl $mybuf = $curwin->Buffer() # returns the buffer object for window
:perl $curbuf->Name() # returns buffer name
:perl $curbuf->Number() # returns buffer number
:perl $curbuf->Count() # returns the number of lines
:perl $l = $curbuf->Get(10) # returns line 10
:perl @l = $curbuf->Get(1 .. 5) # returns lines 1 through 5
:perl $curbuf->Delete(10) # deletes line 10
:perl $curbuf->Delete(10, 20) # delete lines 10 through 20
:perl $curbuf->Append(10, "Line") # appends a line
:perl $curbuf->Append(10, "Line1", "Line2", "Line3") # appends 3 lines
:perl @l = ("L1", "L2", "L3")
:perl $curbuf->Append(10, @l) # appends L1, L2 and L3
:perl $curbuf->Set(10, "Line") # replaces line 10
:perl $curbuf->Set(10, "Line1", "Line2") # replaces lines 10 and 11
:perl $curbuf->Set(10, @l) # replaces 3 lines
```

perl-Msg

VIM::Msg({msg}, {group}?)

Displays the message {msg}. The optional {group} argument specifies a highlight group for Vim to use for the message.

perl-SetOption

VIM::SetOption({arg})

Sets a vim option. {arg} can be any argument that the ":set" command accepts. Note that this means that no spaces are allowed in the argument! See :set .

perl-Buffers

VIM::Buffers([{bn}...])

With no arguments, returns a list of all the buffers in an array context or returns the number of buffers in a scalar context. For a list of buffer names or

numbers `{bn}`, returns a list of the buffers matching `{bn}`, using the same rules as Vim's internal `bufname()` function.

WARNING: the list becomes invalid when `:bwipe` is used. Using it anyway may crash Vim.

perl-Windows

`VIM::Windows([{{wn}}...])` With no arguments, returns a list of all the windows in an array context or returns the number of windows in a scalar context. For a list of window numbers `{wn}`, returns a list of the windows with those numbers.

WARNING: the list becomes invalid when a window is closed. Using it anyway may crash Vim.

perl-DoCommand

`VIM::DoCommand({cmd})` Executes Ex command `{cmd}`.

perl-Eval

`VIM::Eval({expr})` Evaluates `{expr}` and returns (success, value) in list context or just value in scalar context. `success=1` indicates that `val` contains the value of `{expr}`; `success=0` indicates a failure to evaluate the expression. '@x' returns the contents of register x, '&x' returns the value of option x, 'x' returns the value of internal variables x, and '\$x' is equivalent to perl's `$ENV{x}`. All functions accessible from the command-line are valid for `{expr}`. A List is turned into a string by joining the items and inserting line breaks.

perl-Blob

`VIM::Blob({expr})` Return Blob literal string `0zXXXX` from scalar value.

perl-SetHeight

`Window->SetHeight({height})`

Sets the Window height to `{height}`, within screen limits.

perl-GetCursor

`Window->Cursor({row}?, {col}?)`

With no arguments, returns a (row, col) array for the current cursor position in the Window. With `{row}` and `{col}` arguments, sets the Window's cursor position to `{row}` and `{col}`. Note that `{col}` is numbered from 0, Perl-fashion, and thus is one less than the value in Vim's ruler.

`Window->Buffer()`

perl-Buffer

Returns the Buffer object corresponding to the given Window.

perl-Name

`Buffer->Name()`

Returns the filename for the Buffer.

perl-Number

Buffer->Number() Returns the number of the Buffer.

perl-Count

Buffer->Count() Returns the number of lines in the Buffer.

perl-Get

Buffer->Get({lnum}, {lnum}?, ...)
Returns a text string of line {lnum} in the Buffer for each {lnum} specified. An array can be passed with a list of {lnum}'s specified.

perl-Delete

Buffer->Delete({lnum}, {lnum}?)
Deletes line {lnum} in the Buffer. With the second {lnum}, deletes the range of lines from the first {lnum} to the second {lnum}.

perl-Append

Buffer->Append({lnum}, {line}, {line}?, ...)
Appends each {line} string after Buffer line {lnum}. The list of {line}s can be an array.

perl-Set

Buffer->Set({lnum}, {line}, {line}?, ...)
Replaces one or more Buffer lines with specified {lines}s, starting at Buffer line {lnum}. The list of {line}s can be an array. If the arguments are invalid, replacement does not occur.

\$main::curwin
The current window object.

\$main::curbuf
The current buffer object.

script-here

When using a script language in-line, you might want to skip this when the language isn't supported.

```

if has('perl')
    perl << EOF
    print 'perl works'
EOF
endif

```

Note that "EOF" must be at the start of the line without preceding white space.

=====

4. Dynamic loading

perl-dynamic

On MS-Windows and Unix the Perl library can be loaded dynamically. The `:version` output then includes `+perl/dyn` .

This means that Vim will search for the Perl DLL or shared library file only when needed. When you don't use the Perl interface you don't need it, thus you can use Vim without this file.

MS-Windows

You can download Perl from <http://www.perl.org>. The one from ActiveState was used for building Vim.

To use the Perl interface the Perl DLL must be in your search path. If Vim reports it cannot find the perl512.dll, make sure your \$PATH includes the directory where it is located. The Perl installer normally does that. In a console window type "path" to see what directories are used. The `'perldll'` option can be also used to specify the Perl DLL.

The name of the DLL must match the Perl version Vim was compiled with. Currently the name is "perl512.dll". That is for Perl 5.12. To know for sure edit "gvim.exe" and search for "perl\d*.dll\c".

Unix

The `'perldll'` option can be used to specify the Perl shared library file instead of DYNAMIC_PERL_DLL file what was specified at compile time. The version of the shared library must match the Perl version Vim was compiled with.

Note: If you are building Perl locally, you have to use a version compiled with threading support for it for Vim to successfully link against it. You can use the `'-Dusetthreads'` flags when configuring Perl, and check that a Perl binary has it enabled by running `'perl -V'` and verify that `'USE_ITHREADS'` is under "Compile-time options".

=====

The Python Interface to Vim

python Python

- 1. Commands python-commands
- 2. The vim module python-vim
- 3. Buffer objects python-buffer
- 4. Range objects python-range
- 5. Window objects python-window
- 6. Tab page objects python-tabpage
- 7. vim.bindeval objects python-bindeval-objects
- 8. pyeval(), py3eval() Vim functions python-pyeval
- 9. Dynamic loading python-dynamic
- 10. Python 3 python3
- 11. Python X python_x
- 12. Building with Python support python-building

The Python 2.x interface is available only when Vim was compiled with the `+python` feature.

The Python 3 interface is available only when Vim was compiled with the `+python3` feature.

Both can be available at the same time, but read `python-2-and-3` .

NOTE: Python 2 is old and no longer being developed. Using Python 3 is highly recommended. Python 2 support will be dropped when it does not work properly anymore.

=====

- 1. Commands python-commands

`:python` `:py` E263 E264 E887

`:[range]py[thon] {stmt}`

Execute Python statement `{stmt}`. A simple check if the ``:python`` command is working:

`:python print "Hello"`

`:[range]py[thon] << [trim] [{endmarker}]`

`{script}`
`{endmarker}`

Execute Python script `{script}`.

Note: This command doesn't work when the Python feature wasn't compiled in. To avoid errors, see `script-here` .

If `[endmarker]` is omitted from after the "`<<`", a dot `.'` must be used after `{script}`, like for the `:append` and `:insert` commands. Refer to `:let-heredoc` for more information.

This form of the `:python` command is mainly useful for including python code in Vim scripts.

Example:

```
function! IcecreamInitialize()
python << EOF
class StrawberryIcecream:
    def __call__(self):
        print 'EAT ME'
EOF
endfunction
```

To see what version of Python you have:

```
:python print(sys.version)
```

There is no need to import sys, it's done by default.

[python-environment](#)

Environment variables set in Vim are not always available in Python. This depends on how Vim and Python were built. Also see <https://docs.python.org/3/library/os.html#os.environ>

Note: Python is very sensitive to the indenting. Make sure the "class" line and "EOF" do not have any indent.

[:pydo](#)

```
:[range]pydo {body}
```

Execute Python function "def _vim_pydo(line, linenr): {body}" for each line in the [range], with the function arguments being set to the text of each line in turn, without a trailing <EOL>, and the current line number. The function should return a string or None. If a string is returned, it becomes the text of the line in the current turn. The default for [range] is the whole file: "1,\$".

Examples:

```
:pydo return "%s\t%d" % (line[:::-1], len(line))
:pydo if line: return "%4d: %s" % (linenr, line)
```

One can use `:pydo` in possible conjunction with `:py` to filter a range using python. For example:

```
:py3 << EOF
needle = vim.eval('@a')
replacement = vim.eval('@b')

def py_vim_string_replace(str):
    return str.replace(needle, replacement)
EOF
:'<,'>py3do return py_vim_string_replace(line)
```

[:pyfile](#) [:pyf](#)

```
:[range]pyf[ile] {file}
```

Execute the Python script in {file}. The whole argument is used as a single file name.

Both of these commands do essentially the same thing - they execute a piece of Python code, with the "current range" `python-range` set to the given line range.

In the case of `:python`, the code to execute is in the command-line.

In the case of `:pyfile`, the code to execute is the contents of the given file.

Python commands cannot be used in the `sandbox` .

To pass arguments you need to set `sys.argv[]` explicitly. Example:

```
:python sys.argv = ["foo", "bar"]
:pyfile myscript.py
```

Here are some examples

`python-examples`

```
:python from vim import *
:python from string import upper
:python current.line = upper(current.line)
:python print "Hello"
:python str = current.buffer[42]
```

(Note that changes - like the imports - persist from one command to the next, just like in the Python interpreter.)

2. The vim module `python-vim`

Python code gets all of its access to vim (with one exception - see `python-output` below) via the "vim" module. The vim module implements two methods, three constants, and one error object. You need to import the vim module before using it:

```
:python import vim
```

Overview

```
:py print "Hello"           # displays a message
:py vim.command(cmd)        # execute an Ex command
:py w = vim.windows[n]      # gets window "n"
:py cw = vim.current.window # gets the current window
:py b = vim.buffers[n]      # gets buffer "n"
:py cb = vim.current.buffer # gets the current buffer
:py w.height = lines        # sets the window height
:py w.cursor = (row, col)   # sets the window cursor position
:py pos = w.cursor          # gets a tuple (row, col)
:py name = b.name           # gets the buffer file name
:py line = b[n]             # gets a line from the buffer
:py lines = b[n:m]          # gets a list of lines
:py num = len(b)            # gets the number of lines
:py b[n] = str               # sets a line in the buffer
:py b[n:m] = [str1, str2, str3] # sets a number of lines at once
:py del b[n]                 # deletes a line
:py del b[n:m]              # deletes a number of lines
```

Methods of the "vim" module

`vim.command(str)` python-command

Executes the vim (ex-mode) command `str`. Returns `None`.

Examples:

```
:py vim.command("set tw=72")
:py vim.command("%s/aaa/bbb/g")
```

The following definition executes Normal mode commands:

```
def normal(str):
    vim.command("normal "+str)
    # Note the use of single quotes to delimit a string containing
    # double quotes
    normal('"a2dd"aP')
```

E659

The `":python"` command cannot be used recursively with Python 2.2 and older. This only works with Python 2.3 and later:

```
:py vim.command("python print 'Hello again Python'")
```

`vim.eval(str)` python-eval

Evaluates the expression `str` using the vim internal expression evaluator (see [expression](#)). Returns the expression result as:

- a string if the Vim expression evaluates to a string or number
- a list if the Vim expression evaluates to a Vim list
- a dictionary if the Vim expression evaluates to a Vim dictionary

Dictionaries and lists are recursively expanded.

Examples:

```
:" value of the 'textwidth' option
:py text_width = vim.eval("&tw")
:
:" contents of the 'a' register
:py a_reg = vim.eval("@a")
:
:" Result is a string! Use string.atoi() to convert to a number.
:py str = vim.eval("12+12")
:
:py tagList = vim.eval('taglist("eval_expr")')
```

The latter will return a python list of python dicts, for instance:

```
[{'cmd': '/^eval_expr(arg, nextcmd)$/ ', 'static': 0, 'name':
'eval_expr', 'kind': 'f', 'filename': './src/eval.c'}]
```

`vim.bindeval(str)` python-bindeval

Like [python-eval](#), but returns special objects described in [python-bindeval-objects](#). These python objects let you modify ([List](#) or [Dictionary](#)) or call ([Funcref](#)) vim objects.

`vim.strwidth(str)` python-strwidth

Like [strwidth\(\)](#): returns number of display cells `str` occupies, tab is counted as one cell.

`vim.foreach_rtp(callable)` python-foreach_rtp

Call the given callable for each path in ['runtimepath'](#) until either callable returns something but `None`, the exception is raised or there are no longer paths. If stopped in case callable returned non-`None`,

`vim.foreach_rtp` function returns the value returned by callable.

`vim.chdir(*args, **kwargs)`
`vim.fchdir(*args, **kwargs)`

[python-chdir](#)
[python-fchdir](#)

Run `os.chdir` or `os.fchdir`, then all appropriate vim stuff.

Note: you should not use these functions directly, use `os.chdir` and `os.fchdir` instead. Behavior of `vim.fchdir` is undefined in case `os.fchdir` does not exist.

Error object of the "vim" module

`vim.error`

[python-error](#)

Upon encountering a Vim error, Python raises an exception of type `vim.error`.

Example:

```
try:
    vim.command("put a")
except vim.error:
    # nothing in register a
```

Constants of the "vim" module

Note that these are not actually constants - you could reassign them. But this is silly, as you would then lose access to the vim objects to which the variables referred.

`vim.buffers`

[python-buffers](#)

A mapping object providing access to the list of vim buffers. The object supports the following operations:

```
:py b = vim.buffers[i]      # Indexing (read-only)
:py b in vim.buffers        # Membership test
:py n = len(vim.buffers)    # Number of elements
:py for b in vim.buffers:   # Iterating over buffer list
```

`vim.windows`

[python-windows](#)

A sequence object providing access to the list of vim windows. The object supports the following operations:

```
:py w = vim.windows[i]     # Indexing (read-only)
:py w in vim.windows       # Membership test
:py n = len(vim.windows)   # Number of elements
:py for w in vim.windows:  # Sequential access
```

Note: `vim.windows` object always accesses current tab page.

[python-tabpage](#) `.windows` objects are bound to parent [python-tabpage](#) object and always use windows from that tab page (or throw `vim.error` in case tab page was deleted). You can keep a reference to both without keeping a reference to vim module object or [python-tabpage](#), they will not lose their properties in this case.

`vim.tabpages`

[python-tabpages](#)

A sequence object providing access to the list of vim tab pages. The object supports the following operations:

```
:py t = vim.tabpages[i]    # Indexing (read-only)
:py t in vim.tabpages      # Membership test
:py n = len(vim.tabpages)  # Number of elements
```

```
:py for t in vim.tabpages: # Sequential access
```

`vim.current` [python-current](#)

An object providing access (via specific attributes) to various "current" objects available in vim:

<code>vim.current.line</code>	The current line (RW)	String
<code>vim.current.buffer</code>	The current buffer (RW)	Buffer
<code>vim.current.window</code>	The current window (RW)	Window
<code>vim.current.tabpage</code>	The current tab page (RW)	TabPage
<code>vim.current.range</code>	The current line range (RO)	Range

The last case deserves a little explanation. When the `:python` or `:pyfile` command specifies a range, this range of lines becomes the "current range". A range is a bit like a buffer, but with all access restricted to a subset of lines. See [python-range](#) for more details.

Note: When assigning to `vim.current.{buffer,window,tabpage}` it expects valid [python-buffer](#), [python-window](#) or [python-tabpage](#) objects respectively. Assigning triggers normal (with [autocommands](#)) switching to given buffer, window or tab page. It is the only way to switch UI objects in python: you can't assign to [python-tabpage](#).`window` attribute. To switch without triggering autocommands use

```
py << EOF
saved_eventignore = vim.options['eventignore']
vim.options['eventignore'] = 'all'
try:
    vim.current.buffer = vim.buffers[2] # Switch to buffer 2
finally:
    vim.options['eventignore'] = saved_eventignore
EOF
```

`vim.vars` [python-vars](#)
`vim.vvars` [python-vvars](#)

Dictionary-like objects holding dictionaries with global (`g:`) and vim (`v:`) variables respectively. Identical to ``vim.bindeval("g:")``, but faster.

`vim.options` [python-options](#)

Object partly supporting mapping protocol (supports setting and getting items) providing a read-write access to global options.

Note: unlike `:set` this provides access only to global options. You cannot use this object to obtain or set local options' values or access local-only options in any fashion. Raises `KeyError` if no global option with such name exists (i.e. does not raise `KeyError` for [global-local](#) options and global only options, but does for window- and buffer-local ones). Use [python-buffer](#) objects to access to buffer-local options and [python-window](#) objects to access to window-local options.

Type of this object is available via "Options" attribute of vim module.

Output from Python [python-output](#)

Vim displays all Python code output in the Vim message area. Normal output appears as information messages, and error output appears as error messages.

In implementation terms, this means that all output to `sys.stdout` (including the output from `print` statements) appears as information messages, and all output to `sys.stderr` (including error tracebacks) appears as error messages.

Input (via `sys.stdin`, including `input()` and `raw_input()`) is not supported, and may cause the program to crash. This should probably be fixed.

Python `'runtimepath'` handling `python-input`
`python2-directory` `python3-directory` `pythonx-directory`
`python-special-path`

In python `vim.VIM_SPECIAL_PATH` special directory is used as a replacement for the list of paths found in `'runtimepath'`: with this directory in `sys.path` and `vim.path_hooks` in `sys.path_hooks` python will try to load module from `{rtp}/python2` (or `python3`) and `{rtp}/pythonx` (for both python versions) for each `{rtp}` found in `'runtimepath'`.

Implementation is similar to the following, but written in C:

```
from imp import find_module, load_module
import vim
import sys

class VimModuleLoader(object):
    def __init__(self, module):
        self.module = module

    def load_module(self, fullname, path=None):
        return self.module

def _find_module(fullname, oldtail, path):
    idx = oldtail.find('.')
    if idx > 0:
        name = oldtail[:idx]
        tail = oldtail[idx+1:]
        fmr = find_module(name, path)
        module = load_module(fullname[:-len(oldtail)] + name, *fmr)
        return _find_module(fullname, tail, module.__path__)
    else:
        fmr = find_module(fullname, path)
        return load_module(fullname, *fmr)

# It uses vim module itself in place of VimPathFinder class: it does not
# matter for python which object has find_module function attached to as
# an attribute.
class VimPathFinder(object):
    @classmethod
    def find_module(cls, fullname, path=None):
```

```

    try:
        return VimModuleLoader(_find_module(fullname, fullname, path or vim._get_paths()))
    except ImportError:
        return None

    @classmethod
    def load_module(cls, fullname, path=None):
        return _find_module(fullname, fullname, path or vim._get_paths())

def hook(path):
    if path == vim.VIM_SPECIAL_PATH:
        return VimPathFinder
    else:
        raise ImportError

sys.path_hooks.append(hook)

```

vim.VIM_SPECIAL_PATH python-VIM_SPECIAL_PATH
String constant used in conjunction with vim path hook. If path hook installed by vim is requested to handle anything but path equal to vim.VIM_SPECIAL_PATH constant it raises ImportError. In the only other case it uses special loader.

Note: you must not use value of this constant directly, always use vim.VIM_SPECIAL_PATH object.

vim.find_module(...) python-find_module
vim.path_hook(path) python-path_hook
Methods or objects used to implement path loading as described above. You should not be using any of these directly except for vim.path_hook in case you need to do something with sys.meta_path. It is not guaranteed that any of the objects will exist in the future vim versions.

vim._get_paths python-_get_paths
Methods returning a list of paths which will be searched for by path hook. You should not rely on this method being present in future versions, but can use it for debugging.

It returns a list of {rtp}/python2 (or {rtp}/python3) and {rtp}/pythonx directories for each {rtp} in 'runtimepath'.

3. Buffer objects python-buffer

Buffer objects represent vim buffers. You can obtain them in a number of ways:

- via vim.current.buffer (python-current)
- from indexing vim.buffers (python-buffers)
- from the "buffer" attribute of a window (python-window)

Buffer objects have two read-only attributes - name - the full file name for the buffer, and number - the buffer number. They also have three methods (append, mark, and range; see below).

You can also treat buffer objects as sequence objects. In this context, they act as if they were lists (yes, they are mutable) of strings, with each element being a line of the buffer. All of the usual sequence operations, including indexing, index assignment, slicing and slice assignment, work as you would expect. **Note** that the result of indexing (slicing) a buffer is a string (list of strings). This has one unusual consequence - `b[:]` is different from `b`. In particular, `"b[:] = None"` deletes the whole of the buffer, whereas `"b = None"` merely updates the variable `b`, with no effect on the buffer.

Buffer indexes start at zero, as is normal in Python. This differs from vim line numbers, which start from 1. This is particularly relevant when dealing with marks (see below) which use vim line numbers.

The buffer object attributes are:

<code>b.vars</code>	Dictionary-like object used to access <code>buffer-variable</code> <code>s</code> .
<code>b.options</code>	Mapping object (supports item getting, setting and deleting) that provides access to buffer-local options and buffer-local values of <code>global-local</code> options. Use <code>python-window</code> <code>.options</code> if option is window-local, this object will raise <code>KeyError</code> . If option is <code>global-local</code> and local value is missing getting it will return <code>None</code> .
<code>b.name</code>	String, RW. Contains buffer name (full path). Note: when assigning to <code>b.name</code> <code>BufFilePre</code> and <code>BufFilePost</code> autocommands are launched.
<code>b.number</code>	Buffer number. Can be used as <code>python-buffers</code> key. Read-only.
<code>b.valid</code>	True or False. Buffer object becomes invalid when corresponding buffer is wiped out.

The buffer object methods are:

<code>b.append(str)</code>	Append a line to the buffer
<code>b.append(str, nr)</code>	Idem, below line "nr"
<code>b.append(list)</code>	Append a list of lines to the buffer Note that the option of supplying a list of strings to the append method differs from the equivalent method for Python's built-in list objects.
<code>b.append(list, nr)</code>	Idem, below line "nr"
<code>b.mark(name)</code>	Return a tuple (row,col) representing the position of the named mark (can also get the <code>[]"<></code> marks)
<code>b.range(s,e)</code>	Return a range object (see <code>python-range</code>) which represents the part of the given buffer between line numbers <code>s</code> and <code>e</code> <code>inclusive</code> .

Note that when adding a line it must not contain a line break character `'\n'`. A trailing `'\n'` is allowed and ignored, so that you can do:

```
:py b.append(f.readlines())
```

Buffer object type is available using "Buffer" attribute of vim module.

Examples (assume `b` is the current buffer)

```
:py print b.name          # write the buffer file name
:py b[0] = "hello!!!"    # replace the top line
```

```

:py b[:] = None           # delete the whole buffer
:py del b[:]             # delete the whole buffer
:py b[0:0] = [ "a line" ] # add a line at the top
:py del b[2]             # delete a line (the third)
:py b.append("bottom")   # add a line at the bottom
:py n = len(b)           # number of lines
:py (row,col) = b.mark('a') # named mark
:py r = b.range(1,5)     # a sub-range of the buffer
:py b.vars["foo"] = "bar" # assign b:foo variable
:py b.options["ff"] = "dos" # set fileformat
:py del b.options["ar"]  # same as :set autoread<

```

4. Range objects

python-range

Range objects represent a part of a vim buffer. You can obtain them in a number of ways:

- via `vim.current.range` ([python-current](#))
- from a buffer's `range()` method ([python-buffer](#))

A range object is almost identical in operation to a buffer object. However, all operations are restricted to the lines within the range (this line range can, of course, change as a result of slice assignments, line deletions, or the `range.append()` method).

The range object attributes are:

```

r.start      Index of first line into the buffer
r.end        Index of last line into the buffer

```

The range object methods are:

```

r.append(str)  Append a line to the range
r.append(str, nr) Idem, after line "nr"
r.append(list) Append a list of lines to the range
                Note that the option of supplying a list of strings to
                the append method differs from the equivalent method
                for Python's built-in list objects.
r.append(list, nr) Idem, after line "nr"

```

Range object type is available using "Range" attribute of vim module.

Example (assume r is the current range):

```

# Send all lines in a range to the default printer
vim.command("%d,%dhardcopy!" % (r.start+1,r.end+1))

```

5. Window objects

python-window

Window objects represent vim windows. You can obtain them in a number of ways:

- via `vim.current.window` ([python-current](#))
- from indexing `vim.windows` ([python-windows](#))
- from indexing "windows" attribute of a tab page ([python-tabpage](#))
- from the "window" attribute of a tab page ([python-tabpage](#))

You can manipulate window objects only through their attributes. They have no

methods, and no sequence or other interface.

Window attributes are:

buffer (read-only)	The buffer displayed in this window
cursor (read-write)	The current cursor position in the window This is a tuple, (row,col).
height (read-write)	The window height, in rows
width (read-write)	The window width, in columns
vars (read-only)	The window <code>w:</code> variables. Attribute is unassignable, but you can change window variables this way
options (read-only)	The window-local options. Attribute is unassignable, but you can change window options this way. Provides access only to window-local options, for buffer-local use <code>python-buffer</code> and for global ones use <code>python-options</code> . If option is <code>global-local</code> and local value is missing getting it will return None.
number (read-only)	Window number. The first window has number 1. This is zero in case it cannot be determined (e.g. when the window object belongs to other tab page).
row, col (read-only)	On-screen window position in display cells. First position is zero.
tabpage (read-only)	Window tab page.
valid (read-write)	True or False. Window object becomes invalid when corresponding window is closed.

The height attribute is writable only if the screen is split horizontally.
The width attribute is writable only if the screen is split vertically.

Window object type is available using "Window" attribute of vim module.

6. Tab page objects

`python-tabpage`

Tab page objects represent vim tab pages. You can obtain them in a number of ways:

- via `vim.current.tabpage` (`python-current`)
- from indexing `vim.tabpages` (`python-tabpages`)

You can use this object to access tab page windows. They have no methods and no sequence or other interfaces.

Tab page attributes are:

number	The tab page number like the one returned by <code>tabpagenr()</code> .
windows	Like <code>python-windows</code> , but for current tab page.
vars	The tab page <code>t:</code> variables.
window	Current tabpage window.
valid	True or False. Tab page object becomes invalid when corresponding tab page is closed.

TabPage object type is available using "TabPage" attribute of vim module.

7. vim.bindeval objects

[python-bindeval-objects](#)

vim.Dictionary object

[python-Dictionary](#)

Dictionary-like object providing access to vim [Dictionary](#) type.

Attributes:

Attribute	Description								
locked	One of python-.locked <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>zero</td><td>Variable is not locked</td></tr><tr><td>vim.VAR_LOCKED</td><td>Variable is locked, but can be unlocked</td></tr><tr><td>vim.VAR_FIXED</td><td>Variable is locked and can't be unlocked</td></tr></tbody></table> Read-write. You can unlock locked variable by assigning <code>`True`</code> or <code>`False`</code> to this attribute. No recursive locking is supported.	Value	Description	zero	Variable is not locked	vim.VAR_LOCKED	Variable is locked, but can be unlocked	vim.VAR_FIXED	Variable is locked and can't be unlocked
Value	Description								
zero	Variable is not locked								
vim.VAR_LOCKED	Variable is locked, but can be unlocked								
vim.VAR_FIXED	Variable is locked and can't be unlocked								
scope	One of <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>zero</td><td>Dictionary is not a scope one</td></tr><tr><td>vim.VAR_DEF_SCOPE</td><td><code>g:</code> or <code>l:</code> dictionary</td></tr><tr><td>vim.VAR_SCOPE</td><td>Other scope dictionary, see internal-variables</td></tr></tbody></table>	Value	Description	zero	Dictionary is not a scope one	vim.VAR_DEF_SCOPE	<code>g:</code> or <code>l:</code> dictionary	vim.VAR_SCOPE	Other scope dictionary, see internal-variables
Value	Description								
zero	Dictionary is not a scope one								
vim.VAR_DEF_SCOPE	<code>g:</code> or <code>l:</code> dictionary								
vim.VAR_SCOPE	Other scope dictionary, see internal-variables								

Methods (note: methods do not support keyword arguments):

Method	Description
keys()	Returns a list with dictionary keys.
values()	Returns a list with dictionary values.
items()	Returns a list of 2-tuples with dictionary contents.
update(iterable), update(dictionary), update(**kwargs)	Adds keys to dictionary.
get(key[, default=None])	Obtain key from dictionary, returning the default if it is not present.
pop(key[, default])	Remove specified key from dictionary and return corresponding value. If key is not found and default is given returns the default, otherwise raises <code>KeyError</code> .
popitem()	Remove random key from dictionary and return (key, value) pair.
has_key(key)	Check whether dictionary contains specified key, similar to <code>`key in dict`</code> .
<code>__new__()</code> , <code>__new__(iterable)</code> , <code>__new__(dictionary)</code> , <code>__new__(update)</code>	You can use <code>`vim.Dictionary()`</code> to create new vim dictionaries. <code>`d=vim.Dictionary(arg)`</code> is the same as <code>`d=vim.bindeval('{'});d.update(arg)`</code> . Without arguments constructs empty dictionary.

Examples:

```
d = vim.Dictionary(food="bar")      # Constructor
d['a'] = 'b'                        # Item assignment
print d['a']                         # getting item
```

```

d.update({'c': 'd'})           # .update(dictionary)
d.update(e='f')               # .update(**kwargs)
d.update((('g', 'h'), ('i', 'j')) # .update(iterable)
for key in d.keys():          # .keys()
for val in d.values():        # .values()
for key, val in d.items():    # .items()
print isinstance(d, vim.Dictionary) # True
for key in d:                 # Iteration over keys
class Dict(vim.Dictionary):   # Subclassing

```

Note: when iterating over keys you should not modify dictionary.

vim.List object

[python-List](#)

Sequence-like object providing access to vim [List](#) type.

Supports `.locked` attribute, see [python-.locked](#) . Also supports the following methods:

Method	Description
<code>extend(item)</code>	Add items to the list.
<code>__new__()</code> , <code>__new__(iterable)</code>	You can use <code>vim.List()</code> to create new vim lists. <code>vim.List(iterable)</code> is the same as <code>vim.bindeval('[]');l.extend(iterable)</code> . Without arguments constructs empty list.

Examples:

```

l = vim.List("abc")           # Constructor, result: ['a', 'b', 'c']
l.extend(['abc', 'def'])     # .extend() method
print l[1:]                  # slicing
l[:0] = ['ghi', 'jkl']      # slice assignment
print l[0]                   # getting item
l[0] = 'mno'                 # assignment
for i in l:                  # iteration
print isinstance(l, vim.List) # True
class List(vim.List):       # Subclassing

```

vim.Function object

[python-Function](#)

Function-like object, acting like vim [Funcref](#) object. Accepts special keyword argument `self`, see [Dictionary-function](#) . You can also use `vim.Function(name)` constructor, it is the same as `vim.bindeval('function(%s)%json.dumps(name)')`.

Attributes (read-only):

Attribute	Description
<code>name</code>	Function name.
<code>args</code>	<code>None</code> or a python-List object with arguments. Note that this is a copy of the arguments list, constructed each time you request this attribute. Modifications made to the list will be ignored (but not to the containers inside argument list: this is like <code>copy()</code> and not <code>deepcopy()</code>).
<code>self</code>	<code>None</code> or a python-Dictionary object with self dictionary. Note that explicit <code>self</code> keyword used when calling resulting object overrides this attribute.
<code>auto_rebind</code>	Boolean. True if partial created from this Python object

and stored in the Vim script dictionary should be automatically rebound to the dictionary it is stored in when this dictionary is indexed. Exposes Vim internal difference between `dict.func` (auto_rebind=True) and `function(dict.func,dict)` (auto_rebind=False). This attribute makes no sense if `self` attribute is `None`.

Constructor additionally accepts `args`, `self` and `auto_rebind` keywords. If `args` and/or `self` argument is given then it constructs a partial, see `function()`. `auto_rebind` is only used when `self` argument is given, otherwise it is assumed to be `True` regardless of whether it was given or not. If `self` is given then it defaults to `False`.

Examples:

```
f = vim.Function('tr')           # Constructor
print f('abc', 'a', 'b')       # Calls tr('abc', 'a', 'b')
vim.command('''
    function DictFun() dict
        return self
    endfunction
''')
f = vim.bindeval('function("DictFun")')
print f(self={})                # Like call('DictFun', [], {})
print isinstance(f, vim.Function) # True

p = vim.Function('DictFun', self={})
print f()
p = vim.Function('tr', args=['abc', 'a'])
print f('b')
```

8. `pyeval()` and `py3eval()` Vim functions

[python-pyeval](#)

To facilitate bi-directional interface, you can use `pyeval()` and `py3eval()` functions to evaluate Python expressions and pass their values to Vim script. `pyxeval()` is also available.

The Python value "None" is converted to `v:none`.

9. Dynamic loading

[python-dynamic](#)

On MS-Windows and Unix the Python library can be loaded dynamically. The `:version` output then includes `+python/dyn` or `+python3/dyn`.

This means that Vim will search for the Python DLL or shared library file only when needed. When you don't use the Python interface you don't need it, thus you can use Vim without this file.

MS-Windows

To use the Python interface the Python DLL must be in your search path. In a

console window type "path" to see what directories are used. If the DLL is not found in your search path, Vim will check the registry to find the path where Python is installed. The **'pythondll'** or **'pythonthreadll'** option can be also used to specify the Python DLL.

The name of the DLL should match the Python version Vim was compiled with. Currently the name for Python 2 is "python27.dll", that is for Python 2.7. That is the default value for **'pythondll'**. For Python 3 it is python36.dll (Python 3.6). To know for sure edit "gvim.exe" and search for "python\d*.dll\c".

Unix

The **'pythondll'** or **'pythonthreadll'** option can be used to specify the Python shared library file instead of DYNAMIC_PYTHON_DLL or DYNAMIC_PYTHON3_DLL file what were specified at compile time. The version of the shared library must match the Python 2.x or Python 3 version (`v:python3_version`) Vim was compiled with unless using `python3-stable-abi` .

Stable ABI and mixing Python versions

`python-stable` `python-stable-abi` `python3-stable-abi`
If Vim was not compiled with Stable ABI (only available for Python 3), the version of the Python shared library must match the version that Vim was compiled with. Otherwise, mixing versions could result in unexpected crashes and failures. With Stable ABI, this restriction is relaxed, and any Python 3 library with version of at least `v:python3_version` will work. See `has-python` for how to check if Stable ABI is supported, or see if version output includes `+python3/dyn-stable` .
On MS-Windows, **'pythonthreadll'** will be set to "python3.dll". When searching the DLL from the registry, Vim will search the latest version of Python.

=====

10. Python 3

`python3`

`:py3` `:python3`

```
:[range]py3 {stmt}
:[range]py3 << [trim] [{endmarker}]
{script}
{endmarker}
```

```
:[range]python3 {stmt}
:[range]python3 << [trim] [{endmarker}]
{script}
{endmarker}
```

The ``:py3`` and ``:python3`` commands work similar to ``:python``. A simple check if the ``:py3`` command is working:

```
:py3 print("Hello")
```

To see what version of Python you have:

```
:py3 import sys
:py3 print(sys.version)
```

`:py3file`

```
:[range]py3f[file] {file}
```

The `:py3file` command works similar to `:pyfile`.

`:py3do`

```
:[range]py3do {body}
```

The `:py3do` command works similar to `:pydo`.

Vim can be built in four ways (:version output):

1. No Python support (-python, -python3)
2. Python 2 support only (+python or +python/dyn, -python3)
3. Python 3 support only (-python, +python3 or +python3/dyn)
4. Python 2 and 3 support (+python/dyn, +python3/dyn)

Some more details on the special case 4: [python-2-and-3](#)

When Python 2 and Python 3 are both supported they must be loaded dynamically.

When doing this on Linux/Unix systems and importing global symbols, this leads to a crash when the second Python version is used. So either global symbols are loaded but only one Python version is activated, or no global symbols are loaded. The latter makes Python's "import" fail on libraries that expect the symbols to be provided by Vim.

[E836](#) [E837](#)

Vim's configuration script makes a guess for all libraries based on one standard Python library (termios). If importing this library succeeds for both Python versions, then both will be made available in Vim at the same time. If not, only the version first used in a session will be enabled. When trying to use the other one you will get the E836 or E837 error message.

Here Vim's behavior depends on the system in which it was configured. In a system where both versions of Python were configured with `--enable-shared`, both versions of Python will be activated at the same time. There will still be problems with other third party libraries that were not linked to libPython.

To work around such problems there are these options:

1. The problematic library is recompiled to link to the according libpython.so.
2. Vim is recompiled for only one Python version.
3. You undefine `PY_NO_RTLD_GLOBAL` in `auto/config.h` after configuration. This may crash Vim though.

[E880](#)

Raising `SystemExit` exception in python isn't endorsed way to quit vim, use:

```
:py vim.command("qall!")
```

[E1266](#)

This error can occur when Python 3 cannot load the required modules. This means that your Python 3 is not correctly installed or there are some mistakes in your settings. Please check the following items:

1. Make sure that Python 3 is correctly installed. Also check the version of python.
2. Check the `'pythonthreedll'` option.
3. Check the `'pythonthreehome'` option.

4. Check the PATH environment variable if you don't set `'pythonthreadll'`. On MS-Windows, you can use `where.exe` to check which dll will be loaded. E.g.


```
where.exe python310.dll
```
5. Check the PYTHONPATH and PYTHONHOME environment variables.

`has-python`

You can test what Python version is available with:

```
if has('python')
  echo 'there is Python 2.x'
endif
if has('python3')
  echo 'there is Python 3.x'
endif
```

Note however, that when Python 2 and 3 are both available and loaded dynamically, these `has()` calls will try to load them. If only one can be loaded at a time, just checking if Python 2 or 3 are available will prevent the other one from being available.

To avoid loading the dynamic library, only check if Vim was compiled with python support:

```
if has('python_compiled')
  echo 'compiled with Python 2.x support'
  if has('python_dynamic')
    echo 'Python 2.x dynamically loaded'
  endif
endif
if has('python3_compiled')
  echo 'compiled with Python 3.x support'
  if has('python3_dynamic')
    echo 'Python 3.x dynamically loaded'
  endif
endif
```

When loading the library dynamically, Vim can be compiled to support Python 3 Stable ABI (`python3-stable-abi`) which allows you to load a different version of Python 3 library than the one Vim was compiled with. To check it:

```
if has('python3_dynamic')
  if has('python3_stable')
    echo 'support Python 3 Stable ABI.'
  else
    echo 'does not support Python 3 Stable ABI.'
    echo 'only use Python 3 version ' .. v:python3_version
  endif
endif
```

This also tells you whether Python is dynamically loaded, which will fail if the runtime library cannot be found.

```
=====
11. Python X python_x pythonx
```

Because most python code can be written so that it works with Python 2.6+ and

Python 3 the `pyx*` functions and commands have been written. They work exactly the same as the Python 2 and 3 variants, but select the Python version using the `'pyxversion'` setting.

You should set `'pyxversion'` in your `.vimrc` to prefer Python 2 or Python 3 for Python commands. If you change this setting at runtime you may risk that state of plugins (such as initialization) may be lost.

If you want to use a module, you can put it in the `{rtp}/pythonx` directory. See `pythonx-directory`.

The `:pyx` and `:pythonx` commands work similar to `:python`. A simple check if the `:pyx` command is working:

```
:pyx print("Hello")
```

To see what version of Python is being used:

```
:pyx import sys
:pyx print(sys.version)
```

The `:pyxfile` command works similar to `:pyfile`. However you can add one of these comments to force Vim using `:pyfile` or `:py3file`:

```
#!/any string/python2      " Shebang. Must be the first line of the file.
#!/any string/python3      " Shebang. Must be the first line of the file.
# requires python 2.x      " Maximum lines depend on 'modelines'.
# requires python 3.x      " Maximum lines depend on 'modelines'.
```

Unlike normal modelines, the bottom of the file is not checked.

If none of them are found, the `'pyxversion'` setting is used.

W20 W21

If Vim does not support the selected Python version a silent message will be printed. Use `:messages` to read them.

:pyxdo

The `:pyxdo` command works similar to `:pydo`.

has-pythonx

You can test if `pyx*` commands are available with:

```
if has('pythonx')
    echo 'pyx* commands are available. (Python ' .. &pyx .. ')'
endif
```

When compiled with only one of `+python` or `+python3`, the `has()` returns 1. When compiled with both `+python` and `+python3`, the test depends on the `'pyxversion'` setting. If `'pyxversion'` is 0, it tests Python 3 first, and if it is not available then Python 2. If `'pyxversion'` is 2 or 3, it tests only Python 2 or 3 respectively.

Note that for `has('pythonx')` to work it may try to dynamically load Python 3 or 2. This may have side effects, especially when Vim can only load one of the two.

If a user prefers Python 2 and want to fallback to Python 3, he needs to set `'pyxversion'` explicitly in his `.vimrc`. E.g.:

```
if has('python')
    set pyx=2
elseif has('python3')
    set pyx=3
endif
```

12. Building with Python support

[python-building](#)

A few hints for building with Python 2 or 3 support.

UNIX

See src/Makefile for how to enable including the Python interface.

On Ubuntu you will want to install these packages for Python 2:

```
python
python-dev
```

For Python 3:

```
python3
python3-dev
```

For Python 3.6:

```
python3.6
python3.6-dev
```

If you have more than one version of Python 3, you need to link python3 to the one you prefer, before running configure.

The Tcl Interface to Vim

tcl Tcl TCL

1. Commands	tcl-ex-commands
2. Tcl commands	tcl-commands
3. Tcl variables	tcl-variables
4. Tcl window commands	tcl-window-cmds
5. Tcl buffer commands	tcl-buffer-cmds
6. Miscellaneous; Output from Tcl	tcl-misc tcl-output
7. Known bugs & problems	tcl-bugs
8. Examples	tcl-examples
9. Dynamic loading	tcl-dynamic

{only available when Vim was compiled with the |+tcl| feature}

E280

WARNING: There are probably still some bugs. Please send bug reports, comments, ideas etc to <Ingo.Wilken@informatik.uni-oldenburg.de>

```
=====
```

1. Commands	tcl-ex-commands	E571	E572
-------------	-----------------	------	------

:tcl

```
:tcl {cmd}          Execute Tcl command {cmd}. A simple check if `:tcl`
                    is working:
                    :tcl puts "Hello"
```

```
:[range]tcl << [trim] [{endmarker}]
{script}
{endmarker}
```

```
Execute Tcl script {script}.
Note: This command doesn't work when the Tcl feature
wasn't compiled in. To avoid errors, see
script-here .
```

If [endmarker] is omitted from after the "<<", a dot '.' must be used after {script}, like for the :append and :insert commands. Refer to :let-heredoc for more information.

This form of the :tcl command is mainly useful for including tcl code in Vim scripts.

Example:

```
function! DefineDate()
  tcl << EOF
  proc date {} {
    return [clock format [clock seconds]]
  }
EOF
```

endfunction

To see what version of Tcl you have:

```
:tcl puts [info patchlevel]
```

```
:[range]tcl[o] {cmd} :tcldo :tcl
Execute Tcl command {cmd} for each line in [range]
with the variable "line" being set to the text of each
line in turn, and "lnum" to the line number. Setting
"line" will change the text, but note that it is not
possible to add or delete lines using this command.
If {cmd} returns an error, the command is interrupted.
The default for [range] is the whole file: "1,$".
See tcl-var-line and tcl-var-lnum .
```

```
:tclfile :tclf
:tclf[ile] {file} Execute the Tcl script in {file}. This is the same as
":tcl source {file}", but allows file name completion.
```

Note that Tcl objects (like variables) persist from one command to the next, just as in the Tcl shell.

Executing Tcl commands is not possible in the [sandbox](#) .

2. Tcl commands

[tcl-commands](#)

Tcl code gets all of its access to vim via commands in the "::

```
::vim::beep # Guess.
::vim::buffer {n} # Create Tcl command for one buffer.
::vim::buffer list # Create Tcl commands for all buffers.
::vim::command [-quiet] {cmd} # Execute an Ex command.
::vim::expr {expr} # Use Vim's expression evaluator.
::vim::option {opt} # Get vim option.
::vim::option {opt} {val} # Set vim option.
::vim::window list # Create Tcl commands for all windows.
```

Commands:

```
::vim::beep tcl-beep
Honk. Does not return a result.
```

```
::vim::buffer {n} tcl-buffer
::vim::buffer exists {n}
::vim::buffer list
Provides access to vim buffers. With an integer argument, creates a
buffer command (see tcl-buffer-cmds ) for the buffer with that
number, and returns its name as the result. Invalid buffer numbers
result in a standard Tcl error. To test for valid buffer numbers,
vim's internal functions can be used:
set nbufs [::vim::expr bufnr("$")]
```

```
set isvalid [::vim::expr "bufexists($n)"]
```

The "list" option creates a buffer command for each valid buffer, and returns a list of the command names as the result.

Example:

```
set bufs [::vim::buffer list]
foreach b $bufs { $b append end "The End!" }
```

The "exists" option checks if a buffer with the given number exists.

Example:

```
if { [::vim::buffer exists $n] } { ::vim::command ":e #n" }
```

This command might be replaced by a variable in future versions.

See also [tcl-var-current](#) for the current buffer.

```
::vim::command {cmd} tcl-command
```

```
::vim::command -quiet {cmd}
```

Execute the vim (ex-mode) command {cmd}. Any Ex command that affects a buffer or window uses the current buffer/current window. Does not return a result other than a standard Tcl error code. After this command is completed, the "::vim::current" variable is updated.

The "-quiet" flag suppresses any error messages from vim.

Examples:

```
::vim::command "set ts=8"
::vim::command "%s/foo/bar/g"
```

To execute normal-mode commands, use "normal" (see [:normal](#)):

```
set cmd "jj"
::vim::command "normal $cmd"
```

See also [tcl-window-command](#) and [tcl-buffer-command](#).

```
::vim::expr {expr} tcl-expr
```

Evaluates the expression {expr} using vim's internal expression evaluator (see [expression](#)). Any expression that queries a buffer or window property uses the current buffer/current window. Returns the result as a string. A [List](#) is turned into a string by joining the items and inserting line breaks.

Examples:

```
set perl_available [::vim::expr has("perl")]
```

See also [tcl-window-expr](#) and [tcl-buffer-expr](#).

```
::vim::option {opt} tcl-option
```

```
::vim::option {opt} {value}
```

Without second argument, queries the value of a vim option. With this argument, sets the vim option to {value}, and returns the previous value as the result. Any options that are marked as 'local to buffer' or 'local to window' affect the current buffer/current window. The global value is not changed, use the ":set" command for that. For boolean options, {value} should be "0" or "1", or any of the keywords "on", "off" or "toggle". See [option-summary](#) for a list of options.

Example:

```
::vim::option ts 8
```

See also [tcl-window-option](#) and [tcl-buffer-option](#).

```
::vim::window {option} tcl-window
```

Provides access to vim windows. Currently only the "list" option is implemented. This creates a window command (see [tcl-window-cmds](#)) for each window, and returns a list of the command names as the result.

Example:

```
set wins [::vim::window list]
foreach w $wins { $w height 4 }
```

This command might be replaced by a variable in future versions.
See also [tcl-var-current](#) for the current window.

3. Tcl variables

[tcl-variables](#)

The `::vim` namespace contains a few variables. These are created when the Tcl interpreter is called from vim and set to current values.

```
::vim::current      # array containing "current" objects
::vim::lbase        # number of first line
::vim::range        # array containing current range numbers
line                # current line as a string (:tcldo only)
lnum                # current line number (:tcldo only)
```

Variables:

`::vim::current` [tcl-var-current](#)

This is an array providing access to various "current" objects available in vim. The contents of this array are updated after `::vim::command` is called, as this might change vim's current settings (e.g., by deleting the current buffer).

The "buffer" element contains the name of the buffer command for the current buffer. This can be used directly to invoke buffer commands (see [tcl-buffer-cmds](#)). This element is read-only.

Example:

```
::$vim::current(buffer) insert begin "Hello world"
```

The "window" element contains the name of the window command for the current window. This can be used directly to invoke window commands (see [tcl-window-cmds](#)). This element is read-only.

Example:

```
::$vim::current(window) height 10
```

`::vim::lbase` [tcl-var-lbase](#)

This variable controls how Tcl treats line numbers. If it is set to '1', then lines and columns start at 1. This way, line numbers from Tcl commands and vim expressions are compatible. If this variable is set to '0', then line numbers and columns start at 0 in Tcl. This is useful if you want to treat a buffer as a Tcl list or a line as a Tcl string and use standard Tcl commands that return an index ("lsort" or "string first", for example). The default value is '1'. Currently, any non-zero value is treated as '1', but your scripts should not rely on this. See also [tcl-linenumbers](#).

`::vim::range` [tcl-var-range](#)

This is an array with three elements, "start", "begin" and "end". It contains the line numbers of the start and end row of the current range. "begin" is the same as "start". This variable is read-only. See [tcl-examples](#).

line [tcl-var-line](#)

lnum [tcl-var-lnum](#)

These global variables are only available if the ":tcldo" Ex command is being executed. They contain the text and line number of the current line. When the Tcl command invoked by ":tcldo" is completed, the current line is set to the contents of the "line" variable, unless the variable was unset by the Tcl command. The "lnum" variable is read-only. These variables are not in the "::vim" namespace so they can be used in ":tcldo" without much typing (this might be changed in future versions). See also [tcl-linenumbers](#) .

4. Tcl window commands

[tcl-window-cmds](#)

Window commands represent vim windows. They are created by several commands:

```

::vim::window list          tcl-window
"windows" option of a buffer command tcl-buffer-windows

```

The `::vim::current(window)` variable contains the name of the window command for the current window. A window command is automatically deleted when the corresponding vim window is closed.

Let's assume the name of the window command is stored in the Tcl variable "win", i.e. "\$win" calls the command. The following options are available:

```

$win buffer                # Create Tcl command for window's buffer.
$win command {cmd}        # Execute Ex command in windows context.
$win cursor                # Get current cursor position.
$win cursor {var}         # Set cursor position from array variable.
$win cursor {row} {col}   # Set cursor position.
$win delcmd {cmd}         # Call Tcl command when window is closed.
$win expr {expr}          # Evaluate vim expression in windows context.
$win height                # Report the window's height.
$win height {n}           # Set the window's height.
$win option {opt} [val]   # Get/Set vim option in windows context.

```

Options:

```

$win buffer                tcl-window-buffer
Creates a Tcl command for the window's buffer, and returns its name as
the result. The name should be stored in a variable:
    set buf [$win buffer]
$buf is now a valid Tcl command. See tcl-buffer-cmds for the
available options.

```

```

$win cursor                tcl-window-cursor
$win cursor {var}
$win cursor {row} {col}
Without argument, reports the current cursor position as a string.
This can be converted to a Tcl array variable:
    array set here [$win cursor]
"here(row)" and "here(column)" now contain the cursor position.
With a single argument, the argument is interpreted as the name of a
Tcl array variable, which must contain two elements "row" and "column".
These are used to set the cursor to the new position:
    $win cursor here        ;# not $here !
With two arguments, sets the cursor to the specified row and column:
    $win cursor $here(row) $here(column)

```

Invalid positions result in a standard Tcl error, which can be caught with "catch". The row and column values depend on the "::vim::lbase" variable. See [tcl-var-lbase](#) .

`$win delcmd {cmd}` [tcl-window-delcmd](#)
Registers the Tcl command {cmd} as a deletion callback for the window. This command is executed (in the global scope) just before the window is closed. Complex commands should be built with "list":
`$win delcmd [list puts vimerr "window deleted"]`

See also [tcl-buffer-delcmd](#) .

`$win height` [tcl-window-height](#)
`$win height {n}`
Without argument, reports the window's current height. With an argument, tries to set the window's height to {n}, then reports the new height (which might be different from {n}).

`$win command [-quiet] {cmd}` [tcl-window-command](#)
`$win expr {expr}` [tcl-window-expr](#)
`$win option {opt} [val]` [tcl-window-option](#)
These are similar to "::vim::command" etc., except that everything is done in the context of the window represented by \$win, instead of the current window. For example, setting an option that is marked 'local to window' affects the window \$win. Anything that affects or queries a buffer uses the buffer displayed in this window (i.e. the buffer that is represented by "\$win buffer"). See [tcl-command](#) , [tcl-expr](#) and [tcl-option](#) for more information.
Example:

`$win option number on`

5. Tcl buffer commands

[tcl-buffer-cmds](#)

Buffer commands represent vim buffers. They are created by several commands:

<code>::vim::buffer {N}</code>	tcl-buffer
<code>::vim::buffer list</code>	tcl-buffer
"buffer" option of a window command	tcl-window-buffer

The `::vim::current(buffer)` variable contains the name of the buffer command for the current buffer. A buffer command is automatically deleted when the corresponding vim buffer is destroyed. Whenever the buffer's contents are changed, all marks in the buffer are automatically adjusted. Any changes to the buffer's contents made by Tcl commands can be undone with the "undo" vim command (see [undo](#)).

Let's assume the name of the buffer command is stored in the Tcl variable "buf", i.e. "\$buf" calls the command. The following options are available:

<code>\$buf append {n} {str}</code>	# Append a line to buffer, after line {n}.
<code>\$buf command {cmd}</code>	# Execute Ex command in buffers context.
<code>\$buf count</code>	# Report number of lines in buffer.
<code>\$buf delcmd {cmd}</code>	# Call Tcl command when buffer is deleted.
<code>\$buf delete {n}</code>	# Delete a single line.
<code>\$buf delete {n} {m}</code>	# Delete several lines.
<code>\$buf expr {expr}</code>	# Evaluate vim expression in buffers context.

```

$buf get {n}           # Get a single line as a string.
$buf get {n} {m}      # Get several lines as a list.
$buf insert {n} {str} # Insert a line in buffer, as line {n}.
$buf last             # Report line number of last line in buffer.
$buf mark {mark}     # Report position of buffer mark.
$buf name             # Report name of file in buffer.
$buf number          # Report number of this buffer.
$buf option {opt} [val] # Get/Set vim option in buffers context.
$buf set {n} {text}  # Replace a single line.
$buf set {n} {m} {list} # Replace several lines.
$buf windows         # Create Tcl commands for buffer's windows.

```

tcl-linenumbers

Most buffer commands take line numbers as arguments. How Tcl treats these numbers depends on the "::::lbase" variable (see [tcl-var-lbase](#)). Instead of line numbers, several keywords can be also used: "top", "start", "begin", "first", "bottom", "end" and "last".

Options:

```

$buf append {n} {str}           tcl-buffer-append
$buf insert {n} {str}          tcl-buffer-insert
Add a line to the buffer. With the "insert" option, the string
becomes the new line {n}, with "append" it is inserted after line {n}.
Example:

```

```

$buf insert top "This is the beginning."
$buf append end "This is the end."

```

To add a list of lines to the buffer, use a loop:

```

foreach line $list { $buf append $num $line ; incr num }

```

```

$buf count           tcl-buffer-count
Reports the total number of lines in the buffer.

```

```

$buf delcmd {cmd}           tcl-buffer-delcmd
Registers the Tcl command {cmd} as a deletion callback for the buffer.
This command is executed (in the global scope) just before the buffer
is deleted. Complex commands should be built with "list":
$buf delcmd [list puts vimerr "buffer [$buf number] gone"]

```

See also [tcl-window-delcmd](#) .

```

$buf delete {n}           tcl-buffer-delete
$buf delete {n} {m}
Deletes line {n} or lines {n} through {m} from the buffer.
This example deletes everything except the last line:
$buf delete first [expr [$buf last] - 1]

```

```

$buf get {n}           tcl-buffer-get
$buf get {n} {m}
Gets one or more lines from the buffer. For a single line, the result
is a string; for several lines, a list of strings.
Example:

```

```

set topline [$buf get top]

```

```

$buf last           tcl-buffer-last
Reports the line number of the last line. This value depends on the

```

":vim::lbase" variable. See [tcl-var-lbase](#) .

`$buf mark {mark}` [tcl-buffer-mark](#)
Reports the position of the named mark as a string, similar to the cursor position of the "cursor" option of a window command (see [tcl-window-cursor](#)). This can be converted to a Tcl array variable:

```
array set mpos [$buf mark "a"]
```

"mpos(column)" and "mpos(row)" now contain the position of the mark. If the mark is not set, a standard Tcl error results.

`$buf name`
Reports the name of the file in the buffer. For a buffer without a file, this is an empty string.

`$buf number`
Reports the number of this buffer. See [:buffers](#) .
This example deletes a buffer from vim:

```
::vim::command "bdelete [$buf number]"
```

`$buf set {n} {string}` [tcl-buffer-set](#)
`$buf set {n} {m} {list}`
Replace one or several lines in the buffer. If the list contains more elements than there are lines to replace, they are inserted into the buffer. If the list contains fewer elements, any unreplaced line is deleted from the buffer.

`$buf windows` [tcl-buffer-windows](#)
Creates a window command for each window that displays this buffer, and returns a list of the command names as the result.

Example:

```
set winlist [$buf windows]
foreach win $winlist { $win height 4 }
```

See [tcl-window-cmds](#) for the available options.

`$buf command [-quiet] {cmd}` [tcl-buffer-command](#)
`$buf expr {expr}` [tcl-buffer-expr](#)
`$buf option {opt} [val]` [tcl-buffer-option](#)
These are similar to "":vim::command" etc., except that everything is done in the context of the buffer represented by \$buf, instead of the current buffer. For example, setting an option that is marked 'local to buffer' affects the buffer \$buf. Anything that affects or queries a window uses the first window in vim's window list that displays this buffer (i.e. the first entry in the list returned by "\$buf windows"). See [tcl-command](#) , [tcl-expr](#) and [tcl-option](#) for more information.

Example:

```
if { [$buf option modified] } { $buf command "w" }
```

6. Miscellaneous; Output from Tcl

[tcl-misc](#) [tcl-output](#)

The standard Tcl commands "exit" and "catch" are replaced by custom versions. "exit" terminates the current Tcl script and returns to vim, which deletes the Tcl interpreter. Another call to ":tcl" then creates a new Tcl interpreter. "exit" does NOT terminate vim! "catch" works as before, except that it does

not prevent script termination from "exit". An exit code != 0 causes the ex command that invoked the Tcl script to return an error.

Two new I/O streams are available in Tcl, "vimout" and "vimerr". All output directed to them is displayed in the vim message area, as information messages and error messages, respectively. The standard Tcl output streams stdout and stderr are mapped to vimout and vimerr, so that a normal "puts" command can be used to display messages in vim.

7. Known bugs & problems

tcl-bugs

Calling one of the Tcl Ex commands from inside Tcl (via "::

Input from stdin is currently not supported.

8. Examples:

tcl-examples

Here are a few small (and maybe useful) Tcl scripts.

This script sorts the lines of the entire buffer (assume it contains a list of names or something similar):

```
set buf $::vim::current(buffer)
set lines [$buf get top bottom]
set lines [lsort -dictionary $lines]
$buf set top bottom $lines
```

This script reverses the lines in the buffer. [Note](#) the use of "::

```
set buf $::vim::current(buffer)
set t $::vim::lbase
set b [$buf last]
while { $t < $b } {
    set tl [$buf get $t]
    set bl [$buf get $b]
    $buf set $t $bl
    $buf set $b $tl
    incr t
    incr b -1
}
```

This script adds a consecutive number to each line in the current range:

```
set buf $::vim::current(buffer)
set i $::vim::range(start)
```

```

set n 1
while { $i <= $::vim::range(end) } {
    set line [$buf get $i]
    $buf set $i "$n\t$line"
    incr i ; incr n
}

```

The same can also be done quickly with two Ex commands, using ":tcldo":

```

:tcl set n 1
:[range]tcldo set line "$n\t$line" ; incr n

```

This procedure runs an Ex command on each buffer (idea stolen from Ron Aaron):

```

proc eachbuf { cmd } {
    foreach b [::vim::buffer list] {
        $b command $cmd
    }
}

```

Use it like this:

```

:tcl eachbuf %s/foo/bar/g

```

Be careful with Tcl's string and backslash substitution, tough. If in doubt, surround the Ex command with curly braces.

If you want to add some Tcl procedures permanently to vim, just place them in a file (e.g. "~/.vimrc.tcl" on Unix machines), and add these lines to your startup file (usually "~/.vimrc" on Unix):

```

if has("tcl")
    tclfile ~/.vimrc.tcl
endif

```

9. Dynamic loading

tcl-dynamic

On MS-Windows and Unix the Tcl library can be loaded dynamically. The `:version` output then includes `+tcl/dyn`.

This means that Vim will search for the Tcl DLL or shared library file only when needed. When you don't use the Tcl interface you don't need it, thus you can use Vim without this file.

MS-Windows

To use the Tcl interface the Tcl DLL must be in your search path. In a console window type "path" to see what directories are used. The `'tcl.dll'` option can be also used to specify the Tcl DLL.

The name of the DLL must match the Tcl version Vim was compiled with. Currently the name is "tcl86.dll". That is for Tcl 8.6. To know for sure edit "gvim.exe" and search for "tcl\d*.dll\c".

Unix

The **'tcl.dll'** option can be used to specify the Tcl shared library file instead of DYNAMIC_TCL_DLL file what was specified at compile time. The version of the shared library must match the Tcl version Vim was compiled with.

The OLE Interface to Vim

ole-interface

- 1. Activation [ole-activation](#)
- 2. Methods [ole-methods](#)
- 3. The "normal" command [ole-normal](#)
- 4. Registration [ole-registration](#)
- 5. MS Visual Studio integration [MSVisualStudio](#)

{only available when compiled with the `+ole` feature. See `src/if_ole.INSTALL`}

An alternative is using the client-server communication [clientserver](#) .

=====
1. Activation

ole-activation

Vim acts as an OLE automation server, accessible from any automation client, for example, Visual Basic, Python, or Perl. The Vim application "name" (its "ProgID", in OLE terminology) is "Vim.Application".

Hence, in order to start a Vim instance (or connect to an already running instance), code similar to the following should be used:

[Visual Basic]

```
Dim Vim As Object
Set Vim = CreateObject("Vim.Application")
```

[Python]

```
from win32com.client.dynamic import Dispatch
vim = Dispatch('Vim.Application')
```

[Perl]

```
use Win32::OLE;
$vim = new Win32::OLE 'Vim.Application';
```

[C#]

```
// Add a reference to Vim in your project.
// Choose the COM tab.
// Select "Vim Ole Interface 1.1 Type Library"
Vim.Vim vimobj = new Vim.Vim();
```

Vim does not support acting as a "hidden" OLE server, like some other OLE Automation servers. When a client starts up an instance of Vim, that instance is immediately visible. Simply closing the OLE connection to the Vim instance is not enough to shut down the Vim instance - it is necessary to explicitly execute a quit command (for example, `:qa!`, `:wqa`).

=====
2. Methods

ole-methods

Vim exposes four methods for use by clients.

ole-sendkeys

SendKeys(keys) Execute a series of keys.

This method takes a single parameter, which is a string of keystrokes. These keystrokes are executed exactly as if they had been typed in at the keyboard. Special keys can be given using their `<..>` names, as for the right hand side of a mapping. **Note:** Execution of the Ex "normal" command is not supported - see below `ole-normal` .

Examples (Visual Basic syntax)

```
Vim.SendKeys "ihello<Esc>"
Vim.SendKeys "ma1GV4jy`a"
```

These examples assume that Vim starts in Normal mode. To force Normal mode, start the key sequence with `CTRL-\ CTRL-N` as in

```
Vim.SendKeys "<C-\><C-N>ihello<Esc>"
```

`CTRL-\ CTRL-N` returns Vim to Normal mode, when in Insert or Command-line mode. **Note** that this doesn't work halfway a Vim command

ole-eval

Eval(expr) Evaluate an expression.

This method takes a single parameter, which is an expression in Vim's normal format (see `expression`). It returns a string, which is the result of evaluating the expression. A `List` is turned into a string by joining the items and inserting line breaks.

Examples (Visual Basic syntax)

```
Line20 = Vim.Eval("getline(20)")
Twelve = Vim.Eval("6 + 6")                    ' Note this is a STRING
Font = Vim.Eval("&guifont")
```

ole-setforeground

SetForeground() Make the Vim window come to the foreground

This method takes no arguments. No value is returned.

Example (Visual Basic syntax)

```
Vim.SetForeground
```

ole-gethwnd

GetHwnd() Return the handle of the Vim window.

This method takes no arguments. It returns the hwnd of the main Vimwindow. You can use this if you are writing something which needs to manipulate the Vim window, or to track it in the z-order, etc.

Example (Visual Basic syntax)

```
Vim_Hwnd = Vim.GetHwnd
```

3. The "normal" command

ole-normal

Due to the way Vim processes OLE Automation commands, combined with the method of implementation of the Ex command `:normal`, it is not possible to execute the `:normal` command via OLE automation. Any attempt to do so will fail, probably harmlessly, although possibly in unpredictable ways.

There is currently no practical way to trap this situation, and users must simply be aware of the limitation.

4. Registration

ole-registration E243

Before Vim will act as an OLE server, it must be registered in the system registry. In order to do this, Vim should be run with a single parameter of `"-register"`.

```
gvim -register
```

-register

If `gvim` with OLE support is run and notices that no Vim OLE server has been registered, it will present a dialog and offers you the choice to register by clicking "Yes".

In some situations registering is not possible. This happens when the registry is not writable. If you run into this problem you need to run `gvim` as "Administrator".

Once `vim` is registered, the application path is stored in the registry. Before moving, deleting, or upgrading Vim, the registry entries should be removed using the `"-unregister"` switch.

```
gvim -unregister
```

-unregister

The OLE mechanism will use the first registered Vim it finds. If a Vim is already running, this one will be used. If you want to have (several) Vim sessions open that should not react to OLE commands, use the non-OLE version, and put it in a different directory. The OLE version should then be put in a directory that is not in your normal path, so that typing `"gvim"` will start the non-OLE version.

-silent

To avoid the message box that pops up to report the result, prepend `"-silent"`:

```
gvim -silent -register
gvim -silent -unregister
```

5. MS Visual Studio integration

MSVisualStudio

The old "VisVim" integration was removed from Vim in patch 9.0.0698.

Using Vim with Visual Studio .Net

.Net studio has support for external editors. Follow these directions:

In .Net Studio choose from the menu Tools->External Tools...

Add

```
Title      - Vim
Command    - c:\vim\vim63\gvim.exe
Arguments  - --servername VS_NET --remote-silent "+call cursor($(CurLine), $(CurCol))"
Init Dir   - Empty
```

Now, when you open a file in .Net, you can choose from the .Net menu:

Tools->Vim

That will open the file in Vim.

You can then add this external command as an icon and place it anywhere you like. You might also be able to set this as your default editor.

If you refine this further, please post back to the Vim maillist so we have a record of it.

```
--servername VS_NET
```

This will create a new instance of vim called VS_NET. So if you open multiple files from VS, they will use the same instance of Vim. This allows you to have multiple copies of Vim running, but you can control which one has VS files in it.

```
--remote-silent "+call cursor(10, 27)"
                - Places the cursor on line 10 column 27
```

In Vim

```
:h --remote-silent for more details
```

[.Net remarks provided by Dave Fishburn and Brian Sturk]

=====

The Ruby Interface to Vim

ruby Ruby

- 1. Commands ruby-commands
- 2. The Vim module ruby-vim
- 3. Vim::Buffer objects ruby-buffer
- 4. Vim::Window objects ruby-window
- 5. Global variables ruby-globals
- 6. rubyeval() Vim function ruby-rubyeval
- 7. Dynamic loading ruby-dynamic

E266 E267 E268 E269 E270 E271 E272 E273

{only available when Vim was compiled with the |+ruby| feature}

The home page for ruby is <http://www.ruby-lang.org/>. You can find links for downloading Ruby there.



1. Commands ruby-commands

:ruby :rub

:rub[y] {cmd} Execute Ruby command {cmd}. A command to try it out:
:ruby print "Hello"

:rub[y] << [trim] [{endmarker}]
{script}
{endmarker}

Execute Ruby script {script}.

If [endmarker] is omitted, it defaults to a dot '.' like for the :append and :insert commands. Refer to :let-heredoc for more information.

This form of the :ruby command is mainly useful for including ruby code in vim scripts.

Note: This command doesn't work when the Ruby feature wasn't compiled in. To avoid errors, see [script-here](#) .

Example Vim script:

```
function! RedGem()
ruby << EOF
class Garnet
  def initialize(s)
    @buffer = Vim::Buffer.current
```

```

        vimputs(s)
    end
    def vimputs(s)
        @buffer.append(@buffer.count,s)
    end
end
gem = Garnet.new("pretty")
EOF
endfunction

```

To see what version of Ruby you have:

```
:ruby print RUBY_VERSION
```

```

:[:range]rubydo[o] {cmd} Evaluate Ruby command {cmd} for each line in the
                        :rubydo   :rubyd   E265
                        [range], with $_ being set to the text of each line in
                        turn, without a trailing <EOL>. Setting $_ will change
                        the text, but note that it is not possible to add or
                        delete lines using this command.
                        The default for [range] is the whole file: "1,$".

```

```

:rubyf[file] {file} Execute the Ruby script in {file}. This is the same as
                        :rubyfile  :rubyf
                        `:ruby load 'file'`, but allows file name completion.

```

Executing Ruby commands is not possible in the `sandbox` .

```

=====
2. The Vim module :rubyvim

```

Ruby code gets all of its access to vim via the "Vim" module.

Overview:

```

print "Hello"           # displays a message
Vim.command(cmd)       # execute an Ex command
num = Vim::Window.count # gets the number of windows
w = Vim::Window[n]     # gets window "n"
cw = Vim::Window.current # gets the current window
num = Vim::Buffer.count # gets the number of buffers
b = Vim::Buffer[n]     # gets buffer "n"
cb = Vim::Buffer.current # gets the current buffer
w.height = lines       # sets the window height
w.cursor = [row, col]  # sets the window cursor position
pos = w.cursor         # gets an array [row, col]
name = b.name          # gets the buffer file name
line = b[n]            # gets a line from the buffer
num = b.count          # gets the number of lines
b[n] = str             # sets a line in the buffer
b.delete(n)           # deletes a line
b.append(n, str)       # appends a line after n
line = Vim::Buffer.current.line # gets the current line
num = Vim::Buffer.current.line_number # gets the current line number
Vim::Buffer.current.line = "test" # sets the current line number

```

Module Functions:

<code>Vim::message({msg})</code>	Displays the message <code>{msg}</code> .	ruby-message
<code>Vim::blob({arg})</code>	Return <code>Blob</code> literal string from <code>{arg}</code> .	ruby-blob
<code>Vim::set_option({arg})</code>	Sets a vim option. <code>{arg}</code> can be any argument that the <code>":set"</code> command accepts. Note that this means that no spaces are allowed in the argument! See <code>:set</code> .	ruby-set_option
<code>Vim::command({cmd})</code>	Executes Ex command <code>{cmd}</code> .	ruby-command
<code>Vim::evaluate({expr})</code>	Evaluates <code>{expr}</code> using the vim internal expression evaluator (see expression). Returns the expression result as: <ul style="list-style-type: none">- a Integer if the Vim expression evaluates to a number- a Float if the Vim expression evaluates to a float- a String if the Vim expression evaluates to a string- a Array if the Vim expression evaluates to a Vim list- a Hash if the Vim expression evaluates to a Vim dictionary Dictionaries and lists are recursively expanded.	ruby-evaluate

3. Vim::Buffer objects

[ruby-buffer](#)

`Vim::Buffer` objects represent vim buffers.

Class Methods:

<code>current</code>	Returns the current buffer object.
<code>count</code>	Returns the number of buffers.
<code>self[{n}]</code>	Returns the buffer object for the number <code>{n}</code> . The first number is 0.

Methods:

<code>name</code>	Returns the full name of the buffer.
<code>number</code>	Returns the number of the buffer.
<code>count</code>	Returns the number of lines.
<code>length</code>	Returns the number of lines.
<code>self[{n}]</code>	Returns a line from the buffer. <code>{n}</code> is the line number.
<code>self[{n}] = {str}</code>	Sets a line in the buffer. <code>{n}</code> is the line number.

<code>delete({n})</code>	Deletes a line from the buffer. <code>{n}</code> is the line number.
<code>append({n}, {str})</code>	Appends a line after the line <code>{n}</code> .
<code>line</code>	Returns the current line of the buffer if the buffer is active.
<code>line = {str}</code>	Sets the current line of the buffer if the buffer is active.
<code>line_number</code>	Returns the number of the current line if the buffer is active.

4. Vim::Window objects

[ruby-window](#)

Vim::Window objects represent vim windows.

Class Methods:

<code>current</code>	Returns the current window object.
<code>count</code>	Returns the number of windows.
<code>self[{n}]</code>	Returns the window object for the number <code>{n}</code> . The first number is 0.

Methods:

<code>buffer</code>	Returns the buffer displayed in the window.
<code>height</code>	Returns the height of the window.
<code>height = {n}</code>	Sets the window height to <code>{n}</code> .
<code>width</code>	Returns the width of the window.
<code>width = {n}</code>	Sets the window width to <code>{n}</code> .
<code>cursor</code>	Returns a [row, col] array for the cursor position. First line number is 1 and first column number is 0.
<code>cursor = [{row}, {col}]</code>	Sets the cursor position to <code>{row}</code> and <code>{col}</code> .

5. Global variables

[ruby-globals](#)

There are two global variables.

<code>\$curwin</code>	The current window object.
<code>\$curbuf</code>	The current buffer object.

6. rubyeval() Vim function

[ruby-rubyeval](#)

To facilitate bi-directional interface, you can use `rubyeval()` function to evaluate Ruby expressions and pass their values to Vim script.

The Ruby value "true", "false" and "nil" are converted to `v:true`, `v:false` and `v:null`, respectively.

7. Dynamic loading

[ruby-dynamic](#)

On MS-Windows and Unix the Ruby library can be loaded dynamically. The

`:version` output then includes `+ruby/dyn` .

This means that Vim will search for the Ruby DLL file or shared library only when needed. When you don't use the Ruby interface you don't need it, thus you can use Vim even though this library file is not on your system.

MS-Windows

You need to install the right version of Ruby for this to work. You can find the package to download from:

<http://rubyinstaller.org/downloads/>

Currently that is `rubyinstaller-2.2.5.exe`

To use the Ruby interface the Ruby DLL must be in your search path. In a console window type "path" to see what directories are used. The `'rubydll'` option can be also used to specify the Ruby DLL.

The name of the DLL must match the Ruby version Vim was compiled with. Currently the name is `"msvcrt-ruby220.dll"`. That is for Ruby 2.2.X. To know for sure edit `"gvim.exe"` and search for `"ruby\d*.dll\c"`.

If you want to build Vim with RubyInstaller 1.9 or 2.X using MSVC, you need some tricks. See the `src/INSTALLpc.txt` for detail.

If Vim is built with RubyInstaller 2.4 or later, you may also need to add `"C:\Ruby<version>\bin\ruby_builtin_dlls"` to the PATH environment variable.

Unix

The `'rubydll'` option can be used to specify the Ruby shared library file instead of `DYNAMIC_RUBY_DLL` file what was specified at compile time. The version of the shared library must match the Ruby version Vim was compiled with.

=====

Debugger Support Features

[debugger-support](#)

These features are for integration with a debugger or an Integrated Programming Environment (IPE) or Integrated Development Environment (IDE). For the debugger running in a Vim terminal window see [terminal-debugger](#) .

1. Debugger Features [debugger-features](#)
2. Vim Compile Options [debugger-compilation](#)

1. Debugger Features

[debugger-features](#)

The following features are available:

Alternate Command Input	alt-input
Debug Signs	debug-signs
Debug Source Highlight	debug-highlight
Message Footer	gui-footer
Balloon Evaluation	balloon-eval

These features were added specifically for use in the Motif version of gvim. However, the [alt-input](#) and [debug-highlight](#) were written to be usable in both vim and gvim. Some of the other features could be used in the non-GUI vim with slight modifications. However, I did not do this nor did I test the reliability of building for vim or non Motif GUI versions.

1.1 Alternate Command Input

[alt-input](#)

For Vim to work with a debugger there must be at least an input connection with a debugger or external tool. In many cases there will also be an output connection but this isn't absolutely necessary.

The purpose of the input connection is to let the external debugger send commands to Vim. The commands sent by the debugger should give the debugger enough control to display the current debug environment and state.

The current implementation is based on the X Toolkit dispatch loop and the `XtAddInput()` function call.

1.2 Debug Signs

[debug-signs](#)

Many debuggers mark specific lines by placing a small sign or color highlight on the line. The `:sign` command lets the debugger set this graphic mark. Some examples where this feature would be used would be a debugger showing an arrow representing the Program Counter (PC) of the program being debugged. Another example would be a small stop sign for a line with a breakpoint. These visible

highlights let the user keep track of certain parts of the state of the debugger.

This feature can be used with more than debuggers, too. An IPE can use a sign to highlight build errors, searched text, or other things. The sign feature can also work together with the `debug-highlight` to ensure the mark is highly visible.

Debug signs are defined and placed using the `:sign` command.

1.3 Debug Source Highlight

`debug-highlight`

This feature allows a line to have a predominant highlight. The highlight is intended to make a specific line stand out. The highlight could be made to work for both vim and gvim, whereas the debug sign is, in most cases, limited to gvim. The one exception to this is Sun Microsystem's dtterm. The dtterm from Sun has a "sign gutter" for showing signs.

1.4 Message Footer

`gui-footer`

The message footer can be used to display messages from a debugger or IPE. It can also be used to display menu and toolbar tips. The footer area is at the bottom of the GUI window, below the line used to display colon commands.

The display of the footer is controlled by the `'guioptions'` letter 'F'.

1.5 Balloon Evaluation

`balloon-eval`

This feature allows a debugger, or other external tool, to display dynamic information based on where the mouse is pointing. The purpose of this feature was to allow Sun's Visual WorkShop debugger to display expression evaluations. However, the feature was implemented in as general a manner as possible and could be used for displaying other information as well. The functionality is limited though, for advanced popups see `popup-window`.

Another way to use the balloon is with the `'balloonexpr'` option. This is completely user definable.

The Balloon Evaluation has some settable parameters too. For Motif the font list and colors can be set via X resources (`XmNballoonEvalFontList`, `XmNballoonEvalBackground`, and `XmNballoonEvalForeground`).

The `'balloondelay'` option sets the delay before an attempt is made to show a balloon.

The `'ballooneval'` and/or the `'balloonevalterm'` option needs to be set to switch it on.

Balloon evaluation is only available in the GUI when compiled with the `+balloon_eval` feature. For the terminal the `+balloon_eval_term` feature matters.

The Balloon evaluation functions are also used to show a tooltip for the

toolbar. The **'ballooneval'** option does not need to be set for this. But the other settings apply.

2. Vim Compile Options

[debugger-compilation](#)

The debugger features were added for use with Sun's Visual WorkShop Integrated Programming Environment (ipe). However, they were done in as generic a manner as possible so that integration with other debuggers could also use these features.

The following compile time preprocessor variables control the features:

Alternate Command Input	ALT_X_INPUT
Debug Glyphs	FEAT_SIGNS
Debug Highlights	FEAT_SIGNS
Message Footer	FEAT_FOOTER
Balloon Evaluation	FEAT_BEVAL

The support specifically for Sun Visual WorkShop has been removed, since the product no longer exists.

For Sun NetBeans support see [netbeans](#) .

Vim NetBeans Protocol: a socket interface for Vim integration into an IDE.

1. Introduction	netbeans-intro
2. Integration features	netbeans-integration
3. Configuring Vim for NetBeans	netbeans-configure
4. Error Messages	netbeans-messages
5. Running Vim in NetBeans mode	netbeans-run
6. NetBeans protocol	netbeans-protocol
7. NetBeans commands	netbeans-commands
8. Known problems	netbeans-problems
9. Debugging NetBeans protocol	netbeans-debugging
10. NetBeans External Editor	
10.1. Downloading NetBeans	netbeans-download
10.2. NetBeans Key Bindings	netbeans-keybindings
10.3. Preparing NetBeans for Vim	netbeans-preparation
10.4. Obtaining the External Editor Module	obtaining-extended
10.5. Setting up NetBeans to run with Vim	netbeans-setup

{only available when compiled with the |+netbeans_intg| feature}



1. Introduction [netbeans-intro](#)

The NetBeans interface was initially developed to integrate Vim into the NetBeans Java IDE, using the external editor plugin. This NetBeans plugin no longer exists for recent versions of NetBeans but the protocol was developed in such a way that any IDE can use it to integrate Vim.

The NetBeans protocol of Vim is a text based communication protocol, over a classical TCP socket. There is no dependency on Java or NetBeans. Any language or environment providing a socket interface can control Vim using this protocol. There are existing implementations in C, C++, Python and Java. The name NetBeans is kept today for historical reasons.

Active project using the NetBeans protocol of Vim:

- Eclim, <http://eclim.org/>

VimIntegration, description of various projects doing Vim Integration:

<http://www.freehackers.org/VimIntegration>

Projects using the NetBeans protocol of Vim are or were:

- Agide, an IDE for the AAP project, written in Python (now replaced by [:Termdebug](#)): <http://www.a-a-p.org>
- Clewn, a gdb integration into Vim, written in C: <http://clewn.sourceforge.net/>
- Pyclewn, a gdb integration into Vim, written in Python:

- <http://pyclewn.sourceforge.net/>
 - VimWrapper, library to easy Vim integration into IDE:
<http://www.freehackers.org/VimWrapper>
- Outdated projects (links don't work):
- VimPlugin, integration of Vim inside Eclipse:
<http://vimplugin.sourceforge.net/wiki/pmwiki.php>
 - PIDA, IDE written in Python integrating Vim:
<http://pida.co.uk/>

Check the specific project pages to see how to use Vim with these projects.

An alternative is to use a channel, see [channel](#) .

In the rest of this help page, we will use the term "Vim Controller" to describe the program controlling Vim through the NetBeans socket interface.

About the NetBeans IDE

NetBeans is an open source Integrated Development Environment developed jointly by Sun Microsystems, Inc. and the netbeans.org developer community. Initially just a Java IDE, NetBeans has had C, C++, and Fortran support added in recent releases.

For more information visit the main NetBeans web site <http://www.netbeans.org>. The External Editor is now, unfortunately, declared obsolete. See <http://externaleditor.netbeans.org>.

Sun Microsystems, Inc. also ships NetBeans under the name Sun ONE Studio. Visit <http://www.sun.com> for more information regarding the Sun ONE Studio product line.

Current releases of NetBeans provide full support for Java and limited support for C, C++, and Fortran. Current releases of Sun ONE Studio provide full support for Java, C, C++, and Fortran.

2. Integration features [netbeans-integration](#)

The NetBeans socket interface of Vim allows to get information from Vim or to ask Vim to perform specific actions:

- get information about buffer: buffer name, cursor position, buffer content, etc.
- be notified when buffers are open or closed
- be notified of how the buffer content is modified
- load and save files
- modify the buffer content
- installing special key bindings
- raise the window, control the window geometry

For sending key strokes to Vim or for evaluating functions in Vim, you must use the [clientserver](#) interface.

3. Configuring Vim for NetBeans

[netbeans-configure](#)

For more help about installing Vim, please read [usr_90.txt](#) in the Vim User Manual.

On Unix:

When running `configure` without arguments the NetBeans interface should be included. That is, if the `configure` check to find out if your system supports the required features succeeds.

In case you do not want the NetBeans interface you can disable it by uncommenting a line with `--disable-netbeans` in the Makefile.

Currently the NetBeans interface is supported by Vim running in a terminal and by `gvim` when it is run with one of the following GUIs: GTK, GNOME, Windows and Motif.

[netbeans-xpm](#)

If Motif support is required the user must supply XPM libraries. The XPM library is required to show images within Vim with Motif. Without it the toolbar and signs will be disabled.

The XPM library is provided by Arnaud Le Hors of the French National Institute for Research in Computer Science and Control. It can be downloaded from <http://cgit.freedesktop.org/xorg/lib/libXpm>. The current release, as of this writing, is `xpm-3.4k-solaris.tgz`, which is a gzip'ed tar file. If you create the directory `/usr/local/xpm` and `untar` the file there you can use the uncommented lines in the Makefile without changing them. If you use another xpm directory you will need to change the `XPM_DIR` in `src/Makefile`.

On MS-Windows:

The Win32 support is now in beta stage.

To use XPM signs on Win32 (e.g. when using with NetBeans) you can compile XPM by yourself or use precompiled libraries from <http://iamphet.nm.ru/misc/> (for MS Visual C++) or <http://gnuwin32.sourceforge.net> (for MinGW).

Enable debugging:

To enable debugging of Vim and of the NetBeans protocol, the `"NBDEBUG"` macro needs to be defined. Search in the Makefile of the platform you are using for `"NBDEBUG"` to see what line needs to be uncommented. This effectively adds `"-DNBDEBUG"` to the compile command. Also see [netbeans-debugging](#)

4. Error Messages

[netbeans-messages](#)

These error messages are specific to NetBeans socket protocol:

E463

Region is guarded, cannot modify

The Vim Controller has defined guarded areas in the text, which you cannot change. Also sets the current buffer, if necessary.

E532

The defineAnnoType highlighting color name is too long

The maximum length of the "fg" or "bg" color argument in the defineAnnoType command is 32 characters.
New in version 2.5.

E656

Writes of unmodified buffers forbidden

Writes of unmodified buffers that were opened from the Vim Controller are not possible.

E657

Partial writes disallowed

Partial writes for buffers that were opened from the Vim Controller are not allowed.

E658

Connection lost for this buffer

The Vim Controller has become confused about the state of this file. Rather than risk data corruption, it has severed the connection for this file. Vim will take over responsibility for saving changes to this file and the Vim Controller will no longer know of these changes.

E744

Read-only file

Vim normally allows changes to a read-only file and only enforces the read-only rule if you try to write the file. However, NetBeans does not let you make changes to a file which is read-only and becomes confused if Vim does this. So Vim does not allow modifications to files when run in NetBeans mode.

5. Running Vim in NetBeans mode

netbeans-run

There are two different ways to run Vim in NetBeans mode:

- + an IDE may start Vim with the `-nb` command line argument
- + NetBeans can be started from within Vim with the `:nbstart` command

Vim uses a 3 second timeout on trying to make the connection.

netbeans-parameters

Three forms can be used to setup the NetBeans connection parameters.

When started from the command line, the `-nb` command line argument may be:

<code>-nb={fname}</code>	from a file
<code>-nb:{hostname}:{addr}:{password}</code>	directly
<code>-nb</code>	from a file or environment

When started from within Vim, the `:nbstart` optional argument may be:

<code>={fname}</code>	from a file
<code>:{hostname}:{addr}:{password}</code>	directly
<code><MISSING ARGUMENT></code>	from a file or environment

E660 E668

When NetBeans is started from the command line, for security reasons, the best method is to write the information in a file readable only by the user. The name of the file can be passed with the `"-nb={fname}"` argument or, when `"-nb"` is used without a parameter, the environment variable `"__NETBEANS_CONINFO"`. The file must contain these three lines, in any order:

```
host={hostname}
port={addr}
auth={password}
```

Other lines are ignored. The Vim Controller is responsible for deleting the file afterwards.

`{hostname}` is the name of the machine where Vim Controller is running. When omitted the environment variable `"__NETBEANS_HOST"` is used or the default `"localhost"`.

`{addr}` is the port number for the NetBeans interface. When omitted the environment variable `"__NETBEANS_SOCKET"` is used or the default 3219.

`{password}` is the password for connecting to NetBeans. When omitted the environment variable `"__NETBEANS_VIM_PASSWORD"` is used or `"changeme"`.

Vim will initiate a socket connection (client side) to the specified host and port upon startup. The password will be sent with the AUTH event when the connection has been established.

6. NetBeans protocol

netbeans-protocol

The communication between the Vim Controller and Vim uses plain text messages. This protocol was first designed to work with the external editor module of NetBeans. Later it was extended to work with Agide (A-A-P GUI IDE, see <http://www.a-a-p.org>) and then with other IDE. The extensions are marked with "version 2.1".

Version 2.2 of the protocol has several minor changes which should only affect NetBeans users (ie, not Agide users). However, a bug was fixed which could cause confusion. The `netbeans_saved()` function sent a "save" protocol command. In protocol version 2.1 and earlier this was incorrectly interpreted

as a notification that a write had taken place. In reality, it told NetBeans to save the file so multiple writes were being done. This caused various problems and has been fixed in 2.2. To decrease the likelihood of this confusion happening again, `netbeans_saved()` has been renamed to `netbeans_save_buffer()`.

We are now at version 2.5. For the differences between 2.4 and 2.5 search for "2.5" below.

The messages are currently sent over a socket. Since the messages are in plain UTF-8 text this protocol could also be used with any other communication mechanism.

Netbeans messages are processed when Vim is idle, waiting for user input. When Vim is run in non-interactive mode, for example when running an automated test case that sources a Vim script, the idle loop may not be called often enough. In that case, insert `:sleep` commands in the Vim script. The `:sleep` command does invoke Netbeans messages processing.

6.1 Kinds of messages	nb-messages
6.2 Terms	nb-terms
6.3 Commands	nb-commands
6.4 Functions and Replies	nb-functions
6.5 Events	nb-events
6.6 Special messages	nb-special
6.7 Protocol errors	nb-protocol_errors

6.1 Kinds of messages [nb-messages](#)

There are four kinds of messages:

kind	direction	comment
Command	IDE -> editor	no reply necessary
Function	IDE -> editor	editor must send back a reply
Reply	editor -> IDE	only in response to a Function
Event	editor -> IDE	no reply necessary

The messages are sent as a single line with a terminating newline character. Arguments are separated by a single space. The first item of the message depends on the kind of message:

kind	first item	example
Command	bufID:name!seqno	11:showBalloon!123 "text"
Function	bufID:name/seqno	11:getLength/123
Reply	seqno	123 5000
Event	bufID:name=seqno	11:keyCommand=123 "S-F2"

6.2 Terms [nb-terms](#)

bufID Buffer number. A message may be either for a specific buffer or generic. Generic messages use a bufID of zero. **NOTE:** this

buffer ID is assigned by the IDE, it is not Vim's buffer number. The bufID must be a sequentially rising number, starting at one. When the `'switchbuf'` option is set to "usetab" and the "bufID" buffer is not found in the current tab page, the netbeans commands and functions that set this buffer as the current buffer will jump to the first open window that contains this buffer in other tab pages instead of replacing the buffer in the current window.

seqno The IDE uses a sequence number for Commands and Functions. A Reply must use the sequence number of the Function that it is associated with. A zero sequence number can be used for Events (the seqno of the last received Command or Function can also be used).

string Argument in double quotes. Text is in UTF-8 encoding. This means ASCII is passed as-is. Special characters are represented with a backslash:

```
\"      double quote
\n      newline
\r      carriage-return
\t      tab (optional, also works literally)
\\      backslash
```

NUL bytes are not allowed!

boolean Argument with two possible values:

```
T      true
F      false
```

number Argument with a decimal number.

color Argument with either a decimal number, "none" (without the quotes) or the name of a color (without the quotes) defined both in the color list in `highlight-ctermfg` and in the color list in `gui-colors`.
New in version 2.5.

offset A number argument that indicates a byte position in a buffer. The first byte has offset zero. Line breaks are counted for how they appear in the file (CR/LF counts for two bytes).
Note that a multibyte character is counted for the number of bytes it takes.

lnum/col Argument with a line number and column number position. The line number starts with one, the column is the byte position, starting with zero. **Note** that a multibyte character counts for several columns.

pathname String argument: file name with full path.

6.3 Commands

`nb-commands`

actionMenuItem Not implemented.

actionSensitivity
Not implemented.

addAnno serNum typeNum off len
Place an annotation in this buffer.
Arguments:

serNum	number	serial number of this placed annotation, used to be able to remove it
typeNum	number	sequence number of the annotation defined with defineAnnoType for this buffer
off	number	offset where annotation is to be placed
len	number	not used

In version 2.1 "lnum/col" can be used instead of "off".

balloonResult text
Not implemented.

close
Close the buffer. This leaves us without current buffer, very dangerous to use!

create
Creates a buffer without a name. Replaces the current buffer (it's hidden when it was changed).
The Vim Controller should use this as the first command for a file that is being opened. The sequence of commands could be:

```

create
setCaretListener      (ignored)
setModified           (no effect)
setContentTypes      (ignored)
startDocumentListen
setTitle
setFullName

```

defineAnnoType typeNum typeName tooltip glyphFile fg bg
Define a type of annotation for this buffer.
Arguments:

typeNum	number	sequence number (not really used)
typeName	string	name that identifies this annotation
tooltip	string	not used
glyphFile	string	name of icon file
fg	color	foreground color for line highlighting
bg	color	background color for line highlighting

Vim will define a sign for the annotation.
When color is a number, this is the "#rrggbb" Red, Green and Blue values of the color (see [gui-colors](#)) and the highlighting is only defined for gVim.
When color is a name, this color is defined both for Vim running in a color terminal and for gVim.
When both "fg" and "bg" are "none" no line highlighting is used (new in version 2.1).
When "glyphFile" is empty, no text sign is used (new in version 2.1).

When "glyphFile" is one or two characters long, a text sign is defined (new in version 2.1).

Note: the annotations will be defined in sequence, and the sequence number is later used with addAnno.

editFile pathname

Set the name for the buffer and edit the file "pathname", a string argument.

Normal way for the IDE to tell the editor to edit a file.

You must set a bufId different of 0 with this command to assign a bufId to the buffer. It will trigger an event fileOpened with a bufId of 0 but the buffer has been assigned.

If the IDE is going to pass the file text to the editor use these commands instead:

setFullName

insert

initDone

New in version 2.1.

enableBalloonEval

Not implemented.

endAtomic

End an atomic operation. The changes between "startAtomic" and "endAtomic" can be undone as one operation. But it's not implemented yet. Redraw when necessary.

guard off len

Mark an area in the buffer as guarded. This means it cannot be edited. "off" and "len" are numbers and specify the text to be guarded.

initDone

Mark the buffer as ready for use. Implicitly makes the buffer the current buffer. Fires the BufReadPost autocommand event.

insertDone starteol readonly

Sent by Vim Controller to tell Vim an initial file insert is done. This triggers a read message being printed. If "starteol" is "F" then the last line doesn't have a EOL. If "readonly" is "T" then the file is marked as readonly. Prior to version 2.3, no read messages were displayed after opening a file. New in version 2.3.

moveAnnoToFront serNum

Not implemented.

netbeansBuffer isNetbeansBuffer

If "isNetbeansBuffer" is "T" then this buffer is "owned" by NetBeans.

New in version 2.2.

putBufferNumber pathname

Associate a buffer number with the Vim buffer by the name

"pathname", a string argument. To be used when the editor reported editing another file to the IDE and the IDE needs to tell the editor what buffer number it will use for this file. Also marks the buffer as initialized. New in version 2.1.

raise Bring the editor to the foreground. Only when Vim is run with a GUI. New in version 2.1.

removeAnno serNum Remove a previously placed annotation for this buffer. "serNum" is the same number used in addAnno.

save Save the buffer when it was modified. The other side of the interface is expected to write the buffer and invoke "setModified" to reset the "changed" flag of the buffer. The writing is skipped when one of these conditions is true:

- 'write' is not set
- the buffer is read-only
- the buffer does not have a file name
- 'buftype' disallows writing

New in version 2.2.

saveDone Sent by Vim Controller to tell Vim a save is done. This triggers a save message being printed. Prior to version 2.3, no save messages were displayed after a save. New in version 2.3.

setAsUser Not implemented.

setBufferNumber pathname Associate a buffer number with Vim buffer by the name "pathname". To be used when the editor reported editing another file to the IDE and the IDE needs to tell the editor what buffer number it will use for this file. Has the side effect of making the buffer the current buffer. See "putBufferNumber" for a more useful command.

setContenttype Not implemented.

setDot off Make the buffer the current buffer and set the cursor at the specified position. If the buffer is open in another window than make that window the current window. If there are folds they are opened to make the cursor line visible. In version 2.1 "lnum/col" can be used instead of "off".

setExitDelay seconds Set the delay for exiting to "seconds", a number. This delay is used to give the IDE a chance to handle things before really exiting. The default delay is two seconds.

New in version 2.1.
Obsolete in version 2.3.

- `setFullName` `pathname`
Set the file name to be used for a buffer to "pathname", a string argument.
Used when the IDE wants to edit a file under control of the IDE. This makes the buffer the current buffer, but does not read the file. "insert" commands will be used next to set the contents.
- `setLocAndSize` Not implemented.
- `setMark` Not implemented.
- `setModified` `modified`
When the boolean argument "modified" is "T" mark the buffer as modified, when it is "F" mark it as unmodified.
- `setModtime` `time`
Update a buffers modification time after the file has been saved directly by the Vim Controller.
New in version 2.3.
- `setReadOnly` `readonly`
When the boolean argument "readonly" is "T" for True, mark the buffer as readonly, when it is "F" for False, mark it as not readonly. Implemented in version 2.3.
- `setStyle` Not implemented.
- `setTitle` `name`
Set the title for the buffer to "name", a string argument. The title is only used for the Vim Controller functions, not by Vim.
- `setVisible` `visible`
When the boolean argument "visible" is "T", goto the buffer. The "F" argument does nothing.
- `showBalloon` `text`
Show a balloon (popup window) at the mouse pointer position, containing "text", a string argument. The balloon should disappear when the mouse is moved more than a few pixels. Only when Vim is run with a GUI.
New in version 2.1.
- `specialKeys`
Map a set of keys (mostly function keys) to be passed back to the Vim Controller for processing. This lets regular IDE hotkeys be used from Vim.
Implemented in version 2.3.
- `startAtomic` Begin an atomic operation. The screen will not be updated

until "endAtomic" is given.

startCaretListen

Not implemented.

startDocumentListen

Mark the buffer to report changes to the IDE with the "insert" and "remove" events. The default is to report changes.

stopCaretListen

Not implemented.

stopDocumentListen

Mark the buffer to stop reporting changes to the IDE. Opposite of startDocumentListen.

NOTE: if "netbeansBuffer" was used to mark this buffer as a NetBeans buffer, then the buffer is deleted in Vim. This is for compatibility with Sun Studio 10.

unguard off len

Opposite of "guard", remove guarding for a text area. Also sets the current buffer, if necessary.

version

Not implemented.

6.4 Functions and Replies

nb-functions

getDot

Not implemented.

getCursor

Return the current buffer and cursor position.

The reply is:

```
seqno bufID lnum col off
```

seqno = sequence number of the function

bufID = buffer ID of the current buffer (if this is unknown -1 is used)

lnum = line number of the cursor (first line is one)

col = column number of the cursor (in bytes, zero based)

off = offset of the cursor in the buffer (in bytes)

New in version 2.1.

getLength

Return the length of the buffer in bytes.

Reply example for a buffer with 5000 bytes:

```
123 5000
```

TODO: explain use of partial line.

getMark

Not implemented.

getAnno serNum

Return the line number of the annotation in the buffer.

Argument:

```
serNum
```

serial number of this placed annotation

The reply is:

```
123 lnum          line number of the annotation
123 0             invalid annotation serial number
New in version 2.4.
```

getModified When a buffer is specified: Return zero if the buffer does not have changes, one if it does have changes.
When no buffer is specified (buffer number zero): Return the number of buffers with changes. When the result is zero it's safe to tell Vim to exit.
New in version 2.1.

getText Return the contents of the buffer as a string.
Reply example for a buffer with two lines
123 "first line\nsecond line\n"
NOTE: docs indicate an offset and length argument, but this is not implemented.

insert off text
Insert "text" before position "off". "text" is a string argument, "off" a number.
"text" should have a "\n" (newline) at the end of each line. Or "\r\n" when '**fileformat**' is "dos". When using "insert" in an empty buffer Vim will set '**fileformat**' accordingly.
When "off" points to the start of a line the text is inserted above this line. Thus when "off" is zero lines are inserted before the first line.
When "off" points after the start of a line, possibly on the NUL at the end of a line, the first line of text is appended to this line. Further lines come below it.
Possible replies:
123 no problem
123 !message failed
Note that the message in the reply is not quoted.
Also sets the current buffer, if necessary.
Does not move the cursor to the changed text.
Resets undo information.

remove off length
Delete "length" bytes of text at position "off". Both arguments are numbers.
Possible replies:
123 no problem
123 !message failed
Note that the message in the reply is not quoted.
Also sets the current buffer, if necessary.

saveAndExit Perform the equivalent of closing Vim: ":confirm qall".
If there are no changed files or the user does not cancel the operation Vim exits and no result is sent back. The IDE can consider closing the connection as a successful result.
If the user cancels the operation the number of modified buffers that remains is returned and Vim does not exit.
New in version 2.1.

6.5 Events

nb-events

balloonEval off len type

The mouse pointer rests on text for a short while. When "len" is zero, there is no selection and the pointer is at position "off". When "len" is non-zero the text from position "off" to "off" + "len" is selected.

Only sent after "enableBalloonEval" was used for this buffer. "type" is not yet defined.

Not implemented yet.

balloonText text

Used when '**ballooneval**' is set and the mouse pointer rests on some text for a moment. "text" is a string, the text under the mouse pointer.

Only when Vim is run with a GUI.

New in version 2.1.

buttonRelease button lnum col

Report which button was pressed and the location of the cursor at the time of the release. Only for buffers that are owned by the Vim Controller. This event is not sent if the button was released while the mouse was in the status line or in a separator line. If col is less than 1 the button release was in the sign area.

New in version 2.2.

disconnect

Tell the Vim Controller that Vim is exiting and not to try and read or write more commands.

New in version 2.3.

fileClosed Not implemented.

fileModified Not implemented.

fileOpened pathname open modified

A file was opened by the user.

Arguments:

pathname	string	name of the file
open	boolean	always "T"
modified	boolean	always "F"

geometry cols rows x y

Report the size and position of the editor window.

Arguments:

cols	number	number of text columns
rows	number	number of text rows
x	number	pixel position on screen
y	number	pixel position on screen

Only works for Motif.

insert off text

Text "text" has been inserted in Vim at position "off".
Only fired when enabled, see "startDocumentListen".

invokeAction Not implemented.

keyCommand keyName

Reports a special key being pressed with name "keyName", which is a string.

Supported key names:

F1	function key 1
F2	function key 2
...	
F12	function key 12
' '	space (without the quotes)
!	exclamation mark
...	any other ASCII printable character
~	tilde
X	any unrecognized key

The key may be prepended by "C", "S" and/or "M" for Control, Shift and Meta (Alt) modifiers. If there is a modifier a dash is used to separate it from the key name. For example: "C-F2".

ASCII characters are new in version 2.1.

keyAtPos keyName lnum/col

Like "keyCommand" and also report the line number and column of the cursor.

New in version 2.1.

killed

A file was deleted or wiped out by the user and the buffer annotations have been removed. The bufID number for this buffer has become invalid. Only for files that have been assigned a bufID number by the IDE.

newDotAndMark off off

Reports the position of the cursor being at "off" bytes into the buffer. Only sent just before a "keyCommand" event.

quit

Not implemented.

remove off len

Text was deleted in Vim at position "off" with byte length "len".

Only fired when enabled, see "startDocumentListen".

revert

Not implemented.

save

The buffer has been saved and is now unmodified.

Only fired when enabled, see "startDocumentListen".

startupDone

The editor has finished its startup work and is ready for

editing files.
New in version 2.1.

unmodified The buffer is now unmodified.
Only fired when enabled, see "startDocumentListen".

version vers Report the version of the interface implementation. Vim reports "2.4" (including the quotes).

6.6 Special messages nb-special

These messages do not follow the style of the messages above. They are terminated by a newline character.

ACCEPT Not used.

AUTH password editor -> IDE: First message that the editor sends to the IDE. Must contain the password for the socket server, as specified with the `-nb` argument. No quotes are used!

DISCONNECT IDE -> editor: break the connection. The editor will exit. The IDE must only send this message when there are no unsaved changes!

DETACH IDE -> editor: break the connection without exiting the editor. Used when the IDE exits without bringing down the editor as well.
New in version 2.1.

REJECT Not used.

6.7 Protocol errors nb-protocol_errors

These errors occur when a message violates the protocol:

E627 E628 E629 E632 E633 E634 E635 E636
E637 E638 E639 E640 E641 E642 E643 E644 E645 E646
E647 E648 E650 E651 E652

7. NetBeans commands netbeans-commands

`:nbs[tart] {connection}` Start a new Netbeans session with `{connection}` as the socket connection parameters. The format of `{connection}` is described in `netbeans-parameters`. At any time, one may check if the netbeans socket is connected by running the command:
`':echo has("netbeans_enabled")'`

`:nbc[lose]` Close the current NetBeans session. Remove all placed :nbclose

signs.

`:nb[key] {key}` [:nbkey](#)

Pass the `{key}` to the Vim Controller for processing. When a hot-key has been installed with the `specialKeys` command, this command can be used to generate a hotkey message to the Vim Controller. This command can also be used to pass any text to the Vim Controller. It is used by Pyclewn, for example, to build the complete set of gdb commands as Vim user commands. The events `newDotAndMark`, `keyCommand` and `keyAtPos` are generated (in this order).

8. Known problems

[netbeans-problems](#)

NUL bytes are not possible. For editor -> IDE they will appear as NL characters. For IDE -> editor they cannot be inserted.

A NetBeans session may be initiated with Vim running in a terminal, and continued later in a GUI environment after running the `:gui` command. In this case, the highlighting defined for the NetBeans annotations may be cleared when the `:gui` command sources `.gvimrc` and this file loads a colorscheme that runs the command `:highlight clear`.
New in version 2.5.

9. Debugging NetBeans protocol

[netbeans-debugging](#)

To debug the Vim protocol, you must first compile Vim with debugging support and NetBeans debugging support. See [netbeans-configure](#) for instructions about Vim compiling and how to enable debug support.

When running Vim, set the following environment variables:

```
export SPRO_GVIM_DEBUG=netbeans.log
export SPRO_GVIM_DLEVEL=0xffffffff
```

Vim will then log all the incoming and outgoing messages of the NetBeans protocol to the file `netbeans.log`.

The content of `netbeans.log` after a session looks like this:

```
Tue May 20 17:19:27 2008
EVT: 0:startupDone=0
CMD 1: (1) create
CMD 2: (1) setTitle "testfile1.txt"
CMD 3: (1) setFullName "testfile1.txt"
EVT(suppressed): 1:remove=3 0 -1
EVT: 1:fileOpened=0 "d:\\work\\vimWrapper\\vimWrapper2\\pyvimwrapper\\tests\\testfile1.txt"
CMD 4: (1) initDone
FUN 5: (0) getCursor
```

```
REP 5: 1 1 0 0
CMD 6: (2) create
CMD 7: (2) setTitle "testfile2.txt"
CMD 8: (2) setFullName "testfile2.txt"
EVT(suppressed): 2:remove=8 0 -1
EVT: 2:fileOpened=0 "d:\\work\\vimWrapper\\vimWrapper2\\pyvimwrapper\\tests\\testfile2.txt"
CMD 9: (2) initDone
```

10. NetBeans External Editor

NOTE: This information is obsolete! Only relevant if you are using an old version of NetBeans.

10.1. Downloading NetBeans

[netbeans-download](#)

The NetBeans IDE is available for download from netbeans.org. You can download a released version, download sources, or use CVS to download the current source tree. If you choose to download sources, follow directions from netbeans.org on building NetBeans.

Depending on the version of NetBeans you download, you may need to do further work to get the required External Editor module. This is the module which lets NetBeans work with `gvim` (or `xemacs` :-). See <http://externaleditor.netbeans.org> for details on downloading this module if your NetBeans release does not have it.

For C, C++, and Fortran support you will also need the `cpp` module. See <http://cpp.netbeans.org> for information regarding this module.

You can also download Sun ONE Studio from Sun Microsystems, Inc for a 30 day free trial. See <http://www.sun.com> for further details.

10.2. NetBeans Key Bindings

[netbeans-keybindings](#)

Vim understands a number of key bindings that execute NetBeans commands. These are typically all the Function key combinations. To execute a NetBeans command, the user must press the Pause key followed by a NetBeans key binding. For example, in order to compile a Java file, the NetBeans key binding is "F9". So, while in vim, press "Pause F9" to compile a java file. To toggle a breakpoint at the current line, press "Pause Shift F8".

The Pause key is Function key 21. If you don't have a working Pause key and want to use F8 instead, use:

```
:map <F8> <F21>
```

The External Editor module dynamically reads the NetBeans key bindings so vim should always have the latest key bindings, even when NetBeans changes them.

10.3. Preparing NetBeans for Vim

[netbeans-preparation](#)

In order for NetBeans to work with vim, the NetBeans External Editor module must be loaded and enabled. If you have a Sun ONE Studio Enterprise Edition then this module should be loaded and enabled. If you have a NetBeans release you may need to find another way of obtaining this open source module.

You can check if you have this module by opening the Tools->Options dialog and drilling down to the "Modules" list (IDE Configuration->System->Modules). If your Modules list has an entry for "External Editor" you must make sure it is enabled (the "Enabled" property should have the value "True"). If your Modules list has no External Editor see the next section on [obtaining-extended](#).

10.4. Obtaining the External Editor Module

[obtaining-extended](#)

There are 2 ways of obtaining the External Editor module. The easiest way is to use the NetBeans Update Center to download and install the module. Unfortunately, some versions do not have this module in their update center. If you cannot download via the update center you will need to download sources and build the module. I will try and get the module available from the NetBeans Update Center so building will be unnecessary. Also check <http://externaleditor.netbeans.org> for other availability options.

To download the External Editor sources via CVS and build your own module, see <http://externaleditor.netbeans.org> and <http://www.netbeans.org>. Unfortunately, this is not a trivial procedure.

10.5. Setting up NetBeans to run with Vim

[netbeans-setup](#)

Assuming you have loaded and enabled the NetBeans External Editor module as described in [netbeans-preparation](#) all you need to do is verify that the gvim command line is properly configured for your environment.

Open the Tools->Options dialog and open the Editing category. Select the External Editor. The right hand pane should contain a Properties tab and an Expert tab. In the Properties tab make sure the "Editor Type" is set to "Vim". In the Expert tab make sure the "Vim Command" is correct.

You should be careful if you change the "Vim Command". There are command line options there which must be there for the connection to be properly set up. You can change the command name but that's about it. If your gvim can be found by your \$PATH then the Vim Command can start with "gvim". If you don't want gvim searched from your \$PATH then hard code in the full Unix path name. At this point you should get a gvim for any source file you open in NetBeans.

If some files come up in gvim and others (with different file suffixes) come up in the default NetBeans editor you should verify the MIME type in the Expert tab MIME Type property. NetBeans is MIME oriented and the External Editor will only open MIME types specified in this property.

Sign Support Features

[sign-support](#)

- 1. Introduction [sign-intro](#)
- 2. Commands [sign-commands](#)
- 3. Functions [sign-functions-details](#)

`{only available when compiled with the |+signs| feature}`

1. Introduction

[sign-intro](#) [signs](#)

When a debugger or other IDE tool is driving an editor it needs to be able to give specific highlights which quickly tell the user useful information about the file. One example of this would be a debugger which had an icon in the left-hand column denoting a breakpoint. Another example might be an arrow representing the Program Counter (PC). The sign features allow both placement of a sign, or icon, in the left-hand side of the window and definition of a highlight which will be applied to that line. Displaying the sign as an image is most likely only feasible in gvim (although Sun Microsystem's dtterm does support this it's the only terminal emulator I know of which does). A text sign and the highlight should be feasible in any color terminal emulator.

Signs and highlights are not useful just for debuggers. Sun's Visual WorkShop uses signs and highlights to mark build errors and SourceBrowser hits. Additionally, the debugger supports 8 to 10 different signs and highlight colors, see [NetBeans](#) .

There are two steps in using signs:

1. Define the sign. This specifies the image, text and highlighting. For example, you can define a "break" sign with an image of a stop roadsign and text "!!".
2. Place the sign. This specifies the file and line number where the sign is displayed. A defined sign can be placed several times in different lines and files.

[sign-column](#)

When signs are defined for a file, Vim will automatically add a column of two characters to display them in. When the last sign is unplaced the column disappears again. This behavior can be changed with the '[signcolumn](#)' option.

The color of the column is set with the SignColumn highlight group [hl-SignColumn](#) . Example to set the color:

```
:highlight SignColumn guibg=darkgrey
```

If **'cursorline'** is enabled, then the CursorLineSign highlight group is used `hl-CursorLineSign` .

sign-identifier

Each placed sign is identified by a number called the sign identifier. This identifier is used to jump to the sign or to remove the sign. The identifier is assigned when placing the sign using the `:sign-place` command or the `sign_place()` function. Each sign identifier should be a unique number. If multiple placed signs use the same identifier, then jumping to or removing a sign becomes unpredictable. To avoid overlapping identifiers, sign groups can be used. The `sign_place()` function can be called with a zero sign identifier to allocate the next available identifier.

sign-group

Each placed sign can be assigned to either the global group or a named group. When placing a sign, if a group name is not supplied, or an empty string is used, then the sign is placed in the global group. Otherwise the sign is placed in the named group. The sign identifier is unique within a group. The sign group allows Vim plugins to use unique signs without interfering with other plugins using signs.

To place a sign in a popup window the group name must start with "PopUp". Other signs will not show in a popup window. The group name "PopUpMenu" is used by popup windows where **'cursorline'** is set.

sign-priority

Each placed sign is assigned a priority value. When multiple signs are placed on the same line, the attributes of the sign with the highest priority is used independently of the sign group. The default priority for a sign is 10. The priority is assigned at the time of placing a sign.

When two signs with the same priority are present, and one has an icon or text in the signcolumn while the other has line highlighting, then both are displayed.

When the line on which the sign is placed is deleted, the sign is moved to the next line (or the last line of the buffer, if there is no next line). When the delete is undone the sign does not move back.

When a sign with line highlighting and **'cursorline'** highlighting are both present, if the priority is 100 or more then the sign highlighting takes precedence, otherwise the **'cursorline'** highlighting.

=====
2. Commands sign-commands :sig :sign

Here is an example that places a sign "piet", displayed with the text ">>", in line 23 of the current file:

```
:sign define piet text=>> texthl=Search  
:exe ":sign place 2 line=23 name=piet file=" .. expand("%:~p")
```

And here is the command to delete it again:

```
:sign unplace 2
```

Note that the `:sign` command cannot be followed by another command or a comment. If you do need that, use the `:execute` command.

DEFINING A SIGN.

`:sign-define` E255 E160 E612

See `sign_define()` for the equivalent Vim script function.

`:sign define {name} {argument}...`

Define a new sign or set attributes for an existing sign. The `{name}` can either be a number (all digits) or a name starting with a non-digit. Leading zeros are ignored, thus "0012", "012" and "12" are considered the same name. About 120 different signs can be defined.

Accepted arguments:

`icon={bitmap}`

Define the file name where the bitmap can be found. Should be a full path. The bitmap should fit in the place of two characters. This is not checked. If the bitmap is too big it will cause redraw problems. Only GTK 2 can scale the bitmap to fit the space available.

toolkit	supports
GTK 1	pixmap (.xpm)
GTK 2	many
Motif	pixmap (.xpm)
Win32	.bmp, .ico, .cur pixmap (.xpm) <code>+xpm_w32</code>

`linehl={group}`

Highlighting group used for the whole line the sign is placed in. Most useful is defining a background color.

`numhl={group}`

Highlighting group used for the line number on the line where the sign is placed. Overrides `hl-LineNr`, `hl-LineNrAbove`, `hl-LineNrBelow`, and `hl-CursorLineNr`.

`text={text}`

Define the text that is displayed when there is no icon or the GUI is not being used. Only printable characters are allowed and they must occupy one or two display cells. E239

`texthl={group}`

Highlighting group used for the text item.

`culhl={group}`

Highlighting group used for the text item when the cursor is on the same line as the sign and `'cursorline'` is enabled.

Example:

```
:sign define MySign text=>> texthl=Search linehl=DiffText
```

DELETING A SIGN

`:sign-undefine` E155

See `sign_undefine()` for the equivalent Vim script function.

`:sign undefine {name}`

Deletes a previously defined sign. If signs with this `{name}` are still placed this will cause trouble.

Example:

```
:sign undefine MySign
```

LISTING SIGNS

`:sign-list` E156

See `sign_getdefined()` for the equivalent Vim script function.

`:sign list` Lists all defined signs and their attributes.

`:sign list {name}`

Lists one defined sign and its attributes.

PLACING SIGNS

`:sign-place` E158

See `sign_place()` for the equivalent Vim script function.

`:sign place {id} line={lnum} name={name} file={fname}`

Place sign defined as `{name}` at line `{lnum}` in file `{fname}`.
`:sign-fname`

The file `{fname}` must already be loaded in a buffer. The exact file name must be used, wildcards, `$ENV` and `~` are not expanded, white space must not be escaped. Trailing white space is ignored.

The sign is remembered under `{id}`, this can be used for further manipulation. `{id}` must be a number.

It's up to the user to make sure the `{id}` is used only once in each file (if it's used several times unplacing will also have to be done several times and making changes may not work as expected).

The following optional sign attributes can be specified before `"file="`:

```
group={group} Place sign in sign group {group}  
priority={prio} Assign priority {prio} to sign
```

By default, the sign is placed in the global sign group.

By default, the sign is assigned a default priority of 10. To assign a different priority value, use `"priority={prio}"` to specify a value. The priority is used to determine the sign that is displayed when multiple signs are placed on the same line.

Examples:

```
:sign place 5 line=3 name=sign1 file=a.py
:sign place 6 group=g2 line=2 name=sign2 file=x.py
:sign place 9 group=g2 priority=50 line=5
\ name=sign1 file=a.py
```

```
:sign place {id} line={lnum} name={name} [buffer={nr}]
```

Same, but use buffer {nr}. If the buffer argument is not given, place the sign in the current buffer.

Example:

```
:sign place 10 line=99 name=sign3
:sign place 10 line=99 name=sign3 buffer=3
```

E885

```
:sign place {id} name={name} file={fname}
```

Change the placed sign {id} in file {fname} to use the defined sign {name}. See remark above about {fname} :sign-fname . This can be used to change the displayed sign without moving it (e.g., when the debugger has stopped at a breakpoint).

The optional "group={group}" attribute can be used before "file=" to select a sign in a particular group. The optional "priority={prio}" attribute can be used to change the priority of an existing sign.

Example:

```
:sign place 23 name=sign1 file=/path/to/edit.py
```

```
:sign place {id} name={name} [buffer={nr}]
```

Same, but use buffer {nr}. If the buffer argument is not given, use the current buffer.

Example:

```
:sign place 23 name=sign1
:sign place 23 name=sign1 buffer=7
```

REMOVING SIGNS

:sign-unplace E159

See `sign_unplace()` for the equivalent Vim script function.

```
:sign unplace {id} file={fname}
```

Remove the previously placed sign {id} from file {fname}. See remark above about {fname} :sign-fname .

```
:sign unplace {id} group={group} file={fname}
```

Same but remove the sign {id} in sign group {group}.

```
:sign unplace {id} group=* file={fname}
```

Same but remove the sign {id} from all the sign groups.

```
:sign unplace * file={fname}
```

Remove all placed signs in file {fname}.

- `:sign unplace * group={group} file={fname}`
Remove all placed signs in group {group} from file {fname}.
- `:sign unplace * group=* file={fname}`
Remove all placed signs in all the groups from file {fname}.
- `:sign unplace {id} buffer={nr}`
Remove the previously placed sign {id} from buffer {nr}.
- `:sign unplace {id} group={group} buffer={nr}`
Remove the previously placed sign {id} in group {group} from buffer {nr}.
- `:sign unplace {id} group=* buffer={nr}`
Remove the previously placed sign {id} in all the groups from buffer {nr}.
- `:sign unplace * buffer={nr}`
Remove all placed signs in buffer {nr}.
- `:sign unplace * group={group} buffer={nr}`
Remove all placed signs in group {group} from buffer {nr}.
- `:sign unplace * group=* buffer={nr}`
Remove all placed signs in all the groups from buffer {nr}.
- `:sign unplace {id}`
Remove the previously placed sign {id} from all files it appears in.
- `:sign unplace {id} group={group}`
Remove the previously placed sign {id} in group {group} from all files it appears in.
- `:sign unplace {id} group=*`
Remove the previously placed sign {id} in all the groups from all the files it appears in.
- `:sign unplace *`
Remove all placed signs in the global group from all the files.
- `:sign unplace * group={group}`
Remove all placed signs in group {group} from all the files.
- `:sign unplace * group=*`
Remove all placed signs in all the groups from all the files.
- `:sign unplace`
Remove a placed sign at the cursor position. If multiple signs are placed in the line, then only one is removed.
- `:sign unplace group={group}`

Remove a placed sign in group `{group}` at the cursor position.

`:sign unplace group=*`
Remove a placed sign in any group at the cursor position.

LISTING PLACED SIGNS

`:sign-place-list`

See `sign_getplaced()` for the equivalent Vim script function.

`:sign place file={fname}`
List signs placed in file `{fname}`.
See remark above about `{fname}` `:sign-fname` .

`:sign place group={group} file={fname}`
List signs in group `{group}` placed in file `{fname}`.

`:sign place group=* file={fname}`
List signs in all the groups placed in file `{fname}`.

`:sign place buffer={nr}`
List signs placed in buffer `{nr}`.

`:sign place group={group} buffer={nr}`
List signs in group `{group}` placed in buffer `{nr}`.

`:sign place group=* buffer={nr}`
List signs in all the groups placed in buffer `{nr}`.

`:sign place` List placed signs in the global group in all files.

`:sign place group={group}`
List placed signs with sign group `{group}` in all files.

`:sign place group=*`
List placed signs in all sign groups in all files.

JUMPING TO A SIGN

`:sign-jump` E157

See `sign_jump()` for the equivalent Vim script function.

`:sign jump {id} file={fname}`
Open the file `{fname}` or jump to the window that contains `{fname}` and position the cursor at sign `{id}`.
See remark above about `{fname}` `:sign-fname` .
If the file isn't displayed in window and the current file can not be `abandon` ed this fails.

`:sign jump {id} group={group} file={fname}`
Same but jump to the sign in group `{group}`

`:sign jump {id} [buffer={nr}]` E934

Same, but use buffer `{nr}`. This fails if buffer `{nr}` does not have a name. If the buffer argument is not given, use the current buffer.

```
:sign jump {id} group={group} [buffer={nr}]  
Same but jump to the sign in group {group}
```

3. Functions

sign-functions-details

```
sign_define({name} [, {dict}])  
sign_define({list})
```

sign_define()

Define a new sign named `{name}` or modify the attributes of an existing sign. This is similar to the `:sign-define` command.

Prefix `{name}` with a unique text to avoid name collisions. There is no `{group}` like with placing signs.

The `{name}` can be a String or a Number. The optional `{dict}` argument specifies the sign attributes. The following values are supported:

icon	full path to the bitmap file for the sign.
linehl	highlight group used for the whole line the sign is placed in.
numhl	highlight group used for the line number where the sign is placed.
text	text that is displayed when there is no icon or the GUI is not being used.
texthl	highlight group used for the text item
culhl	highlight group used for the text item when the cursor is on the same line as the sign and <code>'cursorline'</code> is enabled.

If the sign named `{name}` already exists, then the attributes of the sign are updated.

The one argument `{list}` can be used to define a list of signs. Each list item is a dictionary with the above items in `{dict}` and a "name" item for the sign name.

Returns 0 on success and -1 on failure. When the one argument `{list}` is used, then returns a List of values one for each defined sign.

Examples:

```
call sign_define("mySign", {  
  \ "text" : "=>",  
  \ "texthl" : "Error",  
  \ "linehl" : "Search"})  
call sign_define([  
  \ {'name' : 'sign1',  
    \ 'text' : '=>'},  
  \ {'name' : 'sign2',
```

```
\ 'text' : '!!!'  
\ ])
```

Can also be used as a `method` :
`GetSignList()->sign_define()`

`sign_getdefined([name])` `sign_getdefined()`
Get a list of defined signs and their attributes.
This is similar to the `:sign-list` command.

If the `{name}` is not supplied, then a list of all the defined signs is returned. Otherwise the attribute of the specified sign is returned.

Each list item in the returned value is a dictionary with the following entries:

<code>icon</code>	full path to the bitmap file of the sign
<code>linehl</code>	highlight group used for the whole line the sign is placed in; not present if not set
<code>name</code>	name of the sign
<code>numhl</code>	highlight group used for the line number where the sign is placed; not present if not set
<code>text</code>	text that is displayed when there is no icon or the GUI is not being used.
<code>texthl</code>	highlight group used for the text item; not present if not set
<code>culhl</code>	highlight group used for the text item when the cursor is on the same line as the sign and <code>'cursorline'</code> is enabled; not present if not set

Returns an empty List if there are no signs and when `{name}` is not found.

Examples:

```
" Get a list of all the defined signs  
echo sign_getdefined()  
  
" Get the attribute of the sign named mySign  
echo sign_getdefined("mySign")
```

Can also be used as a `method` :
`GetSignList()->sign_getdefined()`

`sign_getplaced([buf] [, {dict}])` `sign_getplaced()`
Return a list of signs placed in a buffer or all the buffers.
This is similar to the `:sign-place-list` command.

If the optional buffer name `{buf}` is specified, then only the list of signs placed in that buffer is returned. For the use of `{buf}`, see `bufname()` . The optional `{dict}` can contain the following entries:

<code>group</code>	select only signs in this group
<code>id</code>	select sign with this identifier

lnum select signs placed in this line. For the use of {lnum}, see `line()` .
If {group} is '*', then signs in all the groups including the global group are returned. If {group} is not supplied or is an empty string, then only signs in the global group are returned. If no arguments are supplied, then signs in the global group placed in all the buffers are returned. See `sign-group` .

Each list item in the returned value is a dictionary with the following entries:

bufnr number of the buffer with the sign
signs list of signs placed in {bufnr}. Each list item is a dictionary with the below listed entries

The dictionary for each sign contains the following entries:

group sign group. Set to '' for the global group.
id identifier of the sign
lnum line number where the sign is placed
name name of the defined sign
priority sign priority

The returned signs in a buffer are ordered by their line number and priority.

Returns an empty list on failure or if there are no placed signs.

Examples:

```
" Get a List of signs placed in eval.c in the
" global group
echo sign_getplaced("eval.c")

" Get a List of signs in group 'g1' placed in eval.c
echo sign_getplaced("eval.c", {'group' : 'g1'})

" Get a List of signs placed at line 10 in eval.c
echo sign_getplaced("eval.c", {'lnum' : 10})

" Get sign with identifier 10 placed in a.py
echo sign_getplaced("a.py", {'id' : 10})

" Get sign with id 20 in group 'g1' placed in a.py
echo sign_getplaced("a.py", {'group' : 'g1',
                             \ 'id' : 20})

" Get a List of all the placed signs
echo sign_getplaced()
```

Can also be used as a `method` :

```
GetBufname()->sign_getplaced()
```

```
sign_jump()
```

`sign_jump({id}, {group}, {buf})`

Open the buffer `{buf}` or jump to the window that contains `{buf}` and position the cursor at sign `{id}` in group `{group}`. This is similar to the `:sign-jump` command.

If `{group}` is an empty string, then the global group is used. For the use of `{buf}`, see `bufname()` .

Returns the line number of the sign. Returns -1 if the arguments are invalid.

Example:

```
" Jump to sign 10 in the current buffer
call sign_jump(10, '', '')
```

Can also be used as a `method` :

```
GetSignid()->sign_jump()
```

`sign_place()`

`sign_place({id}, {group}, {name}, {buf} [, {dict}])`

Place the sign defined as `{name}` at line `{lnum}` in file or buffer `{buf}` and assign `{id}` and `{group}` to sign. This is similar to the `:sign-place` command.

If the sign identifier `{id}` is zero, then a new identifier is allocated. Otherwise the specified number is used. `{group}` is the sign group name. To use the global sign group, use an empty string. `{group}` functions as a namespace for `{id}`, thus two groups can use the same IDs. Refer to `sign-identifier` and `sign-group` for more information.

`{name}` refers to a defined sign.

`{buf}` refers to a buffer name or number. For the accepted values, see `bufname()` .

The optional `{dict}` argument supports the following entries:

<code>lnum</code>	line number in the file or buffer <code>{buf}</code> where the sign is to be placed. For the accepted values, see <code>line()</code> .
<code>priority</code>	priority of the sign. See <code>sign-priority</code> for more information.

If the optional `{dict}` is not specified, then it modifies the placed sign `{id}` in group `{group}` to use the defined sign `{name}`.

Returns the sign identifier on success and -1 on failure.

Examples:

```
" Place a sign named sign1 with id 5 at line 20 in
" buffer json.c
call sign_place(5, '', 'sign1', 'json.c',
\ {'lnum' : 20})
```

```

" Updates sign 5 in buffer json.c to use sign2
call sign_place(5, '', 'sign2', 'json.c')

" Place a sign named sign3 at line 30 in
" buffer json.c with a new identifier
let id = sign_place(0, '', 'sign3', 'json.c',
                  \ {'lnum' : 30})

" Place a sign named sign4 with id 10 in group 'g3'
" at line 40 in buffer json.c with priority 90
call sign_place(10, 'g3', 'sign4', 'json.c',
               \ {'lnum' : 40, 'priority' : 90})

```

Can also be used as a [method](#) :

```
GetSignid()->sign_place(group, name, expr)
```

[sign_placelist\(\)](#)

`sign_placelist({list})`

Place one or more signs. This is similar to the [sign_place\(\)](#) function. The `{list}` argument specifies the List of signs to place. Each list item is a dict with the following sign attributes:

<code>buffer</code>	Buffer name or number. For the accepted values, see bufname() .
<code>group</code>	Sign group. <code>{group}</code> functions as a namespace for <code>{id}</code> , thus two groups can use the same IDs. If not specified or set to an empty string, then the global group is used. See sign-group for more information.
<code>id</code>	Sign identifier. If not specified or zero, then a new unique identifier is allocated. Otherwise the specified number is used. See sign-identifier for more information.
<code>lnum</code>	Line number in the buffer where the sign is to be placed. For the accepted values, see line() .
<code>name</code>	Name of the sign to place. See sign_define() for more information.
<code>priority</code>	Priority of the sign. When multiple signs are placed on a line, the sign with the highest priority is used. If not specified, the default value of 10 is used. See sign-priority for more information.

If `{id}` refers to an existing sign, then the existing sign is modified to use the specified `{name}` and/or `{priority}`.

Returns a List of sign identifiers. If failed to place a sign, the corresponding list item is set to -1.

Examples:

```

" Place sign s1 with id 5 at line 20 and id 10 at line
" 30 in buffer a.c
let [n1, n2] = sign_placelist([

```

```

\ {'id' : 5,
\ 'name' : 's1',
\ 'buffer' : 'a.c',
\ 'lnum' : 20},
\ {'id' : 10,
\ 'name' : 's1',
\ 'buffer' : 'a.c',
\ 'lnum' : 30}
\ ])

" Place sign s1 in buffer a.c at line 40 and 50
" with auto-generated identifiers
let [n1, n2] = sign_placelist([
\ {'name' : 's1',
\ 'buffer' : 'a.c',
\ 'lnum' : 40},
\ {'name' : 's1',
\ 'buffer' : 'a.c',
\ 'lnum' : 50}
\ ])

```

Can also be used as a `method` :
`GetSignlist()->sign_placelist()`

```

sign_undefine([name]) sign_undefine()
sign_undefine(list)

```

Deletes a previously defined sign `{name}`. This is similar to the `:sign-undefine` command. If `{name}` is not supplied, then deletes all the defined signs.

The one argument `{list}` can be used to undefine a list of signs. Each list item is the name of a sign.

Returns 0 on success and -1 on failure. For the one argument `{list}` call, returns a list of values one for each undefined sign.

Examples:

```

" Delete a sign named mySign
call sign_undefine("mySign")

" Delete signs 'sign1' and 'sign2'
call sign_undefine(["sign1", "sign2"])

" Delete all the signs
call sign_undefine()

```

Can also be used as a `method` :
`GetSignlist()->sign_undefine()`

```

sign_unplace({group} [, dict]) sign_unplace()

```

Remove a previously placed sign in one or more buffers. This is similar to the `:sign-unplace` command.

`{group}` is the sign group name. To use the global sign group, use an empty string. If `{group}` is set to '*', then all the groups including the global group are used. The signs in `{group}` are selected based on the entries in `{dict}`. The following optional entries in `{dict}` are supported:

 buffer buffer name or number. See `bufname()` .
 id sign identifier

If `{dict}` is not supplied, then all the signs in `{group}` are removed.

Returns 0 on success and -1 on failure.

Examples:

```
" Remove sign 10 from buffer a.vim  
call sign_unplace('', {'buffer' : "a.vim", 'id' : 10})
```

```
" Remove sign 20 in group 'g1' from buffer 3  
call sign_unplace('g1', {'buffer' : 3, 'id' : 20})
```

```
" Remove all the signs in group 'g2' from buffer 10  
call sign_unplace('g2', {'buffer' : 10})
```

```
" Remove sign 30 in group 'g3' from all the buffers  
call sign_unplace('g3', {'id' : 30})
```

```
" Remove all the signs placed in buffer 5  
call sign_unplace('*', {'buffer' : 5})
```

```
" Remove the signs in group 'g4' from all the buffers  
call sign_unplace('g4')
```

```
" Remove sign 40 from all the buffers  
call sign_unplace('*', {'id' : 40})
```

```
" Remove all the placed signs from all the buffers  
call sign_unplace('*')
```

Can also be used as a `method` :

```
GetSigngroup()->sign_unplace()
```

`sign_unplacelist({list})` `sign_unplacelist()`

Remove previously placed signs from one or more buffers. This is similar to the `sign_unplace()` function.

The `{list}` argument specifies the List of signs to remove. Each list item is a dict with the following sign attributes:

 buffer buffer name or number. For the accepted values, see `bufname()` . If not specified, then the specified sign is removed from all the buffers.
 group sign group name. If not specified or set to an empty string, then the global sign group is used. If set to '*', then all the groups

including the global group are used.
id sign identifier. If not specified, then all
the signs in the specified group are removed.

Returns a List where an entry is set to 0 if the corresponding
sign was successfully removed or -1 on failure.

Example:

```
" Remove sign with id 10 from buffer a.vim and sign  
" with id 20 from buffer b.vim  
call sign_unplacelist(  
    \ {'id' : 10, 'buffer' : "a.vim"},  
    \ {'id' : 20, 'buffer' : 'b.vim'},  
    \ ])
```

Can also be used as a `method` :

```
GetSignlist()->sign_unplacelist()
```


Differences between Vim and Vi

vi-differences

This file lists the differences between Vim and Vi/Ex and gives an overview of what is in Vim that is not in Vi.

Vim is mostly POSIX 1003.2-1 compliant. The only command known to be missing is `":open"`. There are probably a lot of small differences (either because Vim is missing something or because Posix is beside the mark).

- | | |
|-----------------------------------|-----------------------------------|
| 1. Simulated command | simulated-command |
| 2. Missing options | missing-options |
| 3. Limits | limits |
| 4. The most interesting additions | vim-additions |
| 5. Other vim features | other-features |
| 6. Supported Vi features | vi-features |
| 7. Command-line arguments | cmdline-arguments |
| 8. POSIX compliance | posix-compliance |

1. Simulated command

[simulated-command](#)

This command is in Vi, but Vim only simulates it:

- | | |
|---------------------------------------|--|
| <code>:[range]o[pen]</code> | Works like <code>:visual</code> : end Ex mode.
{Vi: start editing in open mode} |
| <code>:[range]o[pen] /pattern/</code> | As above, additionally move the cursor to the column where "pattern" matches in the cursor line. |

Vim does not support open mode, since it's not really useful. For those situations where `":open"` would start open mode Vim will leave Ex mode, which allows executing the same commands, but updates the whole screen instead of only one line.

2. Missing options

[missing-options](#)

These options are in the Unix Vi, but not in Vim. If you try to set one of them you won't get an error message, but the value is not used and cannot be printed.

- | | | | |
|-----------------------------|-----------------------|--------------------------|-------------------|
| <code>autoprint (ap)</code> | boolean (default on) | <code>'autoprint'</code> | <code>'ap'</code> |
| <code>beautify (bf)</code> | boolean (default off) | <code>'beautify'</code> | <code>'bf'</code> |
| <code>flash (fl)</code> | boolean (default ??) | <code>'flash'</code> | <code>'fl'</code> |
| <code>graphic (gr)</code> | boolean (default off) | <code>'graphic'</code> | <code>'gr'</code> |
| <code>hardtabs (ht)</code> | number (default 8) | <code>'hardtabs'</code> | <code>'ht'</code> |

	number of spaces that a <Tab> moves on the display		
mesg	boolean (default on)	'mesg'	
novice	boolean (default off)	'novice'	
open	boolean (default on)	'open'	
optimize (op)	boolean (default off)	'optimize'	'op'
redraw	boolean (default off)	'redraw'	
slowopen (slow)	boolean (default off)	'slowopen'	'slow'
sourceany	boolean (default off)	'sourceany'	
w300	number (default 23)	'w300'	
w1200	number (default 23)	'w1200'	
w9600	number (default 23)	'w9600'	

Vi did not allow for changing the termcap entries, you would have to exit Vi, edit the termcap entry and try again. Vim has the `terminal-options` .

3. Limits

limits

Vim has only a few limits for the files that can be edited {Vi: can not handle <Nul> characters and characters above 128, has limited line length, many other limits}.

Maximum line length	2147483647 characters. Longer lines are split.
Maximum number of lines	2147483647 lines.
Maximum file size	2147483647 bytes (2 Gbyte) when a long integer is 32 bits. Much more for 64 bit longs. Also limited by available disk space for the <code>swap-file</code> .
	E75
Length of a file path	Unix and Win32: 1024 characters, otherwise 256 characters (or as much as the system supports).
Length of an expanded string option	Unix and Win32: 1024 characters, otherwise 256 characters
Maximum display width	Unix and Win32: 1024 characters, otherwise 255 characters
Maximum lhs of a mapping	50 characters.
Number of different highlighting types:	over 30000
Range of a Number variable:	-2147483648 to 2147483647 (might be more on 64 bit systems)
Maximum length of a line in a tags file:	512 bytes.

Information for undo and text in registers is kept in memory, thus when making (big) changes the amount of (virtual) memory available limits the number of undo levels and the text that can be kept in registers. Other things are also kept in memory: Command-line history, error messages for Quickfix mode, etc.

Memory usage limits

The option `'maxmem'` ('mm') is used to set the maximum memory used for one buffer (in kilobytes). `'maxmemtot'` is used to set the maximum memory used for all buffers (in kilobytes). The defaults depend on the system used. For the Amiga, `'maxmemtot'` is set depending on the amount of memory available. These are not hard limits, but tell Vim when to move text into a swap file.

If you don't like Vim to swap to a file, set `'maxmem'` and `'maxmemtot'` to a very large value. The swap file will then only be used for recovery. If you don't want a swap file at all, set `'updatecount'` to 0, or use the `"-n"` argument when starting Vim.

4. The most interesting additions

`vim-additions`

Vi compatibility.

`'compatible'`

Although Vim is 99% Vi compatible, some things in Vi can be considered to be a bug, or at least need improvement. But still, Vim starts in a mode which behaves like the "real" Vi as much as possible. To make Vim behave a little bit better, try resetting the `'compatible'` option:

```
:set nocompatible
```

Or start Vim with the `"-N"` argument:

```
vim -N
```

Vim starts with `'nocompatible'` automatically if you have a `.vimrc` file. See [startup](#).

The `'coptions'` option can be used to set Vi compatibility on/off for a number of specific items.

Support for different systems.

Vim can be used on:

- All Unix systems (it works on all systems it was tested on, although the GUI and Perl interface may not work everywhere).
- Amiga (500, 1000, 1200, 2000, 3000, 4000, ...).
- MS-Windows
- VMS
- Macintosh
- IBM OS/390

Note that on some systems features need to be disabled to reduce resource usage. For some outdated systems you need to use an older Vim version.

Multi level persistent undo.

`undo`

`'u'` goes backward in time, `'CTRL-R'` goes forward again. Set option `'undolevels'` to the number of changes to be remembered (default 1000). Set `'undolevels'` to 0 for a Vi-compatible one level undo. Set it to -1 for no undo at all.

When all changes in a buffer have been undone, the buffer is not considered changed anymore. You can exit it with `:q`, without `<!>`.

When undoing a few changes and then making a new change Vim will create a branch in the undo tree. This means you can go back to any state of the text, there is no risk of a change causing text to be lost forever. [undo-tree](#)

The undo information is stored in a file when the `'undofile'` option is set. This means you can exit Vim, start Vim on a previously edited file and undo changes that were made before exiting Vim.

Graphical User Interface (GUI).

`gui`

Included support for GUI: menu's, mouse, scrollbars, etc. You can define your own menus. Better support for CTRL/SHIFT/ALT keys in combination with special keys and mouse. Supported for various

platforms, such as X11 with Motif, GTK, Win32 (Windows XP and later), Amiga and Macintosh.

Multiple windows and buffers.

[windows.txt](#)

Vim can split the screen into several windows, each editing a different buffer or the same buffer at a different location. Buffers can still be loaded (and changed) but not displayed in a window. This is called a hidden buffer. Many commands and options have been added for this facility.

Vim can also use multiple tab pages, each with one or more windows. A line with tab labels can be used to quickly switch between these pages.

[tab-page](#)

Terminal window.

[:terminal](#)

Vim can create a window in which a terminal emulator runs. This can be used to execute an arbitrary command, a shell or a debugger.

Syntax highlighting.

[:syntax](#)

Vim can highlight keywords, patterns and other things. This is defined by a number of [:syntax](#) commands, and can be made to highlight most languages and file types. A number of files are included for highlighting the most common languages, like C, C++, Java, Pascal, Makefiles, shell scripts, etc. The colors used for highlighting can be defined for ordinary terminals, color terminals and the GUI with the [:highlight](#) command. A convenient way to do this is using a [:colorscheme](#) command.

The highlighted text can be exported as HTML. [convert-to-HTML](#)

Other items that can be highlighted are matches with the search string ['hlsearch'](#), matching parens [matchparen](#) and the cursor line and column ['cursorline'](#) ['cursorcolumn'](#).

Text properties

[textprop.txt](#)

Vim supports highlighting text by a plugin. Property types can be specified with [prop_type_add\(\)](#) and properties can be placed with [prop_add\(\)](#).

Spell checking.

[spell](#)

When the ['spell'](#) option is set Vim will highlight spelling mistakes. About 50 languages are currently supported, selected with the ['spelllang'](#) option. In source code only comments and strings are checked for spelling.

Folding.

[folding](#)

A range of lines can be shown as one "folded" line. This allows overviewing a file and moving blocks of text around quickly. Folds can be created manually, from the syntax of the file, by indent, etc.

Diff mode.

[diff](#)

Vim can show two versions of a file with the differences highlighted. Parts of the text that are equal are folded away. Commands can be used to move text from one version to the other.

Plugins.

[add-plugin](#)

The functionality can be extended by dropping a plugin file in the right directory. That's an easy way to start using Vim scripts written by others. Plugins can be for all kind of files, or specifically for a filetype.

Packages make this even easier. [packages](#)

Asynchronous communication and timers. [channel](#) [job](#) [timer](#)

Vim can exchange messages with other processes in the background. This makes it possible to have servers do work and send back the results to Vim. [channel](#)

Vim can start a job, communicate with it and stop it. [job](#)

Timers can fire once or repeatedly and invoke a function to do any work. [timer](#)

Repeat a series of commands.

[q](#)

"q{c}" starts recording typed characters into named register {c}.

A subsequent "q" stops recording. The register can then be executed with the "@{c}" command. This is very useful to repeat a complex action.

Flexible insert mode.

[ins-special-special](#)

The arrow keys can be used in insert mode to move around in the file. This breaks the insert in two parts as far as undo and redo is concerned.

CTRL-O can be used to execute a single Normal mode command. This is almost the same as hitting <Esc>, typing the command and doing [a](#) .

Visual mode.

[Visual-mode](#)

Visual mode can be used to first highlight a piece of text and then give a command to do something with it. This is an (easy to use) alternative to first giving the operator and then moving to the end of the text to be operated upon.

[v](#) and [V](#) are used to start Visual mode. [v](#) works on characters and [V](#) on lines. Move the cursor to extend the Visual area. It is shown highlighted on the screen. By typing "o" the other end of the Visual area can be moved. The Visual area can be affected by an operator:

d	delete
c	change
y	yank
> or <	insert or delete indent
!	filter through external program
=	filter through indent
:	start : command for the Visual lines.
gq	format text to 'textwidth' columns
J	join lines
~	swap case
u	make lowercase
U	make uppercase

{Vi has no Visual mode, the name "visual" is used for Normal mode, to distinguish it from Ex mode}

Block operators.

[visual-block](#)

With Visual mode a rectangular block of text can be selected. Start Visual mode with **CTRL-V**. The block can be deleted ("d"), yanked ("y") or its case can be changed ("~", "u" and "U"). A deleted or yanked block can be put into the text with the "p" and "P" commands.

Help system.

[:help](#)

Help is displayed in a window. The usual commands can be used to move around, search for a string, etc. Tags can be used to jump around in the help files, just like hypertext links. The [:help](#) command takes an argument to quickly jump to the info on a subject. [<F1>](#) is the quick access to the help system. The name of the help index file can be set with the '[helpfile](#)' option.

Command-line editing and history.

[cmdline-editing](#)

You can insert or delete at any place in the command-line using the cursor keys. The right/left cursor keys can be used to move forward/backward one character. The shifted right/left cursor keys can be used to move forward/backward one word. **CTRL-B/CTRL-E** can be used to go to the begin/end of the command-line.

[{Vi: can only alter the last character in the line}](#)

[{Vi: when hitting <Esc> the command-line is executed. This is unexpected for most people; therefore it was changed in Vim. But when the <Esc> is part of a mapping, the command-line is executed. If you want the Vi behaviour also when typing <Esc>, use ":cmap ^V<Esc> ^V^M"}}](#)

[cmdline-history](#)

The command-lines are remembered. The up/down cursor keys can be used to recall previous command-lines. The '[history](#)' option can be set to the number of lines that will be remembered. There is a separate history for commands and for search patterns.

Command-line completion.

[cmdline-completion](#)

While entering a command-line (on the bottom line of the screen)

[<Tab>](#) can be typed to complete

what	example
- command	:e<Tab>
- tag	:ta scr<Tab>
- option	:set sc<Tab>
- option value	:set hf=<Tab>
- file name	:e ve<Tab>
- etc.	

If there are multiple matches, **CTRL-N** (next) and **CTRL-P** (previous) will walk through the matches. [<Tab>](#) works like **CTRL-N**, but wraps around to the first match.

The '[wildchar](#)' option can be set to the character for command-line completion, [<Tab>](#) is the default. **CTRL-D** can be typed after an (incomplete) wildcard; all matches will be listed. **CTRL-A** will insert all matches. **CTRL-L** will insert the longest common part of the matches.

Insert-mode completion.

[ins-completion](#)

In Insert mode the **CTRL-N** and **CTRL-P** keys can be used to complete a

word that appears elsewhere. `i_CTRL-N`
With **CTRL-X** another mode is entered, through which completion can be done for:

<code>i_CTRL-X_CTRL-F</code>	file names
<code>i_CTRL-X_CTRL-K</code>	words from 'dictionary' files
<code>i_CTRL-X_CTRL-T</code>	words from 'thesaurus' files
<code>i_CTRL-X_CTRL-I</code>	words from included files
<code>i_CTRL-X_CTRL-L</code>	whole lines
<code>i_CTRL-X_CTRL-]</code>	words from the tags file
<code>i_CTRL-X_CTRL-D</code>	definitions or macros
<code>i_CTRL-X_CTRL-O</code>	Omni completion: clever completion specifically for a file type

etc.

Long line support. `'wrap'` `'linebreak'`

If the **'wrap'** option is off, long lines will not wrap and only part of them will be shown. When the cursor is moved to a part that is not shown, the screen will scroll horizontally. The minimum number of columns to scroll can be set with the **'sidescroll'** option. The `zh` and `zl` commands can be used to scroll sideways.

Alternatively, long lines are broken in between words when the **'linebreak'** option is set. This allows editing a single-line paragraph conveniently (e.g. when the text is later read into a DTP program). Move the cursor up/down with the `gk` and `gj` commands.

Text formatting. `formatting`

The **'textwidth'** option can be used to automatically limit the line length. This supplements the **'wrapmargin'** option of Vi, which was not very useful. The `gq` operator can be used to format a piece of text (for example, `gqap` formats the current paragraph). Commands for text alignment: `:center`, `:left` and `:right`.

Extended search patterns. `pattern`

There are many extra items to match various text items. Examples:
A `"\n"` can be used in a search pattern to match a line break.
`"x\{2,4}"` matches "x" 2 to 4 times.
`"\s"` matches a white space character.

Directory, remote and archive browsing. `netrw`

Vim can browse the file system. Simply edit a directory. Move around in the list with the usual commands and press `<Enter>` to go to the directory or file under the cursor.

This also works for remote files over ftp, http, ssh, etc.
Zip and tar archives can also be browsed. `tar` `zip`

Edit-compile-edit speedup. `quickfix`

The `:make` command can be used to run the compilation and jump to the first error. A file with compiler error messages is interpreted. Vim jumps to the first error.

Each line in the error file is scanned for the name of a file, line number and error message. The **'errorformat'** option can be set to a list of scanf-like strings to handle output from many compilers.

The `:cn` command can be used to jump to the next error.
`:cl` lists all the error messages. Other commands are available.
The `'makeef'` option has the name of the file with error messages.
The `'makeprg'` option contains the name of the program to be executed with the `:make` command.
The `'shellpipe'` option contains the string to be used to put the output of the compiler into the errorfile.

Finding matches in files. [:vimgrep](#)
Vim can search for a pattern in multiple files. This uses the advanced Vim regexp pattern, works on all systems and also works to search in compressed files.

Improved indenting for programs. ['cindent'](#)
When the `'cindent'` option is on the indent of each line is automatically adjusted. C syntax is mostly recognized. The indent for various styles can be set with `'cinoptions'`. The keys to trigger indenting can be set with `'cinkeys'`.

Comments can be automatically formatted. The `'comments'` option can be set to the characters that start and end a comment. This works best for C code, but also works for e-mail ("`>`" at start of the line) and other types of text. The `=` operator can be used to re-indent lines.

For many other languages an indent plugin is present to support automatic indenting. [30.3](#)

Searching for words in included files. [include-search](#)
The `[i` command can be used to search for a match of the word under the cursor in the current and included files. The `'include'` option can be set to a pattern that describes a command to include a file (the default is for C programs).
The `[I` command lists all matches, the `[_CTRL-I` command jumps to a match.
The `[d`, `[D` and `[_CTRL-D` commands do the same, but only for lines where the pattern given with the `'define'` option matches.

Automatic commands. [autocommand](#)
Commands can be automatically executed when reading a file, writing a file, jumping to another buffer, etc., depending on the file name. This is useful to set options and mappings for C programs, documentation, plain text, e-mail, etc. This also makes it possible to edit compressed files.

Scripts and Expressions. [expression](#)
Commands have been added to form up a powerful script language.
`:if` Conditional execution, which can be used for example to set options depending on the value of `$TERM`.
`:while` Repeat a number of commands.
`:for` Loop over a list.
`:echo` Print the result of an expression.
`:let` Assign a value to an internal variable, option, etc. Variable types are Number, String, List and Dictionary.

`:execute` Execute a command formed by an expression.
`:try` Catch exceptions.
etc., etc. See `eval` .
Debugging and profiling are supported. `debug-scripts` `profile`
If this is not enough, an interface is provided to `Python` , `Ruby` ,
`Tcl` , `Lua` , `Perl` and `MzScheme` .

Viminfo. `viminfo-file`
The command-line history, marks and registers can be stored in a file that is read on startup. This can be used to repeat a search command or command-line command after exiting and restarting Vim. It is also possible to jump right back to where the last edit stopped with `'0` . The `'viminfo'` option can be set to select which items to store in the `.viminfo` file. This is off by default.

Printing. `printing`
The `:hardcopy` command sends text to the printer. This can include syntax highlighting.

Mouse support. `mouse-using`
The mouse is supported in the GUI version, in an xterm for Unix, for BSDs with `sysmouse`, for Linux with `gpm`, and Win32. It can be used to position the cursor, select the visual area, paste a register, etc.

Usage of key names. `<>` `key-notation`
Special keys now all have a name like `<Up>`, `<End>`, etc.
This name can be used in mappings, to make it easy to edit them.

Editing binary files. `edit-binary`
Vim can edit binary files. You can change a few characters in an executable file, without corrupting it. Vim doesn't remove NUL characters (they are represented as `<NL>` internally).
`-b` command-line argument to start editing a binary file
`'binary'` Option set by `-b` . Prevents adding an `<EOL>` for the last line in the file.

Multi-language support. `multi-lang`
Files in double-byte or multibyte encodings can be edited. There is UTF-8 support to be able to edit various languages at the same time, without switching fonts. `UTF-8`
Messages and menus are available in different languages.

Move cursor beyond lines.
When the `'virtualedit'` option is set the cursor can move all over the screen, also where there is no text. This is useful to edit tables and figures easily.

===== `other-features`

5. Other vim features

A random collection of nice extra features.

When Vim is started with `"-s scriptfile"`, the characters read from

"scriptfile" are treated as if you typed them. If end of file is reached before the editor exits, further characters are read from the console.

The "-w" option can be used to record all typed characters in a script file. This file can then be used to redo the editing, possibly on another file or after changing some commands in the script file.

The "-o" option opens a window for each argument. "-o4" opens four windows.

Vi requires several termcap entries to be able to work full-screen. Vim only requires the "cm" entry (cursor motion).

In command mode:

When the **'showcmd'** option is set, the command characters are shown in the last line of the screen. They are removed when the command is finished.

If the **'ruler'** option is set, the current cursor position is shown in the last line of the screen.

"U" still works after having moved off the last changed line and after "u".

Characters with the 8th bit set are displayed. The characters between '~' and 0xa0 are displayed as "~?", "~@", "~A", etc., unless they are included in the **'isprint'** option.

"][" goes to the next ending of a C function ('}' in column 1).

"][" goes to the previous ending of a C function ('}' in column 1).

"]f", "[f" and "gf" start editing the file whose name is under the cursor. **CTRL-W** f splits the window and starts editing the file whose name is under the cursor.

"*" searches forward for the identifier under the cursor, "#" backward.

"K" runs the program defined by the **'keywordprg'** option, with the identifier under the cursor as argument.

%" can be preceded with a count. The cursor jumps to the line that percentage down in the file. The normal "%" function to jump to the matching brace skips braces inside quotes.

With the **CTRL-]** command, the cursor may be in the middle of the identifier.

The used tags are remembered. Commands that can be used with the tag stack are **CTRL-T**, ":pop" and ":tag". ":tags" lists the tag stack.

Vi uses **'wrapscan'** when searching for a tag. When jumping to a tag Vi starts searching in line 2 of another file. It does not find a tag in line 1 of another file when **'wrapscan'** is not set.

The **'tags'** option can be set to a list of tag file names. Thus multiple tag files can be used. For file names that start with "./", the "./" is replaced with the path of the current file. This makes it possible to use a

tags file in the same directory as the file being edited.
{Vi: always uses binary search in some versions}
{Vi does not have the security prevention for commands in tag files}

Previously used file names are remembered in the alternate file name list.
CTRL-^ accepts a count, which is an index in this list.
":files" command shows the list of alternate file names.
"#<N>" is replaced with the <N>th alternate file name in the list.
"#<" is replaced with the current file name without extension.

Search patterns have more features. The <NL> character is seen as part of the search pattern and the substitute string of ":s". Vi sees it as the end of the command.

Searches can put the cursor on the end of a match and may include a character offset.

Count added to "~", ":next", ":Next", "n" and "N".

The command ":next!" with 'autowrite' set does not write the file. In vi the file was written, but this is considered to be a bug, because one does not expect it and the file is not written with ":rewind!".

In Vi when entering a <CR> in replace mode deletes a character only when 'ai' is set (but does not show it until you hit <Esc>). Vim always deletes a character (and shows it immediately).

Added :wnext command. Same as ":write" followed by ":next".

The ":w!" command always writes, also when the file is write protected. In Vi you would have to do ":%!chmod +w %:S" and ":set noro".

When 'tildeop' has been set, "~" is an operator (must be followed by a movement command).

With the "J" (join) command you can reset the 'joinspaces' option to have only one space after a period (Vi inserts two spaces).

"cw" can be used to change white space formed by several characters (Vi is confusing: "cw" only changes one space, while "dw" deletes all white space).
{Vi: "cw" when on a blank followed by other blanks changes only the first blank; this is probably a bug, because "dw" deletes all the blanks}

"o" and "O" accept a count for repeating the insert (Vi clears a part of display).

Flags after Ex commands not supported (no plans to include it).

On non-UNIX systems ":cd" command shows current directory instead of going to the home directory (there isn't one). ":pwd" prints the current directory on all systems.

After a ":cd" command the file names (in the argument list, opened files) still point to the same files. In Vi ":cd" is not allowed in a changed file;

otherwise the meaning of file names change.

`":source!"` command reads Vi commands from a file.

`":mkexrc"` command writes current modified options and mappings to a `".exrc"` file. `":mkvimrc"` writes to a `".vimrc"` file.

No check for "tail recursion" with mappings. This allows things like `":map! foo ^]foo"`.

When a mapping starts with number, vi loses the count typed before it (e.g. when using the mapping `":map g 4G"` the command `"7g"` goes to line 4). This is considered a vi bug. Vim concatenates the counts (in the example it becomes `"74G"`), as most people would expect.

The `":put!` command inserts the contents of a register above the current line.

The `"p"` and `"P"` commands of vi cannot be repeated with `."` when the putted text is less than a line. In Vim they can always be repeated.

`":noremap"` command can be used to enter a mapping that will not be remapped. This is useful to exchange the meaning of two keys. `":cmap"`, `":cunmap"` and `":cnoremap"` can be used for mapping in command-line editing only. `":imap"`, `":iunmap"` and `":inoremap"` can be used for mapping in insert mode only. Similar commands exist for abbreviations: `":noreabbrev"`, `":iabbrev"` `":cabbrev"`, `":iunabbrev"`, `":cunabbrev"`, `":inoreabbrev"`, `":cnoreabbrev"`.

In Vi the command `":map foo bar"` would remove a previous mapping `":map bug foo"`. This is considered a bug, so it is not included in Vim. `":unmap! foo"` does remove `":map! bug foo"`, because unmapping would be very difficult otherwise (this is vi compatible).

The `':'` register contains the last command-line.
The `':'` register contains the current file name.
The `':'` register contains the last inserted text.

`":dis"` command shows the contents of the yank registers.

CTRL-O/CTRL-I can be used to jump to older/newer positions. These are the same positions as used with the `'` command, but may be in another file. The `":jumps"` command lists the older positions.

If the **'shiftround'** option is set, an indent is rounded to a multiple of **'shiftwidth'** with `">"` and `"<"` commands.

The **'scrolljump'** option can be set to the minimum number of lines to scroll when the cursor gets off the screen. Use this when scrolling is slow.

The **'scrolloff'** option can be set to the minimum number of lines to keep above and below the cursor. This gives some context to where you are editing. When set to a large number the cursor line is always in the middle of the window.

Uppercase marks can be used to jump between files. The `":marks"` command lists

all currently set marks. The commands `']` and ``]` jump to the end of the previous operator or end of the text inserted with the `put` command. `['` and ``['` do jump to the start. {Vi: no uppercase marks}

The **'shelltype'** option can be set to reflect the type of shell used on the Amiga.

The **'highlight'** option can be set for the highlight mode to be used for several commands.

The **CTRL-A** (add) and **CTRL-X** (subtract) commands are new. The count to the command (default 1) is added to/subtracted from the number at or after the cursor. That number may be decimal, octal (starts with a '0') or hexadecimal (starts with '0x'). Very useful in macros.

With the `:set` command the prefix `inv` can be used to invert boolean options.

In both Vi and Vim you can create a line break with the `":substitute"` command by using a **CTRL-M**. For Vi this means you cannot insert a real **CTRL-M** in the text. With Vim you can put a real **CTRL-M** in the text by preceding it with a **CTRL-V**.

In Insert mode:

If the **'revins'** option is set, insert happens backwards. This is for typing Hebrew. When inserting normal characters the cursor will not be shifted and the text moves rightwards. Backspace, **CTRL-W** and **CTRL-U** will also work in the opposite direction. **CTRL-B** toggles the **'revins'** option. In replace mode **'revins'** has no effect. Only when enabled at compile time.

The backspace key can be used just like **CTRL-D** to remove auto-indents.

You can backspace, **CTRL-U** and **CTRL-W** over line breaks if the **'backspace'** (bs) option includes `"eol"`. You can backspace over the start of insert if the **'backspace'** option includes `"start"`.

When the **'paste'** option is set, a few options are reset and mapping in insert mode and abbreviation are disabled. This allows for pasting text in windowing systems without unexpected results. When the **'paste'** option is reset, the old option values are restored.

CTRL-T/CTRL-D always insert/delete an indent in the current line, no matter what column the cursor is in.

CTRL-@ (insert previously inserted text) works always (Vi: only when typed as first character).

CTRL-A works like **CTRL-@** but does not leave insert mode.

CTRL-R {register} can be used to insert the contents of a register.

When the **'smartindent'** option is set, C programs will be better auto-indented. With **'cindent'** even more.

CTRL-Y and **CTRL-E** can be used to copy a character from above/below the current cursor position.

After **CTRL-V** you can enter a three digit decimal number. This byte value is inserted in the text as a single character. Useful for international characters that are not on your keyboard.

When the '**expandtab**' (et) option is set, a <Tab> is expanded to the appropriate number of spaces.

The window always reflects the contents of the buffer (Vi does not do this when changing text and in some other cases).

If Vim is compiled with DIGRAPHS defined, digraphs are supported. A set of normal digraphs is included. They are shown with the ":digraph" command. More can be added with ":digraph {char1}{char2} {number}". A digraph is entered with "CTRL-K {char1} {char2}" or "{char1} BS {char2}" (only when '**digraph**' option is set).

When repeating an insert, e.g. "10atest <Esc>" vi would only handle wrapmargin for the first insert. Vim does it for all.

A count to the "i" or "a" command is used for all the text. Vi uses the count only for one line. "3iabc<NL>def<Esc>" would insert "abcabcabc<NL>def" in Vi but "abc<NL>defabc<NL>defabc<NL>def" in Vim.

In Command-line mode:

<Esc> terminates the command-line without executing it. In vi the command line would be executed, which is not what most people expect (hitting <Esc> should always get you back to command mode). To avoid problems with some obscure macros, an <Esc> in a macro will execute the command. If you want a typed <Esc> to execute the command like vi does you can fix this with
":cmap ^V<Esc> ^V<CR>"

General:

The '**timeout**' option is like '**timeoutlen**', but only works for cursor and function keys, not for ordinary mapped characters. The '**timeoutlen**' option gives the number of milliseconds that is waited for. If the '**esckeys**' option is not set, cursor and function keys that start with <Esc> are not recognized in insert mode.

There is an option for each terminal string. Can be used when termcap is not supported or to change individual strings.

The '**fileformat**' option can be set to select the <EOL>: "dos" <CR><NL>, "unix" <NL> or "mac" <CR>.

When the '**fileformats**' option is not empty, Vim tries to detect the type of <EOL> automatically. The '**fileformat**' option is set accordingly.

On systems that have no job control (older Unix systems and non-Unix systems)

the **CTRL-Z**, `:"stop"` or `:"suspend"` command starts a new shell.

If Vim is started on the Amiga without an interactive window for output, a window is opened (and `:sh` still works). You can give a device to use for editing with the `-d` argument, e.g. `-d con:20/20/600/150"`.

The **'columns'** and **'lines'** options are used to set or get the width and height of the display.

Option settings are read from the first and last few lines of the file. Option **'modelines'** determines how many lines are tried (default is 5). **Note** that this is different from the Vi versions that can execute any Ex command in a modeline (a major security problem). [trojan-horse](#)

If the **'insertmode'** option is set (e.g. in `.exerc`), Vim starts in insert mode. And it comes back there, when pressing `<Esc>`.

Undo information is kept in memory. Available memory limits the number and size of change that can be undone. This is hardly a problem on the Amiga and almost never with Unix and Win32.

If the **'backup'** or **'writebackup'** option is set: Before a file is overwritten, a backup file (`.bak`) is made. If the "backup" option is set it is left behind.

Vim creates a file ending in `.swp` to store parts of the file that have been changed or that do not fit in memory. This file can be used to recover from an aborted editing session with `"vim -r file"`. Using the swap file can be switched off by setting the **'updatecount'** option to 0 or starting Vim with the `"-n"` option. Use the **'directory'** option for placing the `.swp` file somewhere else.

Vim is able to work correctly on filesystems with 8.3 file names, also when using messydos or crossdos filesystems on the Amiga, or any 8.3 mounted filesystem under Unix. See **'shortname'** .

Error messages are shown at least one second (Vi overwrites error messages).

If Vim gives the **hit-enter** prompt, you can hit any key. Characters other than `<CR>`, `<NL>` and `<Space>` are interpreted as the (start of) a command. **{Vi: only ":" commands are interpreted}**

The contents of the numbered and unnamed registers is remembered when changing files.

The "No lines in buffer" message is a normal message instead of an error message, since that may cause a mapping to be aborted. **{Vi: error messages may be overwritten with other messages before you have a chance to read them}**

The AUX: device of the Amiga is supported.

=====
6. Supported Vi features vi-features

Vim supports nearly all Vi commands and mostly in the same way. That is when the **'compatible'** option is set and **'coptions'** contains all flags. What the effect is of resetting **'compatible'** and removing flags from **'coptions'** can be found at the help for the specific command.

The help files used to mark features that are in Vim but not in Vi with {not in Vi}. However, since these remarks cluttered the help files we now do it the other way around: Below is listed what Vi already supported. Anything else has been added by Vim.

The following Ex commands are supported by Vi:

```
`:abbreviate`   enter abbreviation
`:append`      append text
`:args`        print the argument list
`:cd`          change directory; Vi: no "cd -"
`:change`      replace a line or series of lines
`:chdir`       change directory
`:copy`        copy lines
`:delete`      delete lines
`:edit`        edit a file
`:exit`        same as `:xit`
`:file`        show or set the current file name; Vi: without the column number
`:global`      execute commands for matching lines
`:insert`      insert text
`:join`        join lines; Vi: not :join!
`:k`           set a mark
`:list`        print lines
`:map`         show or enter a mapping
`:mark`        set a mark
`:move`        move lines
`:Next`        go to previous file in the argument list {Vi: no count}
`:next`        go to next file in the argument list {Vi: no count}
`:number`      print lines with line number
`:open`        start open mode (not implemented in Vim)
`:pop`         jump to older entry in tag stack (only in some versions)
`:preserve`    write all text to swap file {Vi: might also exit}
`:previous`    same as `:Next` {Vi: only in some versions}
`:print`       print lines
`:put`         insert contents of register in the text
`:quit`        quit Vi
`:read`        read file into the text
`:recover`     recover a file from a swap file {Vi: recovers in another way
              and sends mail if there is something to recover}
`:rewind`      go to the first file in the argument list; no ++opt
`:set`         set option; but not `:set inv{option}`, `:set option&`,
              `:set all&`, `:set option+=value`, `:set option^=value`
              `:set option-=value` `:set option<`
`:shell`       escape to a shell
`:source`      read Vi or Ex commands from a file
`:stop`        suspend the editor or escape to a shell
`:substitute`  find and replace text; Vi: no '&', 'i', 's', 'r' or 'I' flag,
```

	confirm prompt only supports 'y' and 'n', no highlighting
<code>`:suspend`</code>	same as <code>":stop"</code>
<code>`:t`</code>	same as <code>":copy"</code>
<code>`:tag`</code>	jump to tag
<code>`:unabbreviate`</code>	remove abbreviation
<code>`:undo`</code>	undo last change {Vi: only one level}
<code>`:unmap`</code>	remove mapping
<code>`:vglobal`</code>	execute commands for not matching lines
<code>`:version`</code>	print version number and other info
<code>`:visual`</code>	same as <code>":edit"</code> , but turns off "Ex" mode
<code>`:wq`</code>	write to a file and quit Vi
<code>`:write`</code>	write to a file
<code>`:xit`</code>	write if buffer changed and quit Vi
<code>`:yank`</code>	yank lines into a register
<code>`:z`</code>	print some lines {not in all versions of Vi}
<code>`:!!`</code>	filter lines or execute an external command
<code>`: "`</code>	comment
<code>`: #`</code>	same as <code>":number"</code>
<code>`: *`</code>	execute contents of a register
<code>`: &`</code>	repeat last <code>":substitute"</code>
<code>`: <`</code>	shift lines one 'shiftwidth' left
<code>`: =`</code>	print the cursor line number
<code>`: >`</code>	shift lines one 'shiftwidth' right
<code>`: @`</code>	execute contents of a register; but not <code>`:@`</code> ; <code>`:@@`</code> only in some versions

Common for these commands is that Vi doesn't support the `++opt` argument on ``:edit`` and other commands that open a file.

The following Normal mode commands are supported by Vi:

note: See the beginning of [normal-index](#) for the meaning of WORD, N, Nmove and etc in the description text.

CTRL-B	scroll N screens Backwards
CTRL-C	interrupt current (search) command
CTRL-D	scroll Down N lines (default: half a screen); Vim scrolls 'scroll' screen lines, Vi scrolls file lines; makes a difference when lines wrap
CTRL-E	scroll N lines upwards (N lines Extra)
CTRL-F	scroll N screens Forward
CTRL-G	display current file name and position
<BS>	same as "h"
CTRL-H	same as "h"
<NL>	same as "j"
CTRL-J	same as "j"
CTRL-L	redraw screen
<CR>	cursor to the first CHAR N lines lower
CTRL-M	same as <CR>
CTRL-N	same as "j"
CTRL-P	same as "k"
CTRL-R	in some Vi versions: same as CTRL-L
CTRL-T	jump to N older Tag in tag list

CTRL-U N lines Upwards (default: half a screen) {Vi used file lines while Vim scrolls 'scroll' screen lines; makes a difference when lines wrap}

CTRL-Y scroll N lines downwards

CTRL-Z suspend program (or start new shell)

CTRL-] :ta to ident under cursor {Vi: identifier after the cursor}

CTRL-^ edit alternate file {Vi: no count}

<Space> same as "l"

! filter Nmove text through the {filter} command

!! filter N lines through the {filter} command

" use register {a-zA-Z0-9.%#:-"} for next delete, yank or put (uppercase to append) ({.%#:-} only work with put)

\$ cursor to the end of Nth next line

% find the next (curly/square) bracket on this line and go to its match, or go to matching comment bracket, or go to matching preprocessor directive (Vi: no count supported)

& repeat last :s

' jump to mark (Vi: only lowercase marks)

(cursor N sentences backward

) cursor N sentences forward

+ same as <CR>

, repeat latest f, t, F or T in opposite direction N times

- cursor to the first CHAR N lines higher

. repeat last change with count replaced with N

/ search forward for the Nth occurrence of {pattern}

0 cursor to the first char of the line

: start entering an Ex command

; repeat latest f, t, F or T N times

< shift Nmove lines one 'shiftwidth' leftwards

<< shift N lines one 'shiftwidth' leftwards

= filter Nmove lines through "indent"

== filter N lines through "indent"

> shift Nmove lines one 'shiftwidth' rightwards

>> shift N lines one 'shiftwidth' rightwards

? search backward for the Nth previous occurrence of {pattern}

@ execute the contents of register {a-z} N times
{Vi: only named registers}

@@ repeat the previous @[a-z] N times

A append text after the end of the line N times

B cursor N WORDS backward

C change from the cursor position to the end of the line

D delete the characters under the cursor until the end of the line and N-1 more lines [into register x]; synonym for "d\$"

E cursor forward to the end of WORD N

F cursor to the Nth occurrence of {char} to the left

G cursor to line N, default last line

H cursor to line N from top of screen

I insert text before the first CHAR on the line N times

J Join N lines; default is 2

L cursor to line N from bottom of screen

M cursor to middle line of screen

N repeat the latest '/' or '?' N times in opposite direction

O begin a new line above the cursor and insert text, repeat N times {Vi: blank [count] screen lines}

P put the text [from register x] before the cursor N times
 {Vi: no count}

Q switch to "Ex" mode

R enter replace mode: overwrite existing characters, repeat the entered text N-1 times

S delete N lines [into register x] and start insert; synonym for "cc".

T cursor till after Nth occurrence of {char} to the left

U undo all latest changes on one line
 {Vi: while not moved off of the last modified line}

W cursor N WORDS forward

X delete N characters before the cursor [into register x]

Y yank N lines [into register x]; synonym for "yy"

ZZ store current file if modified, and exit

[[cursor N sections backward

]] cursor N sections forward

^ cursor to the first CHAR of the line

_ cursor to the first CHAR N - 1 lines lower

` cursor to the mark {a-zA-Z0-9}

a append text after the cursor N times

b cursor N words backward

c delete Nmove text [into register x] and start insert

cc delete N lines [into register x] and start insert

d delete Nmove text [into register x]

dd delete N lines [into register x]

e cursor forward to the end of word N

f cursor to Nth occurrence of {char} to the right

h cursor N chars to the left

i insert text before the cursor N times

j cursor N lines downward

k cursor N lines upward

l cursor N chars to the right

m set mark {A-Za-z} at cursor position

n repeat the latest '/' or '?' N times

o begin a new line below the cursor and insert text
 {Vi: blank [count] screen lines}

p put the text [from register x] after the cursor N times
 {Vi: no count}

r replace N chars with {char} {Vi: CTRL-V <CR> still replaces with a line break, cannot replace something with a <CR>}

s (substitute) delete N characters [into register x] and start insert

t cursor till before Nth occurrence of {char} to the right

u undo changes {Vi: only one level}

w cursor N words forward

x delete N characters under and after the cursor [into register x]

y yank Nmove text [into register x]

yy yank N lines [into register x]

z<CR> current line to the top

z- current line to the bottom

z+ cursor on line N

z^ cursor on line N

{ cursor N paragraphs backward

	cursor to column N
}	cursor N paragraphs forward
~	switch case of N characters under the cursor; Vim: depends on 'tildeop' {Vi: no count, no 'tildeop'}
	same as "x"

The following commands are supported in Insert mode by Vi:

CTRL-@	insert previously inserted text and stop insert {Vi: only when typed as first char, only up to 128 chars}
CTRL-C	quit insert mode, without checking for abbreviation, unless 'insertmode' set.
CTRL-D	delete one shiftwidth of indent in the current line {Vi: CTRL-D works only when used after autoindent}
<BS>	delete character before the cursor {Vi: does not delete autoindents, does not cross lines, does not delete past start position of insert}
CTRL-H	same as <BS>
<Tab>	insert a <Tab> character
CTRL-I	same as <Tab>
<NL>	same as <CR>
CTRL-J	same as <CR>
<CR>	begin new line
CTRL-M	same as <CR>
CTRL-T	insert one shiftwidth of indent in current line {Vi: only when in indent}
CTRL-V {char}	insert next non-digit literally {Vi: no decimal byte entry}
CTRL-W	delete word before the cursor
CTRL-Z	when 'insertmode' set: suspend Vi
<Esc>	end insert mode (unless 'insertmode' set)
CTRL-[same as <Esc>
0 CTRL-D	delete all indent in the current line
^ CTRL-D	delete all indent in the current line, restore it in the next line
	delete character under the cursor

The following options are supported by Vi:

'autoindent'	'ai'	take indent for new line from previous line {Vi does this slightly differently: After the indent is deleted when typing <Esc> or <CR>, the cursor position when moving up or down is after the deleted indent; Vi puts the cursor somewhere in the deleted indent}.
'autowrite'	'aw'	automatically write file if changed
'directory'	'dir'	list of directory names for the swap file {Vi: directory to put temp file in, defaults to "/tmp"}
'edcompatible'	'ed'	toggle flags of ":substitute" command
'errorbells'	'eb'	ring the bell for error messages
'ignorecase'	'ic'	ignore case in search patterns
'lines'		number of lines in the display

'lisp'		automatic indenting for Lisp {Vi: Does it a little bit differently}
'list'		show <Tab> and <EOL>
'magic'		changes special characters in search patterns
'modeline'	'ml'	recognize 'modelines' at start or end of file {called modelines in some Vi versions}
'number'	'nu'	print the line number in front of each line
'paragraphs'	'para'	nrff macros that separate paragraphs
'prompt'	'prompt'	enable prompt in Ex mode
'readonly'	'ro'	disallow writing the buffer {Vim sets 'readonly' when editing a file with <code>`:view`</code> }
'remap'		allow mappings to work recursively
'report'		threshold for reporting nr. of lines changed
'scroll'	'scr'	lines to scroll with CTRL-U and CTRL-D
'sections'	'sect'	nrff macros that separate sections
'shell'	'sh'	name of shell to use for external commands
'shiftwidth'	'sw'	number of spaces to use for (auto)indent step
'showmatch'	'sm'	briefly jump to matching bracket if insert one
'showmode'	'smd'	message on status line to show current mode
'tabstop'	'ts'	number of spaces that <Tab> in file uses
'taglength'	'tl'	number of significant characters for a tag
'tags'	'tag'	list of file names used by the tag command {Vi: default is <code>"tags /usr/lib/tags"</code> }
'tagstack'	'tgst'	push tags onto the tag stack {not in all versions of Vi}
'term'		name of the terminal
'terse'		shorten some messages
'timeout'	'to'	time out on mappings and key codes
'timeoutlen'	'tm'	time for 'timeout' {only in some Vi versions}
'ttytype'	'tty'	alias for 'term'
'verbose'	'vbs'	give informative messages {only in some Vi versions as a boolean option}
'warn'		warn for shell command when buffer was changed
'window'	'wi'	nr of lines to scroll for CTRL-F and CTRL-B {Vi also uses the option to specify the number of displayed lines}
'wrapmargin'	'wm'	chars from the right where wrapping starts {Vi: works differently and less usefully}
'wrapscan'	'ws'	searches wrap around the end of the file
'writeany'	'wa'	write to file with no need for <code>"!"</code> override

Also see [missing-options](#) .

7. Command-line arguments

[cmdline-arguments](#)

Different versions of Vi have different command-line arguments. This can be confusing. To help you, this section gives an overview of the differences.

Five variants of Vi will be considered here:

Elvis	Elvis version 2.1b
Nvi	Nvi version 1.79
Posix	Posix 1003.2
Vi	Vi version 3.7 (for Sun 4.1.x)

Vile Vile version 7.4 (incomplete)
Vim Vim version 5.2

Only Vim is able to accept options in between and after the file names.

`+{command}` Elvis, Nvi, Posix, Vi, Vim: Same as "`-c {command}`".

`-` Nvi, Posix, Vi: Run Ex in batch mode.
Vim: Read file from stdin (use `-s` for batch mode).

`--` Vim: End of options, only file names are following.

`--cmd {command}` Vim: execute `{command}` before sourcing vimrc files.

`--echo-wid` Vim: GTK+ echoes the Window ID on stdout

`--help` Vim: show help message and exit.

`--literal` Vim: take file names literally, don't expand wildcards.

`--nofork` Vim: same as `-f`

`--noplugin[s]` Vim: Skip loading plugins.

`--remote` Vim: edit the files in another Vim server

`--remote-expr {expr}` Vim: evaluate `{expr}` in another Vim server

`--remote-send {keys}` Vim: send `{keys}` to a Vim server and exit

`--remote-silent {file}` Vim: edit the files in another Vim server if possible

`--remote-wait` Vim: edit the files in another Vim server and wait for it

`--remote-wait-silent` Vim: like `--remote-wait`, no complaints if not possible

`--role {role}` Vim: GTK+ 2: set role of main window

`--serverlist` Vim: Output a list of Vim servers and exit

`--servername {name}` Vim: Specify Vim server name

`--socketid {id}` Vim: GTK window socket to run Vim in

`--windowid {id}` Vim: Win32 window ID to run Vim in

`--version` Vim: show version message and exit.

`-?` Vile: print usage summary and exit.

`-a` Elvis: Load all specified file names into a window (use `-o` for Vim).

`-A` Vim: Start in Arabic mode (when compiled with Arabic).

-b {blksize} Elvis: Use {blksize} blocksize for the session file.
 -b Vim: set **'binary'** mode.

-C Vim: Compatible mode.

-c {command} Elvis, Nvi, Posix, Vim: run {command} as an Ex command after loading the edit buffer.
 Vim: allow up to 10 "-c" arguments

-d {device} Vim: Use {device} for I/O (Amiga only). {only when compiled without the +diff feature}
 -d Vim: start with **'diff'** set. vimdiff

-dev {device} Vim: Use {device} for I/O (Amiga only).

-D Vim: debug mode.

-e Elvis, Nvi, Vim: Start in Ex mode, as if the executable is called "ex".

-E Vim: Start in improved Ex mode gQ , like "exim".

-f Vim: Run GUI in foreground (Amiga: don't open new window).
 -f {session} Elvis: Use {session} as the session file.

-F Vim: Start in Farsi mode (when compiled with Farsi).
 Nvi: Fast start, don't read the entire file when editing starts.

-G {gui} Elvis: Use the {gui} as user interface.

-g Vim: Start GUI.
 -g N Vile: start editing at line N

-h Vim: Give help message.
 Vile: edit the help file

-H Vim: start Hebrew mode (when compiled with it).

-i Elvis: Start each window in Insert mode.
 -i {viminfo} Vim: Use {viminfo} for viminfo file.

-L Vim: Same as "-r" {only in some versions of Vi: "List recoverable edit sessions"}.

-l Nvi, Vi, Vim: Set **'lisp'** and **'showmatch'** options.

-m Vim: Modifications not allowed to be written, resets **'write'** option.

-M Vim: Modifications not allowed, resets **'modifiable'** and the **'write'** option.

-N Vim: No-compatible mode.

-n Vim: No swap file used.

-nb[args] Vim: open a NetBeans interface connection

-O[N] Vim: Like -o, but use vertically split windows.

-o[N] Vim: Open [N] windows, or one for each file.

-p[N] Vim: Open [N] tab pages, or one for each file.

-P {parent-title} Win32 Vim: open Vim inside a parent application window

-q {name} Vim: Use {name} for quickfix error file.
-q{name} Vim: Idem.

-R Elvis, Nvi, Posix, Vile, Vim: Set the 'readonly' option.

-r Elvis, Nvi, Posix, Vi, Vim: Recovery mode.

-S Nvi: Set 'secure' option.
-S {script} Vim: source script after starting up.

-s Nvi, Posix, Vim: Same as "-" (silent mode), when in Ex mode.
Elvis: Sets the 'safer' option.
-s {scriptin} Vim: Read from script file {scriptin}; only when not in Ex mode.
-s {pattern} Vile: search for {pattern}

-t {tag} Elvis, Nvi, Posix, Vi, Vim: Edit the file containing {tag}.
-t{tag} Vim: Idem.

-T {term} Vim: Set terminal name to {term}.

-u {vimrc} Vim: Read initializations from {vimrc} file.

-U {gvimrc} Vim: Read GUI initializations from {gvimrc} file.

-v Nvi, Posix, Vi, Vim: Begin in Normal mode (visual mode, in Vi terms).
Vile: View mode, no changes possible.

-V Elvis, Vim: Verbose mode.
-V{nr} Vim: Verbose mode with specified level.

-w {size} Elvis, Posix, Nvi, Vi, Vim: Set value of 'window' to {size}.
-w{size} Nvi, Vi: Same as "-w {size}".
-w {name} Vim: Write to script file {name} (must start with non-digit).

-W {name} Vim: Append to script file {name}.

-x Vi, Vim: Ask for encryption key. See [encryption](#) .

-X Vim: Don't connect to the X server.
-y Vim: Start in easy mode, like `evim` .
-Z Vim: restricted mode
@{cmdfile} Vile: use {cmdfile} as startup file.

8. POSIX compliance

`posix` `posix-compliance`

In 2005 the POSIX test suite was run to check the compatibility of Vim. Most of the test was executed properly. There are the few things where Vim is not POSIX compliant, even when run in Vi compatibility mode.

`$VIM_POSIX`

Set the `$VIM_POSIX` environment variable to have '`coptions`' include the POSIX flags when Vim starts up. This makes Vim run as POSIX as it can. That's a bit different from being Vi compatible.

You can find the Posix specification for Vi here:

<https://pubs.opengroup.org/onlinepubs/9699919799/utilities/vi.html>

And the related Ex specification:

<https://pubs.opengroup.org/onlinepubs/9699919799/utilities/ex.html>

This is where Vim does not behave as POSIX specifies and why:

`posix-screen-size`

The `$COLUMNS` and `$LINES` environment variables are ignored by Vim if the size can be obtained from the terminal in a more reliable way. Add the `|` flag to '`coptions`' to have `$COLUMNS` and `$LINES` overrule sizes obtained in another way.

The `"{"` and `"}"` commands don't stop at a `"{"` in the original Vi, but POSIX specifies it does. Add the `{` flag to '`coptions`' if you want it the POSIX way.

The `"D"`, `"o"` and `"O"` commands accept a count. Also when repeated. Add the `#` flag to '`coptions`' if you want to ignore the count.

The `":cd"` command fails if the current buffer is modified when the `.` flag is present in '`coptions`'.

There is no ATTENTION message, the `"A"` flag is added to '`shortmess`'.

These are remarks about running the POSIX test suite:

- vi test 33 sometimes fails for unknown reasons
- vi test 250 fails; behavior will be changed in a new revision
<http://www.opengroup.org/austin/mailarchives/ag-review/msg01710.html>
(link no longer works, perhaps it's now:
https://www.opengroup.org/sophocles/show_mail.tpl?CALLER=show_archive.tpl&source=L&list=ag-review)
- vi test 310 fails; exit code non-zero when any error occurred?
- ex test 24 fails because test is wrong. Changed between SUSv2 and SUSv3.
- ex tests 47, 48, 49, 72, 73 fail because `.exrc` file isn't read in silent mode and `$EXINIT` isn't used.

- ex tests 76, 78 fail because echo is used instead of printf. (fixed)
Also: problem with \s not changed to space.
- ex test 355 fails because 'window' isn't used for "30z".
- ex test 368 fails because shell command isn't echoed in silent mode.
- ex test 394 fails because "=" command output isn't visible in silent mode.
- ex test 411 fails because test file is wrong, contains stray ':'.
- ex test 475 and 476 fail because reprint output isn't visible in silent mode.
- ex test 480 and 481 fail because the tags file has spaces instead of a tab.
- ex test 502 fails because .exrc isn't read in silent mode.
- ex test 509 fails because .exrc isn't read in silent mode. and exit code is 1 instead of 2.
- ex test 534 fails because .exrc isn't read in silent mode.

VIM REFERENCE MANUAL by Bram Moolenaar

This document lists the incompatible differences between Vim 3.0 and Vim 4.0. Although 4.0 is mentioned here, this is also for version 4.1, 4.2, etc..

This file is important for everybody upgrading from Vim 3.0. Read it carefully to avoid unexpected problems.

' backup ' option default changed	backup-changed
Extension for backup file changed	backup-extension
Structure of swap file changed	swapfile-changed
"-w scriptout" argument changed	scriptout-changed
Backspace and Delete keys	backspace-delete
Escape for changed	escape-bar
Key codes changed	key-codes-changed
Terminal options changed	termcap-changed
' errorformat ' option changed	errorformat-changed
' graphic ' option gone	graphic-option-gone
' yankendofline ' option gone	ye-option-gone
' icon ' and ' title ' default value changed	icon-changed
' highlight ' option changed	highlight-changed
' tildeop ' and ' weirdinvert ' short names changed	short-name-changed
Use of "v", "V" and " CTRL-V " in Visual mode	use-visual-cmds
CTRL-B in Insert mode removed	toggle-revins

'**backup**' option default changed backup-changed

The default value for '**backup**' used to be on. This resulted in a backup file being made when the original file was overwritten.

Now the default for '**backup**' is off. As soon as the writing of the file has successfully finished, the backup file is deleted. If you want to keep the backup file, set '**backup**' on in your vimrc. The reason for this change is that many people complained that leaving a backup file behind is not Vi-compatible. '**backup**'

Extension for backup file changed backup-extension

The extension for the backup file used to be ".bak". Since other programs also use this extension and some users make copies with this extension, it was changed to the less obvious "~". Another advantage is that this takes less space, which is useful when working on a system with short file names. For example, on MS-DOS the backup files for "longfile.c" and "longfile.h" would both become "longfile.bak"; now they will be "longfile.c~" and "longfile.h~".

If you prefer to use ".bak", you can set the '**backupext**' option:

```
:set bex=.bak
```

Structure of swap file changed

swapfile-changed

The contents of the swap file were extended with several parameters. Vim stores the user name and other information about the edited file to make recovery more easy and to be able to know where the swap file comes from. The first part of the swap file can now be understood on a machine with a different byte order or sizeof(int). When you try to recover a file on such a machine, you will get an error message that this is not possible.

Because of this change, swap files cannot be exchanged between 3.0 and 4.0. If you have a swap file from a crashed session with 3.0, use Vim 3.0 to recover the file---don't use 4.0.

swap-file

"-w scriptout" argument changed

scriptout-changed

"vim -w scriptout" used to append to the scriptout file. Since this was illogical, it now creates a new file. An existing file is not overwritten (to avoid destroying an existing file for those who rely on the appending). [This was removed again later]

-w

Backspace and Delete keys

backspace-delete

In 3.0 both the delete key and the backspace key worked as a backspace in insert mode; they deleted the character to the left of the cursor. In 4.0 the delete key has a new function: it deletes the character under the cursor, just like it does on the command-line. If the cursor is after the end of the line and **'bs'** is set, two lines are joined.

 i_

In 3.0 the backspace key was always defined as **CTRL-H** and delete as **CTRL-?**. In 4.0 the code for the backspace and delete key is obtained from termcap or termLib, and adjusted for the "stty erase" value on Unix. This helps people who define the erase character according to the keyboard they are working on.

<BS> i_<BS>

If you prefer backspace and delete in Insert mode to have the old behavior, put this line in your vimrc:

```
inoremap ^? ^H
```

And you may also want to add these, to fix the values for **<BS>** and ****:

```
set t_kb=^H
set t_kD=^?
```

(Enter ^H with **CTRL-V CTRL-H** and ^? with **CTRL-V CTRL-?** or ****.)

If the value for `t_kb` is correct, but the `t_kD` value is not, use the `":fixdel"` command. It will set `t_kD` according to the value of `t_kb`. This is useful if you are using several different terminals. [:fixdel](#)

When `^H` is not recognized as `<BS>` or ``, it is used like a backspace.

Escape for | changed

[escape-bar](#)

When the `'b'` flag is present in `'coptions'`, the backslash cannot be used to escape `'|'` in mapping and abbreviate commands, only `CTRL-V` can. This is Vi-compatible. If you work in Vi-compatible mode and had used `"\|"` to include a bar in a mapping, this needs to be replaced by `"^V|"`. See [:bar](#).

Key codes changed

[key-codes-changed](#)

The internal representation of key codes has changed dramatically. In 3.0 a one-byte code was used to represent a key. This caused problems with different characters sets that also used these codes. In 4.0 a three-byte code is used that cannot be confused with a character. [key-notation](#)

If you have used the single-byte key codes in your `vimrc` for mappings, you will have to replace them with the 4.0 codes. Instead of using the three-byte code directly, you should use the symbolic representation for this in `<>`. See the table below. The table also lists the old name, as it was used in the 3.0 documentation.

The key names in `<>` can be used in mappings directly. This makes it possible to copy/paste examples or type them literally. The `<>` notation has been introduced for this `<>`. The `'B'` and `'<'` flags must not be present in `'coptions'` to enable this to work `'coptions'`.

old name	new name	old code		old MS-DOS code	
		hex	dec	hex	dec
<code><ESC></code>	<code><Esc></code>				
<code><TAB></code>	<code><Tab></code>				
<code><LF></code>	<code><NL></code> <code><NewLine></code> <code><LineFeed></code>				
<code><SPACE></code>	<code><Space></code>				
<code><NUL></code>	<code><Nul></code>				
<code><BELL></code>	<code><Bell></code>				
<code><BS></code>	<code><BS></code> <code><BackSpace></code>				
<code><INSERT></code>	<code><Insert></code>				
<code></code>	<code></code> <code><Delete></code>				
<code><HOME></code>	<code><Home></code>				
<code><END></code>	<code><End></code>				
<code><PAGE_UP></code>	<code><PageUp></code>				
<code><PAGE_DOWN></code>	<code><PageDown></code>				
<code><C_UP></code>	<code><Up></code>	0x80	128	0xb0	176
<code><C_DOWN></code>	<code><Down></code>	0x81	129	0xb1	177
<code><C_LEFT></code>	<code><Left></code>	0x82	130	0xb2	178

<C_RIGHT>	<Right>	0x83	131	0xb3	179
<SC_UP>	<S-Up>	0x84	132	0xb4	180
<SC_DOWN>	<S-Down>	0x85	133	0xb5	181
<SC_LEFT>	<S-Left>	0x86	134	0xb6	182
<SC_RIGHT>	<S-Right>	0x87	135	0xb7	183
<F1>	<F1>	0x88	136	0xb8	184
<F2>	<F2>	0x89	137	0xb9	185
<F3>	<F3>	0x8a	138	0xba	186
<F4>	<F4>	0x8b	139	0xbb	187
<F5>	<F5>	0x8c	140	0xbc	188
<F6>	<F6>	0x8d	141	0xbd	189
<F7>	<F7>	0x8e	142	0xbe	190
<F8>	<F8>	0x8f	143	0xbf	191
<F9>	<F9>	0x90	144	0xc0	192
<F10>	<F10>	0x91	145	0xc1	193
<SF1>	<S-F1>	0x92	146	0xc2	194
<SF2>	<S-F2>	0x93	147	0xc3	195
<SF3>	<S-F3>	0x94	148	0xc4	196
<SF4>	<S-F4>	0x95	149	0xc5	197
<SF5>	<S-F5>	0x96	150	0xc6	198
<SF6>	<S-F6>	0x97	151	0xc7	199
<SF7>	<S-F7>	0x98	152	0xc8	200
<SF8>	<S-F8>	0x99	153	0xc9	201
<SF9>	<S-F9>	0x9a	154	0xca	202
<SF10>	<S-F10>	0x9b	155	0xcb	203
<HELP>	<Help>	0x9c	156	0xcc	204
<UNDO>	<Undo>	0x9d	157	0xcd	205
	(not used)	0x9e	158	0xce	206
	(not used)	0x9f	159	0xcf	207

Terminal options changed

termcap-changed

The names of the terminal options have been changed to match the termcap names of these options. All terminal options now have the name `t_xx`, where `xx` is the termcap name. Normally these options are not used, unless you have a termcap entry that is wrong or incomplete, or you have set the highlight options to a different value.

terminal-options

Note that for some keys there is no termcap name. Use the `<>` type of name instead, which is a good idea anyway.

Note that `"t_ti"` has become `"t_mr"` (invert/reverse output) and `"t_ts"` has become `"t_ti"` (init terminal mode). Be careful when you use `"t_ti"`!

old name	new name	meaning	
<code>t_cdl</code>	<code>t_DL</code>	delete number of lines	<code>t_cdl</code>
<code>t_ci</code>	<code>t_vi</code>	cursor invisible	<code>t_ci</code>
<code>t_cil</code>	<code>t_AL</code>	insert number of lines	<code>t_cil</code>

t_cm		t_cm	move cursor	
t_cri		t_RI	cursor number of chars right	t_cri
t_cv		t_ve	cursor visible	t_cv
t_cvv		t_vs	cursor very visible	t_cvv
t_dl		t_dl	delete line	
t_cs		t_cs	scroll region	
t_ed		t_cl	clear display	t_ed
t_el		t_ce	clear line	t_el
t_il		t_al	insert line	t_il
		t_da	display may be retained above the screen	
		t_db	display may be retained below the screen	
t_ke		t_ke	put terminal out of keypad transmit mode	
t_ks		t_ks	put terminal in keypad transmit mode	
t_ms		t_ms	save to move cursor in highlight mode	
t_se		t_se	normal mode (undo t_so)	
t_so		t_so	shift out (standout) mode	
t_ti		t_mr	reverse highlight	
t_tb		t_md	bold mode	t_tb
t_tp		t_me	highlight end	t_tp
t_sr		t_sr	scroll reverse	
t_te		t_te	out of termcap mode	
t_ts		t_ti	into termcap mode	t_ts_old
t_vb		t_vb	visual bell	
t_csc		t_CS	cursor is relative to scroll region	t_csc
t_ku	t_ku	<Up>	arrow up	
t_kd	t_kd	<Down>	arrow down	
t_kr	t_kr	<Right>	arrow right	
t_kl	t_kl	<Left>	arrow left	
t_sku		<S-Up>	shifted arrow up	t_sku
t_skd		<S-Down>	shifted arrow down	t_skd
t_skr	t_%i	<S-Right>	shifted arrow right	t_skr
t_skL	t_#4	<S-Left>	shifted arrow left	t_skL
t_f1	t_k1	<F1>	function key 1	t_f1
t_f2	t_k2	<F2>	function key 2	t_f2
t_f3	t_k3	<F3>	function key 3	t_f3
t_f4	t_k4	<F4>	function key 4	t_f4
t_f5	t_k5	<F5>	function key 5	t_f5
t_f6	t_k6	<F6>	function key 6	t_f6
t_f7	t_k7	<F7>	function key 7	t_f7
t_f8	t_k8	<F8>	function key 8	t_f8
t_f9	t_k9	<F9>	function key 9	t_f9
t_f10	t_k;	<F10>	function key 10	t_f10
t_sf1		<S-F1>	shifted function key 1	t_sf1
t_sf2		<S-F2>	shifted function key 2	t_sf2
t_sf3		<S-F3>	shifted function key 3	t_sf3
t_sf4		<S-F4>	shifted function key 4	t_sf4
t_sf5		<S-F5>	shifted function key 5	t_sf5
t_sf6		<S-F6>	shifted function key 6	t_sf6
t_sf7		<S-F7>	shifted function key 7	t_sf7
t_sf8		<S-F8>	shifted function key 8	t_sf8
t_sf9		<S-F9>	shifted function key 9	t_sf9
t_sf10		<S-F10>	shifted function key 10	t_sf10
t_help	t_%1	<Help>	help key	t_help

t_undo t_&8 <Undo> undo key t_undo

'errorformat' option changed errorformat-changed

'errorformat' can now contain several formats, separated by commas. The first format that matches is used. The default values have been adjusted to catch the most common formats. errorformat

If you have a format that contains a comma, it needs to be preceded with a backslash. Type two backslashes, because the ":set" command will eat one.

'graphic' option gone graphic-option-gone

The **'graphic'** option was used to make the characters between <~> and 0xa0 display directly on the screen. Now the **'isprint'** option takes care of this with many more possibilities. The default setting is the same; you only need to look into this if you previously set the **'graphic'** option in your vimrc. 'isprint'

'yankendofline' option gone ye-option-gone

The **'yankendofline'** option has been removed. Instead you can just use
:map Y y\$

'icon' and **'title'** default value changed icon-changed

The **'title'** option is now only set by default if the original title can be restored. Avoids "Thanks for flying Vim" titles. If you want them anyway, put ":set title" in your vimrc. 'title'

The default for **'icon'** now depends on the possibility of restoring the original value, just like **'title'**. If you don't like your icon titles to be changed, add this line to your vimrc: 'icon'
:set noicon

'highlight' option changed highlight-changed

The 'i' flag now means italic highlighting, instead of invert. The 'r' flag is used for reverse highlighting, which is what 'i' used to be. Normally you won't see the difference, because italic mode is not supported on most terminals and reverse mode is used as a fallback. 'highlight'

When an occasion is not present in **'highlight'**, use the mode from the default value for **'highlight'**, instead of reverse mode.

'tildeop' and **'weirdinvert'** short names changed

short-name-changed

Renamed **'to'** (abbreviation for **'tildeop'**) to **'top'**.

'tildeop'

Renamed **'wi'** (abbreviation for **'weirdinvert'**) to **'wiv'**.

'weirdinvert'

This was done because Vi uses **'wi'** as the short name for **'window'** and **'to'** as the short name for **'timeout'**. This means that if you try setting these options, you won't get an error message, but the effect will be different.

Use of **"v"**, **"V"** and **"CTRL-V"** in Visual mode

use-visual-cmds

In Visual mode, **"v"**, **"V"**, and **"CTRL-V"** used to end Visual mode. Now this happens only if the Visual mode was in the corresponding type. Otherwise the type of Visual mode is changed. Now only ESC can be used in all circumstances to end Visual mode without doing anything.

v_V

CTRL-B in Insert mode removed

toggle-revins

CTRL-B in Insert mode used to toggle the **'revins'** option. If you don't know this and accidentally hit **CTRL-B**, it is very difficult to find out how to undo it. Since hardly anybody uses this feature, it is disabled by default. If you want to use it, define RIGHTLEFT in feature.h before compiling. **'revins'**

VIM REFERENCE MANUAL by Bram Moolenaar

Welcome to Vim Version 5.0!

This document lists the differences between Vim 4.x and Vim 5.0. Although 5.0 is mentioned here, this is also for version 5.1, 5.2, etc. See [vi_diff.txt](#) for an overview of differences between Vi and Vim 5.0. See [version4.txt](#) for differences between Vim 3.0 and Vim 4.0.

INCOMPATIBLE:	incompatible-5
Default value for 'compatible' changed	cp-default
Text formatting command "Q" changed	Q-command-changed
Command-line arguments changed	cmdline-changed
Autocommands are kept	autocmds-kept
Use of 'hidden' changed	hidden-changed
Text object commands changed	text-objects-changed
X-Windows Resources removed	x-resources
Use of \$VIM	\$VIM-use
Use of \$HOME for MS-DOS and Win32	\$HOME-use
Tags file format changed	tags-file-changed
Options changed	options-changed
CTRL-B in Insert mode gone	i_CTRL-B-gone
NEW FEATURES:	new-5
Syntax highlighting	new-highlighting
Built-in script language	new-script
Perl and Python support	new-perl-python
Win32 GUI version	added-win32-GUI
VMS version	added-VMS
BeOS version	added-BeOS
Macintosh GUI version	added-Mac
More Vi compatible	more-compatible
Read input from stdin	read-stdin
Regular expression patterns	added-regexp
Overloaded tags	tag-overloaded
New commands	new-commands
New options	added-options
New command-line arguments	added-cmdline-args
Various additions	added-various
IMPROVEMENTS	improvements-5
COMPILE TIME CHANGES	compile-changes-5
BUG FIXES	bug-fixes-5
VERSION 5.1	version-5.1
Changed	changed-5.1
Added	added-5.1

Fixed	fixed-5.1
VERSION 5.2	version-5.2
Long lines editable	long-lines
File browser added	file-browser-5.2
Dialogs added	dialogs-added
Popup menu added	popup-menu-added
Select mode added	new-Select-mode
Session files added	new-session-files
User defined functions and commands	new-user-defined
New interfaces	interfaces-5.2
New ports	ports-5.2
Multi-byte support	new-multi-byte
New functions	new-functions-5.2
New options	new-options-5.2
New Ex commands	new-ex-commands-5.2
Changed	changed-5.2
Added	added-5.2
Fixed	fixed-5.2
VERSION 5.3	version-5.3
Changed	changed-5.3
Added	added-5.3
Fixed	fixed-5.3
VERSION 5.4	version-5.4
Runtime directory introduced	new-runtime-dir
Filetype introduced	new-filetype-5.4
Vim script line continuation	new-line-continuation
Improved session files	improved-sessions
Autocommands improved	improved-autocmds-5.4
Encryption	new-encryption
GTK GUI port	new-GTK-GUI
Menu changes	menu-changes-5.4
Viminfo improved	improved-viminfo
Various new commands	new-commands-5.4
Various new options	new-options-5.4
Vim scripts	new-script-5.4
Avoid hit-enter prompt	avoid-hit-enter
Improved quickfix	improved-quickfix
Regular expressions	regexp-changes-5.4
Changed	changed-5.4
Added	added-5.4
Fixed	fixed-5.4
VERSION 5.5	version-5.5
Changed	changed-5.5
Added	added-5.5
Fixed	fixed-5.5
VERSION 5.6	version-5.6
Changed	changed-5.6
Added	added-5.6
Fixed	fixed-5.6

VERSION 5.7	version-5.7
Changed	changed-5.7
Added	added-5.7
Fixed	fixed-5.7

VERSION 5.8	version-5.8
Changed	changed-5.8
Added	added-5.8
Fixed	fixed-5.8

INCOMPATIBLE

incompatible-5

Default value for **'compatible'** changed

cp-default

Vim version 5.0 tries to be more Vi compatible. This helps people who use Vim as a drop-in replacement for Vi, but causes some things to be incompatible with version 4.x.

In version 4.x the default value for the **'compatible'** option was off. Now the default is on. The first thing you will notice is that the "u" command undoes itself. Other side effects will be that mappings may work differently or not work at all.

Since a lot of people switching from Vim 4.x to 5.0 will find this annoying, the **'compatible'** option is switched off if Vim finds a vimrc file. This is a bit of magic to make sure that 90% of the Vim users will not be bitten by this change.

What does this mean?

- If you prefer to run in **'compatible'** mode and don't have a vimrc file, you don't have to do anything.
- If you prefer to run in **'nocompatible'** mode and do have a vimrc file, you don't have to do anything.
- If you prefer to run in **'compatible'** mode and do have a vimrc file, you should put this line first in your vimrc file:
:set compatible
- If you prefer to run in **'nocompatible'** mode and don't have a vimrc file, you can do one of the following:
 - Create an empty vimrc file (e.g.: "~/vimrc" for Unix).
 - Put this command in your .exrc file or \$EXINIT:
:set nocompatible
- Start Vim with the "-N" argument.

If you are new to Vi and Vim, using **'nocompatible'** is strongly recommended, because Vi has a lot of unexpected side effects, which are avoided by this setting. See **'compatible'**.

If you like some things from **'compatible'** and some not, you can tune the compatibility with **'coptions'**.

When you invoke Vim as "ex" or "gex", Vim always starts in compatible mode.

Text formatting command "Q" changed

Q-command-changed

The "Q" command formerly formatted lines to the width the **'textwidth'** option specifies. The command for this is now "gq" (see [gq](#) for more info). The reason for this change is that "Q" is the standard Vi command to enter "Ex" mode, and Vim now does in fact have an "Ex" mode (see [Q](#) for more info).

If you still want to use "Q" for formatting, use this mapping:

```
:noremap Q gq
```

And if you also want to use the functionality of "Q":

```
:noremap gQ Q
```

Command-line arguments changed

cmdline-changed

Command-line file-arguments and option-arguments can now be mixed. You can give options after the file names. Example:

```
vim main.c -g
```

This is not possible when editing a file that starts with a '-'. Use the "--" argument then --- :

```
vim -g -- -main.c
```

"-v" now means to start Ex in Vi mode, use "-R" for read-only mode.

```
old: "vim -v file"      -v
new: "vim -R file"     -R
```

"-e" now means to start Vi in Ex mode, use "-q" for quickfix.

```
old: "vim -e errorfile" -e
new: "vim -q errorfile" -q
```

"-s" in Ex mode now means to run in silent (batch) mode. [-s-ex](#)

"-x" reserved for crypt, use "-f" to avoid starting a new CLI (Amiga).

```
old: "vim -x file"      -x
new: "vim -f file"     -f
```

Vim allows up to ten "+cmd" and "-c cmd" arguments. Previously Vim executed only the last one.

"-n" now overrides any setting for **'updatecount'** in a vimrc file, but not in a gvimrc file.

Autocommands are kept

autocmds-kept

Before version 5.0, autocommands with the same event, file name pattern, and command could appear only once. This was fine for simple autocommands (like setting option values), but for more complicated autocommands, where the same

command might appear twice, this restriction caused problems. Therefore Vim stores all autocommands and keeps them in the order that they are defined.

The most obvious side effect of this change is that when you source a vimrc file twice, the autocommands in it will be defined twice. To avoid this, do one of these:

- Remove any autocommands that might already be defined before defining them. Example:

```
:au! * *.ext
:au BufEnter *.ext ...
```

- Put the autocommands inside an ":if" command. Example:

```
if !exists("did_ext_autocmds")
  let did_ext_autocmds = 1
  autocmd BufEnter *.ext ...
endif
```

- Put your autocommands in a different autocommand group so you can remove them before defining them `:augroup` :

```
augroup uncompress
  au!
  au BufReadPost *.gz ...
augroup END
```

Use of **'hidden'** changed

hidden-changed

In version 4.x, only some commands used the **'hidden'** option. Now all commands uses it whenever a buffer disappears from a window.

Previously you could do `":buf xxx"` in a changed buffer and that buffer would then become hidden. Now you must set the **'hidden'** option for this to work.

The new behavior is simpler: whether Vim hides buffers no longer depends on the specific command that you use.

- with **'hidden'** not set, you never get hidden buffers. Exceptions are the `":hide"` and `":close!"` commands and, in rare cases, where you would otherwise lose changes to the buffer.
- With **'hidden'** set, you almost never unload a buffer. Exceptions are the `":bunload"` or `":bdel"` commands.

`":buffer"` now supports a `"!":` abandon changes in current buffer. So do `":bnext"`, `":brewind"`, etc.

Text object commands changed

text-objects-changed

Text object commands have new names. This allows more text objects and makes characters available for other Visual mode commands. Since no more single characters were available, text objects names now require two characters. The first one is always `'i'` or `'a'`.

OLD	NEW		
a	aw	a word	v_aw
A	aW	a WORD	v_aW
s	as	a sentence	v_as
p	ap	a paragraph	v_ap
S	ab	a () block	v_ab
P	aB	a {} block	v_aB

There is another set of text objects that starts with "i", for "inner". These select the same objects, but exclude white space.

X-Windows Resources removed

[x-resources](#)

Vim no longer supports the following X resources:

- boldColor
- italicColor
- underlineColor
- cursorColor

Vim now uses highlight groups to set colors. This avoids the confusion of using a bold Font, which would imply a certain color. See [:highlight](#) and [gui-resources](#) .

Use of \$VIM

[\\$VIM-use](#)

Vim now uses the VIM environment variable to find all Vim system files. This includes the global vimrc, gvimrc, and menu.vim files and all on-line help and syntax files. See [\\$VIM](#) . Starting with version 5.4, [\\$VIMRUNTIME](#) can also be used.

For Unix, Vim sets a default value for \$VIM when doing "make install". When \$VIM is not set, its default value is the directory from **'helpfile'**, excluding "/doc/help.txt".

Use of \$HOME for MS-DOS and Win32

[\\$HOME-use](#)

The MS-DOS and Win32 versions of Vim now first check \$HOME when searching for a vimrc or exrc file and for reading/storing the viminfo file. Previously Vim used \$VIM for these systems, but this causes trouble on a system with several users. Now Vim uses \$VIM only when \$HOME is not set or the file is not found in \$HOME. See [_vimrc](#) .

Tags file format changed

[tags-file-changed](#)

Only tabs are allowed to separate fields in a tags file. This allows for spaces in a file name and is still Vi compatible. In previous versions of Vim, any white space was allowed to separate the fields. If you have a file

which doesn't use a single tab between fields, edit the tags file and execute this command:

```
:%s/\(\S*\)\s\+\(\S*\)\s\+\(.*\)/\1\t2\t3/
```

Options changed

options-changed

The default value of **'errorfile'** has changed from "errors.vim" to "errors.err". The reason is that only Vim scripts should have the ".vim" extensions.

The ":make" command no longer uses the **'errorfile'** option. This prevents the output of the ":make" command from overwriting a manually saved error file. ":make" uses the **'makeef'** option instead. This also allows for generating a unique name, to prevent concurrently running ":make" commands from overwriting each other's files.

With **'insertmode'** set, a few more things change:

- <Esc> in Normal mode goes to Insert mode.
- <Esc> in Insert mode doesn't leave Insert mode.
- When doing ":set im", go to Insert mode immediately.

Vim considers a buffer to be changed when the **'fileformat'** (formerly the **'textmode'** option) is different from the buffer's initial format.

CTRL-B in Insert mode gone

i_CTRL-B-gone

When Vim was compiled with the **+rightleft** feature, you could use **CTRL-B** to toggle the **'revins'** option. Unfortunately, some people hit the 'B' key accidentally when trying to type **CTRL-V** or **CTRL-N** and then didn't know how to undo this. Since toggling the **'revins'** option can easily be done with the mapping below, this use of the **CTRL-B** key is disabled. You can still use the CTRL_ key for this **i_CTRL_**.

```
:imap <C-B> <C-O>:set revins!<CR>
```

NEW FEATURES

new-5

Syntax highlighting

new-highlighting

Vim now has a very flexible way to highlighting just about any type of file. See **syntax**. Summary:

```
:syntax on
```

Colors and attributes can be set for the syntax highlighting, and also for other highlighted items with the ':' flag in the **'highlight'** option. All highlighted items are assigned a highlight group which specifies their highlighting. See **:highlight**. The default colors have been improved.

You can use the "Normal" group to set the default fore/background colors for a color terminal. For the GUI, you can use this group to specify the font, too.

The "2html.vim" script can be used to convert any file that has syntax highlighting to HTML. The colors will be exactly the same as how you see them in Vim. With a HTML viewer you can also print the file with colors.

Built-in script language

[new-script](#)

A few extra commands and an expression evaluator enable you to write simple but powerful scripts. Commands include ":if" and ":while". Expressions can manipulate numbers and strings. You can use the '=' register to insert directly the result of an expression. See [expression](#) .

Perl and Python support

[new-perl-python](#)

Vim can call Perl commands with ":perldo", ":perl", etc. See [perl](#) .
Patches made by Sven Verdoolaege and Matt Gerassimoff.

Vim can call Python commands with ":python" and ":pyfile". See [python](#) .

Both of these are only available when enabled at compile time.

Win32 GUI version

[added-win32-GUI](#)

The GUI has been ported to MS-Windows 95 and NT. All the features of the X11 GUI are available to Windows users now. [gui-w32](#)
This also fixes problems with running the Win32 console version under Windows 95, where console support has always been bad.
There is also a version that supports OLE automation interface. [if_ole.txt](#)
Vim can be integrated with Microsoft Developer Studio using the VisVim DLL.
It is possible to produce a DLL version of gvim with Borland C++ (Aaron).

VMS version

[added-VMS](#)

Vim can now also be used on VMS systems. Port done by Henk Elbers.
This has not been tested much, but it should work.
Sorry, no documentation!

BeOS version

[added-BeOS](#)

Vim can be used on BeOS systems (including the BeBox). (Olaf Seibert)
See [os_beos.txt](#) .

Macintosh GUI version

[added-Mac](#)

Vim can now be used on the Macintosh. (Dany St-Amant)
It has not been tested much yet, be careful!
See `os_mac.txt` .

More Vi compatible

[more-compatible](#)

There is now a real Ex mode. Started with the "Q" command, or by calling the executable "ex" or "gex". [Ex-mode](#)

Always allow multi-level undo, also in Vi compatible mode. When the 'u' flag in '[cptions](#)' is included, **CTRL-R** is used for repeating the undo or redo (like "." in Nvi).

Read input from stdin

[read-stdin](#)

When using the "-" command-line argument, Vim reads its text input from stdin. This can be used for putting Vim at the end of a pipe:

```
grep "^a.*" *.c | vim -  
See -- .
```

Regular expression patterns

[added-regexp](#)

Added specifying a range for the number of matches of an atom: "\{a,b}". [/\{](#)
Added the "shortest match" regexp "\{-}" (Webb).
Added "\s", matches a white character. Can replace "[\t]". [/\s](#)
Added "\S", matches a non-white character. Can replace "[^ \t]". [/\S](#)

Overloaded tags

[tag-overloaded](#)

When using a language like C++, there can be several tags for the same tagname. Commands have been added to be able to jump to any of these overloaded tags:

```
:tselect      List matching tags, and jump to one of them.  
:stselect     Idem, and split window.  
g_CTRL-]     Do ":tselect" with the word under the cursor.
```

After ":ta {tagname}" with multiple matches:

```
:tnext        Go to next matching tag.  
:tprevious    Go to previous matching tag.  
:trewind      Go to first matching tag.  
:tlast        Go to last matching tag.
```

The ":tag" command now also accepts wildcards. When doing command-line completion on tags, case-insensitive matching is also available (at the end).

New commands

new-commands

<code>:amenu</code>	Define menus for all modes, inserting a CTRL-O for Insert mode, ESC for Visual and CTRL-C for Cmdline mode. "amenu" is used for the default menus and the Syntax menu.
<code>:augroup</code>	Set group to be used for following autocommands. Allows the grouping of autocommands to enable deletion of a specific group.
<code>:crewind</code>	Go to first error.
<code>:clast</code>	Go to last error.
<code>:doautoall</code>	Execute autocommands for all loaded buffers.
<code>:echo</code>	Echo its argument, which is an expression. Can be used to display messages which include variables.
<code>:execute</code>	Execute its argument, which is an expression. Can be used to built up an Ex command with anything.
<code>:hide</code>	Works like ":close".
<code>:if</code>	Conditional execution, for built-in script language.
<code>:intro</code>	Show introductory message. This is always executed when Vim is started without file arguments.
<code>:let</code>	Assign a value to an internal variable.
<code>:omap</code>	Map only in operator-pending mode. Makes it possible to map text-object commands.
<code>:redir</code>	Redirect output of messages to a file.
<code>:update</code>	Write when buffer has changed.
<code>:while</code>	While-loop for built-in script language.

Visual mode:

<code>v_0</code>	"0" in Visual block mode, moves the cursor to the other corner horizontally.
<code>v_D</code>	"D" in Visual block mode deletes till end of line.

Insert mode:

<code>i_CTRL-]</code>	Triggers abbreviation, without inserting any character.
-----------------------	---

New options

added-options

'background'	Used for selecting highlight color defaults. Also used in "syntax.vim" for selecting the syntax colors. Often set automatically, depending on the terminal used.
'complete'	Specifies how Insert mode completion works.
'eventignore'	Makes it possible to ignore autocommands temporarily.
'fileformat'	Current file format. Replaces 'textmode' .
'fileformats'	Possible file formats. Replaces 'textauto' . New is that this also supports Macintosh format: A single <CR> separates lines. The default for 'fileformats' for MS-DOS, Win32 and OS/2 is "dos,unix", also when 'compatible' set. Unix type files didn't work anyway when 'fileformats' was empty.
'guicursor'	Set the cursor shape and blinking in various modes. Default is to adjust the cursor for Insert and Replace mode, and when an operator is pending. Blinking is default on.
'fkmap'	Farsi key mapping.
'hlsearch'	Highlight all matches with the last used search pattern.
'hkmap'	Phonetic Hebrew mapping. (Ilya Dogolazky)
'iconstring'	Define the name of the icon, when not empty. (Version 5.2: the string is used literally, a newline can be used to make two lines.)
'lazyredraw'	Don't redraw the screen while executing macros, registers or other not typed commands.
'makeef'	Errorfile to be used for ":make". "##" is replaced with a unique number. Avoids that two Vim sessions overwrite each others errorfile. The Unix default is "/tmp/vim##.err"; for Amiga "t:vim##.Err, for others "vim##.err".
'matchtime'	1/10s of a second to show a matching paren, when 'showmatch' is set. Like Nvi.
'mousehide'	Hide mouse pointer in GUI when typing text.
'nrformats'	Defines what bases Vim will consider for numbers when using the CTRL-A and CTRL-X commands. Default: "hex,octal".
'shellxquote'	Add extra quotes around the whole shell command, including redirection.
'softtabstop'	Make typing behave like tabstop is set at this value, without changing the value of 'tabstop' . Makes it more easy to keep 'ts' at 8, while still getting four spaces for a <Tab>.
'titlestring'	String for the window title, when not empty. (Version 5.2:

this string is used literally, a newline can be used to make two lines.)

'verbose' Level of verbosity. Makes it possible to show which .vimrc, .exrc, .viminfo files etc. are used for initializing. Also to show autocommands that are being executed. Can also be set by using the "-V" command-line argument.

New command-line arguments

added-cmdline-args

- U Set the gvimrc file to be used. Like "-u" for the vimrc.
- V Set the **'verbose'** option. E.g. "vim -V10".
- N Start in non-compatible mode.
- C Start in compatible mode.
- Z Start in restricted mode, disallow shell commands. Can also be done by calling the executable "rvim".
- h Show usage information and exit.

Various additions

added-various

Added support for SNIFF+ connection (submitted by Toni Leherbauer). Vim can be used as an editor for SNIFF. No documentation available...

For producing a bug report, the bugreport.vim script has been included. Can be used with ":so \$VIMRUNTIME/bugreport.vim", which creates the file "bugreport.txt" in the current directory. [bugs](#)

Added range to ":normal" command. Now you can repeat the same command for each line in the range. [:normal-range](#)

Included support for the Farsi language (Shiran). Only when enabled at compile time. See [farsi](#) .

IMPROVEMENTS

improvements-5

Performance:

- When **'showcmd'** was set, mappings would execute much more slowly because the output would be flushed very often. Helps a lot when executing the "life" macros with **'showcmd'** set.
- Included patches for binary searching in tags file (David O'Neill). Can be disabled by resetting the **'tagbsearch'** option.
- Don't update the ruler when repeating insert (slowed it down a lot).
- For Unix, file name expansion is now done internally instead of starting a shell for it.

- Expand environment variables with `expand_env()`, instead of calling the shell. Makes `":so $VIMRUNTIME/syntax/syntax.vim"` a LOT faster.
- Reduced output for cursor positioning: Use CR-LF for moving to first few columns in next few lines; Don't output CR twice when using `termios`.
- Optimized cursor positioning. Use CR, BS and NL when it's shorter than absolute cursor positioning.
- Disable redrawing while repeating insert `"1000ii<Esc>"`.
- Made `"d$"` or `"D"` for long lines a lot faster (delete all characters at once, instead of one by one).
- Access option table by first letter, instead of searching from start.
- Made setting special highlighting attributes a lot faster by using `highlight_attr[]`, instead of searching in the `'highlight'` string.
- Don't show the mode when redrawing is disabled.
- When setting an option, only redraw the screen when required.
- Improved performance of Ex commands by using a lookup table for the first character.

Options:

- `'cinoptions'` Added 'g' flag, for C++ scope declarations.
- `'cptions'` Added 'E' flag: Disallow yanking, deleting, etc. empty text area. Default is to allow empty yanks. When 'E' is included, `"y$"` in an empty line now is handled as an error (Vi compatible).
 - Added 'j' flag: Only add two spaces for a join after a '.', not after a '?' or '!'.
 - Added 'A' flag: don't give ATTENTION message.
 - Added 'L' flag: When not included, and `'list'` is set, `'textwidth'` formatting works like `'list'` is not set.
 - Added 'W' flag: Let `":w!"` behave like Vi: don't overwrite readonly files, or a file owned by someone else.
- `'highlight'` Added '@' flag, for '@' characters after the last line on the screen, and '\$' at the end of the line when `'list'` is set.
 - Added 'i' flag: Set highlighting for `'incsearch'`. Default uses "IncSearch" highlight group, which is linked to "Visual". Disallow 'h' flag in `'highlight'` (wasn't used anymore since 3.0).
- `'guifont'` Win32 GUI only: When set to "*" brings up a font requester.
- `'guipty'` Default on, because so many people need it.
- `'path'` Can contain wildcards, and "*" for searching a whole tree.
- `'shortmess'` Added 'I' flag to avoid the intro message.
- `'vminfo'` Added '%' flag: Store buffer list in viminfo file.

- Increased defaults for `'maxmem'` and `'maxmemtot'` for Unix and Win32. Most machines have much more RAM now that prices have dropped.
- Implemented `":set all&"`, set all options to their default value. `:set`

Swap file:

- Don't create a swap file for a readonly file. Then create one on the first change. Also create a swapfile when the amount of memory used is getting too high. `swap-file`
- Make swap file "hidden", if possible. On Unix this is done by prepending a dot to the swap file name. When long file names are used, the DJGPP and Win32 versions also prepend a dot, in case a file on a mounted Unix file system is edited. `:swapname` On MSDOS the hidden file attribute is NOT

- set, because this causes problems with share.exe.
- **'updatecount'** always defaults to non-zero, also for Vi compatible mode. This means there is a swap file, which can be used for recovery.

Tags:

- Included ctags 2.0 (Darren Hiebert). The syntax for static tags changed from


```
{tag}:{fname} {fname} {command}
```

 to


```
{tag} {fname} {command};" file:
```

 Which is both faster to parse, shorter and Vi compatible. The old format is also still accepted, unless disabled in src/feature.h (see OLD_STATIC_TAGS).
 [tags-file-format](#)
- Completion of tags now also includes static tags for other files, at the end.
- Included "shtags" from Stephen Riehm.
- When finding a matching tag, but the file doesn't exist, continue searching for another match. Helps when using the same tags file (with links) for different versions of source code.
- Give a tag with a global match in the current file a higher priority than a global match in another file.

Included xxd version V1.8 (Juergen Weigert).

Autocommands:

- VimLeave autocommands are executed after writing the viminfo file, instead of before. [VimLeave](#)
- Allow changing autocommands while executing them. This allows for self-modifying autocommands. (idea from Goldberg)
- When using autocommands with two or more patterns, could not split ":if/endif" over two lines. Now all matching autocommands are executed in one do_cmdline().
- Autocommands no longer change the command repeated with ".".
- Search patterns are restored after executing autocommands. This avoids that the **'hlsearch'** highlighting is messed up by autocommands.
- When trying to execute an autocommand, also try matching the pattern with the short file name. Helps when short file name is different from full file name (expanded symbolic links). [autocmd-patterns](#)
- Made the output of ":autocmd" shorter and look better.
- Expand `<sfile>` in an ":autocmd" when it is defined. [<sfile>](#)
- Added "nested" flag to ":autocmd", allows nesting. [autocmd-nested](#)
- Added `[group]` argument to ":autocmd". Overrides the currently set group. [autocmd-groups](#)
- new events:

BufUnload	before a buffer is unloaded
BufDelete	before a buffer is deleted from the buffer list
FileChangedShell	when a file's modification time has changed after executing a shell command
User	user-defined autocommand
- When **'modified'** was set by a BufRead* autocommand, it was reset again afterwards. Now the ":set modified" is remembered.

GUI:

- Improved GUI scrollbar handling when redrawing is slower than the scrollbar

- events are generated.
- "vim -u NONE" now also stops loading the .gvimrc and other GUI inits. `-u`
Use "-U" to use another gvimrc file. `-U`
 - Handle **CTRL-C** for external command, also for systems where "setsid()" is supported.
 - When starting the GUI, restrict the window size to the screen size.
 - The default menus are read from \$VIMRUNTIME/menu.vim. This allows for a customized default menu. `menu.vim`
 - Improved the default menus. Added File/Print, a Window menu, Syntax menu, etc.
 - Added priority to the ":menu" command. Now each menu can be put in a place where you want it, independent of the order in which the menus are defined. `menu-priority`

Give a warning in the intro screen when running the Win32 console version on Windows 95 because there are problems using this version under Windows 95. `win32-problems`

Added 'e' flag for ":substitute" command: Don't complain when not finding a match (Campbell). `:s`

When using search commands in a mapping, only the last one is kept in the history. Avoids that the history is trashed by long mappings.

Ignore characters after "ex", "view" and "gvim" when checking startup mode. Allows the use of "gvim5" et. al. `gvim` "gview" starts the GUI in readonly mode. `gview`

When resizing windows, the cursor is kept in the same relative position, if possible. (Webb)

":all" and ":ball" no longer close and then open a window for the same buffer. Avoids losing options, jumplist, and other info.

"-f" command-line argument is now ignored if Vim was compiled without GUI. `-f`

In Visual block mode, the right mouse button picks up the nearest corner.

Changed default mappings for DOS et al. Removed the DOS-specific mappings, only use the Windows ones. Added Shift-Insert, Ctrl-Insert, Ctrl-Del and Shift-Del.

Changed the numbers in the output of ":jumps", so you can see where `{count}` **CTRL-O** takes you. `:jumps`

Using "~" for \$HOME now works for all systems. `$HOME`

Unix: Besides using **CTRL-C**, also use the INTR character from the tty settings. Somebody has INTR set to DEL.

Allow a `<NL>` in a ":help" command argument to end the help command, so another command can follow.

Doing "%" on a line that starts with " #if" didn't jump to matching "#else". Don't recognize "#if", "#else" etc. for '%' when 'cpo' contains the '%' flag.
%

Insert mode expansion with "CTRL-N", "CTRL-P" and "CTRL-X" improved
ins-completion :

- 'complete' option added.
- When 'nowrapscan' is set, and no match found, report the searched direction in the error message.
- Repeating CTRL-X commands adds following words/lines after the match.
- When adding-expansions, accept single character matches.
- Made repeated CTRL-X CTRL-N not break undo, and "." repeats the whole insertion. Also fixes not being able to backspace over a word that has been inserted with CTRL-N.

When copying characters in Insert mode from previous/next line, with CTRL-E or CTRL-Y, 'textwidth' is no longer used. i_CTRL-E

Commands that move in the arglist, like ":n" and ":rew", keep the old cursor position of the file (this is mostly Vi compatible).

Vim now remembers the '<' and '>' marks for each buffer. This fixes a problem that a line-delete in one buffer invalidated the '<' and '>' marks in another buffer. '<

For MSDOS, Unix and OS/2: When \$VIM not set, use the path from the executable. When using the executable path for \$VIM, remove "src/" when present. Should make Vim find the docs and syntax files when it is run directly after compiling. \$VIM

When quitting Visual mode with <Esc>, the cursor is put at start of the Visual area (like after executing an operator).

Win32 and Unix version: Removed 1100 character limit on external commands.

Added possibility to include a space in a ":edit +command" argument, by putting a backslash before it. +cmd

After recovery, BufReadPost autocommands are applied. :recover

Added color support for "os2ansi", OS/2 console. (Slootman)

Allow "%:p:h" when % is empty. :_%

Included "<sfile>": file name from the ":source" command. <sfile>

Added "<Bslash>" special character. Helps for avoiding multiple backslashes in mappings and menus.

In a help window, a double-click jumps to the tag under the cursor (like CTRL-]).

<C-Left> and <C-Right> now work like <S-Left> and <S-Right>, move a word forward/backward (Windows compatible). <C-Left>

Removed the requirement for a `":version"` command in a `.vimrc` file. It wasn't used for anything. You can use `":if"` to handle differences between versions.
`:version`

For MS-DOS, Win32 and OS/2: When comparing file names for autocommands, don't make a difference between `'/'` and `'\ '` for path separator.

New termcap options:

`"mb"`: blink. Can only be used by assigning it to one of the other highlight options. `t_mb`

`"bc"`: backspace character. `t_bc`

`"nd"`: Used for moving the cursor right in the GUI, to avoid removing one line of pixels from the last bold character. `t_nd`

`"xs"`: highlighting not erased by overwriting, for hpterm. Combined with `'weirdinvert'`. Visual mode works on hpterm now. `t_xs`

Unix: Set time of patch and backup file same as original file. (Hiebert).

Amiga: In QuickFix mode no longer opens another window. Shell commands can be used now.

Added decmouse patches from David Binette. Can now use Dec and Netterm mouse. But only when enabled at compile time.

Added `'#'` register: Alternate file name `quote#` . Display `'#'` register with `":dis"` command. `:display`

Removed `':'` from `'isfname'` default for Unix. Check for `"://"` in a file name anyway. Also check for `":\\"`, for MS-DOS.

Added count to `"K"` keyword command, when `'keywordprg'` is `"man"`, is inserted in the man command. `"2K"` results in `"!man 2 <word>"`. `K`

When using `"gf"` on a relative path name, remove `"/"` from the file name, like it's done for file names in the tags file. `gf`

When finishing recording, don't make the recorded register the default put register.

When using `"!!"`, don't put `":5,5!"` on the command-line, but `":.!"`. And some other enhancements to replace the line number with `."` or `"$"` when possible.

MSDOS et al.: Renamed `$VIM/viminfo` to `$VIM/_viminfo`. It's more consistent: `.vimrc/_vimrc` and `.viminfo/_viminfo`

For systems where case doesn't matter in file names (MSDOS, Amiga), ignore case while sorting file names. For buffer names too.

When reading from stdin doesn't work, read from stderr (helps for `"foo | xargs vim"`).

32 bit MS-DOS version: Replaced `csdpmi3` by `csdpmi4`.

Changed <C-Left> and <C-Right> to skip a WORD instead of a word.

Warning for changed modified time when overwriting a file now also works on other systems than Unix.

Unix: Changed the defaults for configure to be the same as the defaults for Makefile: include GUI, Perl, and Python.

Some versions of Motif require "-lXpm". Added check for this in configure.

Don't add "-L/usr/lib" to the link line, causes problems on a few systems.

COMPILE TIME CHANGES

compile-changes-5

When compiling, allow a choice for minimal, normal or maximal features in an easy way, by changing a single line in src/feature.h.

The DOS16 version has been compiled with minimal features to avoid running out of memory too quickly.

The Win32, DJGPP, and OS/2 versions use maximal features, because they have enough memory.

The Amiga version is available with normal and maximal features.

Added "make test" to Unix version Makefile. Allows for a quick check if most "normal" commands work properly. Also tests a few specific commands.

Added setlocale() with codepage support for DJGPP version.

autoconf:

- Added autoconf check for -lXdmcp.
- Included check for -lXmu, no longer needed to edit the Makefile for this.
- Switched to autoconf 2.12.
- Added configure check for <poll.h>. Seems to be needed when including Perl on Linux?
- term lib is now checked before termcap.
- Added configure check for strncasecmp(), stricmp() and strnicmp(). Added vim_stricmp() for when there's no library function for stricmp().
- Use "datadir" in configure, instead of our own check for HELPDIR.

Removed "make proto" from Makefile.manx. Could not make it work without a lot of #ifdefs.

Removed "proto/" from paths in proto.h. Needed for the Mac port.

Drastically changed Makefile.mint. Now it includes the Unix Makefile.

Added support for Dos16 in Makefile.b32 (renamed Makefile.b32 to Makefile.bor)

All source files are now edited with a tabstop of 8 instead of 4, which is better when debugging and using other tools. 'softtabstop' is set to 4, to make editing easier.

Unix: Added "link.sh" script, which removes a few unnecessary libraries from the link command.

Don't use HPUX digraphs by default, but only when HPUX_DIGRAPHS is defined.
[digraphs-default](#)

BUG FIXES

[bug-fixes-5](#)

Note: Some of these fixes may only apply to test versions which were created after version 4.6, but before 5.0.

When doing `":bdel"`, try going to the next loaded buffer. Don't rewind to the start of the buffer list.

`mch_isdir()` for Unix returned TRUE for "" on some systems.

Win32: `'shell'` set to `"mksnt/sh.exe"` breaks `":!"` commands. Don't use backslashes in the temp file names.

On linux, with a FAT file system, could get spurious "file xxx changed since editing started" messages, because the time is rounded off to two seconds unexpectedly.

Crash in GUI, when selecting a word (double click) and then extend until an empty line.

For systems where `isdigit()` can't handle characters > 255, `get_number()` caused a crash when moving the mouse during the prompt for recovery.

In Insert mode, `"CTRL-O P"` left the cursor on the last inserted character. Now the cursor is left after the last putted character.

When quickfix found an error type other than 'e' or 'w', it was never printed.

A setting for `'errorfile'` in a `.vimrc` overruled the `"-q errorfile"` argument.

Some systems create a file when generating a temp file name. Filtering would then create a backup file for this, which was never deleted. Now no backup file is made when filtering.

`simplify_filename()` could remove a `".."` after a link, resulting in the wrong file name. Made `simplify_filename` also work for MSDOS. Don't use it for Amiga, since it doesn't have `"../"`.

`otherfile()` was unreliable when using links. Could think that reading/writing was for a different file, when it was the same.

Pasting with mouse in Replace mode didn't replace anything.

Window height computed wrong when resizing a window with an autocommand (could cause a crash).

`":s!foo!bar!"` wasn't possible (Vi compatible).

do_bang() freed memory twice when called recursively, because of autocommands (test11). Thanks to Electric Fence!

"v\$d" on an empty line didn't remove the "-- VISUAL --" mode message from the command-line, and inverted the cursor.

":mkexrc" didn't check for failure to open the file, causing a crash. (Felderhoff).

Win32 mch_write() wrote past fixed buffer, causing terminal keys no longer to be recognized. Both console and GUI version.

Athena GUI: Crash when removing a menu item. Now Vim doesn't crash, but the reversing of the menu item is still wrong.

Always reset **'list'** option for the help window.

When **'scrolloff'** is non-zero, a **'showmatch'** could cause the shown match to be in the wrong line and the window to be scrolled (Acevedo).

After ":set all&", **'lines'** and **'ttytype'** were still non-default, because the defaults never got set. Now the defaults for **'lines'** and **'columns'** are set after detecting the window size. **'term'** and **'ttytype'** defaults are set when detecting the terminal type.

For (most) non-Unix systems, don't add file names with illegal characters when expanding. Fixes "cannot open swapfile" error when doing ":e *.burp", when there is no match.

In X11 GUI, drawing part of the cursor obscured the text. Now the text is drawn over the cursor, like when it fills the block. (Seibert)

when started with "-c cmd -q errfile", the cursor would be left in line 1. Now a ":cc" is done after executing "cmd".

":ilist" never ignored case, even when **'ignorecase'** set.

"vim -r file" for a readonly file, then making a change, got ATTENTION message in insert mode, display mixed up until <Esc> typed. Also don't give ATTENTION message after recovering a file.

The abbreviation ":ab #i #include" could not be removed.

CTRL-L completion (longest common match) on command-line didn't work properly for case-insensitive systems (MS-DOS, Windows, etc.). (suggested by Richard Kilgore).

For terminals that can hide the cursor ("vi" termcap entry), resizing the window caused the cursor to disappear.

Using an invalid mark in an Ex address didn't abort the command.

When **'smarttab'** set, would use **'shiftround'** when inserting a TAB after a space. Now it always rounds to a tabstop.

Set '[' and ']' marks for ":copy", ":move", ":append", ":insert", ":substitute" and ":change". (Acevedo).

"d\$" in an empty line still caused an error, even when 'E' is not in **'coptions'**.

Help files were stored in the viminfo buffer list without a path.

GUI: Displaying cursor was not synchronized with other displaying. Caused several display errors. For example, when the last two lines in the file start with spaces, "dd" on the last line copied text to the (then) last line.

Win32: Needed to type CTRL-SHIFT-- to get CTRL-__.

GUI: Moving the cursor forwards over bold text would remove one column of bold pixels.

X11 GUI: When a bold character in the last column was scrolled up or down, one column of pixels would not be copied.

Using <BS> to move the cursor left can sometimes erase a character. Now use "le" termcap entry for this.

Keyword completion with regexp didn't work. e.g., for "b.*crat".

Fixed: With **CTRL-O** that jumps to another file, cursor could end up just after the line.

Amiga: '\$' was missing from character recognized as wildcards, causing \$VIM sometimes not to be expanded.

":change" didn't adjust marks for deleted lines.

":help [range]" didn't work. Also for [pattern], [count] and [quotex].

For **'cindent'**, typing "class::method" doesn't align like a label when the second ':' is typed.

When inserting a CR with **'cindent'** set (and a bunch of other conditions) the cursor went to a wrong location.

'cindent' was wrong for a line that ends in '}'.

'cindent' was wrong after "else {".

While editing the cmdline in the GUI, could not use the mouse to select text from the command-line itself.

When deleting lines, marks in tag stack were only adjusted for the current window, not for other windows on the same buffer.

Tag guessing could find a function "some_func" instead of the "func" we were looking for.

Tags file name relative to the current file didn't work.

":g/pat2/s//pat2/g", causing the number of subs to be reported, used to cause a scroll up. Now you no longer have to hit <CR>.

X11 GUI: Selecting text could cause a crash.

32 bit DOS version: **CTRL-C** in external command killed Vim. When SHELL is set to "sh.exe", external commands didn't work. Removed using of command.com, no longer need to set '**shellquote**'.

Fixed crash when using ":g/pat/i".

Fixed (potential) crash for X11 GUI, when using an X selection. Was giving a pointer on the stack to a callback function, now it's static.

Using "#" and "*" with an operator didn't work. E.g. "c#".

Command-line expansion didn't work properly after ":*". (Acevedo)

Setting '**weirdinvert**' caused highlighting to be wrong in the GUI.

":e +4 #" didn't work, because the "4" was in unallocated memory (could cause a crash).

Cursor position was wrong for ":e #", after ":e #" failed, because of changes to the buffer.

When doing ":buf N", going to a buffer that was edited with ":view", the readonly flag was reset. Now make a difference between ":e file" and ":buf file": Only set/reset '**ro**' for the first one.

Avoid **hit-enter** prompt when not able to write viminfo on exit.

When giving error messages in the terminal where the GUI was started, GUI escape codes would be written to the terminal. In an xterm this could be seen as a '\$' after the message.

Mouse would not work directly after ":gui", because full_screen isn't set, which causes starttermcap() not to do its work.

'incsearch' did not scroll the window in the same way as the actual search. When '**nowrap**' set, incsearch didn't show a match when it was off the side of the screen. Now it also shows the whole match, instead of just the cursor position (if possible).

":unmap", ":unab" and ":unmenu" did not accept a double quote, it was seen as the start of a comment. Now it's Vi compatible.

Using <Up><Left><Left><Up> in the command-line, when there is no previous cmdline in the history, inserted a NUL on the command-line.

"i<Esc>" when on a <Tab> in column 0 left the cursor in the wrong place.

GUI Motif: When adding a lot of menu items, the menu bar goes into two rows. Deleting menu items, reducing the number of rows, now also works.

With `":g/pat/s//foo/c"`, a match in the first line was scrolled off of the screen, so you could not see it.

When using `":s//c"`, with `'nowrap'` set, a match could be off the side of the screen, so you could not see it.

When `'helpfile'` was set to a fixed, non-absolute path in `feature.h`, Vim would crash. `mch_Fullname` can now handle file names in read-only memory. (Lottem)

When using `CTRL-A` or `CTRL-@` in Insert mode, there could be strange effects when using `CTRL-D` next. Also, when repeating inserted text that included `"@ CTRL-D"` or `"^ CTRL-D"` this didn't work. (Acevedo)

Using `CTRL-D` after using `CTRL-E` or `CTRL-Y` in Insert mode that inserted a `'@'` or `'^'`, removed the `'@'` or `'^'` and more indent.

The command `"2".p"` caused the last inserted text to be executed as commands. (Acevedo)

Repeating the insert of `"CTRL-V 048"` resulted in `"^@"` to be inserted.

Repeating Insert completion could fail if there are special characters in the text. (Acevedo)

`":normal /string<CR>"` caused the window to scroll. Now all `":normal"` commands are executed without scrolling messages.

Redo of `CTRL-E` or `CTRL-Y` in Insert mode interpreted special characters as commands.

Line wrapping for `'tw'` was done one character off for insert expansion inserts.

`buffer_exists()` function didn't work properly for buffer names with a symbolic link in them (e.g. when using `buffer_exists(#)`).

Removed the `"MOTIF_COMMENT"` construction from Makefile. It now works with FreeBSD make, and probably with NeXT make too.

Matching the `'define'` and `'include'` arguments now honor the settings for `'ignorecase'`. (Acevedo)

When one file shown in two windows, Visual selection mixed up cursor position in current window and other window.

When doing `":e file"` from a help file, the `'isk'` option wasn't reset properly, because of a modeline in the help file.

When doing `":e!"`, a cursor in another window on the same buffer could become invalid, leading to `"ml_get: invalid lnum"` errors.

Matching buffer name for when expanded name has a different path from not expanded name (Brugnara).

Normal mappings didn't work after an operator. For example, with `":map Q gq"`,

"QQ" didn't work.

When ":make" resulted in zero errors, a "No Errors" error message was given (which breaks mappings).

When ":sourcing" a file, line length was limited to 1024 characters. **CTRL-V** before `<EOL>` was not handled Vi compatible. (Acevedo)

Unexpected exit for X11 GUI, caused by SAVE_YOURSELF event. (Heimann)

CTRL-X CTRL-I only found one match per line. (Acevedo)

When using an illegal **CTRL-X** key in Insert mode, the **CTRL-X** mode message was stuck.

Finally managed to ignore the "Quit" menu entry of the Window manager! Now Vim only exists when there are no changed buffers.

Trying to start the GUI when \$DISPLAY is not set resulted in a crash. When \$DISPLAY is not set and gvim starts vim, title was restored to "Thanks for flying Vim".

When \$DISPLAY not set, starting "gvim" (dropping back to vim) and then selecting text with the mouse caused a crash.

"J", with **'joinspaces'** set, on a line ending in ". ", caused one space too many to be added. (Acevedo)

In insert mode, a **CTRL-R {regname}** which didn't insert anything left the '"' on the screen.

":z10" didn't work. (Clapp)

"Help "*" didn't work.

Renamed a lot of functions, to avoid clashes with POSIX name space.

When adding characters to a line, making it wrap, the following lines were sometimes not shifted down (e.g. after a tag jump).

CTRL-E, with **'so'** set and cursor on last line, now does not move cursor as long as the last line is on the screen.

When there are two windows, doing "**^W+^W-**" in the bottom window could cause the status line to be doubled (not redrawn correctly).

This command would hang: ":n ``cat``". Now connect stdin of the external command to /dev/null, when expanding.

Fixed lalloc(0,) error for ":echo %:e:r". (Acevedo)

The "+command" argument to ":split" didn't work when there was no file name.

When selecting text in the GUI, which is the output of a command-line command or an external command, the inversion would sometimes remain.

GUI: "-mh 70" argument was broken. Now, when menuheight is specified, it is not changed anymore.

GUI: When using the scrollbar or mouse while executing an external command, this caused garbage characters.

Showmatch sometimes jumped to the wrong position. Was caused by a call to findmatch() when redrawing the display (when syntax highlighting is on).

Search pattern "\ (a *)\ {3} did not work correctly, also matched "a a". Problem with brace_count not being decremented.

Wildcard expansion added too many non-matching file names.

When **'iskeyword'** contains characters like '~', "*" and "#" didn't work properly. (Acevedo)

On Linux, on a FAT file system, modification time can change by one second. Avoid a "file has changed" warning for a one second difference.

When using the page-switching in an xterm, Vim would position the cursor on the last line of the window on exit. Also removed the cursor positioning for ":@" commands.

":g/pat/p" command (partly) overwrote the command. Now the output is on a separate line.

With **'ic'** and **'scs'** set, a search for "Keyword", ignore-case matches were highlighted too.

"^" on a line with only white space, put cursor beyond the end of the line.

When deleting characters before where insertion started ('bs' == 2), could not use abbreviations.

CTRL-E at end of file puts cursor below the file, in Visual mode, when **'so'** is non-zero. **CTRL-E** didn't work when **'so'** is big and the line below the window wraps. **CTRL-E**, when **'so'** is non-zero, at end of the file, caused jumping up-down.

":retab" didn't work well when **'list'** is set.

Amiga: When inserting characters at the last line on the screen, causing it to wrap, messed up the display. It appears that a '\n' on the last line doesn't always cause a scroll up.

In Insert mode "0<C-D><C-D>" deleted an extra character, because Vim thought that the "0" was still there. (Acevedo)

"z{count}l" ignored the count. Also for "zh" et. al. (Acevedo)

"S" when **'autoindent'** is off didn't delete leading white space.

"/<Tab>" landed on the wrong character when **'incsearch'** is set.

Asking a yes/no question could cause a `hit-enter` prompt.

When the file consists of one long line (>4100 characters), making changes caused various errors and a crash.

DJGPP version could not save long lines (>64000) for undo.

"yw" on the last char in the file didn't work. Also fixed "6x" at the end of the line. "6X" at the start of a line fails, but does not break a mapping. In general, a movement for an operator doesn't beep or flush a mapping, but when there is nothing to operate on it beeps (this is Vi compatible).

"m'" and "m`" now set the "'" mark at the cursor position.

Unix: Resetting of signals for external program didn't work, because SIG_DFL and NULL are the same! For "!!yes|dd count=1|", the yes command kept on running.

Partly fixed: Unix GUI: Typeahead while executing an external command was lost. Now it's not lost while the command is producing output.

Typing `<S-Tab>` in Insert mode, when it isn't mapped, inserted "`<S-Tab>`". Now it works like a normal `<Tab>`, just like `<C-Tab>` and `<M-Tab>`.

Redrawing ruler didn't check for old value correctly (caused UMR warnings in Purify).

Negative array index in `finish_viminfo_history()`.

`":g/^/d|mo $"` deleted all the lines. The `":move"` command now removes the `:global` mark from the moved lines.

Using `"vG"` while the last line in the window is a `"@"` line, didn't update correctly. Just the `"v"` showed `"~"` lines.

`"daw"` on the last char of the file, when it's a space, moved the cursor beyond the end of the line.

When `'hlsearch'` was set or reset, only the current buffer was redrawn, while this affects all windows.

`CTRL-^`, positioning the cursor somewhere from 1/2 to 1 1/2 screen down the file, put the cursor at the bottom of the window, instead of halfway.

When scrolling up for `":append"` command, not all windows were updated correctly.

When `'hlsearch'` is set, and an auto-indent is highlighted, pressing `<Esc>` didn't remove the highlighting, although the indent was deleted.

When `'ru'` set and `'nosc'`, using `"$j"` showed a wrong ruler.

Under Xfree 3.2, Shift-Tab didn't work (wrong keysym is used).

Mapping `<S-Tab>` didn't work. Changed the key translations to use the shortest key code possible. This makes the termcode translations and mappings more consistent. Now all modifiers work in all combinations, not only with `<Tab>`, but also with `<Space>`, `<CR>`, etc.

For Unix, restore three more signals. And Vim catches SIGINT now, so **CTRL-C** in Ex mode doesn't make Vim exit.

`"a5Y"` yanked 25 lines instead of 5.

`"vrxxx<Esc>"` in an empty line could not be undone.

A **CTRL-C** that breaks `:"make"` caused the errorfile not to be read (annoying when you want to handle what `:"make"` produced so far).

`:"0;/pat"` didn't find "pat" in line 1.

Search for `"/test/s+1"` at first char of file gave bottom-top message, or didn't work at all with `'nowrapscan'`.

Bug in viminfo history. Could cause a crash on exit.

`:"print"` didn't put cursor on first non-blank in line.

`:"0r !cat </dev/null"` left cursor in line zero, with very strange effects.

With `'showcmd'` set and `'timeoutlen'` set to a few seconds, trick to position the cursor leftwards didn't work.

AIX stty settings were restored to cs5 instead of cs8 (Winn).

File name completion didn't work for "zsh" versions that put spaces between file names, instead of NULs.

Changed `"XawChain*"` to `"XtChain*"`, should work for more systems.

Included quite a few fixes for rightleft mode (Lottem).

Didn't ask to `hit-enter` when GUI is started and error messages are printed.

When trying to edit a file in a non-existent directory, ended up with editing "No file".

`"gqap"` to format a paragraph did too much redrawing.

When `'hlsearch'` set, only the current window was updated for a new search pattern.

Sometimes error messages on startup didn't cause a `hit-enter` prompt, because of autocommands containing an empty line.

Was possible to select part of the window in the border, below the command line.

'< and '> marks were not at the correct position after linewise Visual selection.

When translating a help argument to "CTRL-x", prepend or append a '_', when applicable.

Blockwise visual mode wasn't correct when moving vertically over a special character (displayed as two screen characters).

Renamed "struct option" to "struct vimoption" to avoid name clash with GNU getopt().

":abclear" didn't work (but ":iabclear" and ":cabclear" did work).

When 'nowrap' used, screen wasn't always updated correctly.

"vim -c split file" displayed extra lines.

After starting the GUI, searched the termcap for a "gui" term.

When 'hls' used, search for "^\$" caused a hang.

When 'hls' was set, an error in the last regexp caused trouble.

Unix: Only output an extra <EOL> on exit when outputted something in the alternate screen, or when there is a message that needs to be cleared.

"/a\{" did strange things, depending on previous search.

"c}" only redrew one line (with -u NONE).

For mappings, CTRL-META-A was shown as <M-^A> instead of <MC-A>, while :map only accepts <MC-A>. Now <M-C-A> is shown.

Unix: When using full path name in a tags file, which contains a link, and 'hidden' set and jumping to a tag in the current file, would get bogus ATTENTION message. Solved by always expanding file names, even when starting with '/'.

'hlsearch' highlighting of special characters (e.g., a TAB) didn't highlight the whole thing.

"r<CR>" didn't work correctly on the last char of a line.

Sometimes a window resize or other signal caused an endless loop, involving set_winsize().

"vim -r" didn't work, it would just hang (using tgetent() while 'term' is empty).

"gk" while 'nowrap' set moved two lines up.

When windows are split, a message that causes a scroll-up messed up one of the windows, which required a CTRL-L to be typed.

Possible endless loop when using shell command in the GUI.

Menus defined in the .vimrc were removed when GUI started.

Crash when pasting with the mouse in insert mode.

Crash with ":unmenu *" in .gvimrc for Athena.

"5>>" shifted 5 lines 5 times, instead of 1 time.

CTRL-C when getting a prompt in ":global" didn't interrupt.

When **'so'** is non-zero, and moving the scrollbar completely to the bottom, there was a lot of flashing.

GUI: Scrollbar ident must be long for DEC Alpha.

Some functions called vim_regcomp() without setting reg_magic, which could lead to unpredictable maginness.

Crash when clicking around the status line, could get a selection with a backwards range.

When deleting more than one line characterwise, the last character wasn't deleted.

GUI: Status line could be overwritten when moving the scrollbar quickly (or when **'wd'** is non-zero).

An ESC at the end of a ":normal" command caused a wait for a terminal code to finish. Now, a terminal code is not recognized when its start comes from a mapping or ":normal" command.

Included patches from Robert Webb for GUI. Layout of the windows is now done inside Vim, instead of letting the layout manager do this. Makes Vim work with Lesstif!

UMR warning in set_expand_context().

Memory leak: b_winlnum list was never freed.

Removed TIOCLSET/TIOCLGET code from os_unix.c. Was changing some of the terminal settings, and looked like it wasn't doing anything good. (suggested by Juergen Weigert).

Ruler overwrote "is a directory" message. When starting up, and **'cmdheight'** set to > 1, first message could still be in the last line.

Removed prototype for putenv() from proto.h, it's already in osdef2.h.in.

In replace mode, when moving the cursor and then backspacing, wrong characters were inserted.

Win32 GUI was checking for a **CTRL-C** too often, making it slow.

Removed mappings for MS-DOS that were already covered by commands.

When visually selecting all lines in a file, cursor at last line, then "J". Gave ml_get errors. Was a problem with scrolling down during redrawing.

When doing a linewise operator, and then an operator with a mouse click, it was also linewise, instead of characterwise.

When **'list'** is set, the column of the ruler was wrong.

Spurious error message for `"/\(b\+\\)*"`.

When visually selected many lines, message from `":w file"` disappeared when redrawing the screen.

`":set <M-b>=^[b"`, then insert `^[b"`, waited for another character. And then inserted `"<M-b>"` instead of the real `<M-b>` character. Was trying to insert `K_SPECIAL x NUL`.

CTRL-W] didn't use count to set window height.

GUI: `"-font"` command-line argument didn't override **'guifont'** setting from `.gvimrc`. (Acevedo)

GUI: clipboard wasn't used for `"*y"`. And some more Win32/X11 differences fixed for the clipboard (Webb).

Jumping from one help file to another help file, with **'compatible'** set, removed the **'help'** flag from the buffer.

File-writable bit could be reset when using `":w!"` for a readonly file.

There was a wait for **CTRL-O** n in Insert mode, because the search pattern was shown.

Reduced wait, to allow reading a message, from 10 to 3 seconds. It seemed nothing was happening.

`":recover"` found same swap file twice.

GUI: `"*yy` only worked the second time (when pasting to an xterm)."

DJGPP version (dos32): The system flags were cleared.

Dos32 version: Underscores were sometimes replaced with y-umlaut (Levin).

Version 4.1 of ncurses can't handle `tputs("", ..)`. Avoid calling `tputs()` with an empty string.

`<S-Tab>` in the command-line worked like **CTRL-P** when no completion started yet. Now it does completion, last match first.

Unix: Could get annoying "can't write viminfo" message after doing "su". Now

the viminfo file is overwritten, and the user set back to the original one.

":set term=builtin_gui" started the GUI in a wrong way. Now it's not allowed anymore. But "vim -T gui" does start the GUI correctly now.

GUI: Triple click after a line only put last char in selection, when it is a single character word.

When the window is bigger than the screen, the scrolling up of messages was wrong (e.g. ":vers", ":hi"). Also when the bottom part of the window was obscured by another window.

When using a wrong option only an error message is printed, to avoid that the usage information makes it scroll off the screen.

When exiting because of not being able to read from stdin, didn't preserve the swap files properly.

Visual selecting all chars in more than one line, then hit "x" didn't leave an empty line. For one line it did leave an empty line.

Message for which autocommand is executing messed up file write message (for FileWritePost event).

"vim -h" included "-U" even when GUI is not available, and "-l" when lisp is not available.

Crash for ":he <C-A>" (command-line longer than screen).

":s/this/that/gc", type "y" two times, then undo, did reset the modified option, even though the file is still modified.

Empty lines in a tags file caused a ":tag" to be aborted.

When hitting 'q' at the more prompt for ":menu", still scrolled a few lines.

In an xterm that uses the bold trick a single row of characters could remain after an erased bold character. Now erase one extra char after the bold char, like for the GUI.

":pop!" didn't work.

When the reading a buffer was interrupted, ":w" should not be able to overwrite the file, ":w!" is required.

":cf%" caused a crash.

":gui longfilename", when forking is enabled, could leave part of the longfilename at the shell prompt.

VERSION 5.1

version-5.1

Improvements made between version 5.0 and 5.1.

This was mostly a bug-fix release, not many new features.

Changed

changed-5.1

The `expand()` function now separates file names with `<NL>` instead of a space. This avoids problems for file names with embedded spaces. To get the old result, use `substitute(expand(foo), "\n", " ", "g")`.

For Insert-expanding dictionaries allow a backslash to be used for wildchars. Allows expanding "ze\kra", when **'isk'** includes a backslash.

New icon for the Win32 GUI.

`":tag", ":tselect` etc. only use the argument as a regexp when it starts with `('/')`. Avoids that `":tag xx~"` gives an error message: "No previous sub.regexp". Also, when the `:tag` argument contained wildcard characters, it was not Vi compatible.

When using `('/')`, the argument is taken literally too, with a higher priority, so it's found before wildcard matches.

Only when the `('/')` is used are matches with different case found, even though **'ignorecase'** isn't set.

Changed `"g^]"` to only do `":tselect"` when there is more than one matching tag.

Changed some of the default colors, because they were not very readable on a dark background.

A character offset to a search pattern can move the cursor to the next or previous line. Also fixes that `"/pattern/e+2"` got stuck on "pattern" at the end of a line.

Double-clicks in the status line do no longer start Visual mode. Dragging a status line no longer stops Visual mode.

Perl interface: `Buffers()` and `Windows()` now use more logical arguments, like they are used in the rest of Vim (Moore).

Init `''` mark to the first character of the first line. Makes it possible to use `''` in an autocommand without getting an error message.

Added

added-5.1

`"shell_error"` internal variable: result of last shell command.

`":echohl"` command: Set highlighting for `":echo"`.

`'S'` flag in **'highlight'** and `StatusLineNC` highlight group: highlighting for status line of not-current window. Default is to use bold for current window.

Added `buffer_name()` and `buffer_number()` functions (Aaron).
Added `flags` argument `"g"` to `substitute()` function (Aaron).
Added `winheight()` function.

Win32: When an external command starts with `"start "`, no console is opened for it (Aaron).

Win32 console: Use termcap codes for bold/reverse based on the current console attributes.

Configure check for `"strip"`. (Napier)

CTRL-R CTRL-R x in Insert mode: Insert the contents of a register literally, instead of as typed.

Made a few "No match" error messages more informative by adding the pattern that didn't match.

`"make install"` now also copies the macro files.

`tools/tcltags`, a shell script to generate a tags file from a TCL file.

`"--with-tlib"` setting for configure. Easy way to use `termlib`: `./configure --with-tlib=termlib`.

`'u'` flag in `'cino'` for setting the indent for contained `()` parts.

When Win32 OLE version can't load the registered type library, ask the user if he wants to register Vim now. (Erhardt)

Win32 with OLE: When registered automatically, exit Vim.

Included `VisVim 1.1b`, with a few enhancements and the new icon (Heiko Erhardt).

Added patch from Vince Negri for Win32s support. Needs to be compiled with VC 4.1!

Perl interface: Added `$curbuf`. Rationalized `Buffers()` and `Windows()`. (Moore) Added `"group"` argument to `Msg()`.

Included Perl files in DOS source archive. Changed `Makefile.bor` and `Makefile.w32` to support building a Win32 version with Perl included.

Included new `Makefile.w32` from Ken Scott. Now it's able to make all Win32 versions, including OLE, Perl and Python.

Added **CTRL-W g]** and **CTRL-W g ^**: split window and do `g]` or `g^`].

Added `"g]"` to always do `":tselect"` for the indent under the cursor.

Added `":tjump"` and `":stjump"` commands.

Improved listing of `":tselect"` when tag names are a bit long.

Included patches for the Macintosh version. Also for Python interface. (St-Amant)

":buf foo" now also restores cursor column, when the buffer was used before.

Adjusted the Makefile for different final destinations for the syntax files and scripts (for Debian Linux).

Amiga: \$VIM can be used everywhere. When \$VIM is not defined, "VIM:" is used. This fixes that "VIM:" had to be assigned for the help files, and \$VIM set for the syntax files. Now either of these work.

Some xterms send vt100 compatible function keys F1-F4. Since it's not possible to detect this, recognize both type of keys and translate them to <F1> - <F4>.

Added "VimEnter" autocommand. Executed after loading all the startup stuff.

BeOS version now also runs on Intel CPUs (Seibert).

Fixed

fixed-5.1

":ts" changed position in the tag stack when cancelled with <CR>.

":ts" changed the cursor position for **CTRL-T** when cancelled with <CR>.

":tn" would always jump to the second match. Was using the wrong entry in the tag stack.

Doing "tag foo", then ":tselect", overwrote the original cursor position in the tag stack.

"make install" changed the vim.1 manpage in a wrong way, causing "doc/doc" to appear for the documentation files.

When compiled with MAX_FEAT, xterm mouse handling failed. Was caused by DEC mouse handling interfering.

Was leaking memory when using selection in X11.

CTRL-D halfway a command-line left some characters behind the first line(s) of the listing.

When expanding directories for ":set path=", put two extra backslashes before a space in a directory name.

When '**lisp**' set, first line of a function would be indented. Now its indent is set to zero. And use the indent of the first previous line that is at the same () level. Added test33.

"so<Esc>u" in an empty file didn't work.

DOS: "seek error in swap file write" errors, when using DOS 6.2 share.exe, because the swap file was made hidden. It's no longer hidden.

":global" command would sometimes not execute on a matching line. Happened when a data block is full in ml_replace().

For AIX use a tgetent buffer of 2048 bytes, instead of 1024.

Win32 gvim now only sets the console size for external commands to 25x80 on Windows 95, not on NT.

Win32 console: Dead key could cause a crash, because of a missing "WINAPI" (Deshpande).

The right mouse button started Visual mode, even when 'mouse' is empty, and in the command-line, a left click moved the cursor when 'mouse' is empty. In Visual mode, 'n' in 'mouse' would be used instead of 'v'.

A blinking cursor or focus change cleared a non-Visual selection.

CTRL-Home and **CTRL-End** didn't work for MS-DOS versions.

Could include NUL in 'iskeyword', causing a crash when doing insert mode completion.

Use _dos_commit() to flush the swap file to disk for MSDOS 16 bit version.

In mappings, **CTRL-H** was replaced by the backspace key code. This caused problems when it was used as text, e.g. ":map _U :%s/.^H//g<CR>".

":set t_Co=0" was not handled like a normal term. Now it's translated into ":set t_Co=", which works.

For ":syntax keyword" the "transparent" option did work, although not mentioned in the help. But synID() returned wrong name.

"gqG" in a file with one-word-per-line (e.g. a dictionary) was very slow and not interruptible.

"gq" operator inserted screen lines in the wrong situation. Now screen lines are inserted or deleted when this speeds up displaying.

cindent was wrong when an "if" contained "(".

'r' flag in 'viminfo' was not used for '%'. Could get files in the buffer list from removable media.

Win32 GUI with OLE: if_ole_vc.mak could not be converted into a project. Hand-edited to fix this...

With 'nosol' set, doing "\$kdw" below an empty line positioned the cursor at the end of the line.

Dos32 version changed "\dir\file" into "/dir/file", to work around a DJGPP bug. That bug appears to have been fixed, therefore this translation has been removed.

"/^*" didn't work (find '*' in first column).

"<afile>" was not always set for autocommands. E.g., for ":au BufEnter *

```
let &tags = expand("<afile>:p:h") . "/tags".
```

In an xterm, the window may be a child of the outer xterm window. Use the parent window when getting the title and icon names. (Smith)

When starting with "gvim -bg black -fg white", the value of **'background'** is only set after reading the .gvimrc file. This causes a ":syntax on" to use the wrong colors. Now allow using ":gui" to open the GUI window and set the colors. Previously ":gui" in a gvimrc crashed Vim.

tempname() returned the same name all the time, unless the file was actually created. Now there are at least 26 different names.

File name used for <afile> was sometimes full path, sometimes file name relative to current directory.

When **'background'** was set after the GUI window was opened, it could change colors that were set by the user in the .gvimrc file. Now it only changes colors that have not been set by the user.

Ignore special characters after a CSI in the GUI version. These could be interpreted as special characters in a wrong way. (St-Amant)

Memory leak in farsi code, when using search or ":s" command. Farsi string reversing for a mapping was only done for new mappings. Now it also works for replacing a mapping.

Crash in Win32 when using a file name longer than _MAX_PATH. (Aaron)

When BufDelete autocommands were executed, some things for the buffer were already deleted (esp. Perl stuff).

Perl interface: Buffer specific items were deleted too soon; fixes "screen no longer exists" messages. (Moore)

The Perl functions didn't set the **'modified'** flag.

link.sh did not return an error on exit, which may cause Vim to start installing, even though there is no executable to install. (Riehm)

Vi incompatibility: In Vi "." redoes the "y" command. Added the 'y' flag to **'coptions'**. Only for **'compatible'** mode.

":echohl" defined a new group, when the argument was not an existing group.

"syn on" and ":syn off" could move the cursor, if there is a hidden buffer that is shorter than the current cursor position.

The " mark was not set when doing ":b file".

When a "nextgroup" is used with "skipwhite" in syntax highlighting, space at the end of the line made the nextgroup also be found in the next line.

":he g<CTRL-D>", then ":" and backspace to the start didn't redraw.

X11 GUI: "gvim -rv" reversed the colors twice on Sun. Now Vim checks if the result is really reverse video (background darker than foreground).

"cat link.sh | vim -" didn't set syntax highlighting.

Win32: Expanding "file.sw?" matched ".file.swp". This is an error of FindnextFile() that we need to work around. (Kilgore)

"ggq" gave an "Invalid lnum" error on the last line. Formatting with "gq" didn't format the first line after a change of comment leader.

There was no check for out-of-memory in win_alloc().

"vim -h" didn't mention "--register" and "--unregister" for the OLE version.

Could not increase 'cmdheight' when the last window is only one line. Now other windows are also made smaller, when necessary.

Added a few {} to avoid "suggest braces around" warnings from gcc 2.8.x. Changed return type of main() from void to int. (Nam)

Using '~' twice in a substitute pattern caused a crash.

"syn on" and ":syn off" could scroll the window, if there is a hidden buffer that is shorter than the current cursor position.

":if 0 | if 1 | endif | endif" didn't work. Same for ":while" and "elseif".

With two windows on modified files, with 'autowrite' set, cursor in second window, ":qa" gave a warning for the file in the first window, but then auto-wrote the file in the second window. (Webb)

Win32 GUI scrollbar could only handle 32767 lines. Also makes the intellimouse wheel use the configurable number of scrolls. (Robinson)

When using 'patchmode', and the backup file is on another partition, the file copying messed up the write-file message.

GUI X11: Alt-Backspace and Alt-Delete didn't work.

"`0" could put the cursor after the last character in the line, causing trouble for other commands, like "i".

When completing tags in insert mode with ^X^], some matches were skipped, because the compare with other tags was wrong. E.g., when "mnuFileSave" was already there, "mnuFile" would be skipped. (Negri)

When scrolling up/down, a syntax item with "keepend" didn't work properly. Now the flags are also stored for the syntax state at the start of each line.

When 'ic' was changed while 'hlsearch' is on, there was no redraw to show the effect.

Win32 GUI: Don't display "No write since last chance" in a message box, but in the Vim window.

VERSION 5.2

version-5.2

Improvements made between version 5.1 and 5.2.

Long lines editable

long-lines

A single long line that doesn't fit in the window doesn't show a line of @@@ anymore. Redrawing starts at a character further on in the line, such that the text around the cursor can be seen. This makes it possible to edit these long lines when wrapping is on.

File browser added

file-browser-5.2

The Win32, Athena and Motif GUI bring up a file requester if the user asks to `":browse"` for the `":e"`, `":w"`, `":r"`, `":so"`, `":redirect"` and `":mkexrc/vimrc/vsess"` commands. `":browse e /foo/bar"` opens the requester in the `/foo/bar` directory, so you can have nice mapping rhs's like `":browse so $vim/macros"`. If no initial dir specified for `":browse e"`, can be compiled to either begin in the current directory, or that of the current buffer. (Negri and Kahn)

Added the **'browsedir'** option, with value "current", "last" or "buffer". Tells whether a browse dialog starts in last used dir, dir of current buffer, or current dir. `":browse w"` is unaffected.

The default menus have been changed to use the `":browse"` command.

Dialogs added

dialogs-added

Added the `":confirm"` command. Works on `":e"`, `":q"`, `":w"`, `":cl"`. Win32, Athena and Motif GUI uses a window-dialog. All other platforms can use prompt in command-line. `":confirm qa"` offers a choice to save all modified files.

`confirm()` function: allows user access to the confirm engine.

Added 'v' flag to **'guioptions'**. When included, a vertical button layout is always used for the Win32 GUI dialog. Otherwise, a horizontal layout is preferred.

Win32 GUI: `":promptfind"` and `":promptrepl"` pop up a dialog to find/replace. To be used from a menu entry. (Negri)

Popup menu added

popup-menu-added

When the **'mousemodel'** option is set to "popup", the right mouse button displays the top level menu headed with "PopUp" as pop-up context menu. The "PopUp" menu is not displayed in the normal menu bar. This currently only works for Win32 and Athena GUI.

Select mode added

new-Select-mode

A new mode has been added: "Select mode". It is like Visual mode, but typing a printable character replaces the selection.

- **CTRL-G** can be used to toggle between Visual mode and Select mode.
- **CTRL-O** can be used to switch from Select mode to Visual mode for one command.
- Added **'selectmode'** option: tells when to start Select mode instead of Visual mode.
- Added **'mousemodel'** option: Change use of mouse buttons.
- Added **'keymodel'** option: tells to use shifted special keys to start a Visual or Select mode selection.
- Added **":behave"**. Can be used to quickly set **'selectmode'**, **'mousemodel'** and **'keymodel'** for MS-Windows and xterm behavior.
- The xterm-like selection is now called modeless selection.
- Visual mode mappings and menus are used in Select mode. They automatically switch to Visual mode first. Afterwards, reselect the area, unless it was deleted. The "gV" command can be used in a mapping to skip the reselection.
- Added the "gh", "gH" and "g^H" commands: start Select (highlight) mode.
- Backspace in Select mode deletes the selected area.

"mswin.vim" script. Sets behavior mostly like MS-Windows.

Session files added

new-session-files

":mks[ession]" acts like "mkvimrc", but also writes the full filenames of the currently loaded buffers and current directory, so that **":so'**ing the file re-loads those files and **cd'**s to that directory. Also stores and restores windows. File names are made relative to session file.

The **'sessionoptions'** option sets behavior of **":mksession"**. (Negri)

User defined functions and commands

new-user-defined

Added user defined functions. Defined with **":function"** until **":endfunction"**. Called with **"Func()"**. Allows the use of a variable number of arguments. Included support for local variables **"l:name"**. Return a value with **":return"**. See **:function**.

Call a function with **":call"**. When using a range, the function is called for each line in the range. **:call**

"macros/justify.vim" is an example of using user defined functions.

User functions do not change the last used search pattern or the command to be redone with **":."**.

'**maxfuncdepth**' option. Restricts the depth of function calls. Avoids trouble (crash because of out-of-memory) when a function uses endless recursion.

User definable Ex commands: ":command", ":delcommand" and ":comclear".
(Moore) See [user-commands](#) .

New interfaces

[interfaces-5.2](#)

Tcl interface. (Wilken) See [tcl](#) .
Uses the ":tcl", ":tcldo" and "tclfile" commands.

Cscope support. (Kahn) (Sekera) See [cscope](#) .
Uses the ":cscope" and ":cstag" commands. Uses the options '**cscopeprg**', '**cscopetag**', '**cscopetagorder**' and '**cscopeverbose**'.

New ports

[ports-5.2](#)

Amiga GUI port. (Nielsen) Not tested much yet!

RISC OS version. (Thomas Leonard) See [riscos](#) .
This version can run either with a GUI or in text mode, depending upon where it is invoked.
Deleted the "os_archie" files, they were not working anyway.

Multi-byte support

[new-multi-byte](#) [new-multibyte](#)

MultiByte support for Win32 GUI. (Baek)
The '**fileencoding**' option decides how the text in the file is encoded.
":ascii" works for multibyte characters. Multi-byte characters work on Windows 95, even when using the US version. (Aaron)
Needs to be enabled in feature.h.
This has not been tested much yet!

New functions

[new-functions-5.2](#)

browse()	puts up a file requester when available. (Negri)
escape()	escapes characters in a string with a backslash.
fnamemodify()	modifies a file name.
input()	asks the user to enter a line. (Aaron) There is a separate history for lines typed for the input() function.
argc()	
argv()	can be used to access the argument list.
winbufnr()	buffer number of a window. (Aaron)
winnr()	window number. (Aaron)
matchstr()	Return matched string.
setline()	Set a line to a string value.

'allowrevins'	Enable the <code>CTRL-_</code> command in Insert and Command-line mode.
'browsedir'	Tells in which directory a browse dialog starts.
'confirm'	when set, <code>:q :w</code> and <code>:e</code> commands always act as if <code>":confirm"</code> is used. (Negri)
'cscopeprg'	
'cscopetag'	
'cscopetagorder'	
'cscopeverbose'	Set the <code>cscope</code> behavior.
'filetype'	RISC-OS specific type of file.
'grepformat'	
'grepprg'	For the <code>:grep</code> command.
'keymodel'	Tells to use shifted special keys to start a Visual or Select mode selection.
'listchars'	Set character to show in <code>'list'</code> mode for end-of-line, tabs and trailing spaces. (partly by Smith) Also sets character to display if a line doesn't fit when <code>'nowrap'</code> is set.
'matchpairs'	Allows matching <code>'<'</code> with <code>'>'</code> , and other single character pairs.
'mousefocus'	Window focus follows mouse (partly by Terhaar). Changing the focus with a keyboard command moves the pointer to that window. Also move the pointer when changing the window layout (split window, change window height, etc.).
'mousemodel'	Change use of mouse buttons.
'selection'	When set to <code>"inclusive"</code> or <code>"exclusive"</code> , the cursor can go one character past the end of the line in Visual or Select mode. When set to <code>"old"</code> the old behavior is used. When <code>"inclusive"</code> , the character under the cursor is included in the operation. When using <code>"exclusive"</code> , the new <code>"ve"</code> entry of <code>'guicursor'</code> is used. The default is a vertical bar.
'selectmode'	Tells when to start Select mode instead of Visual mode.
'sessionoptions'	Sets behavior of <code>":mksession"</code> . (Negri)
'showfulltag'	When completing a tag in Insert mode, show the tag search pattern (tidied up) as a choice as well (if there is one).
'swapfile'	Whether to use a swap file for a buffer.
'syntax'	When it is set, the syntax by that name is loaded. Allows for setting a specific syntax from a modeline.
'ttermouse'	Allows using xterm mouse codes for terminals which name doesn't start with <code>"xterm"</code> .
'wildignore'	List of patterns for files that should not be completed at all.
'wildmode'	Can be used to set the type of expansion for <code>'wildchar'</code> . Replaces the <code>CTRL-T</code> command for command line completion. Don't beep when listing all matches.
'winaltkeys'	Win32 and Motif GUI. When <code>"yes"</code> , ALT keys are handled entirely by the window system. When <code>"no"</code> , ALT keys are never used by the window system. When <code>"menu"</code> it depends on whether a key is a menu shortcut.
'winminheight'	Minimal height for each window. Default is 1. Set to 0 if you want zero-line windows. Scrollbar is removed for

zero-height windows. (Negri)

New Ex commands

new-ex-commands-5.2

<code>:badd</code>	Add file name to buffer list without side effects. (Negri)
<code>:behave</code>	Quickly set MS-Windows or xterm behavior.
<code>:browse</code>	Use file selection dialog.
<code>:call</code>	Call a function, optionally with a range.
<code>:cnewer</code>	
<code>:colder</code>	To access a stack of quickfix error lists.
<code>:comclear</code>	Clear all user-defined commands.
<code>:command</code>	Define a user command.
<code>:continue</code>	Go back to <code>":while"</code> .
<code>:confirm</code>	Ask confirmation if something unexpected happens.
<code>:cscope</code>	Execute cscope command.
<code>:cstag</code>	Use cscope to jump to a tag.
<code>:delcommand</code>	Delete a user-defined command.
<code>:delfunction</code>	Delete a user-defined function.
<code>:endfunction</code>	End of user-defined function.
<code>:function</code>	Define a user function.
<code>:grep</code>	Works similar to <code>":make"</code> . (Negri)
<code>:mksession</code>	Create a session file.
<code>:nohlsearch</code>	Stop 'hlsearch' highlighting for a moment.
<code>:Print</code>	This is Vi compatible. Does the same as <code>":print"</code> .
<code>:promptfind</code>	Search dialog (Win32 GUI).
<code>:promptrepl</code>	Search/replace dialog (Win32 GUI).
<code>:return</code>	Return from a user-defined function.
<code>:simalt</code>	Win32 GUI: Simulate alt-key pressed. (Negri)
<code>:smagic</code>	Like <code>":substitute"</code> , but always use 'magic' .
<code>:snomagic</code>	Like <code>":substitute"</code> , but always use 'nomagic' .
<code>:tcl</code>	Execute TCL command.
<code>:tcldo</code>	Execute TCL command for a range of lines.
<code>:tclfile</code>	Execute a TCL script file.
<code>:tearoff</code>	Tear-off a menu (Win32 GUI).
<code>:tmenu</code>	
<code>:tunmenu</code>	Win32 GUI: menu tooltips. (Negri)
<code>:star :*</code>	Execute a register.

Changed

changed-5.2

Renamed functions:

<code>buffer_exists()</code>	-> <code>bufexists()</code>
<code>buffer_name()</code>	-> <code>bufname()</code>
<code>buffer_number()</code>	-> <code>bufnr()</code>
<code>file_readable()</code>	-> <code>filereadable()</code>
<code>highlight_exists()</code>	-> <code>hlexists()</code>
<code>highlightID()</code>	-> <code>hlID()</code>
<code>last_buffer_nr()</code>	-> <code>bufnr("\$")</code>

The old ones are still there, for backwards compatibility.

The `CTRL-_` command in Insert and Command-line mode is only available when the new `'allowrevins'` option is set. Avoids that people who want to type `SHIFT-_` accidentally enter reverse Insert mode, and don't know how to get out.

When a file name path in `":tselect"` listing is too long, remove a part in the middle and put `"..."` there.

Win32 GUI: Made font selector appear inside Vim window, not just any odd place. (Negri)

`":bn"` skips help buffers, unless currently in a help buffer. (Negri)

When there is a status line and only one window, don't show `'^'` in the status line of the current window.

`":*"` used to be used for `"'<,>'"`, the Visual area. But in Vi it's used as an alternative for `":@"`. When `'coptions'` includes `'*'` this is Vi compatible.

When `'insertmode'` is set, using `CTRL-O` to execute a mapping will work like `'insertmode'` was not set. This allows "normal" mappings to be used even when `'insertmode'` is set.

When `'mouse'` was set already (e.g., in the `.vimrc` file), don't automatically set `'mouse'` when the GUI starts.

Removed the `'N'`, `'I'` and `'A'` flags from the `'mouse'` option.

Renamed "toggle option" to "boolean option". Some people thought that `":set xyz"` would toggle `'xyz'` on/off each time.

The internal variable `"shell_error"` contains the error code from the shell, instead of just 0 or 1.

When inserting or replacing, typing `CTRL-V CTRL-<CR>` used to insert `"<C-CR>"`. That is not very useful. Now the CTRL key is ignored and a `<CR>` is inserted. Same for all other "normal" keys with modifiers. Mapping these modified key combinations is still possible.

In Insert mode, `<C-CR>` and `<S-Space>` can be inserted by using `CTRL-K` and then the special character.

Moved "quotes" file to `doc/quotes.txt`, and "todo" file to `doc/todo.txt`. They are now installed like other documentation files.

`winheight()` function returns `-1` for a non-existing window. It used to be zero, but that is a valid height now.

The default for `'selection'` is "inclusive", which makes a difference when using `"$"` or the mouse to move the cursor in Visual mode.

`":q!"` does not exit when there are changed buffers which are hidden. Use `":qa!"` to exit anyway.

Disabled the Perl/Python/Tcl interfaces by default. Not many people use them

and they make the executable a lot bigger. The internal scripting language is now powerful enough for most tasks.

The strings from the **'titlestring'** and **'iconstring'** options are used untranslated for the Window title and icon. This allows for including a `<CR>`. Previously a `<CR>` would be shown as `^M` (two characters).

When a mapping is started in Visual or Select mode which was started from Insert mode (the mode shows "(insert) Visual"), don't return to Insert mode until the mapping has ended. Makes it possible to use a mapping in Visual mode that also works when the Visual mode was started from Select mode.

Menus in `$VIMRUNTIME/menu.vim` no longer overrule existing menus. This helps when defining menus in the `.vimrc` file, or when sourcing `mswin.vim`.

Unix: Use `/var/tmp` for `.swp` files, if it exists. Files there survive a reboot (at least on Linux).

Added

added-5.2

`--with-motif-lib` configure argument. Allows for using a static Motif library.

Support for mapping numeric keypad `+, -, *, /` keys. (Negri)
When not mapped, they produce the normal character.

Win32 GUI: When directory dropped on gVim, `cd` there and edit new buffer. (Negri)

Win32 GUI: Made **CTRL-Break** work as interrupt, so that **CTRL-C** can be used for mappings.

In the output of `":map"`, highlight the `"*` to make clear it's not part of the rhs. (Roemer)

When showing the Visual area, the cursor is not switched off, so that it can be located. The Visual area is now highlighted with a grey background in the GUI. This makes the cursor visible when it's also reversed.

Win32: When started with single full pathname (e.g. via double-clicked file), `cd` to that file's directory. (Negri)

Win32 GUI: Tear-off menus, with `":tearoff <menu-name>"` command. (Negri)
't' option to **'guioptions'**: Add tearoff menu items for Win32 GUI and Motif. It's included by default.

Win32 GUI: tearoff menu with submenus is indicated with a `">>"`. (Negri)

Added `^Kaa` and `^KAA` digraphs.

Added "euro" symbol to `digraph.c`. (Corry)

Support for Motif menu shortcut keys, using `'&'` like MS-Windows (Ollis).
Other GUIs ignore `'&'` in a menu name.

DJGPP: Faster screen updating (John Lange).

Clustering of syntax groups ":syntax cluster" (Bigham).
Including syntax files: ":syntax include" (Bigham).

Keep column when switching buffers, when '**nosol**' is set (Radics).

Number function for Perl interface.

Support for Intellimouse in Athena GUI. (Jensen)

":sleep" also accepts an argument in milliseconds, when "m" is used.

Added 'p' flag in '**guioptions**': Install callbacks for enter/leave window events. Makes cursor blinking work for Terhaar, breaks it for me.

"--help" and "--version" command-line arguments.

Non-text in ":list" output is highlighted with NonText.

Added text objects: "i(" and "i)" as synonym for "ib". "i{" and "i}" as synonym for "iB". New: "i<" and "i>", to select <thing>. All this also for "a" objects.

'O' flag in '**shortmess**': message for reading a file overwrites any previous message. (Negri)

Win32 GUI: 'T' flag in '**guioptions**': switch toolbar on/off.
Included a list with self-made toolbar bitmaps. (Negri)

Added menu priority for sub-menus. Implemented for Win32 and Motif GUI.
Display menu priority with ":menu" command.
Default and Syntax menus now include priority for items. Allows inserting menu items in between the default ones.

When the '**number**' option is on, highlight line numbers with the LineNr group.

"Ignore" highlight group: Text highlighted with this is made blank. It is used to hide special characters in the help text.

Included Exuberant Ctags version 2.3, with C++ support, Java support and recurse into directories. (Hiebert)

When a tags file is not sorted, and this is detected (in a simplistic way), an error message is given.

":unlet" accepts a "!", to ignore non-existing variables, and accepts more than one argument. (Roemer)
Completion of variable names for ":unlet". (Roemer)

When there is an error in a function which is called by another function, show the call stack in the error message.

New file name modifiers:

":.": reduce file name to be relative to current dir.
":~": reduce file name to be relative to home dir.
":s?pat?sub?": substitute "pat" with "sub" once.
":gs?pat?sub?": substitute "pat" with "sub" globally.

New configure arguments: --enable-min-features and --enable-max-features.
Easy way to switch to minimum or maximum features.

New compile-time feature: modify_fname. For file name modifiers, e.g,
"%:p:h". Can be disabled to save some code (16 bit DOS).

When using whole-line completion in Insert mode, and 'cindent' is set, indent the line properly.

MSDOS and Win32 console: 'guicursor' sets cursor thickness. (Negri)

Included new set of Farsi fonts. (Shiran)

Accelerator text now also works in Motif. All menus can be defined with & for mnemonic and TAB for accelerator text. They are ignored on systems that don't support them.

When removing or replacing a menu, compare the menu name only up to the <Tab> before the mnemonic.

'i' and 'I' flags after ":substitute": ignore case or not.

"make install" complains if the runtime files are missing.

Unix: When finding an existing swap file that can't be opened, mention the owner of the file in the ATTENTION message.

The 'i', 't' and 'k' options in 'complete' now also print the place where they are looking for matches. (Acevedo)

"gJ" command: Join lines without inserting a space.

Setting 'keywordprg' to "man -s" is handled specifically. The "-s" is removed when no count given, the count is added otherwise. Configure checks if "man -s 2 read" works, and sets the default for 'keywordprg' accordingly.

If you do a ":bd" and there is only one window open, Vim tries to move to a buffer of the same type (i.e. non-help to non-help, help to help), for consistent behavior to :bnext/:bprev. (Negri)

Allow "<Nop>" to be used as the rhs of a mapping. ":map xx <Nop>", maps "xx" to nothing at all.

In a ":menu" command, "<Tab>" can be used instead of a real tab, in the menu path. This makes it more easy to type, no backslash needed.

POSIX compatible character classes for regexp patterns: [:alnum:], [:alpha:], [:blank:], [:cntrl:], [:digit:], [:graph:], [:lower:], [:print:], [:punct:], [:space:], [:upper:] and [:xdigit:]. (Briscoe)

regex character classes (for fast syntax highlight matching):

digits:	<code>\d [0-9]</code>	<code>\D</code> not digit (Roemer)
hex:	<code>\x [0-9a-fA-F]</code>	<code>\X</code> not hex
octal:	<code>\o [0-7]</code>	<code>\O</code> not octal
word:	<code>\w [a-zA-Z0-9_]</code>	<code>\W</code> not word
head:	<code>\h [a-zA-Z_]</code>	<code>\H</code> not head
alphabetic:	<code>\a [a-zA-Z]</code>	<code>\A</code> not alphabetic
lowercase:	<code>\l [a-z]</code>	<code>\L</code> not lowercase
uppercase:	<code>\u [A-Z]</code>	<code>\U</code> not uppercase

":set" now accepts "+=", "|^=" and "--=": add or remove parts of a string option, add or subtract a number from a number option. A comma is automagically inserted or deleted for options that are a comma-separated list.

Filetype feature, for autocommands. Uses a file type instead of a pattern to match a file. Currently only used for RISC OS. (Leonard)

In a pattern for an autocommand, environment variables can be used. They are expanded when the autocommand is defined.

"BufFilePre" and "BufFilePost" autocommand events: Before and after applying the ":file" command to change the name of a buffer.

"VimLeavePre" autocommand event: before writing the .viminfo file.

For autocommands argument: `<abuf>` is buffer number, like `<afile>`.

Made syntax highlighting a bit faster when scrolling backwards, by keeping more syncing context.

Win32 GUI: Made scrolling faster by avoiding a redraw when deleting or inserting screen lines.

GUI: Made scrolling faster by not redrawing the scrollbar when the thumb moved less than a pixel.

Included ":highlight" in bugreport.vim.

Created install.exe program, for simplistic installation on DOS and MS-Windows.

New register: '_', the black hole. When writing to it, nothing happens. When reading from it, it's always empty. Can be used to avoid a delete or change command to modify the registers, or reduce memory use for big changes.

CTRL-V xff enters character by hex number. **CTRL-V** o123 enters character by octal number. (Aaron)

Improved performance of syntax highlighting by skipping check for "keepend" when there isn't any.

Moved the mode message ("-- INSERT --") to the last line of the screen. When '**cmdheight**' is more than one, messages will remain readable.

When listing matching files, they are also sorted on '**suffices**', such that

they are listed in the same order as **CTRL-N** retrieves them.

synIDattr() takes a third argument (optionally), which tells for which terminal type to get the attributes for. This makes it possible to run 2html.vim outside of gvim (using color names instead of #RRGGBB).

Memory profiling, only for debugging. Prints at exit, and with "g^A" command. (Kahn)

DOS: When using a file in the current drive, remove the drive name: "A:\dir\file" -> "\dir\file". This helps when moving a session file on a floppy from "A:\dir" to "B:\dir".

Increased number of remembered jumps from 30 to 50 per window.

Command to temporarily disable 'hls' highlighting until the next search: ":nohlsearch".

"gp" and "gP" commands: like "p" and "P", but leave the cursor just after the inserted text. Used for the **CTRL-V** command in MS-Windows mode.

Fixed

fixed-5.2

Win32 GUI: Could draw text twice in one place, for fake-bold text. Removed this, Windows will handle the bold text anyway. (Negri)

patch 5.1.1: Win32s GUI: pasting caused a crash (Negri)

patch 5.1.2: When entering another window, where characters before the cursor have been deleted, could have a cursor beyond the end of the line.

patch 5.1.3: Win32s GUI: Didn't wait for external command to finish. (Negri)

patch 5.1.4: Makefile.w32 can now also be used to generate the OLE version (Scott).

patch 5.1.5: Crashed when using syntax highlighting: cursor on a line that doesn't fit in the window, and splitting that line in two.

patch 5.1.6: Visual highlighting bug: After ":set nowrap", go to end of line (so that the window scrolls horizontally), ":set wrap". Following Visual selection was wrong.

patch 5.1.7: When 'tagbsearch' off, and 'ignorecase' off, still could do binary searching.

patch 5.1.8: Win32 GUI: dragging the scrollbar didn't update the ruler.

patch 5.1.9: Using ":gui" in .vimrc, caused xterm cursor to disappear.

patch 5.1.10: A **CTRL-N** in Insert mode could cause a crash, when a buffer without a name exists.

patch 5.1.11: "make test" didn't work in the shadow directory. Also adjusted "make shadow" for the links in the ctags directory.

patch 5.1.12: "buf 123foo" used "123" as a count, instead as the start of a buffer name.

patch 5.1.13: When completing file names on the command-line, reallocating the command-line may go wrong.

patch 5.1.14: ":[nvc]unmenu" removed menu for all modes, when full menu patch specified.

Graceful handling of NULLs in drag-dropped file list. Handle passing NULL to Fullname_save(). (Negri)

Win32: "!!start" to invoke a program without opening a console, swapping screens, or waiting for completion in either console or gui version, e.g. you can type "!!start winfile". ALSO fixes "can't delete swapfile after spawning a shell" bug. (enhancement of Aaron patch) (Negri)

Win32 GUI: Fix **CTRL-X** default keymapping to be more Windows-like. (Negri)

Shorten filenames on startup. If in /foo/bar, entering "vim ../bar/bang.c" displays "bang.c" in status bar, not "/foo/bar/bang.c" (Negri)

Win32 GUI: No copy to Windows clipboard when it's not desired.

Win32s: Fix pasting from clipboard - made an assumption not valid under Win32s. (Negri)

Win32 GUI: Speed up calls to gui_mch_draw_string() and cursor drawing functions. (Negri)

Win32 GUI: Middle mouse button emulation now works in GUI! (Negri)

Could skip messages when combining commands in one line, e.g.:
":echo "hello" | write".

Perl interpreter was disabled before executing VimLeave autocommands. Could not use ":perl" in them. (Aaron)

Included patch for the Intellimouse (Aaron/Robinson).

Could not set '**ls**' to one, when last window has only one line. (Mitterand)

Fixed a memory leak when removing menus.

After ":only" the ruler could overwrite a message.

Dos32: removed changing of __system_flags. It appears to work better when it's left at the default value.

p_aleph was an int instead of along, caused trouble on systems where

sizeof(int) != sizeof(long). (Schmidt)

Fixed enum problems for Ultrix. (Seibert)

Small redraw problem: "dd" on last line in file cleared wrong line.

Didn't interpret "cmd | endif" when "cmd" starts with a range. E.g. "if 0 | .d | endif".

Command "+|" on the last line of the file caused ml_get errors.

Memory underrun in eval_vars(). (Aaron)

Don't rename files in a difficult way, except on Windows 95 (was also done on Windows NT).

Win32 GUI: An external command that produces an error code put the error message in a dialog box. had to close the window and close the dialog. Now the error code is displayed in the console. (Negri)

"comctl32.lib" was missing from the GUI libraries in Makefile.w32. (Battle)

In Insert mode, when entering a window in Insert mode, allow the cursor to be one char beyond the text.

Renamed machine dependent rename() to mch_rename(). Define mch_rename() to rename() when it works properly.

Rename vim_chdir() to mch_chdir(), because it's machine dependent.

When using an arglist, and editing file 5 of 4, ":q" could cause "-1 more files to edit" error.

In if_python.c, VimCommand() caused an assertion when a do_cmdline() failed. Moved the Python_Release_Vim() to before the VimErrorCheck(). (Harkins)

Give an error message for an unknown argument after "--". E.g. for "vim --xyz".

The FileChangedShell autocommand didn't set <afile> to the name of the changed file.

When doing ":e file", causing the attention message, there sometimes was no hit-enter prompt. Caused by empty line or "endif" at end of sourced file.

A large number of patches for the VMS version. (Hunsaker)

When **CTRL-L** completion (find longest match) results in a shorter string, no completion is done (happens with ":help").

Crash in Win32 GUI version, when using an Ex "@" command, because LinePointers[] was used while not initialized.

Win32 GUI: allow mapping of Alt-Space.

Output from "vim -h" was sent to stderr. Sending it to stdout is better, so one can use "vim -h | more".

In command-line mode, ":vi[!]" should reload the file, just like ":e[!]"
In Ex mode, ":vi" stops Ex mode, but doesn't reload the file. This is Vi compatible.

When using a ":set ls=1" in the .gvimrc file, would get a status line for a single window. (Robinson)

Didn't give an error message for ":set ai,xx". (Roemer)

Didn't give an error message for ":set ai?xx", ":set ai&xx", ":set ai!xx".

Non-Unix systems: That a file exists but is unreadable is recognized as "new file". Now check for existence when file can't be opened (like Unix).

Unix: osdef.sh didn't handle declarations where the function name is at the first column of the line.

DJGPP: Shortening of file names didn't work properly, because get_cwd() returned a path with backslashes. (Negri)

When using a '**comments**' part where a space is required after the middle part, always insert a space when starting a new line. Helps for C comments, below a line with "/****".

Replacing path of home directory with "~/ " could be wrong for file names with embedded spaces or commas.

A few fixes for the Sniff interface. (Leherbauer)

When asking to hit 'y' or 'n' (e.g. for ":3,ld"), using the mouse caused trouble. Same for ":s/x/y/c" prompt.

With '**nowrap**' and '**list**', a Tab halfway on the screen was displayed as blanks, instead of the characters specified with '**listchars**'. Also for other characters that take more than one screen character.

When setting '**guifont**' to an unknown font name, the previous font was lost and a default font would be used. (Steed)

DOS: Filenames in the root directory didn't get shortened properly. (Negri)

DJGPP: making a full path name out of a file name didn't work properly when there is no _fullpath() function. (Negri)

Win32 console: ":sh" caused a crash. (Negri)

Win32 console: Setting '**lines**' and/or '**columns**' in the _vimrc failed miserably (could hang Windows 95). (Negri)

Win32: The change-drive function was not correct, went to the wrong drive. (Tsindlekht)

GUI: When editing a command line in Ex mode, Tabs were sometimes not backspaced properly, and unprintable characters were displayed directly. non-GUI can still be wrong, because a system function is called for this.

":set" didn't stop after an error. For example ":set no ai" gave an error for "no", but still set "ai". Now ":set" stops after the first error.

When running configure for ctags, \$LDLFLAGS wasn't passed to it, causing trouble for IRIX.

"@%" and "@#" when file name not set gave an error message. Now they just return an empty string. (Steed)

CTRL-X and **CTRL-A** didn't work correctly with negative hex and octal numbers. (Steed)

":echo" always started with a blank.

Updating GUI cursor shape didn't always work (e.g., when blinking is off).

In silent Ex mode ("ex -s" or "ex <file") ":s///p" didn't print a line. Also a few other commands that explicitly print a text line didn't work. Made this Vi compatible.

Win32 version of _chdrive() didn't return correct value. (Tsindlekht)

When using 't' in **'complete'** option, no longer give an error message for a missing tags file.

Unix: tgoto() can return NULL, which was not handled correctly in configure.

When doing ":help" from a buffer where **'binary'** is set, also edited the help file in binary mode. Caused extra ^Ms for DOS systems.

Cursor position in a file was reset to 1 when closing a window.

":!ls" in Ex mode switched off echo.

When doing a double click in window A, while currently in window B, first click would reset double click time, had to click three times to select a word.

When using <F11> in mappings, ":mkexrc" produced an exrc file that can't be used in Vi compatible mode. Added setting of **'cpo'** to avoid this. Also, add a **CTRL-V** in front of a '<', to avoid a normal string to be interpreted as a special key name.

Gave confusing error message for ":set guifont=-*-lucida-*": first "font is not fixed width", then "Unknown font".

Some options were still completely left out, instead of included as hidden options.

While running the X11 GUI, ignore SIGHUP signals. Avoids a crash after executing an external command (in rare cases).

In `os_unixx.h`, `signal()` was defined to `sigset()`, while it already was.

Memory leak when executing autocommands (was reported as a memory leak in syntax highlighting).

Didn't print source of error sometimes, because pointers were the same, although names were different.

Avoid a number of UMR errors from Purify (third argument to `open()`).

A swap file could still be created just after setting `'updatecount'` to zero, when there is an empty buffer and doing `":e file"`. (Kutschera)

Test 35 failed on 64 bit machines. (Schild)

With `"p"` and `"P"` commands, redrawing was slow.

Awk script for html documentation didn't work correctly with AIX awk. Replaced `"[,.);\\]]"` with `"[] ,.);]"`. (Briscoe)
The `makehtml.awk` script had a small problem, causing extra lines to be inserted. (Briscoe)

`"gqqq"` could not be repeated. Repeating for `"gugu"` and `"gUgU"` worked in a wrong way. Also made `"gqq"` work to be consistent with `"guu"`.

C indent was wrong after `"case ':':"`.

`":au BufReadPre *.c put"`: Line from `put` text was deleted, because the buffer was still assumed to be empty.

Text pasted with the Edit/Paste menu was subject to `'textwidth'` and `'autoindent'`. That was inconsistent with using the mouse to paste. Now `"*p` is used.

When using `CTRL-W CTRL-]` on a word that's not a tag, and then `CTRL-]` on a tag, window was split.

`":ts"` got stuck on a tags line that has two extra fields.

In Insert mode, with `'showmode'` on, `<C-O><C-G>` message was directly overwritten by mode message, if preceded with search command warning message.

When putting the result of an expression with `"=<expr>p`, newlines were inserted like `^@` (NUL in the file). Now the string is split up in lines at the newline.

`putenv()` was declared with `"const char *"` in `pty.c`, but with `"char *"` in `osdef2.h.in`. Made the last one also `"const char *"`.

`":help {word}"`, where `+{word}` is a feature, jumped to the feature list instead of where the command was explained. E.g., `":help browse"`, `":help autocmd"`.

Using the "\<xx>" form in an expression only got one byte, even when using a special character that uses several bytes (e.g., "\<F9>").
Changed "\<BS>" to produce **CTRL-H** instead of the special key code for the backspace key. "\" produces 0x7f.

":mkvimrc" didn't write a command to set '**compatible**' or '**nocompatible**'.

The shell syntax didn't contain a "syn sync maxlines" setting. In a long file without recognizable items, syncing took so long it looked like Vim hangs. Added a maxlines setting, and made syncing interruptible.

The "gs" command didn't flush output before waiting.

Memory leaks for:

```
":if 0 | let a = b . c | endif"  
"let a = b[c]"  
":so {file}" where {file} contains a ":while"
```

GUI: allocated fonts were never released. (Leonard)

Makefile.bor:

- Changed \$(DEFINES) into a list of "-D" options, so that it can also be used for the resource compiler. (not tested!)
- "bcc.cfg" was used for all configurations. When building for another configuration, the settings for the previous one would be used. Moved "bcc.cfg" to the object directory. (Geddes)
- Included targets for vimrun, install, ctags and xxd. Changed the default to use the Borland DLL Runtime Library, makes Vim.exe a log smaller. (Aaron)

"2*" search for the word under the cursor with "2" prepended. (Leonard)

When deleting into a specific register, would still overwrite the non-Win32 GUI selection. Now ""x"*P works.

When deleting into the "" register, would write to the last used register. Now ""x always writes to the unnamed register.

GUI Athena: A submenu with a '!' in it didn't work. E.g.,
":amenu Syntax.XY\Z.foo lll".

When first doing ":tag foo" and then ":tnext" and/or ":tselect" the order of matching tags could change, because the current file is different. Now the existing matches are kept in the same order, newly found matches are added after them, not matter what the current file is.

":ta" didn't find the second entry in a tags file, if the second entry was longer than the first one.

When using ":set si tw=7" inserting "foo {^P}" made the "}" inserted at the wrong position. can_si was still TRUE when the cursor is not in the indent of the line.

Running an external command in Win32 version had the problem that Vim exits

when the X on the console is hit (and confirmed). Now use the "vimrun" command to start the external command indirectly. (Negri)

Win32 GUI: When running an external filter, do it in a minimized DOS box. (Negri)

":let" listed variables without translation into printable characters.

Win32 console: When resizing the window, switching back to the old size (when exiting or executing an external command) sometimes failed. (Negri)

This appears to also fix a "non fixable" problem:

Win32 console in NT 4.0: When running Vim in a cmd window with a scrollbar, the scrollbar disappeared and was not restored when Vim exits. This does work under NT 3.51, it appears not to be a Vim problem.

When executing BufDelete and BufUnload autocommands for a buffer without a name, the name of the current buffer was used for <afile>.

When jumping to a tag it reported "tag 1 of >2", while in fact there could be only two matches. Changed to "tag 1 of 2 or more".

":tjump tag" did a linear search in the tags file, which can be slow.

Configure didn't find "LibXm.so.2.0", a Xm library with a version number.

Win32 GUI: When using a shifted key with ALT, the shift modifier would remain set, even when it was already used by changing the used key. E.g., "<M-S-9>" resulted in "<M-S-(>", but it should be "<M-(>". (Negri)

A call to ga_init() was often followed by setting growsize and itemsize. Created ga_init2() for this, which looks better. (Aaron)

Function filereadable() could call fopen() with an empty string, which might be illegal.

X Windows GUI: When executing an external command that outputs text, could write one character beyond the end of a buffer, which caused a crash. (Kohan)

When using "*" or "#" on a string that includes '/' or '?' (when these are included in 'isk'), they were not escaped. (Parmelan)

When adding a ToolBar menu in the Motif GUI, the submenu_id field was not cleared, causing random problems.

When adding a menu, the check if this menu (or submenu) name already exists didn't compare with the simplified version (no mnemonic or accelerator) of the new menu. Could get two menus with the same name, e.g., "File" and "&File".

Breaking a line because of 'textwidth' at the last line in the window caused a redraw of the whole window instead of a scroll. Speeds up normal typing with 'textwidth' a lot for slow terminals.

An invalid line number produced an "invalid range" error, even when it wasn't to be executed (inside "if 0").

When the unnamed, first buffer is re-used, the "BufDelete" autocommand was not called. It would stick in a buffer list menu.

When doing "%" on the NUL after the line, a "{" or "}" in the last character of the line was not found.

The Insert mode menu was not used for the "s" command, the Operator-pending menu was used instead.

With '**compatible**' set, some syntax highlighting was not correct, because of using "[\t]" for a search pattern. Now use the regexps for syntax highlighting like the '**coptions**' option is empty (as was documented already).

When using "map <M-Space> ms" or "map <Space> sss" the output of ":map" didn't show any lhs for the mapping (if '**isprint**' includes 160). Now always use <Space> and <M-Space>, even when they are printable.

Adjusted the Syntax menu, so that the lowest entry fits on a small screen (for Athena, where menus don't wrap).

When using **CTRL-E** or **CTRL-Y** in Insert mode for characters like 'o', 'x' and digits, repeating the insert didn't work.

The file "tools/ccfilter.README.txt" could not be unpacked when using short file names, because of the two dots. Renamed it to "tools/ccfilter_README.txt".

For a dark '**background**', using Blue for Directory and SpecialKey highlight groups is not very readable. Use Cyan instead.

In the function uc_scan_attr() in ex_docmd.c there was a goto that jumped into a block with a local variable. That's illegal for some compilers.

Win32 GUI: There was a row of pixels at the bottom of the window which was not drawn. (Aaron)

Under DOS, editing "filename/" created a swap file of "filename/.swp". Should be "filename/_swp".

Win32 GUI: pointer was hidden when executing an external command.

When '**so**' is 999, "J" near the end of the file didn't redisplay correctly.

":0a" inserted after the first line, instead of before the first line.

Unix: Wildcard expansion didn't handle single quotes and {} patterns. Now ":file 'window.c'" removes the quotes and ":e 'main*.c'" works (literal '*'). ":file {o}{n}{e}" now results in file name "one".

Memory leak when setting a string option back to its default value.

=====

VERSION 5.3	version-5.3
--------------------	-------------

Version 5.3 was a bug-fix version of 5.2. There are not many changes.
Improvements made between version 5.2 and 5.3:

Changed

changed-5.3

Renamed "IDE" menu to "Tools" menu.

Added

added-5.3

Win32 GUI: Give a warning when Vim is activated, and one of the files changed since editing started. (Negri)

Fixed

fixed-5.3

5.2.1: Win32 GUI: space for external command was not properly allocated, could cause a crash. (Aaron) This was the reason to bring out 5.3 quickly after 5.2.

5.2.2: Some commands didn't complain when used without an argument, although they need one: ":badd", ":browse", ":call", ":confirm", ":behave", ":delfunction", ":delcommand" and ":tearoff".
":endfunction" outside of a function gave wrong error message: "Command not implemented". Should be ":endfunction not inside a function".

5.2.3: Win32 GUI: When gvim was installed in "Program files", or another path with a space in it, executing external commands with vimrun didn't work.

5.2.4: Pasting with the mouse in Insert mode left the cursor on the last pasted character, instead of behind it.

5.2.5: In Insert mode, cursor after the end of the line, a shift-cursor-left didn't include the last character in the selection.

5.2.6: When deleting text from Insert mode (with "<C-O>D" or the mouse), which includes the last character in the line, the cursor could be left on the last character in the line, instead of just after it.

5.2.7: Win32 GUI: scrollbar was one pixel too big.

5.2.8: Completion of "PopUp" menu showed the derivatives "PopUpC", "PopUPi", etc. ":menu" also showed these.

5.2.9: When using two input() functions on a row, the prompt would not be drawn in column 0.

5.2.10: A loop with input() could not be broken with **CTRL-C**.

5.2.11: ":call asdf" and ":call asdf(" didn't give an error message.

5.2.12: Recursively using `":normal"` crashes Vim after a while. E.g.:
`":map gq :normal gq<CR>"`

5.2.13: Syntax highlighting used `'iskeyword'` from wrong buffer. When using `":help"`, then `"/\k*` in another window with `'hlsearch'` set.

5.2.14: When using `":source"` from a function, global variables would not be available unless `"g:"` was used.

5.2.15: XPM files can have the extension `".pm"`, which is the same as for Perl modules. Added `"syntax/pmfile.vim"` to handle this.

5.2.16: On Win32 and Amiga, `"echo expand("%:p:h)"` removed one `dirname` in an empty buffer. `mch_Fullname()` didn't append a slash at the end of a directory name.

Should include the character under the cursor in the Visual area when using `'selection'` "exclusive". This wasn't done for `"%", "e", "E", "t"` and `"f"`.

`""p` would always put register `0`, instead of the unnamed (last used) register. Reverse the change that `""x` doesn't write in the unnamed (last used) register. It would always write in register `0`, which isn't very useful. Use `""-x` for the paste mappings in Visual mode.

When there is one long line on the screen, and `'showcmd'` is off, `"0$"` didn't redraw the screen.

Win32 GUI: When using `'mousehide'`, the pointer would flicker when the cursor shape is changed. (Negri)

When cancelling Visual mode, and the cursor moves to the start, the wanted column wasn't set, `"k"` or `"j"` moved to the wrong column.

When using `":browse"` or `":confirm"`, was checking for a comment and separating bar, which can break some commands.

Included fixes for Macintosh. (Kielhorn)

VERSION 5.4

version-5.4

Version 5.4 adds new features, useful changes and a lot of bug fixes.

Runtime directory introduced

new-runtime-dir

The distributed runtime files are now in `$VIMRUNTIME`, the user files in `$VIM`. You normally don't set `$VIMRUNTIME` but let Vim find it, by using `$VIM/vim{version}`, or use `$VIM` when that doesn't exist. This allows for separating the user files from the distributed files and makes it more easy to upgrade to another version. It also makes it possible to keep two versions of Vim around, each with their own runtime files.

In the Unix distribution the runtime files have been moved to the "runtime" directory. This makes it possible to copy all the runtime files at once, without the need to know what needs to be copied.

The archives for DOS, Windows, Amiga and OS/2 now have an extra top-level "vim" directory. This is to make clear that user-modified files should be put here. The directory that contains the executables doesn't have '-' or '.' characters. This avoids strange extensions.

The \$VIM and \$VIMRUNTIME variables are set when they are first used. This allows them to be used by Perl, for example.

The runtime files are also found in a directory called "\$VIM/runtime". This helps when running Vim after just unpacking the runtime archive. When using an executable in the "src" directory, Vim checks if "vim54" or "runtime" can be added after removing it. This make the runtime files be found just after compiling.

A default for \$VIMRUNTIME can be given in the Unix Makefile. This is useful if \$VIM doesn't point to above the runtime directory but to e.g., "/etc/".

Filetype introduced

[new-filetype-5.4](#)

Syntax files are now loaded with the new FileType autocommand. Old "mysyntaxfile" files will no longer work. [filetypes](#)

The scripts for loading syntax highlighting have been changed to use the new Syntax autocommand event.

This combination of Filetype and Syntax events allows tuning the syntax highlighting a bit more, also when selected from the Syntax menu. The FileType autocommand can also be used to set options and mappings specifically for that type of file.

The "\$VIMRUNTIME/filetype.vim" file is not loaded automatically. The ":filetype on" command has been added for this. ":syntax on" also loads it.

The '[filetype](#)' option has been added. It is used to trigger the FileType autocommand event, like the '[syntax](#)' option does for the Syntax event.

":set syntax=OFF" and ":set syntax=ON" can be used (in a modeline) to switch syntax highlighting on/off for the current file.

The Syntax menu commands have been moved to \$VIMRUNTIME/menu.vim. The Syntax menu is included both when ":filetype on" and when ":syntax manual" is used.

Renamed the old '[filetype](#)' option to '[osfiletype](#)'. It was only used for RISCOS. '[filetype](#)' is now used for the common file type.

Added the ":syntax manual" command. Allows manual selection of the syntax to be used, e.g., from a modeline.

Vim script line continuation

new-line-continuation

When an Ex line starts with a backslash, it is concatenated to the previous line. This avoids the need for long lines. [line-continuation](#) (Roemer)

Example:

```
if has("dialog_con") ||
  \ has("dialog_gui")
  :let result = confirm("Enter your choice",
                      \ "&Yes\n&No\n&Maybe",
                      \ 2)
endif
```

Improved session files

improved-sessions

New words for **'sessionoptions'**:

- "help" Restore the help window.
- "blank" Restore empty windows.
- "winpos" Restore the Vim window position. Uses the new ":winpos" command
- "buffers" Restore hidden and unloaded buffers. Without it only the buffers in windows are restored.
- "slash" Replace backward by forward slashes in file names.
- "globals" Store global variables.
- "unix" Use unix file format (<NL> instead of <CR><NL>)

The ":mksession" and **'sessionoptions'** are now in the +mksession feature.

The top line of the window is also restored when using a session file.

":mksession" and ":mkvimrc" don't store **'fileformat'**, it should be detected when loading a file.

(Most of this was done by Vince Negri and Robert Webb)

Autocommands improved

improved-autocmds-5.4

New events:

- FileType** When the file type has been detected.
- FocusGained** When Vim got input focus. (Negri)
- FocusLost** When Vim lost input focus. (Negri)
- BufCreate** Called just after a new buffer has been created or has been renamed. (Madsen)
- CursorHold** Triggered when no key has been typed for **'updatetime'**. Can be used to do something with the word under the cursor. (Negri)
Implemented CursorHold autocommand event for Unix. (Zellner)
Also for Amiga and MS-DOS.
- GUIEnter** Can be used to do something with the GUI window after it has

`BufHidden` been created (e.g., a `":winpos 100 50"`).
When a buffer becomes hidden. Used to delete the
option-window when it becomes hidden.

Also trigger `BufDelete` just before a buffer is going to be renamed. (Madsen)

The `"<amatch>"` pattern can be used like `"<afile>"` for autocommands, except that it is the matching value for the `FileType` and `Syntax` events.

When `":let @/ = <string>"` is used in an autocommand, this last search pattern will be used after the autocommand finishes.

Made loading autocommands a bit faster. Avoid doing `strlen()` on each exiting pattern for each new pattern by remembering the length.

Encryption

[new-encryption](#)

Files can be encrypted when writing and decrypted when reading. Added the `'key'` option, `"-x"` command line argument and `":X"` command. [encryption](#) (based on patch from Mohsin Ahmed)

When reading a file, there is an automatic detection whether it has been encrypted. Vim will then prompt for the key.

Note that the encryption method is not compatible with Vi. The encryption is not unbreakable. This allows it to be exported from the US.

GTK GUI port

[new-GTK-GUI](#)

New GUI port for GTK+. Includes a toolbar, menu tearoffs, etc. [gui-gtk](#)
Added the `:helpfind` command. (Kahn and Dalecki)

Menu changes

[menu-changes-5.4](#)

Menus can now also be used in the console. It is enabled by the new `'wildmenu'` option. This shows matches for command-line completion like a menu. This works as a minimal file browser.

The new `:emenu` command can be used to execute a menu item.

Uses the last status line to list items, or inserts a line just above the command line. (Negri)

The `'wildcharx'` option can be used to trigger `'wildmenu'` completion from a mapping.

When compiled without menus, this can be detected with `has("menu")`. Also show this in the `":version"` output. Allow compiling GUI versions without menu

support. Only include toolbar support when there is menu support.

Moved the "Window" menu all the way to the right (priority 70). Looks more familiar for people working with MS-Windows, shouldn't matter for others.

Included "Buffers" menu. Works with existing autocommands and functions. It can be disabled by setting the "no_buffers_menu" variable. (Aaron and Madsen)

Win32 supports separators in a menu: "-.*-". (Geddes)
Menu separators for Motif now work too.

Made Popup menu for Motif GUI work. (Madsen)

'M' flag in **'guioptions'**: Don't source the system menu.

All the menu code has been moved from gui.c to menu.c.

Viminfo improved

[improved-viminfo](#)

New flags for **'viminfo'**:

'!' Store global variables in the viminfo file if they are in uppercase letters. (Negri)

'h' Do ":nohlsearch" when loading a viminfo file.

Store search patterns in the viminfo file with their offset, magic, etc. Also store the flag whether **'hlsearch'** highlighting is on or off (which is not used if the 'h' flag is in **'viminfo'**).

Give an error message when setting **'viminfo'** without commas.

Various new commands

[new-commands-5.4](#)

Operator **g?** : rot13 encoding. (Negri)

zH and **zL** commands: Horizontal scrolling by half a page.

gm move cursor to middle of screen line. (Ideas by Campbell)

Operations on Visual blocks: **v_b_I** , **v_b_A** , **v_b_c** , **v_b_C** , **v_b_r** , **v_b_<** and **v_b_>** . (Kelly)

New command: **CTRL-\ CTRL-N**, which does nothing in Normal mode, and goes to Normal mode when in Insert or Command-line mode. Can be used by VisVim or other OLE programs to make sure Vim is in Normal mode, without causing a beep.
CTRL-_CTRL-N

":cscope kill" command to use the connection filename. **:cscope** (Kahn)

:startinsert command: Start Insert mode next.

:history command, to show all four types of histories. (Roemer)

[m , [M ,]m and]M commands, for jumping backward/forward to start/end of method in a (Java) class.

":@*" executes the * register. :@ (Acevedo)

go and :goto commands: Jump to byte offset in the file.

gR and gr command: Virtual Replace mode. Replace characters without changing the layout. (Webb)

":cd -" changes to the directory from before the previous ":cd" command.
:cd- (Webb)

Tag preview commands :ptag . Shows the result of a ":tag" in a dedicated window. Can be used to see the context of the tag (e.g., function arguments). (Negri)

:pclose command, and CTRL-W CTRL-Z: Close preview window. (Moore)

'previewheight' option, height for the preview window.

Also :ppop , :ptnext , :ptprevious , :ptNext , :ptrewind , :ptlast .

:find and :sfind commands: Find a file in 'path', (split window) and edit it.

The :options command opens an option window that shows the current option values. Or use ":browse set" to open it. Options are grouped by function. Offers short help on each option. Hit <CR> to jump to more help. Edit the option value and hit <CR> on a "set" line to set a new value.

Various new options

new-options-5.4

Scroll-binding: 'scrollbind' and 'scrollopt' options. Added :syncbind command. Makes windows scroll the same amount (horizontally and/or vertically). (Ralston)

'conskey' option for MS-DOS. Use direct console I/O. This should work with telnet (untested!).

'statusline' option: Configurable contents of the status line. Also allows showing the byte offset in the file. Highlighting with %1* to %9*, using the new highlight groups User1 to User9. (Madsen)

'rulerformat' option: Configurable contents of the ruler, like 'statusline'. (Madsen)

'write' option: When off, writing files is not allowed. Avoids overwriting a file even with ":w!". The -m command line option resets 'write'.

'clipboard' option: How the clipboard is used. Value "unnamed": Use unnamed register like "*". (Cortopassi) Value "autoselect": Like what 'a' in

'guioptions' does but works in the terminal.

'guifontset' option: Specify fonts for the +fontset feature, for the X11 GUI versions. Allows using normal fonts when vim is compiled with this feature. (Nam)

'guiheadroom' option: How much room to allow above/below the GUI window. Used for Motif, Athena and GTK.

Implemented **'tagstack'** option: When off, pushing tags onto the stack is disabled (Vi compatible). Useful for mappings.

'shellslash' option. Only for systems that use a backslash as a file separator. This option will use a forward slash in file names when expanding it. Useful when **'shell'** is sh or csh.

'pastetoggle' option: Key sequence that toggles **'paste'**. Works around the problem that mappings don't work in Insert mode when **'paste'** is set.

'display' option: When set to "lastline", the last line fills the window, instead of being replaced with "@" lines. Only the last three characters are replaced with "@@@", to indicate that the line has not finished yet.

'switchbuf' option: Allows re-using existing windows on a buffer that is being jumped to, or split the window to open a new buffer. (Roemer)

'titleold' option. Replaces the fixed string "Thanks for flying Vim", which is used to set the title when exiting. (Schild)

Vim scripts

[new-script-5.4](#)

The `exists()` function can also check for existence of a function. (Roemer)
An internal function is now found with a binary search, should be a bit faster. (Roemer)

New functions:

- `getwinposx()` and `getwinposy()` : get Vim window position. (Webb)
- `histnr()` , `histadd()` , `histget()` and `histdel()` : Make history available. (Roemer)
- `maparg()` : Returns rhs of a mapping. Based on a patch from Vikas.
- `mapcheck()` : Check if a map name matches with an existing one.
- `visualmode()` : Return type of last Visual mode. (Webb)
- `libcall()` : Call a function in a library. Currently only for Win32. (Negri)
- `bufwinnr()` : find window that contains the specified buffer. (Roemer)
- `bufloaded()` : Whether a buffer exists and is loaded.
- `localtime()` and `getftime()` : wall clock time and last modification time of a file (Webb)
- `glob()` : expand file name wildcards only.
- `system()` : get the raw output of an external command. (based on a patch from Aaron).
- `strtrans()` : Translate String into printable characters. Used for 2html.vim script.
- `append()` : easy way to append a line of text in a buffer.

Changed functions:

- Optional argument to `strftime()` to give the time in seconds. (Webb)
- `expand()` now also returns names for files that don't exist.

Allow numbers in the name of a user command. (Webb)

Use "v:" for internal Vim variables: "v:errmsg", "v:shell_error", etc. The ones from version 5.3 can be used without "v:" too, for backwards compatibility.

New variables:

"v:warningmsg" and "v:statusmsg" internal variables. Contain the last given warning and status message. `v:warningmsg` `v:statusmsg` (Madsen)

"v:count1" variable: like "v:count", but defaults to one when no count is used. `v:count1`

When compiling without expression evaluation, "if 1" can be used around the not supported commands to avoid it being executed. Works like in Vim 4.x. Some of the runtime scripts gave errors when used with a Vim that was compiled with minimal features. Now "if 1" is used around code that is not always supported.

When evaluating an expression with `&&` and `||`, skip the parts that will not influence the outcome. This makes it faster and avoids error messages. (Webb)
Also optimized the skipping of expressions inside an "if 0".

Avoid hit-enter prompt

`avoid-hit-enter`

Added 'T' flag to '`shortmess`': Truncate all messages that would cause the hit-enter prompt (unless that would happen anyway).
The 'O' flag in '`shortmess`' now also applies to quickfix messages, e.g., from the `:cn` command.

The default for '`shortmess`' is now "filnxtTo0", to make most messages fit on the command line, and not cause the hit-enter prompt.

Previous messages can be viewed with the new `:messages` command.

Some messages are shown fully, even when '`shortmess`' tells to shorten messages, because the user is expected to want to see them in full: **CTRL-G** and some quickfix commands.

Improved quickfix

`improved-quickfix`

Parse change-directory lines for gmake: "make[1]: Entering directory '`name`'".
Uses "%D" and "%X" in '`errorformat`'.
Also parse "Making `{target}` in `{dir}`" messages from make. Helps when not using GNU make. (Schandl)

Use '`isfname`' for "%f" in '`errorformat`'.

Parsing of multi-line messages. [errorformat-multi-line](#)

Allow a range for the `:clist` command. (Roemer)

Support for "global" file names, for error formats that output the file name once for several errors. (Roemer)

`:cnfile` jumps to first error in next file.

"\$*" in `'makeprg'` is replaced by arguments to `:"make"`. (Roemer)

Regular expressions

[regexp-changes-5.4](#)

In a regexp, a '\$' before "\)" is also considered to be an end-of-line. `/$`
In patterns "^" after "\|" or "\(" is a start-of-line. `/^` (Robinson)

In a regexp, in front of "\)" and "\|" both "\$" and "\\$" were considered end-of-line. Now use "\$" as end-of-line and "\\$" for a literal dollar. Same for '^' after "\(" and "\|". `/\$$` `/\^`

Some search patterns can be extremely slow, even though they are not really illegal. For example: `"\([^a-z]\+\)\+Q"`. Allow interrupting any regexp search with **CTRL-C**.

Register `"/:` last search string (read-only). (Kohan) Changed to use last used search pattern (like what `'hlsearch'` uses). Can set the search pattern with `:"let @/ = {expr}"`.

Added character classes to search patterns, to avoid the need for removing the 'l' flag from `'coptions'`: `[:tab:]`, `[:return:]`, `[:backspace:]` and `[:escape:]`.

By adding a '?' after a comparative operator in an expression, the comparison is done by ignoring case. `expr==?`

Other improvements made between version 5.3 and 5.4

Changed

[changed-5.4](#)

Unix: Use \$TMPDIR for temporary files, if it is set and exists.

Removed "Empty buffer" message. It isn't useful and can cause a hit-enter prompt. (Negri)

"ex -" now reads commands from stdin and works in silent mode. This is to be compatible with the original "ex" command that is used for scripts.

Default range for `:"tcldo"` is the whole file.

Cancelling Visual mode with ESC moved the cursor. There appears to be no reason for this. Now leave the cursor where it is.

The ":grep" and ":make" commands see " as part of the arguments, instead of the start of a comment.

In expressions the "=~" and "!~" operators no longer are affected by **'ignorecase'**.

Renamed vimrc_example to vimrc_example.vim and gvimrc_example to gvimrc_example.vim. Makes them being recognized as vim scripts.

"gd" no longer starts searching at the end of the previous function, but at the first blank line above the start of the current function. Avoids that using "gd" in the first function finds global a variable.

Default for **'complete'** changed from ".,b" to ".,w,b,u,t,i". Many more matches will be found, at the cost of time (the search can be interrupted).

It is no longer possible to set 'shell*' options from a modeline. Previously only a warning message was given. This reduces security risks.

The ordering of the index of documentation files was changed to make it more easy to find a subject.

On MS-DOS and win32, when \$VIM was not set, \$HOME was used. This caused trouble if \$HOME was set to e.g., "C:\\" for some other tool, the runtime files would not be found. Now use \$HOME only for _vimrc, _gvimrc, etc., not to find the runtime file.

When **'tags'** is "./{fname}" and there is no file name for the current buffer, just use it. Previously it was skipped, causing "vim -t {tag}" not to find many tags.

When trying to select text in the **'scrolloff'** area by mouse dragging, the resulting scrolling made this difficult. Now **'scrolloff'** is temporarily set to 0 or 1 to avoid this. But still allow scrolling in the top line to extend to above the displayed text.

Default for **'comments'** now includes "sl:/*,mb: *,ex:*/", to make javadoc comments work. Also helps for C comments that start with "/******".

CTRL-X CTRL-] Insert mode tag expansion tried to expand to all tags when used after a non-ID character, which can take a very long time. Now limit this to 200 matches. Also used for command-line tag completion.

The OS/2 distribution has been split in two files. It was too big to fit on a floppy. The same runtime archive as for the PC is now used.

In the documentation, items like <a-z> have been replaced with {a-z} for non-optional arguments. This avoids confusion with key names: <C-Z> is a **CTRL-Z**, not a character between C and Z, that is {C-Z}.

Added

added-5.4

Color support for the iris-ansi builtin termcap entry. (Tubman)

Included VisVim version 1.3a. (Erhardt)

Win32 port for SNIFF+ interface. (Leherbauer)

Documentation file for sniff interface: if_sniff.txt. (Leherbauer)

Included the "SendToVim" and "OpenWithVim" programs in the OleVim directory. To be used with the OLE version of gvim under MS-Windows. (Schaller)

Included Exuberant Ctags version 3.2.4 with Eiffel support. (Hiebert)

When a file that is being edited is deleted, give a warning (like when the time stamp changed).

Included newer versions of the HTML-generating Awk and Perl scripts. (Colombo)

Linux console mouse support through "gpm". (Tsindlekht)

Security fix: Disallow changing **'secure'** and **'exrc'** from a modeline. When **'secure'** is set, give a warning for changing options that contain a program name.

Made the Perl interface work with Perl 5.005 and threads. (Verdoolaege)

When giving an error message for an ambiguous mapping, include the offending mapping. (Roemer)

Command line editing:

- Command line completion of mappings. (Roemer)
- Command line completion for ":function", ":delfunction", ":let", ":call", ":if", etc. (Roemer)
- When using **CTRL-D** completion for user commands that have "-complete=tag_listfiles" also list the file names. (Madsen)
- Complete the arguments of the ":command" command. (Webb)
- **CTRL-R** . in command line inserts last inserted text. **CTRL-F**, **CTRL-P**, **CTRL-W** and **CTRL-A** after **CTRL-R** are used to insert an object from under the cursor. (Madsen)

Made the text in uganda.txt about copying Vim a bit more clear.

Updated the Vim tutor. Added the "vimtutor" command, which copies the tutor and starts Vim on it. "make install" now also copies the tutor.

In the output of ":clist" the current entry is highlighted, with the 'i' highlighting (same as used for **'incsearch'**).

For the ":clist" command, you can scroll backwards with "b" (one screenful), "u" (half a screenful) and "k" (one line).

Multi-byte support:

- X-input method for multibyte characters. And various fixes for multibyte support. (Nam)
- Hangul input method feature: `hangul`. (Nam)
- Cleaned up configuration of multibyte support, XIM, fontset and Hangul input. Each is now configurable separately.
- Changed check for GTK_KEYBOARD to HANGUL_KEYBOARD_TYPE. (Nam)
- Added doc/hangulin.txt: Documentation for the Hangul input code. (Nam)
- XIM support for GTK+. (Nam)
- First attempt to include support for SJIS encoding. (Nagano)
- When a double-byte character doesn't fit at the end of the line, put a "~" there and print it on the next line.
- Optimize output of multibyte text. (Park)
- Win32 IME: preedit style is like over-the-spot. (Nagano)
- Win32 IME: IME mode change now done with ImmSetOpenStatus. (Nagano)
- GUI Athena: file selection dialog can display multibyte characters. (Nagano)
- Selection reply for XA_TEXT as XA_STRING. (Nagano)

"runtime/macros/diffwin.vim". Mappings to make a diff window. (Campbell)

Added ".obj" to the `'suffixes'` option.

Reduced size of syntax/synload.vim by using the `":SynAu"` user command.

Automated numbering of Syntax menu entries in menu.vim.

In the Syntax menu, insert separators between syntax names that start with a different letter. (Geddes)

Xterm:

- Clipboard support when using the mouse in an xterm. (Madsen)
- When using the xterm mouse, track dragging of the mouse. Use xterm escape sequences when possible. It is more precise than other methods, but requires a fairly recent xterm version. It is enabled with "xterm2" in `'ttymouse'`. (Madsen)
- Check xterm patch level, to set the value of `'ttymouse'`. Has only been added to xterm recently (patch level > 95). Uses the new `'t_RV'` termcap option. Set `'ttymouse'` to "xterm2" when a correct response is recognized. Will make xterm mouse dragging work better.
- Support for shifted function keys on xterm. Changed codes for shifted cursor keys to what the xterm actually produces. Added codes for shifted `<End>` and `<Home>`.
- Added `'t_WP'` to set the window position in pixels and `'t_WS'` to set the window size in characters. Xterm can now move (used for `":winpos"`) and resize (use for `":set lines="` and `":set columns="`).

X11:

- When in Visual mode but not owning the selection, display the Visual area with the VisualNOS group to show this. (Madsen)
- Support for requesting the type of clipboard support. Used for AIX and dtterm. (Wittig)
- Support compound_text selection (even when compiled without multibyte).

Swap file:

- New variation for naming swap files: Replace path separators into %, place

all swap files in one directory. Used when a name in **'dir'** ends in two path separators. (Madsen)

- When a swap file is found, show whether it contains modifications or not in the informative message. (Madsen)
- When dialogs are supported, use a dialog to ask the user what to do when a swapfile already exists.

"popup_setpos" in **'mousemodel'** option. Allows for moving the cursor when using the right mouse button.

When a buffer is deleted, the selection for which buffer to display instead now uses the most recent entry from the jump list. (Madsen)

When using **CTRL-O/CTRL-I**, skip deleted buffers.

A percentage is shown in the ruler, when there is room.

Used autoconf 1.13 to generate configure.

Included get_lisp_indent() from Dirk van Deun. Does better Lisp indenting when 'p' flag in **'cptions'** is not included.

Made the 2html.vim script quite a bit faster. (based on ideas from Geddes)

Unix:

- Included the name of the user that compiled Vim and the system name it was compiled on in the version message.
- "make install" now also installs the "tools" directory. Makes them available for everybody.
- "make check" now does the same as "make test". "make test" checks for Visual block mode shift, insert, replace and change.
- Speed up comparing a file name with existing buffers by storing the device/inode number with the buffer.
- Added configure arguments "--disable-gtk", "--disable-motif" and "--disable-athena", to be able to disable a specific GUI (when it doesn't work).
- Renamed the configure arguments for disabling the check for specific GUIs. Should be clearer now. (Kahn)
- On a Digital Unix system ("OSF1") check for the curses library before termcap and termcap. (Schild)
- "make uninstall_runtime" will only delete the version-specific files. Can be used to delete the runtime files of a previous version.

Macintosh: (St-Amant)

- Dragging the scrollbar, like it's done for the Win32 GUI. Moved common code from gui_w32.c to gui.c
- Added dialogs and file browsing.
- Resource fork preserved, warning when it will be lost.
- Copy original file attributes to newly written file.
- Set title/notitle bug solved.
- Filename completion improved.
- Grow box limit resize to a char by char size.
- Use of rgb.txt for more colors (but give back bad color).
- Apple menu works (beside the about...).

- Internal border now vim compliant.
- Removing a menu doesn't crash anymore.
- Weak-linking of Python 1.5.1 (only on PPC). Python is supported when the library is available.
- If an error is encountered when sourcing the users .vimrc, the alert box now shows right away with the OK button defaulted. There's no more "Delete"-key sign at the start of each line
- Better management of environment variables. Now \$VIM is calculated only once, not regenerated every time it is used.
- No more CPU hog when in background.
- In a sourced Vim script the Mac file format can be recognized, just like DOS file format is.

When both "unix" and "mac" are present in '**fileformats**', prefer "mac" format when there are more CR than NL characters.

When using "mac" fileformat, use CR instead of a NL, because NL is used for NUL. Will preserve all characters in a file. (Madsen)

The DOS install.exe now contains checks for an existing installation. It avoids setting \$VIM and \$PATH again.

The install program for Dos/Windows can now install Vim in the popup menu, by adding two registry keys.

Port to EGCS/mingw32. New Makefile.ming. (Aaron)

DOS 16 bit: Don't include cursor shape stuff. Save some bytes.

TCL support to Makefile.w32. (Duperval)

OS/2: Use argv[0] to find runtime files.

When using "gf" to go to a buffer that has already been used, jump to the line where the cursor last was.

Colored the output of ":tselect" a bit more. Different highlighting between tag name and file name. Highlight field name ("struct:") separately from argument.

Backtick expansion for non-Unix systems. Based on a patch from Aaron.

Allows the use of things like ":n `grep -l test *.c`" and "echo expand('`ls m*`')".

Check for the '**complete**' option when it is set. (Acevedo)

'd' flag in '**complete**' searches for defined names or macros.

While searching for Insert mode completions in include files and tags files, check for typeahead, so that you can use matches early. (Webb)

The '.' flag in '**complete**' now scans the current buffer completely, ignoring '**nowrapscan**'. (Webb)

Added '~' flag to '**whichwrap**'. (Acevedo)

When ending the Visual mode (e.g., with ESC) don't grab ownership of the selection.

In a color terminal, "fg" and "bg" can be used as color names. They stand for the "Normal" colors.

A few cscope cleanups. (Kahn)

Included changed vimspell.sh from Schemenauer.

Concatenation of strings in an expression with "." is a bit faster. (Roemer)

The ":redir" command can now redirect to a register: ":redir @r". (Roemer)

Made the output of ":marks" and ":jumps" look similar. When the mark is in the current file, show the text at the mark. Also for ":tags".

When configure finds ftello() and fseeko(), they are used in tag.c (for when you have extremely big tags files).

Configure check for "-F0limit,2000" argument for the compiler. (Borsenkow)

GUI:

- When using ":gui" in a non-GUI Vim, give a clear error message.
- "gvim -v" doesn't start the GUI (if console support is present).
- When in Ex mode, use non-Visual selection for the whole screen.
- When starting with "gvim -f" and using ":gui" in the .gvimrc file, Vim forked anyway. Now the "-f" flag is remembered for ":gui". Added "gui -b" to run gvim in the background anyway.

Motif GUI:

- Check for "-lXp" library in configure (but it doesn't work yet...).
- Let configure check for Lesstif in "/usr/local/Lesstif/Motif*". Changed the order to let a local Motif version override a system standard version.

Win32 GUI:

- When using "-register" or "-unregister" in the non-OLE version, give an error message.
- Use GTK toolbar icons. Make window border look better. Use sizing handles on the lower left&right corners of the window. (Negri)
- When starting an external command with "!:start" and the command can not be executed, give an error message. (Webb)
- Use sizing handles for the grey rectangles below the scrollbars. Can draw toolbar in flat mode now, looks better. (Negri)
- Preparations for MS-Windows 3.1 addition. Mostly changing WIN32 to MSWIN and USE_GUI_WIN32 to USE_GUI_MSWIN. (Negri)

Avoid allocating the same string four times in buflist_findpat(). (Williams)

Set title and icon text with termcap options '**t_ts**', '**t_fs**', '**t_IS**' and '**t_IE**'. Allows doing this on any terminal that supports setting the title and/or icon text. (Schild)

New 'x' flag in '**comments**': Automatically insert the end part when its last character is typed. Helps to close a /* */ comment in C. (Webb)

When expand() has a second argument which is non-zero, don't use '**suffices**'

and **'wildignore'**, return all matches.

'O' flag in **'cptions'** When not included, Vim will not overwrite a file, if it didn't exist when editing started but it does exist when the buffer is written to the file. The file must have been created outside of Vim, possibly without the user knowing it. When this is detected after a shell command, give a warning message.

When editing a new file, **CTRL-G** will show [New file]. When there were errors while reading the file, **CTRL-G** will show [Read errors].

":wall" can now use a dialog and file-browsing when needed.

Grouped functionality into new features, mainly to reduce the size of the minimal version:

+linebreak: **'showbreak'**, **'breakat'** and **'linebreak'**
+visualextra: "I"nsert and "A"ppend in Visual block mode, "c"hange all lines in a block, ">" and "<": Shifting a block, "r": Replacing a Visual area with one character.
+comments: **'comments'**
+cmdline_info: **'ruler'** and **'showcmd'**. Replaces +showcmd.
"+title" Don't add code to set title or icon for MSDOS, this was not possible anyway.
+cmdline_compl Disable commandline completion at compile time, except for files, directories and help items.

Moved features from a list of function calls into an array. Should save a bit of space.

While entering the body of a function, adjust indent according to "if" and "while" commands.

VMS: Adjusted os_vms.mms a bit according to suggestions from Arpadffy.

The flags in the **'comments'** option can now include an offset. This makes it possible to align **"/*****"**, **"/* xxx"** and **"/*"** comments with the same **'comments'** setting. The default value for **'comments'** uses this.

Added 'O' flag: Don't use this part for the "O" command. Useful for "set com=s0:* \ -,m0:* \ \ ,ex0:*/"

FileType autocommands recognize ".bak", ".orig" and "~" extensions and remove them to find the relevant extension.

The tutorial for writing a Vim script file has been extended.

Some more highlighting in help files, for items that are not typed literally.

Can use **"CTRL-W CTRL-G"** like **"CTRL-W g"**.

"make test" for OS/2.

Adjusted configure to automatically use the GUI for BeOS.

Fixed

fixed-5.4

5.3.1: When using an autocommand for BufWritePre that changes the name of the buffer, freed memory would be used. (Geddes)

Mac: Compiler didn't understand start of skip_class_name().

Win32 GUI:

- When cancelling the font requester, don't give an error message.
- When a tearoff-menu is open and its menu is deleted, Vim could crash. (Negri)
- There was a problem on Windows 95 with (un)maximizing the window. (Williams)
- when **'mousehide'** is set, the mouse would stay hidden when a menu is dropped with the keyboard. (Ralston)
- The tempname() function already created the file. Caused problems when using ":w". Now the file is deleted.
- Cursor disappeared when ending up in the top-left character on the screen after scrolling. (Webb)
- When adding a submenu for a torn-off menu, it was not updated.
- Menu tooltip was using the toolbar tooltip. (Negri)
- Setting **'notitle'** didn't remove the title. (Steed)
- Using "!:start cmd" scrolled the screen one line up, and didn't wait for return when the command wasn't found.

Cscope interface: Sorting of matches was wrong. Starting the interface could fail. (Kahn)

Motif GUI: Could not compile with Motif 1.1, because some tear-off functionality was not in #ifdefs.

Configure could sometimes not compile or link the test program for sizeof(int) properly. This caused alignment problems for the undo structure allocations. Added a safety check that SIZEOF_INT is not zero.

Added configure check to test if strings.h can be included after string.h. Some systems can't handle it.

Some systems need both string.h and strings.h included. Adjusted vim.h for that. Removed including string.h from os_unix.h, since it's already in vim.h. (Savage)

AIX: defining _NO_PROTO in os_unix.h causes a conflict between string.h and strings.h, but after the configure check said it was OK. Also define _NO_PROTO for AIX in the configure check. (Winn)

When closing a window with **CTRL-W c**, the value of **'hidden'** was not taken into account, the buffer was always unloaded. (Negri)

Unix Makefile: "make install" always tried to rename an older executable and remove it. This caused an error message when it didn't exit. Added a check for the existence of an old executable.

The command line for "make install" could get too long, because of the many syntax files. Now first do a "cd" to reduce the length.

On RISCOS and MSDOS, reading a file could fail, because the short filename was used, which can be wrong after a "!:cd".

In the DOS versions, the wrong install.exe was included (required Windows). Now the install.exe version is included that is the same as the Vim version. This also supports long file names where possible.

When recording, and stopping while in Insert mode with **CTRL-O** q, the **CTRL-O** would also be recorded.

32bit DOS version: "vim \file", while in a subdirectory, resulted in "new file" for "file" in the local directory, while "\file" did exist. When "file" in the current directory existed, this didn't happen.

MSDOS: Mouse could not go beyond 80 columns in 132 columns mode. (Young)

"make test" failed in the RedHat RPM, because compatible is off by default.

In Insert mode **<C-O><C-W><C-W>** changes to other window, but the status bars were not updated until another character was typed.

MSDOS: environment options in lowercase didn't work, although they did in the Win32 versions. (Negri)

After ":%nohlsearch", a tag command switched highlighting back on.

When using "append" command as the last line in an autocommand, Vim would crash.

RISCOS: The scroll bumpers (?) were not working properly. (Leonard)

"zl" and "zh" could move the cursor, but this didn't set the column in which e.g., "k" would move the cursor.

When doing ":%set all%" the value of **'scroll'** was not set correctly. This caused an error message when later setting any other number option.

When **'hlsearch'** highlighting has been disabled with ":%nohlsearch", incremental searching would switch it back on too early.

When listing tags for ":%tselect", and using a non-search command, and the last character was equal to the first (e.g., "99"), the last char would not be shown.

When searching for tags with ":%tag" Vim would assume that all matches had been found when there were still more (e.g. from another tags file).

Win32: Didn't recognize "c:\\" (e.g., in tags file) as absolute path when upper/lowercase was different.

Some xterms (Debian) send **<Esc>OH** for HOME and **<Esc>OF** for END. Added these to the builtin-xterm.

In ex mode, any CR was seen as the end of the line. Only a NL should be

handled that way. broke ":s/foo/some^Mtext/".

In menu.vim, a vmenu was used to override an amenu. That didn't work, because the system menu file doesn't overwrite existing menus. Added explicit vunmenu to solve this.

Configure check for terminal library could find a library that doesn't work at runtime (Solaris: shared library not found). Added a check that a program with tgoto() can run correctly.

Unix: "echo -n" in the Makefile doesn't work on all systems, causing errors compiling pathdef.c. Replaced it with "tr".

Perl: DO_JOIN was redefined by Perl. Undefined it in the perl files.

Various XIM and multibyte fixes:

- Fix user cannot see his language while he is typing his language with off-the-spot method. (Nagano)
- Fix preedit position using text/edit area (using gui.wid). (Nagano)
- remove 'fix dead key' codes. It was needed since XNFocusWindow was "x11_window", XNFocusWindow is now gui.wid. (Nagano)
- Remove some compile warnings and fix typos. (Namsh)
- For status area, check the gtk+ version while Vim runs. I believe it is better than compile time check. (Namsh)
- Remove one FIXME for gtk+-xim. (Namsh)
- XIM: Dead keys didn't work for Czech. (Vyskovsky)
- Multibyte: If user input only 3byte such as mb1_mb2_eng or eng_mb1_mb2 VIM could convert it to special character. (Nam)
- Athena/Motif with XIM: fix preedit area. (Nam)
- XIM: Composed strings were sometimes ignored. Vim crashed when compose string was longer than 256 bytes. IM's geometry control is fixed. (Nam, Nagano)
- Win32 multibyte: hollowed cursor width on a double byte char was wrong. (Nagano)
- When there is no GUI, selecting XIM caused compilation problems. Automatically disable XIM when there is no GUI in configure.
- Motif and Athena: When compiled with XIM, but the input method was not enabled, there would still be a status line. Now the status line is gone if the input method doesn't work. (Nam)

Win32: tooltip was not removed when selecting a parent menu (it was when selecting a menu entry). (Negri)

Unix with X: Some systems crash on exit, because of the XtCloseDisplay() call. Removed it, it should not be necessary when exiting.

Win32: Crash on keypress when compiled with Borland C++. (Aaron)

When checking for Motif library files, prefer the same location as the include files (with "include" replaced with "lib") above another entry.

Athena GUI: Changed "XtOffset()" in gui_at_fs.c to "XtOffsetOf()", like it's used in gui_x11.c.

Win32: When testing for a timestamp of a file on floppy, would get a dialog box when the floppy has been removed. Now return with an error. (Negri)

Win32 OLE: When forced to come to the foreground, a minimized window was still minimized, now it's restored. (Zivkov)

There was no check for a positive **'shiftwidth'**. A negative value could cause a hangup, a zero value a crash.

Athena GUI: horizontal scrollbar wasn't updated correctly when clicking right or left of the thumb.

When making a Visual-block selection in one window, and trying to scroll another, could cause errors for accessing non-existent line numbers.

When **'matchpairs'** contains "':"`, jumping from the ` to the ' didn't work properly.

Changed '\'" to '"' to make it compatible with old C compilers.

The command line expansion for mappings caused a script with a TAB between lhs and rhs of a map command to fail. Assume the TAB is to separate lhs and rhs when there are no mappings to expand.

When editing a file with very long lines with **'scrolloff'** set, "j" would sometimes end up in a line which wasn't displayed.

When editing a read-only file, it was completely read into memory, even when it would not fit. Now create a swap file for a read-only file when running out of memory while reading the file.

When using ":set cino={s,e-s", a line after "} else {" was not indented properly. Also added a check for this in test3.in.

The Hebrew mapping for the command line was remembered for the next command line. That isn't very useful, a command is not Hebrew. (Kol)

When completing file names with embedded spaces, like "Program\ files", this didn't work. Also for user commands. Moved backslash_half() down to mch_expandpath().

When using "set mouse=a" in Ex mode, mouse events were handled like typed text. Then typing "quit" screwed up the mouse behavior of the xterm.

When repeating an insert with "." that contains a **CTRL-Y**, a number 5 was inserted as "053".

Yanking a Visual area, with the cursor past the line, didn't move the cursor back onto the line. Same for "~", "u", "U" and "g?"

Win32: Default for **'grepprog'** could be "findstr /n" even though there is no findstr.exe (Windows 95). Check if it exists, and fall back to "grep -n" if it doesn't.

Because `gui_mouse_moved()` inserted a leftmouse click in the input buffer, remapping a leftmouse click caused strange effects. Now Insert another code in the input buffer. Also insert a leftmouse release, to avoid the problem with `":map <LeftMouse> l"` that the next release is seen as the release for the focus click.

With `'wrap'` on, when using a line that doesn't fit on the screen, if the start of the Visual area is before the start of the screen, there was no highlighting. Also, `'showbreak'` doesn't work properly.

DOS, Win32: A pattern `"[0-9]\+"` didn't work in autocommands.

When creating a swap file for a buffer which isn't the current buffer, could get a mixup of short file name, resulting in a long file name when a short file name was required. `makeswapname()` was calling `modname()` instead of `buf_modname()`.

When a function caused an error, and the error message was very long because of recursiveness, this would cause a crash.

`'suffixes'` were always compared with matching case. For MS-DOS, Win32 and OS/2 case is now ignored.

The use of `CHARBITS` in `regexp.c` didn't work on some Linux. Don't use it.

When generating a script file, `'cpo'` was made empty. This caused backslashes to disappear from mappings. Set it to `"B"` to avoid that.

Lots of typos in the documentation. (Campbell)

When editing an existing (hidden) buffer, jump to the last used cursor position. (Madsen)

On a Sun the xterm screen was not restored properly when suspending. (Madsen)

When `$VIMINIT` is processed, `'nocompatible'` was only set after processing it.

Unix: Polling for a character wasn't done for GPM, Sniff and Xterm clipboard all together. Cleaned up the code for using `select()` too.

When executing external commands from the GUI, some typeahead was lost. Added some code to regain as much typeahead as possible.

When the window height is 5 lines or fewer, `<PageDown>` didn't use a one-line overlap, while `<PageUp>` does. Made sure that `<PageUp>` uses the same overlap as `<PageDown>`, so that using them both always displays the same lines.

Removed a few unused functions and variables (found with `lint`).

Dictionary completion didn't use `'infercase'`. (Raul)

Configure tests failed when the Perl library was not in `LD_LIBRARY_PATH`. Don't use the Perl library for configure tests, add it to the linker line only when linking Vim.

When using ncurses/terminfo, could get a '**t_Sf**' and '**t_Sb**' termcap entry that has "%d" instead of "%p1%d". The light background colors didn't work then.

GTK GUI with ncurses: Crashed when starting up in tputs(). Don't use tputs() when the GUI is active.

Could use the ":let" command to set the "count", "shell_error" and "version" variables, but that didn't work. Give an error message when trying to set them.

On FreeBSD 3.0, tclsh is called tclsh8.0. Adjusted configure.in to find it.

When Vim is linked with -lncurses, but python uses -ltermcap, this causes trouble: "OOPS". Configure now removes the -ltermcap.

:@ and :* didn't work properly, because the " was recognized as the start of a comment.

Win32s GUI: Minimizing the console where a filter command runs in caused trouble for detecting that the filter command has finished. (Negri)

After executing a filter command from an xterm, the mouse would be disabled. It would work again after changing the mode.

Mac GUI: Crashed in newenv(). (St-Amant)

The menus and mappings in mswin.vim didn't handle text ending in a NL correctly. (Acevedo)

The ":k" command didn't check if it had a valid argument or extra characters. Now give a meaningful error message. (Webb)

On SGI, the signal function doesn't always have three arguments. Check for struct sigcontext to find out. Might still be wrong...

Could crash when using '**hlsearch**' and search pattern is "^".

When search patterns were saved and restored, status of no_hlsearch was not also saved and restored (from ":nohlsearch" command).

When using setline() to make a line shorter, the cursor position was not adjusted.

MS-DOS and Win95: When trying to edit a file and accidentally adding a slash or backslash at the end, the file was deleted. Probably when trying to create the swap file. Explicitly check for a trailing slash or backslash before trying to read a file.

X11 GUI: When starting the GUI failed and received a deadly signal while setting the title, would lock up when trying to exit, because the title is reset again. Avoid using mch_settitle() recursively.

X11 GUI: When starting the GUI fails, and then trying it again, would crash,

because `argv[]` has been freed and `x11_display` was reset to `NULL`.

Win32: When `$HOME` was set, would put `"~user"` in the swap file, which would never compare with a file name, and never cause the attention message. Put the full path in the swap file instead.

Win32 console: There were funny characters at the end of the `"vim -r"` swap files message (direct output of `CR CR LF`).

DOS 32 bit: `"vim -r"` put the text at the top of the window.

GUI: With `'mousefocus'` set, got mouse codes as text with `"!sleep 100"` or `"Q"`.

Motif and Win32 GUI: When changing `'guifont'` to a font of the same size the screen wasn't redrawn.

Unix: When using `":make"`, jumping to a file `b.c`, which is already open as a symbolic link `a.c`, opened a new buffer instead of using the existing one.

Inserting text in the current buffer while sourcing the `.vimrc` file would cause a crash or hang. The memfile for the current buffer was never allocated. Now it's allocated as soon as something is written in the buffer.

DOS 32 bit: `"lightblue"` background worked for text, but not drawn parts were black.

DOS: Colors of console were not restored upon exiting.

When recording, with `'cmdheight'` set to 2 and typing `Esc>` in Insert mode caused the `"recording"` message to be doubled.

Spurious `"file changed"` messages could happen on Windows. Now tolerate a one second difference, like for Linux.

GUI: When returning from Ex mode, scrollbars were not updated.

Win32: Copying text to the clipboard containing a `<CR>`, pasting it would replace it with a `<NL>` and drop the next character.

Entering a double byte character didn't work if the second byte is in `[xXo0]`. (Eric Lee)

`vim_realloc` was both defined and had a prototype in `proto/misc2.pro`. Caused conflicts on Solaris.

A pattern in an autocommand was treated differently on DOS et al. than on Unix. Now it's the same, also when using backslashes.

When using `<Tab>` twice for command line completion, without a match, the `<Tab>` would be inserted. (Negri)

Bug in MS-Visual C++ 6.0 when compiling `ex_docmd.c` with optimization. (Negri)

Testing the result of `mktemp()` for failure was wrong. Could cause a crash.

(Peters)

GUI: When checking for a ".gvimrc" file in the current directory, didn't check for a "_gvimrc" file too.

Motif GUI: When using the popup menu and then adding an item to the menu bar, the menu bar would get very high.

Mouse clicks and special keys (e.g. cursor keys) quit the more prompt and dialogs. Now they are ignored.

When at the more-prompt, xterm selection didn't work. Now use the 'r' flag in **'mouse'** also for the more-prompt.

When selecting a Visual area of more than 1023 lines, with **'guioptions'** set to "a", could mess up the display because of a message in free_yank(). Removed that message, except for the Amiga.

Moved auto-selection from ui_write() to the screen update functions. Avoids unexpected behavior from a low-level function. Also makes the different feedback of owning the selection possible.

Vi incompatibility: Using "i<CR>" in an indent, with **'ai'** set, used the original indent instead of truncating it at the cursor. (Webb)

":echo x" didn't stop at "q" for the more prompt.

Various fixes for Macintosh. (St-Amant)

When using **'selectmode'** set to "exclusive", selecting a word and then using **CTRL-]** included the character under the cursor.

Using ":let a:name" in a function caused a crash. (Webb)

When using ":append", an empty line didn't scroll up.

DOS etc.: A file name starting with '!' didn't work. Added '!' to default for **'isfname'**.

BeOS: Compilation problem with prototype of skip_class_name(). (Price)

When deleting more than one line, e.g., with "de", could still use "U" command, which didn't work properly then.

Amiga: Could not compile ex_docmd.c, it was getting too big. Moved some functions to ex_cmds.c.

The expand() function would add a trailing slash for directories.

Didn't give an error message when trying to assign a value to an argument of a function. (Webb)

Moved including sys/psem.h to after termios.h. Needed for Sinix.

OLE interface: Don't delete the object in CVimCF::Release() when the reference count becomes zero. (Cordell)
VisVim could still crash on exit. (Erhardt)

"case a: case b:" (two case statements in one line) aligned with the second case. Now it uses one 'sw' for indent. (Webb)

Font initialisation wasn't right for Athena/Motif GUI. Moved the call to highlight_gui_started() gui_mch_init() to gui_mch_open(). (Nam)

In Replace mode, backspacing over a TAB before where the replace mode started while 'sts' is different from 'ts', would delete the TAB.

Win32 console: When executing external commands and switching between the two console screens, Vim would copy the text between the buffers. That caused the screen to be messed up for backtick expansion.

":winpos -1" then ":winpos" gave wrong error message.

Windows commander creates files called c:\tmp\%wc\abc.txt. Don't remove the backslash before the \$. Environment variables were not expanded anyway, because of the backslash before the dollar.

Using "--=" with ":set" could remove half a part when it contains a "\",
E.g., ":set path+=a\\,b" and then "set path--=b" removed ",b".

When Visually selecting lines, with 'selection' set to "inclusive", including the last char of the line, "<<" moved an extra line. Also for other operators that always work on lines.

link.sh changed "-lnsl_s" to "_s" when looking for "nsl" to be removed. Now it only removes whole words.

When jumped to a mark or using "fz", and there is an error, the current column was lost. E.g. when using "\$fzj".

The "g **CTRL-G**" command could not be interrupted, even though it can take a long time.

Some terminals do have <F4> and <xF4>. <xF4> was always interpreted as <F4>. Now map <xF4> to <F4>, so that the user can override this.

When compiling os_win32.c with MIN_FEAT the apply_autocmds() should not be used. (Aaron)

This autocommand looped forever: ":au FileChangedShell * ++nested e <file>"
Now FileChangeShell never nests. (Roemer)

When evaluating an ":elseif" that was not going to matter anyway, ignore errors. (Roemer)

GUI Lesstif: Tearoff bar was the last item, instead of the first.

GUI Motif: Colors of tear-off widgets was wrong when 't' flag added to

'**guioptions**' afterwards. When 't' flag in '**guioptions**' is excluded, would still get a tearoff item in a new menu.

An inode number can be "long long". Use ino_t instead of long. Added configure check for ino_t.

Binary search for tags was using a file offset "long" instead of "off_t".

Insert mode completion of tags was not using '**ignorecase**' properly.

In Insert mode, the <xFn> keys were not properly mapped to <Fn> for the default mappings. Also caused errors for ":mkvimrc" and ":mksession".

When jumping to another window while in Insert mode, would get the "warning: changing readonly file" even when not making a change.

A '(' or '{' inside a trailing "/*" comment would disturb C-indenting. When using two labels below each other, the second one was not indented properly. Comments could mess up C-indenting in many places. (Roemer)

Could delete or redefine a function while it was being used. Could cause a crash.

In a function it's logical to prepend "g:" to a system variable, but this didn't work. (Roemer)

Hangul input: Buffer would overflow when user inputs invalid key sequence. (Nam)

When BufLoad or BufEnter autocommands change the topline of the buffer in the window, it was overruled and the cursor put halfway the window. Now only put the cursor halfway if the autocommands didn't change the topline.

Calling exists("&option") always returned 1. (Roemer)

Win32: Didn't take actually available memory into account. (Williams)

White space after an automatically inserted comment leader was not removed when '**ai**' is not set and <CR> hit just after inserting it. (Webb)

A few menus had duplicated accelerators. (Roemer)

Spelling errors in documentation, quite a few "the the". (Roemer)

Missing prototypes for Macintosh. (Kielhorn)

Win32: When using '**shellquote**' or '**shellxquote**', the "!start cmd" wasn't executed in a disconnected process.

When resizing the window, causing a line before the cursor to wrap or unwrap, the cursor was displayed in the wrong position.

There was quite a bit of dead code when compiling with minimal features.

When doing a ":%s///" command that makes lines shorter, such that lines above

the final cursor position no longer wrap, the cursor position was not updated.

get_id_list() could allocate an array one too small, when a "contains=" item has a wildcard that matches a group name that is added just after it. E.g.: "contains=a.*b,axb". Give an error message for it.

When yanking a Visual area and using the middle mouse button -> crash. When clipboard doesn't work, now make "*" always use "".

Win32: Using ":buf a\ b\file" didn't work, it was interpreted as "ab\file".

Using ":ts ident", then hit <CR>, with 'cmdheight' set to 2: command line was not cleared, the tselect prompt was on the last but one line.

mksession didn't restore the cursor column properly when it was after a tab. Could not get all windows back when using a smaller terminal screen. Didn't restore all windows when "winsize" was not in 'sessionoptions'. (Webb)

Command line completion for ":buffer" depended on 'ignorecase' for Unix, but not for DOS et al. Now don't use 'ignorecase', but let it depend on whether file names are case sensitive or not (like when expanding file names).

Win32 GUI: (Negri)

- Redrawing the background caused flicker when resizing the window. Removed _OnEraseBG(). Removed CS_HREDRAW and CS_VREDRAW flags from the sndclass.style.
- Some parts of the window were drawn in grey, instead of using the color from the user color scheme.
- Dropping a file on gvim didn't activate the window.
- When there is no menu ('guioptions' excludes 'm'), never use the ALT key for it.

GUI: When resizing the window, would make the window height a bit smaller. Now round off to the nearest char cell size. (Negri)

In Vi the ")" and "(" commands don't stop at a single space after a dot. Added 'J' flag in 'cptions' to make this behave Vi compatible. (Roemer)

When saving a session without any buffers loaded, there would be a ":normal" command without arguments in it. (Webb)

Memory leaks fixed: (Madsen)

- eval.c: forgot to release func structure when func deleted
- ex_docmd.c: forgot to release string after "<sfile>"
- misc1.c: leak when completion pattern had no matches.
- os_unix.c: forgot to release regexp after file completions

Could crash when using a buffer without a name. (Madsen)

Could crash when doing file name completion, because of backslash_half(). (Madsen)

":@a" would do mappings on register a, which is not Vi compatible. (Roemer)

":g/foo.*()/s/foobar/_&/gc" worked fine, but then "n" searched for "foobar"

and displayed `"/foo.*()`". (Roemer)

OS/2: `get_cmd_output()` was not included. Didn't check for `$VIM/.vimrc` file.

Command line completion of options didn't work after `"+="` and `"-="`.

Unix configure: Test for `memmove()/bcopy()/memcpy()` tried redefining these functions, which could fail if they are defined already. Use `mch_memmove()` to redefine.

Unix: `":let a = expand("`xterm`&")` started an xterm asynchronously, but `":let a = expand("`xterm&`")` generated an error message, because the redirection was put after the `'&'`.

Win32 GUI: Dialog buttons could not be selected properly with cursor keys, when the default is not the first button. (Webb)

The "File has changed since editing started" (when regaining focus) could not always be seen. (Webb)

When starting with `"ex filename"`, the file message was overwritten with the "entering Ex mode" message.

Output of `":tselect"` listed name of file directly from the tags file. Now it is corrected for the position of the tags file.

When **'backspace'** is 0, could backspace over autoindent. Now it is no longer allowed (Vi compatible).

In Replace mode, when **'noexpandtab'** and **'smarttab'** were set, and inserting Tabs, backspacing didn't work correctly for Tabs inserted at the start of the line (unless **'sts'** was set too). Also, when replacing the first non-blank after which is a space, rounding the indent was done on the first non-blank instead of on the character under the cursor.

When **'sw'** at 4, **'ts'** at 8 and **'smarttab'** set: When a tab was appended after four spaces (they are replaced with a tab) couldn't backspace over the tab.

In Insert mode, with **'bs'** set to 0, couldn't backspace to before autoindent, even when it was removed with **CTRL-D**.

When repeating an insert command where a `<BS>`, `<Left>` or other key causes an error, would flush buffers and remain in Insert mode. No longer flush buffers, only beep and continue with the insert command.

Dos and Win32 console: Setting `t_me` didn't work to get another color. Made this works backwards compatible.

For Turkish (`LANG = "tr"`) uppercase `'i'` is not an `'I'`. Use ASCII uppercase translation in `vim_strup()` to avoid language problems. (Komur)

Unix: Use `usleep()` or `nanosleep()` for `mch_delay()` when available. Hopefully this avoids a hangup in `select(0, ..)` for Solaris 2.6.

Vim would crash when using a script file with 'let &sp = "| tee"', starting vim with "vim -u test", then doing ":set sp=". The P_WAS_SET flag wasn't set for a string option, could cause problems with any string option.

When using "cmd | vim -", stdin is not a terminal. This gave problems with GPM (Linux console mouse) and when executing external commands. Now close stdin and re-open it as a copy of stderr.

Syntax highlighting: A "nextgroup" item was not properly stored in the state list. This caused missing of next groups when not redrawing from start to end, but starting halfway.

Didn't check for valid values of **'ttymouse'**.

When executing an external command from the GUI, waiting for the child to terminate might not work, causing a hang. (Parmelan)

"make uninstall" didn't delete the vimrc_example.vim and gvimrc_example.vim files and the vimtutor.

Win32: "expand("%:p:h)") with no buffer name removed the directory name. "fnamemodify("", ":p)") did not add a trailing slash, fname_case() removed it.

Fixed: When **'hlsearch'** was set and the 'c' flag was not in **'coptions'**: highlighting was not correct. Now overlapping matches are handled correctly.

Athena, Motif and GTK GUI: When started without focus, cursor was shown as if with focus.

Don't include **'shellpipe'** when compiled without quickfix, it's not used. Don't include **'dictionary'** option when compiled without the +insert_expand feature.

Only include the **'shelltype'** option for the Amiga.

When making a change to a line, with **'hlsearch'** on, causing it to wrap, while executing a register, the screen would not be updated correctly. This was a generic problem in update_screenline() being called while must_redraw is VALID.

Using ":bdelete" in a BufUnload autocommand could cause a crash. The window height was added to another window twice in close_window().

Win32 GUI: When removing a menu item, the tearoff wasn't updated. (Negri)

Some performance bottlenecks removed. Allocating memory was not efficient. For Win32 checking for available memory was slow, don't check it every time now. On NT obtaining the user name takes a long time, cache the result (for all systems).

fnamemodify() with an argument ":~:." or "::~~" didn't work properly.

When editing a new file and exiting, the marks for the buffer were not saved in the viminfo file.

":confirm only" didn't put up a dialog.

These text objects didn't work when **'selection'** was "exclusive": va(vi(va{ vi{ va< vi< vi[va[.

The dialog for writing a readonly file didn't have a valid default. (Negri)

The line number used for error messages when sourcing a file was reset when modelines were inspected. It was wrong when executing a function.

The file name and line number for an error message wasn't displayed when it was the same as for the last error, even when this was long ago. Now reset the name/lnum after a hit-enter prompt.

In a session file, a "%" in a file name caused trouble, because fprintf() was used to write it to the file.

When skipping statements, a mark in an address wasn't skipped correctly: "ka|if 0 'ad|else|echo endif". (Roemer)

":wall" could overwrite a not-edited file without asking.

GUI: When \$DISPLAY was not set or starting the GUI failed in another way, the console mode then started with wrong colors and skipped initializations. Now do an early check if the GUI can be started. Don't source the menu.vim or gvimrc when it will not. Also do normal terminal initializations if the GUI might not start.

When using a BufEnter autocommand to position the cursor and scroll the window, the cursor was always put at the last used line and halfway the window anyhow.

When **'wildmode'** was set to "longest,list:full", ":e *.c<Tab><Tab>" didn't list the matches. Also avoid that listing after a "longest" lists the wrong matches when the first expansion changed the string in front of the cursor.

When using ":insert", ":append" or ":change" inside a while loop, was not able to break out of it with a **CTRL-C**.

Win32: ":e ." took an awful long time before an error message when used in "C:\". Was caused by adding another backslash and then trying to get the full name for "C:\\".

":winpos -10 100" was working like ":winpos -10 -10", because a pointer was not advanced past the '-' sign.

When obtaining the value of a hidden option, would give an error message. Now just use a zero value.

OS/2: Was using argv[0], even though it was not a useful name. It could be just "vim", found in the search path.

Xterm: ":set columns=78" didn't redraw properly (when lines wrap/unwrap) until after a delay of **'updatetime'**. Didn't check for the size-changed signal.

'**scrollbind**' didn't work in Insert mode.
Horizontal scrollbinding didn't always work for "0" and "\$" commands (e.g., when '**showcmd**' was off).

When compiled with minimal features but with GUI, switching on the mouse in an xterm caused garbage, because the mouse codes were not recognized. Don't enable the mouse when it can't be recognized. In the GUI it also didn't work, the arguments to the mouse code were not interpreted.

When '**showbreak**' used, in Insert mode, when the cursor is just after the last character in the line, which is also the in the rightmost column, the cursor position would be like the '**showbreak**' string is shown, but it wasn't.

Autocommands could move the cursor in a new file, so that **CTRL-W** i didn't show the right line. Same for when using a filemark to jump to another file.

When redefining the argument list, the title used for other windows could be showing the wrong info about the position in the argument list. Also update this for a ":split" command without arguments.

When editing file 97 of 13, ":Next" didn't work. Now it goes to the last file in the argument list.

Insert mode completion (for dictionaries or included files) could not be interrupted by typing an <Esc>. Could get hit-enter prompt after line completion, or whenever the informative message would get too long.

When using the ":edit" command to re-edit the same file, an autocommand to jump to the last cursor position caused the cursor to move. Now set the last used cursor position to avoid this.

When '**comments**' has a part that starts with white space, formatting the comment didn't work.

At the ":tselect" prompt Normal mode mappings were used. That has been disabled.

When '**selection**' is not "old", some commands still didn't allow the cursor past the end-of-line in Visual mode.

Athena: When a menu was deleted, it would appear again (but not functional) when adding another menu. Now they don't reappear anymore (although they are not really deleted either).

Borland C++ 4.x had an optimizer problem in fill_breakat_flags(). (Negri)

"ze" didn't work when '**number**' was on. (Davis)

Win32 GUI: Intellimouse code didn't work properly on Windows 98. (Robinson)

A few files were including proto.h a second time, after vim.h had already done that, which could cause problems with the vim_realloc() macro.

Win32 console: <M-x> or ALT-x was not recognized. Also keypad '+', '-' and '*'. (Negri)

MS-DOS: <M-x> didn't work, produced a two-byte code. Now the alphabetic and number keys work. (Negri)

When finding a lot of matches for a tag completion, the check for avoiding double matches could take a lot of time. Add a line_breakcheck() to be able to interrupt this. (Deshpande)

When the command line was getting longer than the screen, the more-prompt would be given regularly, and the cursor position would be wrong. Now only show the part of the command line that fits on the screen and force the cursor to be positioned on the visible part. There can be text after the cursor which isn't editable.

At the more prompt and with the console dialog, a cursor key was interpreted as <Esc> and OA. Now recognize special keys in get_keystroke(). Ignore mouse and scrollbar events.

When typing a BS after inserting a middle comment leader, typing the last char of the end comment leader still changed it into the end comment leader. (Webb)

When a file system is full, writing to a swap file failed. Now first try to write one block to the file. Try next entry in 'dir' if it fails.

When "~" is in 'whichwrap', doing "~" on last char of a line didn't update the display.

Unix: Expanding wildcards for ":file {\}" didn't work, because "\" was translated to }" before the shell got it. Now don't remove backslashes when wildcards are going to be expanded.

Unix: ":e /tmp/\$uid" didn't work. When expanding environment variables in a file name doesn't work, use the shell to expand the file name. ":e /tmp/\$tty" still doesn't work though.

"make test" didn't always work on DOS/Windows for test30, because it depended on the external "echo" command.

The link.sh script used "make" instead of \$MAKE from the Makefile. Caused problems for generating pathdef.c when "make" doesn't work properly.

On versions that can do console and GUI: In the console a typed CSI code could cause trouble.

The patterns in expression evaluation didn't ignore the 'l' flag in 'coptions'. This broke the working of <CR> in the options window.

When 'hls' off and 'ai' on, "O<Esc>" did remove the indent, but it was still highlighted red for trailing space.

Win32 GUI: Dropping an encrypted file on a running gvim didn't work right. Vim would loop while outputting "*" characters. vgetc() was called recursively, thus it returns NUL. Added safe_vgetc(), which reads input directly from the

user in this situation.

While reading text from stdin, only an empty screen was shown. Now show that Vim is reading from stdin.

The cursor shape wasn't set properly when returning to Insert mode, after using a **CTRL-O** /asdf command which fails. It would be OK after a few seconds. Now it's OK right away.

The **'isfname'** default for DOS/Windows didn't include the '@' character. File names that contained "dir\@file" could not be edited.

Win32 console: <C-S-Left> could cause a crash when compiled with Borland or egcs. (Aaron)

Unix and VMS: "#if HAVE_DIRENT_H" caused problems for some compilers. Use "#ifdef HAVE_DIRENT_H" instead. (Jones)

When a matching tag is in the current file but has a search pattern that doesn't match, the cursor would jump to the first line.

Unix: Dependencies for pty.c were not included in Makefile. Dependency of ctags/config.h was not included (only matters for parallel make).

Removed a few Uninitialized Memory Reads (potential crashes). In do_call() calling clear_var() when not evaluating. In win32_expandpath() and dos_expandpath() calling backslash_halve() past the end of a file name.

Removed memory leaks: Set_vim_var_string() never freed the value. The next_list for a syntax keyword was never freed.

On non-Unix systems, using a file name with wildcards without a match would silently fail. E.g., ":e *.sh". Now give a "No match" error message.

The life/life.mac, urm/urm.mac and hanoi/hanoi.mac files were not recognized as Vim scripts. Renamed them to *.vim.

[Note: some numbered patches are not relevant when upgrading from version 5.3, they have been removed]

Patch 5.4m.1

Problem: When editing a file with a long name, would get the hit-enter prompt, even though all settings are such that the name should be truncated to avoid that. filemess() was printing the file name without truncating it.

Solution: Truncate the message in filemess(). Use the same code as for msg_trunc_attr(), which is moved to the new function msg_may_trunc().

Files: src/message.c, src/proto/message.pro, src/fileio.c

Patch 5.4m.3

Problem: The Motif libraries were not found by configure for Digital Unix.

Solution: Add "/usr/shlib" to the search path. (Andy Kahn)

Files: src/configure.in, src/configure

Patch 5.4m.5

Problem: Win32 GUI: When using the Save-As menu entry and selecting an existing file in the file browser, would get a dialog to confirm overwriting twice. (Ed Krall)

Solution: Removed the dialog from the file browser. It would be nicer to set the "forceit" flag and skip Vim's ":confirm" dialog, but it requires quite a few changes to do that.

Files: src/gui_w32.c

Patch 5.4m.6

Problem: Win32 GUI: When reading text from stdin, e.g., "cat foo | gvim -", a message box would pop up with "-stdin-" (when exiting). (Michael Schaap)

Solution: Don't switch off termcap mode for versions that are GUI-only. They use another terminal to read from stdin.

Files: src/main.c, src/fileio.c

Patch 5.4m.7

Problem: Unix: running configure with --enable-gtk-check, --enable-motif-check, --enable-athena-check or --enable-gtktest had the reverse effect. (Thomas Koehler)

Solution: Use \$enable_gtk_check variable correctly in AC_ARG_ENABLE().

Files: src/configure.in, src/configure

Patch 5.4m.9

Problem: Multi-byte: With wrapping lines, the cursor was sometimes 2 characters to the left. Syntax highlighting was wrong when a double-byte character was split for a wrapping line. When **'showbreak'** was on the splitting also didn't work.

Solution: Adjust getvcol() and win_line(). (Chong-Dae Park)

Files: src/charset.c, src/screen.c

Patch 5.4m.11

Problem: The ":call" command didn't check for illegal trailing characters. (Stefan Roemer)

Solution: Add the check in do_call().

Files: src/eval.c

Patch 5.4m.13

Problem: With the ":s" command:

1. When performing a substitute command, the mouse would be disabled and enabled for every substitution.
2. The cursor position could be beyond the end of the line. Calling line_breakcheck() could try to position the cursor, which causes a crash in the Win32 GUI.
3. When using ":s" in a ":g" command, the cursor was not put on the first non-white in the line.
4. There was a hit-enter prompt when confirming the substitution and the replacement was a bit longer.

Solution:

1. Only disable/enable the mouse when asking for confirmation.
2. Always put the cursor on the first character, it is going to be moved to the first non-blank anyway.
Don't use the cursor position in gui_mch_draw_hollow_cursor(),

- get the character from the screen buffer.
3. Added `global_need_beginline` flag to call `beginline()` after `":g"` has finished all substitutions.
 4. Clear the `need_wait_return` flag after prompting the user.

Files: `src/ex_cmds.c`, `src/gui_w32.c`

Patch 5.4m.14

Problem: When doing `"vim xxx"`, `":opt"`, `":only"` and then `":e xxx"` we end up with two swapfiles for `"xxx"`. That is caused by the `":bdel"` command which is executed when unloading the option-window. Also, there was no check if closing a buffer made the new one invalid, this could cause a crash.

Solution: When closing a buffer causes the current buffer to be deleted, use the new buffer to replace it. Also detect that the new buffer has become invalid as a side effect of closing the current one. Make autocommand that calls `":bdel"` in `optwin.vim` nested, so that the buffer loading it triggers also executes autocommands. Also added a test for this in `test13`.

Files: `runtime/optwin.vim`, `src/buffer.c`, `src/ex_cmds.c`, `src/globals.h`
`src/testdir/test13.in`, `src/testdir/test13.ok`

Patch 5.4m.15

Problem: When using a `BufEnter` autocommand to reload the syntax file, conversion to HTML caused a crash. (Sung-Hyun Nam)

Solution: When using `":syntax clear"` the current stack of syntax items was not cleared. This will cause memory to be used that has already been freed. Added call to `invalidate_current_state()` in `syntax_clear()`.

Files: `src/syntax.c`

Patch 5.4m.17

Problem: When omitting a `)'` in an expression it would not be seen as a failure. When detecting an error inside `()`, there would be an error message for a missing `)'` too. When using `":echo 1+|echo 2"` there was no error message. (Roemer) When using `":exe 1+"` there was no error message. When using `":return 1+"` there was no error message.

Solution: Fix `do_echo()`, `do_execute()` and `do_return()` to give an error message when `eval1()` returns FAIL. Fix `eval6()` to handle trailing `)'` correctly and return FAIL when it's missing.

Files: `src/eval.c`

Patch 5.4m.18

Problem: When using `input()` from inside an expression entered with **CTRL-R** = on the command line, there could be a crash. And the resulting command line was wrong.

Solution: Added `getcmdline_prompt()`, which handles recursive use of `getcmdline()` correctly. It also sets the command line prompt. Removed `cmdline_prompt()`. Also use `getcmdline_prompt()` for getting the crypt key in `get_crypt_key()`.

Files: `src/proto/ex_getln.pro`, `src/ex_getln.c`, `src/eval.c`, `src/misc2.c`

Patch 5.4m.21

Problem: When starting up, the screen structures were first allocated at the minimal size, then initializations were done with Rows possibly different from screen_Rows. Caused a crash in rare situations (GTK with XIM and fontset).

Solution: Call screenalloc() in main() only after calling ui_get_winsize(). Also avoids a potential delay because of calling screenclear() while "starting" is non-zero.

Files: src/main.c

Patch 5.4m.22

Problem: In the GUI it was possible that the screen was resized and the screen structures re-allocated while redrawing the screen. This could cause a crash (hard to reproduce). The call sequence goes through update_screen() .. syntax_start() .. ui_breakcheck() .. gui_resize_window() .. screenalloc().

Solution: Set updating_screen while redrawing. If the window is resized remember the new size and handle it only after redrawing is finished. This also fixes that resizing the screen while still redrawing (slow syntax highlighting) would not work properly. Also disable display_hint, it was never used.

Files: src/globals.h, src/gui.c, src/screen.c, src/proto/gui.pro

Patch 5.4m.23

Problem: When using expand("<word>") when there was no word under the cursor, would get an error message. Same for <WORD> and <file>.

Solution: Don't give an error message, return an empty string.

Files: src/eval.c

Patch 5.4m.24

Problem: ":help \|" didn't find anything. It was translated to "/\|".

Solution: Translate "\|" into "\\bar". First check the table for specific translations before checking for "\x".

Files: src/ex_cmds.c

Patch 5.4m.25

Problem: Unix: When using command line completion on files that contain '''', '''' or '|' the file name could not be used. Adding this file name to the Buffers menu caused an error message.

Solution: Insert a backslash before these three characters. Adjust Mungename() function to insert a backslash before '|'.

Files: src/ex_getln.c, runtime/menu.vim

Patch 5.4m.26

Problem: When using a mapping of two function keys, e.g., <F1><F1>, and only the first char of the second key has been read, the mapping would not be recognized. Noticed on some Unix systems with xterm.

Solution: Add 'K' flag to 'coptions' to wait for the whole key code, even when halfway a mapping.

Files: src/option.h, src/term.c

Patch 5.4m.27

Problem: When making test33 without the lisp feature it hangs. Interrupting

the execution of the script then might cause a crash.
Solution: In inchar(), after closing a script, don't use buf[] anymore.
close_script() has freed typebuf[] and buf[] might be pointing
inside typebuf[].

Avoid that test33 hangs when the lisp feature is missing.
Files: src/getchar.c src/testdir/test33.in

"os2" was missing from the feature list. Useful for has("os2").

BeOS:

- Included patches from Richard Offer for BeOS R4.5.
- menu code didn't work right. Crashed in the Buffers menu. The window title wasn't set. (Offer)

Patch 5.4n.3

Problem: C-indenting was wrong after " } else". The white space was not skipped. Visible when 'cino' has "+10".

Solution: Skip white space before calling cin_iselse(). (Norbert Zeh)

Files: src/misc1.c

Patch 5.4n.4

Problem: When the 't' flag in 'coptions' is included, after a ":nohlsearch" the search highlighting would not be enabled again after a tag search. (Norbert Zeh)

Solution: When setting the new search pattern in jumpton_tag(), don't restore no_hlsearch.

Files: src/tag.c

Patch 5.4n.5

Problem: When using ":normal" from a CursorHold autocommand Vim hangs. The autocommand is executed down from vgetc(). Calling vgetc() recursively to execute the command doesn't work then.

Solution: Forbid the use of ":normal" when vgetc_busy is set. Give an error message when this happens.

Files: src/ex_docmd.c, runtime/doc/autocmd.txt

Patch 5.4n.6

Problem: "gv" could reselect a Visual that starts and/or ends past the end of a line. (Robert Webb)

Solution: Check that the start and end of the Visual area are on a valid character by calling adjust_cursor().

Files: src/normal.c

Patch 5.4n.8

Problem: When a mark was on a non existing line (e.g., when the .viminfo was edited), jumping to it caused ml_get errors. (Alexey Marinichev).

Solution: Added check_cursor_lnum() in nv_gomark().

Files: src/normal.c

Patch 5.4n.9

Problem: ":-2" moved the cursor to a negative line number. (Ralf Schandl)

Solution: Give an error message for a negative line number.

Files: src/ex_docmd.c

Patch 5.4n.10

Problem: Win32 GUI: At the hit-enter prompt, it was possible to scroll the text. This erased the prompt and made Vim look like it is in Normal mode, while it is actually still waiting for a <CR>.

Solution: Disallow scrolling at the hit-enter prompt for systems that use on the fly scrolling.

Files: src/message.c

Patch 5.4n.14

Problem: Win32 GUI: When using ":winsize 80 46" and the height is more than what fits on the screen, the window size was made smaller than asked for (that's OK) and Vim crashed (that's not OK)>

Solution: Call check_winsize() from gui_set_winsize() to resize the windows.

Files: src/gui.c

Patch 5.4n.16

Problem: Win32 GUI: The <F10> key both selected the menu and was handled as a key hit.

Solution: Apply 'winaltkeys' to <F10>, like it is used for Alt keys.

Files: src/gui_w32.c

Patch 5.4n.17

Problem: Local buffer variables were freed when the buffer is unloaded. That's not logical, since options are not freed. (Ron Aaron)

Solution: Free local buffer variables only when deleting the buffer.

Files: src/buffer.c

Patch 5.4n.19

Problem: Doing ":e" (without argument) in an option-window causes trouble. The mappings for <CR> and <Space> are not removed. When there is another buffer loaded, the swap file for it gets mixed up. (Steve Mueller)

Solution: Also remove the mappings at the BufUnload event, if they are still present.

When re-editing the same file causes the current buffer to be deleted, don't try editing it.

Also added a test for this situation.

Files: runtime/optwin.vim, src/ex_cmds.c, src/testdir/test13.in, src/testdir/test13.ok

Patch 5.4n.24

Problem: BeOS: configure never enabled the GUI, because \$with_x was "no". Unix prototypes caused problems, because Display and Widget are undefined.

Freeing fonts on exit caused a crash.

Solution: Only disable the GUI when \$with_x is "no" and \$BEOS is not "yes". Add dummy defines for Display and Widget in proto.h.

Don't free the fonts in gui_exit() for BeOS.

Files: src/configure.in, src/configure, src/proto.h, src/gui.c.

The runtime/vim48x48.xpm icon didn't have a transparent background. (Schild)

Some versions of the mingw32/egcs compiler didn't have WINBASEAPI defined.
(Aaron)

VMS:

- mch_setenv() had two arguments instead of three.
- The system vimrc and gvimrc files were called ".vimrc" and ".gvimrc".
Removed the dot.
- call to RealWaitForChar() had one argument too many. (Campbell)
- WaitForChar() is static, removed the prototype from proto/os_vms.pro.
- Many file accesses failed, because Unix style file names were used.
Translate file names to VMS style by using vim_fopen().
- Filtering didn't work, because the temporary file name was generated wrong.
- There was an extra newline every 9192 characters when writing a file. Work
around it by writing line by line. (Campbell)
- os_vms.c contained "# typedef int DESC". Should be "typedef int DESC;".
Only mattered for generating prototypes.
- Added file name translation to many places. Made easy by defining macros
mch_access(), mch_fopen(), mch_fstat(), mch_lstat() and mch_stat().
- Set default for '**tagbsearch**' to off, because binary tag searching apparently
doesn't work for VMS.
- make mch_get_host_name() work with /dec and /standard=vaxc. (Campbell)

Patch 5.4o.2

Problem: Crash when using "gf" on "file.c://comment here". (Scott Graham)
Solution: Fix wrong use of pointers in get_file_name_in_path().
Files: src/window.c

Patch 5.4o.3

Problem: The horizontal scrollbar was not sized correctly when '**number**' is
set and '**wrap**' not set.
Athena: Horizontal scrollbar wasn't updated when the cursor was
positioned with a mouse click just after dragging.
Solution: Subtract 8 from the size when '**number**' set and '**wrap**' not set.
Reset gui.dragged_sb when a mouse click is received.
Files: src/gui.c

Patch 5.4o.4

Problem: When running in an xterm and \$WINDOWID is set to an illegal value,
Vim would exit with "Vim: Got X error".
Solution: When using the display which was opened for the xterm clipboard,
check if x11_window is valid by trying to obtain the window title.
Also add a check in setup_xterm_clip(), for when using X calls to
get the pointer position in an xterm.
Files: src/os_unix.c

Patch 5.4o.5

Problem: Motif version with Lesstif: When removing the menubar and then
using a menu shortcut key, Vim would crash. (raf)
Solution: Disable the menu mnemonics when the menu bar is removed.
Files: src/gui_motif.c

Patch 5.4o.9

Problem: The DOS install.exe program used the "move" program. That doesn't

work on Windows NT, where "move" is internal to cmd.exe.
Solution: Don't use an external program for moving the executables. Use C functions to copy the file and delete the original.
Files: src/dosinst.c

Motif and Athena obtained the status area height differently from GTK. Moved status_area_enabled from global.h to gui_x11.c and call xim_get_status_area_height() to get the status area height.

Patch 5.4p.1

Problem: When using auto-select, and the "gv" command is used, would not always obtain ownership of the selection. Caused by the Visual area still being the same, but ownership taken away by another program.

Solution: Reset the clipboard Visual mode to force updating the selection.
Files: src/normal.c

Patch 5.4p.2

Problem: Motif and Athena with XIM: Typing 3-byte <multibyte><multibyte><space> doesn't work correctly with Ami XIM.

Solution: Avoid using key_sym XK_VoidSymbol. (Nam)
Files: src/multbyte.c, src/gui_x11.c

Patch 5.4p.4

Problem: Win32 GUI: The scrollbar values were reduced for a file with more than 32767 lines. But this info was kept global for all scrollbars, causing a mixup between the windows. Using the down arrow of a scrollbar in a large file didn't work. Because of round-off errors there is no scroll at all.

Solution: Give each scrollbar its own scroll_shift field. When the down arrow is used, scroll several lines.
Files: src/gui.h, src/gui_w32.c

Patch 5.4p.5

Problem: When changing buffers in a BufDelete autocommand, there could be ml_line errors and/or a crash. (Schandl) Was caused by deleting the current buffer.

Solution: When the buffer to be deleted unexpectedly becomes the current buffer, don't delete it. Also added a check for this in test13.

Files: src/buffer.c, src/testdir/test13.in, src/testdir/test13.ok

Patch 5.4p.7

Problem: Win32 GUI: When using 'mousemodel' set to "popup_setpos" and clicking the right mouse button outside of the selected area, the selected area wasn't removed until the popup menu has gone. (Aaron)

Solution: Set the cursor and update the display before showing the popup menu.
Files: src/normal.c

Patch 5.4p.8

Problem: The generated bugreport didn't contain information about \$VIMRUNTIME and whether runtime files actually exist.

Solution: Added a few checks to the bugreport script.
Files: runtime/bugreport.vim

Patch 5.4p.9

Problem: The windows install.exe created a wrong entry in the popup menu. The "%1" was "". The full directory was included, even when the executable had been moved elsewhere. (Ott)
Solution: Double the '%' to get one from printf. Only include the path to gvim.exe when it wasn't moved and it's not in \$PATH.
Files: src/dosinst.c

Patch 5.4p.10

Problem: Win32: On top of 5.4p.9: The "Edit with Vim" entry sometimes used a short file name for a directory.
Solution: Change the "%1" to "%L" in the registry entry.
Files: src/dosinst.c

Patch 5.4p.11

Problem: Motif, Athena and GTK: When closing the GUI window when there is a changed buffer, there was only an error message and Vim would not exit.
Solution: Put up a dialog, like for ":confirm qa". Uses the code that was already used for MS-Windows.
Files: src/gui.c, src/gui_w32.c

Patch 5.4p.12

Problem: Win32: Trying to expand a string that is longer than 256 characters could cause a crash. (Steed)
Solution: For the buffer in win32_expandpath() don't use a fixed size array, allocate it.
Files: src/os_win32.c

MSDOS: Added "-Wall" to Makefile.djg compile flags. Function prototypes for fname_case() and mch_update_cursor() were missing. "fd" was unused in mf_sync(). "puiLocation" was unused in myputch(). "newcmd" unused in mch_call_shell() for DJGPP version.

VERSION 5.5

version-5.5

Version 5.5 is a bug-fix version of 5.4.

Changed

changed-5.5

The DJGPP version is now compiled with "-O2" instead of "-O4" to reduce the size of the executables.

Moved the src/STYLE file to runtime/doc/develop.txt. Added the design goals to it.

'backspace' is now a string option. See patch 5.4.15.

Added

added-5.5

Included Exuberant Ctags version 3.3. (Darren Hiebert)

In runtime/mswin.vim, map **CTRL-Q** to **CTRL-V**, so that **CTRL-Q** can be used everywhere to do what **CTRL-V** used to do.

Support for decompression of bzip2 files in vimrc_example.vim.

When a patch is included, the patch number is entered in a table in version.c. This allows skipping a patch without breaking a next one.

Support for mouse scroll wheel in X11. See patch 5.5a.14.

line2byte() can be used to get the size of the buffer. See patch 5.4.35.

The **CTRL-R CTRL-O r** and **CTRL-R CTRL-P r** commands in Insert mode are used to insert a register literally. See patch 5.4.48.

Uninstall program for MS-Windows. To be able to remove the registry entries for "Edit with Vim". It is registered to be run from the "Add/Remove programs" application. See patch 5.4.x7.

Fixed

fixed-5.5

When using vimrc_example.vim: An error message when the cursor is on a line higher than the number of lines in the compressed file. Move the autocommand for jumping to the last known cursor position to after the decompressing autocommands.

":mkexrc" and ":mksession" wrote the current value of **'textmode'**. That may mark a file as modified, which causes problems. This is a buffer-specific setting, it should not affect all files.

"vim --version" wrote two empty lines.

Unix: The alarm signal could kill Vim. It is generated by the Perl alarm() function. Ignore SIGALRM.

Win32 GUI: Toolbar still had the yellow bitmap for running a Vim script.

BeOS: "tmo" must be bigtime_t, instead of double. (Seibert)

Patch 5.4.1

Problem: Test11 fails when \$GZIP is set to "-v". (Matthew Jackson)

Solution: Set \$GZIP to an empty string.

Files: src/testdir/test11.in

Patch 5.4.2

Problem: Typing <Esc> at the crypt key prompt caused a crash. (Kallingal)

Solution: Check for a NULL pointer returned from `get_crypt_key()`.
Files: `src/fileio.c`

Patch 5.4.3

Problem: Python: Trying to use the name of an unnamed buffer caused a crash. (Daniel Burrows)

Solution: Check for `b_fname` being a NULL pointer.
Files: `src/if_python.c`

Patch 5.4.4

Problem: Win32: When compiled without toolbar, but the 'T' flag is in '**guioptions**', there would be an empty space for the toolbar.

Solution: Add two `#ifdefs` where checking for the 'T' flag. (Vince Negri)
Files: `src/gui.c`

Patch 5.4.5

Problem: Athena GUI: Using the `Buffers.Refresh` menu entry caused a crash. Looks like any `":unmenu"` command may cause trouble.

Solution: Disallow `":unmenu"` in the Athena version. Disable the `Buffers` menu, because the `Refresh` item would not work.
Files: `src/menu.c`, `runtime/menu.vim`

Patch 5.4.6

Problem: GTK GUI: Using `":gui"` in the `.gvimrc` file caused an error. Only happens when the GUI forks.

Solution: Don't fork in a recursive call of `gui_start()`.
Files: `src/gui.c`

Patch 5.4.7

Problem: Typing 'q' at the more prompt for the ATTENTION message causes the file loading to be interrupted. (Will Day)

Solution: Reset `got_int` after showing the ATTENTION message.
Files: `src/memline.c`

Patch 5.4.8

Problem: Edit some file, `":he"`, `":opt"`: options from help window are shown, but pressing space updates from the other window. (Phillipps)
Also: When there are changes in the option-window, `":q!"` gives an error message.

Solution: Before creating the option-window, go to a non-help window. Use `":bdel!"` to delete the buffer.

Files: `runtime/optwin.vim`

Patch 5.4.9

Just updates `version.h`. The real patch has been moved to 5.4.x1. This patch is just to keep the version number correct.

Patch 5.4.10

Problem: GTK GUI: When `$DISPLAY` is invalid, `"gvim -f"` just exits. It should run in the terminal.

Solution: Use `gtk_init_check()` instead of `gtk_init()`.
Files: `src/gui_gtk_x11.c`

Patch 5.4.11

Problem: When using the 'S' flag in '**coptions**', '**tabstop**' is not copied to the next buffer for some commands, e.g., ":buffer".
Solution: When the BCO_NOHELP flag is given to buf_copy_options(), still copy the options used by do_help() when neither the "from" or "to" buffer is a help buffer.
Files: src/option.c

Patch 5.4.12

Problem: When using '**smartindent**', there would be no extra indent if the current line did not have any indent already. (Hanus Adler)
Solution: There was a wrongly placed "else", that previously matched with the "if" that set trunc_line. Removed the "else" and added a check for trunc_line to be false.
Files: src/misc1.c

Patch 5.4.13

Problem: New SGI C compilers need another option for optimisation.
Solution: Add a check in configure for "-OPT:Olimit". (Chin A Young)
Files: src/configure.in, src/configure

Patch 5.4.14

Problem: Motif GUI: When the popup menu is present, a tiny window appears on the desktop for some users.
Solution: Set the menu widget ID for a popup menu to 0. (Thomas Koehler)
Files: src/gui_motif.c

Patch 5.4.15

Problem: Since '**backspace**' set to 0 has been made Vi compatible, it is no longer possible to only allow deleting autoindent.
Solution: Make '**backspace**' a list of parts, to allow each kind of backspacing separately.
Files: src/edit.c, src/option.c, src/option.h, src/proto/option.pro, runtime/doc/option.txt, runtime/doc/insert.txt

Patch 5.4.16

Problem: Multibyte: Locale zh_TW.Big5 was not checked for in configure.
Solution: Add zh_TW.Big5 to configure check. (Chih-Tsun Huang)
Files: src/configure.in, src/configure

Patch 5.4.17

Problem: GUI: When started from inside gvim with ":%!gvim", Vim would not start. ":%!gvim -f" works fine.
Solution: After forking, wait a moment in the parent process, to give the child a chance to set its process group.
Files: src/gui.c

Patch 5.4.18

Problem: Python: The clear_history() function also exists in a library.
Solution: Rename clear_history() to clear_hist().
Files: src/ex_getln.c, src/eval.c, src/proto/ex_getln.pro

Patch 5.4.19

Problem: In a terminal with 25 lines, there is a more prompt after the ATTENTION message. When hitting 'q' here the dialog prompt

Solution: doesn't appear and file loading is interrupted. (Will Day)
Don't allow quitting the printing of a message for the dialog prompt. Added the msg_noquit_more flag for this.
Files: src/message.c

Patch 5.4.20

Problem: GTK: When starting gvim, would send escape sequences to the terminal to switch the cursor off and on.
Solution: Don't call msg_start() if the GUI is expected to start.
Files: src/main.c

Patch 5.4.21

Problem: Motif: Toplevel menu ordering was wrong when using tear-off items.
Solution: Don't add one to the index for a toplevel menu.
Files: src/gui_motif.c

Patch 5.4.22

Problem: In Insert mode, <C-Left>, <S-Left>, <C-Right> and <S-Right> didn't update the column used for vertical movement.
Solution: Set curwin->w_set_curswant for those commands.
Files: src/edit.c

Patch 5.4.23

Problem: When a Visual selection is lost to another program, and then the same text is Visually selected again, the clipboard ownership wasn't regained.
Solution: Set clipboard.vmode to NUL to force regaining the clipboard.
Files: src/normal.c

Patch 5.4.24

Problem: Encryption: When using ":r file" while 'key' has already entered, the 'key' option would be messed up. When writing the file it would be encrypted with an unknown key and lost! (Brad Despres)
Solution: Don't free cryptkey when it is equal to the 'key' option.
Files: src/fileio.c

Patch 5.4.25

Problem: When 'cindent' is set, but 'autoindent' isn't, comments are not properly indented when starting a new line. (Mitterand)
Solution: When there is a comment leader for the new line, but 'autoindent' isn't set, do C-indenting.
Files: src/misc1.c

Patch 5.4.26

Problem: Multi-byte: a multibyte character is never recognized in a file name, causing a backslash before it to be removed on Windows.
Solution: Assume that a leading-byte character is a file name character in vim_isfilec().
Files: src/charset.c

Patch 5.4.27

Problem: Entries in the PopUp[nvic] menus were added for several modes, but only deleted for the mode they were used for. This resulted in the entry remaining in the PopUp menu.

When removing a PopUp[nvic] menu, the name had been truncated, could result in greying-out the whole PopUp menu.

Solution: Remove entries for all modes from the PopUp[nvic] menus. Remove the PopUp[nvic] menu entries first, before the name is changed.

Files: src/menu.c

Patch 5.4.28

Problem: When using a BufWritePre autocommand to change **'fileformat'**, the new value would not be used for writing the file.

Solution: Check **'fileformat'** after executing the autocommands instead of before.

Files: src/fileio.c

Patch 5.4.29

Problem: Athena GUI: When removing the 'g' flag from **'guioptions'**, using a menu can result in a crash.

Solution: Always grey-out menus for Athena, don't hide them.

Files: src/menu.c

Patch 5.4.30

Problem: BeOS: Suspending Vim with **CTRL-Z** didn't work (killed Vim). The first character typed after **":sh"** goes to Vim, instead of the started shell.

Solution: Don't suspend Vim, start a new shell. Kill the async read thread when starting a new shell. It will be restarted later. (Will Day)

Files: src/os_unix.c, src/ui.c

Patch 5.4.31

Problem: GUI: When **'mousefocus'** is set, moving the mouse over where a window boundary was, causes a hit-enter prompt to be finished. (Jeff Walker)

Solution: Don't use **'mousefocus'** at the hit-enter prompt. Also ignore it for the more prompt and a few other situations. When an operator is pending, abort it first.

Files: src/gui.c

Patch 5.4.32

Problem: Unix: \$LDLDFLAGS was not passed to configure.

Solution: Pass \$LDLDFLAGS to configure just like \$CFLAGS. (Jon Miner)

Files: src/Makefile

Patch 5.4.33

Problem: Unix: After expanding an environment variable with the shell, the next expansion would also use the shell, even though it is not needed.

Solution: Reset "recursive" before returning from gen_expand_wildcards().

Files: src/misc1.c

Patch 5.4.34 (also see 5.4.x5)

Problem: When editing a file, and the file name is relative to a directory above the current directory, the file name was made absolute. (Gregory Margo)

Solution: Add an argument to shorten_fname() which indicates if all file names should be shortened, or only absolute names. In main() only

use shorten_fname() to shorten absolute names.
Files: src/ex_docmd.c, src/fileio.c, src/main.c, src/proto/fileio.pro

Patch 5.4.35

Problem: There is no function to get the current file size.
Solution: Allow using line2byte() with the number of lines in the file plus one. This returns the offset of the line past the end of the file, which is the file size plus one.
Files: src/eval.c, runtime/doc/eval.txt

Patch 5.4.36

Problem: Comparing strings while ignoring case didn't work correctly for some machines. (Mide Steed)
Solution: vim_stricmp() and vim_strnicmp() only returned 0 or 1. Changed them to return -1 when the first argument is smaller.
Files: src/misc2.c

Patch 5.4.37 (also see 5.4.40 and 5.4.43)

Problem: Long strings from the viminfo file are truncated.
Solution: When writing a long string to the viminfo file, first write a line with the length, then the string itself in a second line.
Files: src/eval.c, src/ex_cmds.c, src/ex_getln.c, src/mark.c, src/ops.c, src/search.c, src/proto/ex_cmds.pro, runtime/syntax/viminfo.vim

Patch 5.4.38

Problem: In the option-window, ":set go&" resulted in 'go' being handled like a boolean option.
Mappings for <Space> and <CR> were overruled by the option-window.
Solution: When the value of an option isn't 0 or 1, don't handle it like a boolean option.
Save and restore mappings for <Space> and <CR> when entering and leaving the option-window.
Files: runtime/optwin.vim

Patch 5.4.39

Problem: When setting a hidden option, spaces before the equal sign were not skipped and cause an error message. E.g., ":set csprg =cmd".
Solution: When skipping over a hidden option, check for a following "=val" and skip it too.
Files: src/option.c

Patch 5.4.40 (depends on 5.4.37)

Problem: Compiler error for "atol(p + 1)". (Axel Kielhorn)
Solution: Add a typecast: "atol((char *)p + 1)".
Files: src/ex_cmds.c

Patch 5.4.41

Problem: Some commands that were not included would give an error message, even when after "if 0".
Solution: Don't give an error message for an unsupported command when not executing the command.
Files: src/ex_docmd.c

Patch 5.4.42

Problem: ":w" would also cause a truncated message to appear in the message history.
Solution: Don't put a kept message in the message history when it starts with "<".
Files: src/message.c

Patch 5.4.43 (depends on 5.4.37)

Problem: Mixing long lines with multiple lines in a register causes errors when writing the viminfo file. (Robinson)
Solution: When reading the viminfo file to skip register contents, skip lines that start with "<".
Files: src/ops.c

Patch 5.4.44

Problem: When '**whichwrap**' includes '~', a "~" command that goes on to the next line cannot be properly undone. (Zellner)
Solution: Save each line for undo in n_swapchar().
Files: src/normal.c

Patch 5.4.45 (also see 5.4.x8)

Problem: When expand("\$ASDF") fails, there is an error message.
Solution: Remove the global expand_interactively. Pass a flag down to skip the error message.
Also: expand("\$ASDF") returns an empty string if \$ASDF isn't set. Previously it returned "\$ASDF" when '**shell**' is "sh".
Also: system() doesn't print an error when the command returns an error code.
Files: many

Patch 5.4.46

Problem: Backspacing did not always use '**softtabstop**' after hitting <CR>, inserting a register, moving the cursor, etc.
Solution: Reset inserted_space much more often in edit().
Files: src/edit.c

Patch 5.4.47

Problem: When executing BufWritePre or BufWritePost autocommands for a hidden buffer, the cursor could be moved to a non-existing position. (Vince Negri)
Solution: Save and restore the cursor and topline for the current window when it is going to be used to execute autocommands for a hidden buffer. Use an existing window for the buffer when it's not hidden.
Files: src/fileio.c

Patch 5.4.48

Problem: A paste with the mouse in Insert mode was not repeated exactly the same with ".". For example, when '**autoindent**' is set and pasting text with leading indent. (Perry)
Solution: Add the **CTRL-R CTRL-O r** and **CTRL-R CTRL-P r** commands in Insert mode, which insert the contents of a register literally.
Files: src/edit.c, src/normal.c, runtime/doc/insert.txt

Patch 5.4.49

Problem: When pasting text with [<MiddleMouse>, the cursor could end up after the last character of the line.
Solution: Correct the cursor position for the change in indent.
Files: src/ops.c

Patch 5.4.x1 (note: Replaces patch 5.4.9)

Problem: Win32 GUI: menu hints were never used, because WANT_MENU is not defined until vim.h is included.
Solution: Move the #ifdef WANT_MENU from where MENUHINTS is defined to where it is used.
Files: src/gui_w32.c

Patch 5.4.x2

Problem: BeOS: When pasting text, one character was moved to the end.
Solution: Re-enable the BeOS code in fill_input_buf(), and fix timing out with acquire_sem_etc(). (Will Day)
Files: src/os_beos.c, src/ui.c

Patch 5.4.x3

Problem: Win32 GUI: When dropping a directory on a running gvim it crashes.
Solution: Avoid using a NULL file name. Also display a message to indicate that the current directory was changed.
Files: src/gui_w32.c

Patch 5.4.x4

Problem: Win32 GUI: Removing an item from the popup menu doesn't work.
Solution: Don't remove the item from the menubar, but from the parent popup menu.
Files: src/gui_w32.c

Patch 5.4.x5 (addition to 5.4.34)

Files: src/gui_w32.c

Patch 5.4.x6

Problem: Win32: Expanding (dir)name starting with a dot doesn't work. (McCormack) Only when there is a path before it.
Solution: Fix the check, done before expansion, if the file name pattern starts with a dot.
Files: src/os_win32.c

Patch 5.4.x7

Problem: Win32 GUI: Removing "Edit with Vim" from registry is difficult.
Solution: Add uninstall program to remove the registry keys. It is installed in the "Add/Remove programs" list for ease of use.
Also: don't set \$VIM when the executable is with the runtime files.
Also: Add a text file with a step-by-step description of how to uninstall Vim for DOS and Windows.
Files: src/uninstal.c, src/dosinst.c, src/Makefile.w32, uninstal.txt

Patch 5.4.x8 (addition to 5.4.45)

Files: many

Patch 5.4.x9

Problem: Win32 GUI: After executing an external command, focus is not

always regained (when using focus-follows-mouse).
Solution: Add SetFocus() in mch_system(). (Mike Steed)
Files: src/os_win32.c

Patch 5.5a.1

Problem: ":let @* = @:" did not work. The text was not put on the
I clipboard. (Fisher)
Solution: Own the clipboard and put the text on it.
Files: src/ops.c

Patch 5.5a.2

Problem: append() did not mark the buffer modified. Marks below the
new line were not adjusted.
Solution: Fix the f_append() function.
Files: src/eval.c

Patch 5.5a.3

Problem: Editing compressed ".gz" files doesn't work on non-Unix systems,
because there is no "mv" command.
Solution: Add the rename() function and use it instead of ":%mv".
Also: Disable the automatic jump to the last position, because it
changes the jumplist.
Files: src/eval.c, runtime/doc/eval.txt, runtime/vimrc_example.vim

Patch 5.5a.4

Problem: When using whole-line completion in insert mode while the cursor
is in the indent, get "out of memory" error. (Stekrt)
Solution: Don't allocate a negative amount of memory in ins_complete().
Files: src/edit.c

Patch 5.5a.5

Problem: Win32: The '**path**' option can hold only up to 256 characters,
because _MAX_PATH is 256. (Robert Webb)
Solution: Use a fixed path length of 1024.
Files: src/os_win32.h

Patch 5.5a.6

Problem: Compiling with gcc on Win32, using the Unix Makefile, didn't work.
Solution: Add \$(SUFFIX) to all places where an executable is used. Also
pass it to ctags. (Reynolds)
Files: src/Makefile

Patch 5.5a.7

Problem: When using "cat | vim -" in an xterm, the xterm version reply
would end up in the file.
Solution: Read the file from stdin before switching the terminal to RAW
mode. Should also avoid problems with programs that use a
specific terminal setting.
Also: when using the GUI, print "Reading from stdin..." in the GUI
window, to give a hint why it doesn't do anything.
Files: src/main.c, src/fileio.c

Patch 5.5a.8

Problem: On multi-threaded Solaris, suspending doesn't work.
Solution: Call pause() when the SIGCONT signal was not received after sending the SIGTSTP signal. (Nagano)
Files: src/os_unix.c

Patch 5.5a.9

Problem: **'winaltkeys'** could be set to an empty argument, which is illegal.
Solution: Give an error message when doing ":set winaltkeys=".
Files: src/option.c

Patch 5.5a.10

Problem: Win32 console: Using ALTGR on a German keyboard to produce "}" doesn't work, because the 8th bit is set when ALT is pressed.
Solution: Don't set the 8th bit when ALT and CTRL are used. (Leipert)
Files: src/os_win32.c

Patch 5.5a.11

Problem: Tcl: Configure always uses tclsh8.0.
Also: Loading a library doesn't work.
Solution: Add "--with-tclsh" configure argument to allow specifying another name for the tcl shell.
Call Tcl_Init() in tclinit() to make loading libraries work.
(Johannes Zellner)
Files: src/configure.in, src/configure, src/if_tcl.c

Patch 5.5a.12

Problem: The "user_commands" feature is called "user-commands".
Solution: Replace "user-commands" with "user_commands". (Kim Sung-bom)
Keep "user-commands" for the has() function, to remain backwards compatible with 5.4.
Files: src/eval.c, src/version.c

Patch 5.5a.13

Problem: OS/2: When \$HOME is not defined, "C:/" is used for the viminfo file. That is very wrong when OS/2 is on another partition.
Solution: Use \$VIM for the viminfo file when it is defined, like for MSDOS.
Also: Makefile.os2 didn't depend on os_unix.h.
Files: src/os_unix.h, src/Makefile.os2

Patch 5.5a.14

Problem: Athena, Motif and GTK: The Mouse scroll wheel doesn't work.
Solution: Interpret a click of the wheel as a key press of the <MouseDown> or <MouseUp> keys. Default behavior is to scroll three lines, or a full page when Shift is used.
Files: src/edit.c, src/ex_getln.c, src/gui.c, src/gui_gtk_x11.c, src/gui_x11.c, src/keymap.h, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/proto/normal.pro, src/vim.h, runtime/doc/scroll.txt

Patch 5.5a.15

Problem: Using **CTRL-A** in Insert mode doesn't work correctly when the insert started with the <Insert> key. (Andreas Rohrschneider)
Solution: Replace <Insert> with "i" before setting up the redo buffer.
Files: src/normal.c

Patch 5.5a.16

Problem: VMS: GUI does not compile and run.

Solution: Various fixes. (Zoltan Arpadffy)
Moved functions from os_unix.c to ui.c, so that VMS can use them too: open_app_context(), x11_setup_atoms() and clip_x11* functions.
Made xterm_dpy global, it's now used by ui.c and os_unix.c.
Use gethostname() always, sys_hostname doesn't exist.

Files: src/globals.h, src/gui_x11.c, src/os_vms.mms, src/os_unix.c, src/os_vms.c, src/ui.c, src/proto/os_unix.pro, src/proto/ui.pro

Renamed AdjustCursorForMultiByteCharacter() to AdjustCursorForMultiByteChar() to avoid symbol length limit of 31 characters. (Steve P. Wall)

Patch 5.5b.1

Problem: SASC complains about dead assignments and implicit type casts.

Solution: Removed the dead assignments. Added explicit type casts.

Files: src/buffer.c, src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/memline.c, src/menu.c, src/misc1.c, src/normal.c, src/ops.c, src/quickfix.c, src/screen.c

Patch 5.5b.2

Problem: When using "**CTRL-O** 0" in Insert mode, hit <Esc> and then "o" in another line truncates that line. (Devin Weaver)

Solution: When using a command that starts Insert mode from **CTRL-O**, reset "restart_edit" first. This avoids that edit() is called with a mix of starting a new edit command and restarting a previous one.

Files: src/normal.c

VERSION 5.6

version-5.6

Version 5.6 is a bug-fix version of 5.5.

Changed

changed-5.6

Small changes to OleVim files. (Christian Schaller)

Inserted "/* */" between patch numbers in src/version.c. This allows for one line of context, which some versions of patch need.

Reordered the Syntax menu to avoid long submenus. Removed keyboard shortcuts for alphabetical items to avoid a clash with fixed items.

Added

added-5.6

Included Exuberant Ctags version 3.4. (Darren Hiebert)

OpenWithVim in Python. (Christian Schaller)

Win32 GUI: gvimext.dll, for the context menu "Edit with Vim" entry. Avoids the reported problems with the MS Office taskbar. Now it's a Shell Extension. (Tianmiao Hu)

New syntax files:

abel	Abel (John Cook)
aml	Arc Macro Language (Nikki Knuit)
apachestyle	Apache-style config file (Christian Hammers)
cf	Cold Fusion (Jeff Lanzarotta)
ctrlh	files with CTRL-H sequences (Bram Moolenaar)
cupl	CUPL (John Cook)
cuplsim	CUPL simulation (John Cook)
erlang	Erlang (Kresimir Marzic)
gedcom	Gedcom (Paul Johnson)
icon	Icon (Wendell Turner)
ist	MakeIndex style (Peter Meszaros)
jsp	Java Server Pages (Rafael Garcia-Suarez)
rcslog	Rcslog (Joe Karthaus)
remind	Remind (Davide Alberani)
sqr	Structured Query Report Writer (Paul Moore)
tads	TADS (Amir Karger)
texinfo	Texinfo (Sandor Kopanyi)
xpm2	X Pixmap v2 (Steve Wall)

The 'C' flag in '**coptions**' can be used to switch off concatenation for sourced lines. See patch 5.5.013 below. [line-continuation](#)

"excludenl" argument for the ":syntax" command. See patch 5.5.032 below. [:syn-excludenl](#)

Implemented [z+](#) and [z^](#) commands. See patch 5.5.050 below.

Vim logo in Corel Draw format. Can be scaled to any resolution.

Fixed

[fixed-5.6](#)

Using this mapping in Select mode, terminated completion:

":vnoremap <C-N> <Esc>a<C-N>" (Benji Fisher)

Ignore K_SELECT in ins_compl_prep().

VMS (Zoltan Arpadffy, David Elins):

- ioctl() in pty.c caused trouble, #ifdef VMS added.
- Cut & paste mismatch corrected.
- Popup menu line crash corrected. (Patch 5.5.047)
- Motif directories during open and save as corrected.
- Handle full file names with version numbers. (Patch 5.5.046)
- Directory handling (CD command etc.)
- Corrected file name conversion VMS to Unix and v.v.
- Recovery was not working.
- Terminal and signal handling was outdated compared to os_unix.c.
- Improved os_vms.txt.

Configure used `fprintf()` instead of `printf()` to check for `__DATE__` and `__TIME__`. (John Card II)

BeOS: Adjust computing the `char_height` and `char_ascent`. Round them up separately, avoids redrawing artifacts. (Mike Steed)

Fix a few multibyte problems in `menu_name_skip()`, `set_reg_ic()`, `searchc()` and `findmatchlimit()`. (Taro Muraoka)

GTK GUI:

- With GTK 1.2.5 and later the scrollbars were not redrawn correctly.
- Adjusted the `gtk_form_draw()` function.
- SNIFF connection didn't work.
- **'mousefocus'** was not working. (Dalecki)
- Some keys were not working with modifiers: Shift-Tab, Ctrl-Space and `CTRL-@`.

Patch 5.5.001

Problem: Configure in the top directory did not pass on an argument with a space correctly. For example `./configure --previs="/My home"`. (Stephane Chazelas)

Solution: Use `"$@"` instead of `'$*'` to pass on the arguments.

Files: `configure`

Patch 5.5.002

Problem: Compilation error for using `"fds[] & POLLIN"`. (Jeff Walker)

Solution: Use `"fds[].revents & POLLIN"`.

Files: `src/os_unix.c`

Patch 5.5.003

Problem: The autoconf check for `sizeof(int)` is wrong on machines where `sizeof(size_t) != sizeof(int)`.

Solution: Use our own configure check. Also fixes the warning for cross-compiling.

Files: `src/configure.in`, `src/configure`

Patch 5.5.004

Problem: On Unix it's not possible to interrupt `":sleep 100"`.

Solution: Switch terminal to cooked mode while asleep, to allow a SIGINT to wake us up. But switch off echo, added `TMODE_SLEEP`.

Files: `src/term.h`, `src/os_unix.c`

Patch 5.5.005

Problem: When using `<f-args>` with a user command, an empty argument to the command resulted in one empty string, while no string was expected.

Solution: Catch an empty argument and pass no argument to the function. (Paul Moore)

Files: `src/ex_docmd.c`

Patch 5.5.006

Problem: Python: When platform-dependent files are in another directory than the platform-independent files it doesn't work.

Solution: Also check the executable directory, and add it to CFLAGS. (Tessa Lau)
Files: src/configure.in, src/configure

Patch 5.5.007 (extra)

Problem: Win32 OLE: Occasional crash when exiting while still being used via OLE.

Solution: Move OleUninitialize() to before deleting the application object. (Vince Negri)

Files: src/if_ole.cpp

Patch 5.5.008

Problem: 10000@@ takes a long time and cannot be interrupted.

Solution: Check for **CTRL-C** typed while in the loop to push the register.

Files: src/normal.c

Patch 5.5.009

Problem: Recent Sequent machines don't link with "-linet". (Kurtis Rader)

Solution: Remove configure check for Sequent.

Files: src/configure.in, src/configure

Patch 5.5.010

Problem: Ctags freed a memory block twice when exiting. When out of memory, a misleading error message was given.

Solution: Update to ctags 3.3.2. Also fixes a few other problems. (Darren Hiebert)

Files: src/ctags/*

Patch 5.5.011

Problem: After "**CTRL-V** s", the cursor jumps back to the start, while all other operators leave the cursor on the last changed character. (Xiangjiang Ma)

Solution: Position cursor on last changed character, if possible.

Files: src/ops.c

Patch 5.5.012

Problem: Using **CTRL-]** in Visual mode doesn't work when the text includes a space (just where it's useful). (Stefan Bittner)

Solution: Don't escape special characters in a tag name with a backslash.

Files: src/normal.c

Patch 5.5.013

Problem: The ":append" and ":insert" commands allow using a leading backslash in a line. The ":source" command concatenates those lines. (Heinlein)

Solution: Add the 'C' flag in '**coptions**' to switch off concatenation.

Files: src/ex_docmd.c, src/option.h, runtime/doc/options.txt, runtime/filetype.vim, runtime/scripts.vim

Patch 5.5.014

Problem: When executing a register with ":@", the ":append" command would get text lines with a ':' prepended. (Heinlein)

Solution: Remove the ':' characters.

Files: src/ex_docmd.c, src/ex_getln.c, src/globals.h

Patch 5.5.015

Problem: When using ":g/pat/p", it's hard to see where the output starts, the ":g" command is overwritten. Vi keeps the ":g" command.

Solution: Keep the ":g" command, but allow overwriting it with the report for the number of changes.

Files: src/ex_cmds.c

Patch 5.5.016 (extra)

Problem: Win32: Using regedit to install Vim in the popup menu requires the user to confirm this in a dialog.

Solution: Use "regedit /s" to avoid the dialog

Files: src/dosinst.c

Patch 5.5.017

Problem: If an error occurs when closing the current window, Vim could get stuck in the error handling.

Solution: Don't set curwin to NULL when closing the current window.

Files: src/window.c

Patch 5.5.018

Problem: Absolute paths in shell scripts do not always work.

Solution: Use /usr/bin/env to find out the path.

Files: runtime/doc/vim2html.pl, runtime/tools/efm_filter.pl, runtime/tools/shtags.pl

Patch 5.5.019

Problem: A function call in **'statusline'** stops using ":q" twice from exiting, when the last argument hasn't been edited.

Solution: Don't decrement quitmore when executing a function. (Madsen)

Files: src/ex_docmd.c

Patch 5.5.020

Problem: When the output of **CTRL-D** completion in the commandline goes all the way to the last column, there is an empty line.

Solution: Don't add a newline when the cursor wrapped already. (Madsen)

Files: src/ex_getln.c

Patch 5.5.021

Problem: When checking if a file name in the tags file is relative, environment variables were not expanded.

Solution: Expand the file name before checking if it is relative. (Madsen)

Files: src/tag.c

Patch 5.5.022

Problem: When setting or resetting **'paste'** the ruler wasn't updated.

Solution: Update the status lines when **'ruler'** changes because of **'paste'**.

Files: src/option.c

Patch 5.5.023

Problem: When editing a new file and autocommands change the cursor position, the cursor was moved back to the first non-white, unless **'startofline'** was reset.

Solution: Keep the new column, just like the line number.

Files: src/ex_cmds.c

Patch 5.5.024 (extra)

Problem: Win32 GUI: When using confirm() to put up a dialog without a default button, the dialog would not have keyboard focus. (Krishna)

Solution: Always set focus to the dialog window. Only set focus to a button when a default one is specified.

Files: src/gui_w32.c

Patch 5.5.025

Problem: When using "keepend" in a syntax region, a contained match that includes the end-of-line could still force that region to continue, if there is another contained match in between.

Solution: Check the keepend_level in check_state_ends().

Files: src/syntax.c

Patch 5.5.026

Problem: When starting Vim in a white-on-black xterm, with 'bg' set to "dark", and then starting the GUI with ":gui", setting 'bg' to "light" in the gvimrc, the highlighting isn't set. (Tsjokwing)

Solution: Set the highlighting when 'bg' is changed in the gvimrc, even though full_screen isn't set.

Files: src/option.c

Patch 5.5.027

Problem: Unix: os_unix.c doesn't compile when XTERM_CLIP is used but WANT_TITLE isn't. (Barnum)

Solution: Move a few functions that are used by the X11 title and clipboard and put another "#if" around it.

Files: src/os_unix.c

Patch 5.5.028 (extra)

Problem: Win32 GUI: When a file is dropped on Win32 gvim while at the ":" prompt, the file is edited but the command line is actually still there, the cursor goes back to command line on the next command. (Krishna)

Solution: When dropping a file or directory on gvim while at the ":" prompt, insert the name of the file/directory. Allows using the file/directory name for any Ex command.

Files: src/gui_w32.c

Patch 5.5.029

Problem: "das" at the end of the file didn't delete the last character of the sentence.

Solution: When there is no character after the sentence, make the operation inclusive in current_sent().

Files: src/search.c

Patch 5.5.030

Problem: Unix: in os_unix.c, "term_str" is used, which is also defined in vim.h as a macro. (wuxin)

Solution: Renamed "term_str" to "buf" in do_xterm_trace().

Files: src/os_unix.c

Patch 5.5.031 (extra)

Problem: Win32 GUI: When exiting Windows, gvim will leave swap files behind and will be killed ungracefully. (Krishna)

Solution: Catch the WM_QUERYENDSESSION and WM_ENDSESSION messages and try to exit gracefully. Allow the user to cancel the shutdown if there is a changed buffer.

Files: src/gui_w32.c

Patch 5.5.032

Problem: Patch 5.5.025 wasn't right. And C highlighting was still not working correctly for a #define.

Solution: Added "excludenl" argument to ":syntax", to be able not to extend a containing item when there is a match with the end-of-line.

Files: src/syntax.c, runtime/doc/syntax.txt, runtime/syntax/c.vim

Patch 5.5.033

Problem: When reading from stdin, a long line in viminfo would mess up the file message. readfile() uses IObuff for keep_msg, which could be overwritten by anyone.

Solution: Copy the message from IObuff to msg_buf and set keep_msg to that. Also change vim_fgets() to not use IObuff any longer.

Files: src/fileio.c

Patch 5.5.034

Problem: "gvim -rv" caused a crash. Using '**t_Co**' before it's set.

Solution: Don't try to initialize the highlighting before it has been initialized from main().

Files: src/syntax.c

Patch 5.5.035

Problem: GTK with XIM: Resizing with status area was messy, and ":set guioptions+=b" didn't work.

Solution: Make status area a separate widget, but not a separate window. (Chi-Deok Hwang)

Files: src/gui_gtk_f.c, src/gui_gtk_x11.c, src/multbyte.c

Patch 5.5.036

Problem: The GZIP_read() function in \$VIMRUNTIME/vimrc_example.vim to uncompress a file did not do detection for '**fileformat**'. This is because the filtering is done with '**binary**' set.

Solution: Split the filtering into separate write, filter and read commands.

Files: runtime/vimrc_example.vim

Patch 5.5.037

Problem: The "U" command didn't mark the buffer as changed. (McCormack)

Solution: Set the '**modified**' flag when using "U".

Files: src/undo.c

Patch 5.5.038

Problem: When typing a long ":" command, so that the screen scrolls up, causes the hit-enter prompt, even though the user just typed return to execute the command.

Solution: Reset need_wait_return if (part of) the command was typed in

Files: getcmdline().
 src/ex_getln.c

Patch 5.5.039

Problem: When using a custom status line, "%a" (file # of #) reports the index of the current window for all windows.
Solution: Pass a window pointer to append_arg_number(), and pass the window being updated from build_stl_str_hl(). (Stephen P. Wall)
Files: src/buffer.c, src/screen.c, src/proto/buffer.pro

Patch 5.5.040

Problem: Multi-byte: When there is some error in xim_real_init(), it can close XIM and return. After this there can be a segv.
Solution: Test "xic" for being non-NULL, don't set "xim" to NULL. Also try to find more matches for supported styles. (Sung-Hyun Nam)
Files: src/multbyte.c

Patch 5.5.041

Problem: X11 GUI: CTRL_ requires the SHIFT key only on some machines.
Solution: Translate CTRL-- to CTRL_-. (Robert Webb)
Files: src/gui_x11.c

Patch 5.5.042

Problem: X11 GUI: keys with ALT were assumed to be used for the menu, even when the menu has been disabled by removing 'm' from 'guioptions'.
Solution: Ignore keys with ALT only when gui.menu_is_active is set. (Raf)
Files: src/gui_x11.c

Patch 5.5.043

Problem: GTK: Handling of fontset fonts was not right when 'guifontset' contains exactly 14 times '-'.
Solution: Avoid setting fonts when working with a fontset. (Sung-Hyun Nam)
Files: src/gui_gtk_x11.c

Patch 5.5.044

Problem: pltags.pl contains an absolute path "/usr/local/bin/perl". That might not work everywhere.
Solution: Use "/usr/bin/env perl" instead.
Files: runtime/tools/pltags.pl

Patch 5.5.045

Problem: Using "this_session" variable does not work, requires preceding it with "v:". Default filename for ":mksession" isn't mentioned in the docs. (Fisher)
Solution: Support using "this_session" to be backwards compatible.
Files: src/eval.c, runtime/doc/options.txt

Patch 5.5.046 (extra)

Problem: VMS: problems with path and filename.
Solution: Truncate file name at last ';', etc. (Zoltan Arpadffy)
Files: src/buffer.c, src/fileio.c, src/gui_motif.c, src/os_vms.c, src/proto/os_vms.pro

Patch 5.5.047

Problem: VMS: Crash when using the popup menu
Solution: Turn the #define MENU_MODE_CHARS into an array. (Arpadffy)
Files: src/structs.h, src/menu.c

Patch 5.5.048

Problem: HP-UX 11: Compiling doesn't work, because both string.h and strings.h are included. (Squassabia)
Solution: The configure test for including both string.h and strings.h must include <Xm/Xm.h> first, because it causes problems.
Files: src/configure.in, src/configure, src/config.h.in

Patch 5.5.049

Problem: Unix: When installing Vim, the protection bits of files might be influenced by the umask.
Solution: Add \$(FILEMOD) to Makefile. (Shetye)
Files: src/Makefile

Patch 5.5.050

Problem: "z+" and "z^" commands are missing.
Solution: Implemented "z+" and "z^".
Files: src/normal.c, runtime/doc/scroll.txt, runtime/doc/index.txt

Patch 5.5.051

Problem: Several Unix systems have a problem with the optimization limits check in configure.
Solution: Removed the configure check, let the user add it manually in Makefile or the environment.
Files: src/configure.in, src/configure, src/Makefile

Patch 5.5.052

Problem: Crash when using a cursor key at the ATTENTION prompt. (Alberani)
Solution: Ignore special keys at the console dialog. Also ignore characters > 255 for other uses of tolower() and toupper().
Files: src/menu.c, src/message.c, src/misc2.c

Patch 5.5.053

Problem: Indenting is wrong after a function when 'cino' has "fs". Another problem when 'cino' has "{s".
Solution: Put line after closing "}" of a function at the left margin. Apply ind_open_extra in the right way after a '{'.
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 5.5.054

Problem: Unix: ":e #" doesn't work if the alternate file name contains a space or backslash. (Hudacek)
Solution: When replacing "#", "%" or other items that stand for a file name, prepend a backslash before special characters.
Files: src/ex_docmd.c

Patch 5.5.055

Problem: Using "<C-V>\$r-" in blockwise Visual mode replaces one character beyond the end of the line. (Zivkov)
Solution: Only replace existing characters.
Files: src/ops.c

Patch 5.5.056

Problem: After "z20<CR>" messages were printed at the old command line position once. (Veselinovic)
Solution: Set msg_row and msg_col when changing cmdline_row in win_setheight().
Files: src/window.c

Patch 5.5.057

Problem: After "S<Esc>" it should be possible to restore the line with "U". (Veselinovic)
Solution: Don't call u_clearline() in op_delete() when changing only one line.
Files: src/ops.c

Patch 5.5.058

Problem: Using a long search pattern and then "n" causes the hit-enter prompt. (Krishna)
Solution: Truncate the echoed pattern, like other messages. Moved code for truncating from msg_attr() to msg_strtrunc().
Files: src/message.c, src/proto/message.pro, src/search.c

Patch 5.5.059

Problem: GTK GUI: When \$term is invalid, using "gvim" gives an error message, even though \$term isn't really used. (Robbins)
Solution: When the GUI is about to start, skip the error messages for a wrong \$term.
Files: src/term.c

Patch 5.5.060 (extra)

Problem: Dos 32 bit: When a directory in '**backupdir**' doesn't exist, ":w" causes the file to be renamed to "axlqwqhy.ba~". (Matzdorf)
Solution: The code to work around a LFN bug in Windows 95 doesn't handle a non-existing target name correctly. When renaming fails, make sure the file has its original name. Also do this for the Win32 version, although it's unlikely that it runs into this problem.
Files: src/os_msdos.c, src/os_win32.c

Patch 5.5.061

Problem: When using "\:" in a modeline, the backslash is included in the option value. (Mohsin)
Solution: Remove one backslash before the ':' in a modeline.
Files: src/buffer.c, runtime/doc/options.txt

Patch 5.5.062 (extra)

Problem: Win32 console: Temp files are created in the root of the current drive, which may be read-only. (Peterson)
Solution: Use the same mechanism of the GUI version: Use \$TMP, \$TEMP or the current directory. Cleaned up vim_tempname() a bit.
Files: src/fileio.c, src/os_win32.h, runtime/doc/os_dos.txt

Patch 5.5.063

Problem: When using whole-line completion in Insert mode, '**cindent**' is applied, even after changing the indent of the line.

Solution: Don't reindent the completed line after inserting/removing indent.
(Robert Webb)
Files: src/edit.c

Patch 5.5.064

Problem: has("sniff") doesn't work correctly.
Solution: Return 1 when Vim was compiled with the +sniff feature. (Pruemmer)
Files: src/eval.c

Patch 5.5.065

Problem: When dropping a file on Vim, the '**shellslash**' option is not effective. (Krishna)
Solution: Fix the slashes in the dropped file names according to '**shellslash**'.
Files: src/ex_docmd.c, runtime/doc/options.txt

Patch 5.5.066

Problem: For systems with backslash in file name: Setting a file name option to a value starting with "\\machine" removed a backslash.
Solution: Keep the double backslash for "\\machine", but do change "\\\\"machine" to "\\machine" for backwards compatibility.
Files: src/option.c, runtime/doc/options.txt

Patch 5.5.067

Problem: With '**hlsearch**' set, the pattern ">" doesn't highlight the first match in a line. (Benji Fisher)
Solution: Fix highlighting an empty match. Also highlight the first character in an empty line for "\$".
Files: src/screen.c

Patch 5.5.068

Problem: Crash when a ":while" is used with an argument that has an error. (Sylvain Viart)
Solution: Was using an uninitialized index in the cs_line[] array. The crash only happened when the index was far off. Made sure the uninitialized index isn't used.
Files: src/ex_docmd.c

Patch 5.5.069

Problem: Shifting lines in blockwise Visual mode didn't set the '**modified**' flag.
Solution: Do set the '**modified**' flag.
Files: src/ops.c

Patch 5.5.070

Problem: When editing a new file, creating that file outside of Vim, then editing it again, ":w" still warns for overwriting an existing file. (Nam)
Solution: The BF_NEW flag in the "b_flags" field wasn't cleared properly.
Files: src/buffer.c, src/fileio.c

Patch 5.5.071

Problem: Using a matchgroup in a ":syn region", which is the same syntax group as the region, didn't stop a contained item from matching in

the start pattern.

Solution: Also push an item on the stack when the syntax ID of the matchgroup is the same as the syntax ID of the region.

Files: src/syntax.c

Patch 5.5.072 (extra)

Problem: Dos 32 bit: When setting '**columns**' to a too large value, Vim may crash, and the DOS console too.

Solution: Check that the value of '**columns**' isn't larger than the number of columns that the BIOS reports.

Files: src/os_msdos.c, src/proto/os_msdos.pro, src/option.c

Patch 5.5.073 (extra)

Problem: Win 32 GUI: The Find and Find/Replace dialogs didn't show the "match case" checkbox. The Find/Replace dialog didn't handle the "match whole word" checkbox.

Solution: Support the "match case" and "match whole word" checkboxes.

Files: src/gui_w32.c

Patch 5.6a.001

Problem: Using <C-End> with a count doesn't work like it does with "G". (Benji Fisher)

Solution: Accept a count for <C-End> and <C-Home>.

Files: src/normal.c

Patch 5.6a.002

Problem: The script for conversion to HTML was an older version.

Solution: Add support for running 2html.vim on a color terminal.

Files: runtime/syntax/2html.vim

Patch 5.6a.003

Problem: Defining a function inside a function didn't give an error message. A missing ":endfunction" doesn't give an error message.

Solution: Allow defining a function inside a function.

Files: src/eval.c, runtime/doc/eval.txt

Patch 5.6a.004

Problem: A missing ":endwhile" or ":endif" doesn't give an error message. (Johannes Zellner)

Solution: Check for missing ":endwhile" and ":endif" in sourced files. Add missing ":endif" in file selection macros.

Files: src/ex_docmd.c, runtime/macros/file_select.vim

Patch 5.6a.005

Problem: '**hlsearch**' was not listed alphabetically. The value of '**toolbar**' was changed when '**compatible**' is set.

Solution: Moved entry of '**hlsearch**' in options[] table down. Don't reset '**toolbar**' option to the default value when '**compatible**' is set.

Files: src/option.c

Patch 5.6a.006

Problem: Using a backwards range inside ":if 0" gave an error message.

Solution: Don't complain about a range when it is not going to be used.

(Stefan Roemer)
Files: src/ex_docmd.c

Patch 5.6a.007

Problem: ":let" didn't show internal Vim variables. (Ron Aaron)
Solution: Do show ":v" variables for ":let" and ":let v:name".
Files: src/eval.c

Patch 5.6a.008

Problem: Selecting a syntax from the Syntax menu gives an error message.
Solution: Replace "else if" in SetSyn() with "elseif". (Ronald Schild)
Files: runtime/menu.vim

Patch 5.6a.009

Problem: When compiling with +extra_search but without +syntax, there is a compilation error in screen.c. (Axel Kielhorn)
Solution: Adjust the #ifdef for declaring and initializing "line" in win_line(). Also solve compilation problem when +statusline is used without +eval. Another one when +cmdline_compl is used without +eval.
Files: src/screen.c, src/misc2.c

Patch 5.6a.010

Problem: In a function, ":startinsert!" does not append to the end of the line if a ":normal" command was used to move the cursor. (Fisher)
Solution: Reset "w_set_curswant" to avoid that w_curswant is changed again.
Files: src/ex_docmd.c

Patch 5.6a.011 (depends on 5.6a.004)

Problem: A missing ":endif" or ":endwhile" in a function doesn't give an error message.
Solution: Give that error message.
Files: src/ex_docmd.c

Patch 5.6a.012 (depends on 5.6a.008)

Problem: Some Syntax menu entries caused a hit-enter prompt.
Solution: Call a function to make the command shorter. Also rename a few functions to avoid name clashes.
Files: runtime/menu.vim

Patch 5.6a.013

Problem: Command line completion works different when another completion was done earlier. (Johannes Zellner)
Solution: Reset wim_index when starting a new completion.
Files: src/ex_getln.c

Patch 5.6a.014

Problem: Various warning messages when compiling and running lint with different combinations of features.
Solution: Fix the warning messages.
Files: src/eval.c, src/ex_cmds.c, src/ex_docmd.c, src/gui_gtk_x11.c, src/option.c, src/screen.c, src/search.c, src/syntax.c, src/feature.h, src/globals.h

Patch 5.6a.015

Problem: The vimtutor command doesn't always know the value of \$VIMRUNTIME.
Solution: Let Vim expand \$VIMRUNTIME, instead of the shell.
Files: src/vimtutor

Patch 5.6a.016 (extra)

Problem: Mac: Window size is restricted when starting. Cannot drag the window all over the desktop.
Solution: Get real screen size instead of assuming 640x400. Do not use a fixed number for the drag limits. (Axel Kielhorn)
Files: src/gui_mac.c

Patch 5.6a.017

Problem: The "Paste" entry in popup menu for Visual, Insert and Cmdline mode is in the wrong position. (Stol)
Solution: Add priority numbers for all Paste menu entries.
Files: runtime/menu.vim

Patch 5.6a.018

Problem: GTK GUI: submenu priority doesn't work.
Help dialog could be destroyed too soon.
When closing a dialog window (e.g. the "ATTENTION" one), Vim would just hang.
When GTK theme is changed, Vim doesn't adjust to the new colors.
Argument for ":promptfind" isn't used.
Solution: Fixed the mentioned problems.
Made the dialogs look&feel nicer.
Moved functions to avoid the need for a forward declaration.
Fixed reentrancy of the file browser dialog.
Added drag&drop support for GNOME.
Init the text for the Find/replace dialog from the last used search string. Set "match whole word" toggle button correctly.
Made repeat rate for drag outside of window depend on the distance from the window. (Marcin Dalecki)
Made the drag in Visual mode actually work.
Removed recursiveness protection from gui_mch_get_rgb(), it might cause more trouble than it solves.
Files: src/ex_docmd.c, src/gui_gtk.c, src/gui_gtk_x11.c, src/ui.c, src/proto/ui.pro, src/misc2.c

Patch 5.6a.019

Problem: When trying to recover through NFS, which uses a large block size, Vim might think the swap file is empty, because mf_blocknr_max is zero. (Scott McDermott)
Solution: When computing the number of blocks of the file in mf_open(), round up instead of down.
Files: src/memfile.c

Patch 5.6a.020

Problem: GUI GTK: Could not set display for gvim.
Solution: Add "--display" and "--display" arguments. (Marcin Dalecki)
Files: src/gui_gtk_x11.c

Patch 5.6a.021

Problem: Recovering still may not work when the block size of the device where the swap file is located is larger than 4096.
Solution: Read block 0 with the minimal block size.
Files: src/memline.c, src/memfile.c, src/vim.h

Patch 5.6a.022 (extra)

Problem: Win32 GUI: When an error in the vimrc causes a dialog to pop up (e.g., for an existing swap file), Vim crashes. (David Elins)
Solution: Before showing a dialog, open the main window.
Files: src/gui_w32.c

Patch 5.6a.023

Problem: Using expand("%:gs??/?") causes a crash. (Ron Aaron)
Solution: Check for running into the end of the string in do_string_sub().
Files: src/eval.c

Patch 5.6a.024

Problem: Using an autocommand to delete a buffer when leaving it can cause a crash when jumping to a tag. (Franz Gorkotte)
Solution: In do_tag(), store tagstacklen before jumping to another buffer. Check tagstackidx after jumping to another buffer. Add extra check in win_split() if tagname isn't NULL.
Files: src/tag.c, src/window.c

Patch 5.6a.025 (extra)

Problem: Win32 GUI: The tables for toupper() and tolower() are initialized too late. (Mike Steed)
Solution: Move the initialization to win32_init() and call it from main().
Files: src/main.c, src/os_w32.c, src/proto/os_w32.pro

Patch 5.6a.026

Problem: When the SNIFF connection is open, shell commands hang. (Pruemmer)
Solution: Skip a second wait() call if waitpid() already detected that the child has exited.
Files: src/os_unix.c

Patch 5.6a.027 (extra)

Problem: Win32 GUI: The "Edit with Vim" popup menu entry causes problems for the Office toolbar.
Solution: Use a shell extension dll. (Tianmiao Hu)
Added it to the install and uninstal programs, replaces the old "Edit with Vim" menu registry entries.
Files: src/dosinst.c, src/uninstal.c, gvimext/*, runtime/doc/gui_w32.txt

Patch 5.6a.028 (extra)

Problem: Win32 GUI: Dialogs and tear-off menus can't handle multibyte characters.
Solution: Adjust nCopyAnsiToWideChar() to handle multibyte characters correctly.
Files: src/gui_w32.c

=====

VERSION 5.7 version-5.7

Version 5.7 is a bug-fix version of 5.6.

Changed

changed-5.7

Renamed src/INSTALL.mac to INSTALL_mac.txt to avoid it being recognized with a wrong file type. Also renamed src/INSTALL.amiga to INSTALL_ami.txt.

Added

added-5.7

New syntax files:

stp	Stored Procedures (Jeff Lanzarotta)
snnstnet, snnspat, snnsres	SNNS (Davide Alberani)
mel	MEL (Robert Minsk)
ruby	Ruby (Mirko Nasato)
tli	TealInfo (Kurt W. Andrews)
ora	Oracle config file (Sandor Kopanyi)
abaqus	Abaqus (Carl Osterwisch)
jproperties	Java Properties (Simon Baldwin)
apache	Apache config (Allan Kelly)
csp	CSP (Jan Bredereke)
samba	Samba config (Rafael Garcia-Suarez)
kscript	KDE script (Thomas Capricelli)
hb	Hyper Builder (Alejandro Forero Cuervo)
fortran	Fortran (rewritten) (Ajit J. Thakkar)
sml	SML (Fabrizio Zeno Cornelli)
cvs	CVS commit (Matt Dunford)
aspperl	ASP Perl (Aaron Hope)
bc	BC calculator (Vladimir Scholtz)
latte	Latte (Nick Moffitt)
wml	WML (Gerfried Fuchs)

Included Exuberant ctags 3.5.1. (Darren Hiebert)

"display" and "fold" arguments for syntax items. For future extension, they are ignored now.

strftime() function for the Macintosh.

macros/explorer.vim: A file browser script (M A Aziz Ahmed)

Fixed

fixed-5.7

The 16 bit MS-DOS version is now compiled with Bcc 3.1 instead of 4.0. The executable is smaller.

When a "make test" failed, the output file was lost. Rename it to test99.failed to be able to see what went wrong.

After sourcing bugreport.vim, it's not clear that bugreport.txt has been

written in the current directory. Edit bugreport.txt to avoid that.

Adding IME support when using Makefile.w32 didn't work. (Taro Muraoka)

Win32 console: Mouse drags were passed on even when the mouse didn't move.

Perl interface: In Buffers(), type of argument to SvPV() was int, should be STRLEN. (Tony Leneis)

Problem with prototype for index() on AIX 4.3.0. Added check for _AIX43 in os_unix.h. (Jake Hamby)

Mappings in mswin.vim could break when some commands are mapped. Add "nore" to most mappings to avoid re-mapping.

modify_fname() made a copy of a file name for ":p" when it already was a full path name, which is a bit slow.

Win32 with Borland C++ 5.5: Pass the path to the compiler on to xxd and ctags, to avoid depending on \$PATH. Fixed "make clean".

Many fixes to Macintosh specific parts: (mostly by Dany StAmant)

- Only one Help menu.
- No more crash when removing a menu item.
- Support as External Editor for Codewarrior (still some little glitches).
- Popup menu support.
- Fixed crash when pasting after application switch.
- Color from rgb.txt properly displayed.
- **'isprint'** default includes all chars above '~'. (Axel Kielhorn)
- mac_expandpath() was leaking memory.
- Add digraphs table. (Axel Kielhorn)
- Multi-byte support: (Kenichi Asai)
 - Switch keyscript when going in/out of Insert mode.
 - Draw multibyte character correctly.
 - Don't use mblen() but highest bit of char to detect multibyte char.
 - Display value of multibyte in statusline (also for other systems).
- mouse button was not initialized properly to MOUSE_LEFT when USE_CTRLCLICKMENU not defined.
- With Japanese SJIS characters: Make "w", "b", and "e" work properly. (Kenichi Asai)
- Replaced old CodeWarrior file os_mac.CW9.hqx with os_mac.cw5.sit.hqx.

Fixes for VMS: (Zoltan Arpadffy) (also see patch 5.6.045 below)

- Added Makefile_vms.mms and vimrc.vms to src/testdir to be able to run the tests.
- Various fixes.
- Set **'undolevels'** to 1000 by default.
- Made mch_settitle() equivalent to the one in os_unix.c.

RiscOS: A few prototypes for os_riscos.c were outdated. Generate prototypes automatically.

Previously released patches:

Patch 5.6.001

Problem: When using "set bs=0 si cin", Inserting "#<BS>" or "<BS>" which reduces the indent doesn't delete the "#" or "}". (Lorton)
Solution: Adjust ai_col in ins_try_si().
Files: src/edit.c

Patch 5.6.002

Problem: When using the vim.vim syntax file, a comment with all uppercase characters causes a hang.
Solution: Adjust pattern for vimCommentTitle (Charles Campbell)
Files: runtime/syntax/vim.vim

Patch 5.6.003

Problem: GTK GUI: Loading a user defined toolbar bitmap gives a warning about the colormap. Probably because the window has not been opened yet.
Solution: Use gdk_pixmap_colormap_create_from_xpm() to convert the xpm file. (Keith Radebaugh)
Files: src/gui_gtk.c

Patch 5.6.004 (extra)

Problem: Win32 GUI with IME: When setting '**guifont**' to "*", the font requester appears twice.
Solution: In gui_mch_init_font() don't call get_logfont() but copy norm_logfont from fh. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 5.6.005

Problem: When '**winminheight**' is zero, **CTRL-W** - with a big number causes a crash. (David Kotchan)
Solution: Check for negative window height in win_setheight().
Files: src/window.c

Patch 5.6.006

Problem: GTK GUI: Bold font cannot always be used. Memory is freed too early in gui_mch_init_font().
Solution: Move call to g_free() to after where sdup is used. (Artem Hodyush)
Files: src/gui_gtk_x11.c

Patch 5.6.007 (extra)

Problem: Win32 IME: Font is not changed when screen font is changed. And IME composition window does not trace the cursor.
Solution: Initialize IME font. When cursor is moved, set IME composition window with ImeSetCompositionWindow(). Add call to ImmReleaseContext() in several places. (Taro Muraoka)
Files: src/gui.c, src/gui_w32.c, src/proto/gui_w32.pro

Patch 5.6.008 (extra)

Problem: Win32: When two files exist with the same name but different case (through NFS or Samba), fixing the file name case could cause the wrong one to be edited.
Solution: Prefer a perfect match above a match while ignoring case in fname_case(). (Flemming Madsen)

Files: src/os_win32.c

Patch 5.6.009 (extra)

Problem: Win32 GUI: Garbage in Windows Explorer help line when selecting "Edit with Vim" popup menu entry.

Solution: Only return the help line when called with the GCS_HELPTEXT flag. (Tianmiao Hu)

Files: GvimExt/gvimext.cpp

Patch 5.6.010

Problem: A file name which contains a TAB was not read correctly from the viminfo file and the ":ls" listing was not aligned properly.

Solution: Parse the buffer list lines in the viminfo file from the end backwards. Count a Tab for two characters to align the ":ls" list.

Files: src/buffer.c

Patch 5.6.011

Problem: When 'columns' is huge (using a tiny font) and 'statusline' is used, Vim can crash.

Solution: Limit maxlen to MAXPATHL in win_redr_custom(). (John Mullin)

Files: src/screen.c

Patch 5.6.012

Problem: When using "zsh" for /bin/sh, toolcheck may hang until "exit" is typed. (Kuratczyk)

Solution: Add "-c exit" when checking for the shell version.

Files: src/toolcheck

Patch 5.6.013

Problem: Multibyte char in tooltip is broken.

Solution: Consider multibyte char in replace_termcodes(). (Taro Muraoka)

Files: src/term.c

Patch 5.6.014

Problem: When cursor is at the end of line and the character under cursor is a multibyte character, "yl" doesn't yank 1 multibyte-char. (Takuhiro Nishioka)

Solution: Recognize a multibyte-char at end-of-line correctly in oneright(). (Taro Muraoka)

Also: make "+quickfix" in ":version" output appear alphabetically.

Files: src/edit.c

Patch 5.6.015

Problem: New xterm delete key sends <Esc>[3~ by default.

Solution: Added <kDel> and <kIns> to make the set of keypad keys complete.

Files: src/edit.c, src/ex_getln.c, src/keymap.h, src/misc1.c, src/misc2.c, src/normal.c, src/os_unix.c, src/term.c

Patch 5.6.016

Problem: When deleting a search string from history from inside a mapping, another entry is deleted too. (Benji Fisher)

Solution: Reset last_maptick when deleting the last entry of the search history. Also: Increment maptick when starting a mapping from typed characters to avoid a just added search string being

overwritten or removed from history.
Files: src/ex_getln.c, src/getchar.c

Patch 5.6.017

Problem: ":s/e/\^M/" should replace an "e" with a **CTRL-M**, not split the line. (Calder)

Solution: Replace the backslash with a **CTRL-V** internally. (Stephen P. Wall)

Files: src/ex_cmds.c

Patch 5.6.018

Problem: ":help [:digit:]" takes a long time to jump to the wrong place.

Solution: Insert a backslash to avoid the special meaning of '[]'.

Files: src/ex_cmds.c

Patch 5.6.019

Problem: "snd.c", "snd.java", etc. were recognized as "mail" filetype.

Solution: Make pattern for mail filetype more strict.

Files: runtime/filetype.vim

Patch 5.6.020 (extra)

Problem: The DJGPP version eats processor time (Walter Briscoe).

Solution: Call `__dpmi_yield()` in the busy-wait loop.

Files: src/os_msdos.c

Patch 5.6.021

Problem: When '**selection**' is "exclusive", a double mouse click in Insert mode doesn't select last char in line. (Lutz)

Solution: Allow leaving the cursor on the NUL past the line in this case.

Files: src/edit.c

Patch 5.6.022

Problem: ":e \^{<Tab>}" expands to ":e ~\^{\$ceelen}", which doesn't work.

Solution: Re-insert the backslash before the '~'.

Files: src/ex_getln.c

Patch 5.6.023 (extra)

Problem: Various warnings for the Ming compiler.

Solution: Changes to avoid the warnings. (Bill McCarthy)

Files: src/ex_cmds.c, src/gui_w32.c, src/os_w32exe.c, src/os_win32.c, src/syntax.c, src/vim.rc

Patch 5.6.024 (extra)

Problem: Win32 console: Entering **CTRL-^{_}** requires the shift key. (Kotchan)

Solution: Specifically catch keycode 0xBD, like the GUI.

Files: src/os_win32.c

Patch 5.6.025

Problem: GTK GUI: Starting the GUI could be interrupted by a SIGWINCH. (Nils Lohner)

Solution: Repeat the read() call to get the gui_in_use value when interrupted by a signal.

Files: src/gui.c

Patch 5.6.026 (extra)

Problem: Win32 GUI: Toolbar bitmaps are searched for in \$VIMRUNTIME/bitmaps, while GTK looks in \$VIM/bitmaps. (Keith Radebaugh)
Solution: Use \$VIM/bitmaps for both, because these are not part of the distribution but defined by the user.
Files: src/gui_w32.c, runtime/doc/gui.txt

Patch 5.6.027

Problem: TCL: Crash when using a Tcl script (reported for Win32).
Solution: Call Tcl_FindExecutable() in main(). (Brent Fulgham)
Files: src/main.c

Patch 5.6.028

Problem: Xterm patch level 126 sends codes for mouse scroll wheel. Fully works with xterm patch level 131.
Solution: Recognize the codes for button 4 (0x60) and button 5 (0x61).
Files: src/term.c

Patch 5.6.029

Problem: GTK GUI: Shortcut keys cannot be used for a dialog. (Johannes Zellner)
Solution: Add support for shortcut keys. (Marcin Dalecki)
Files: src/gui_gtk.c

Patch 5.6.030

Problem: When closing a window and 'ea' is set, Vim can crash. (Yasuhiro Matsumoto)
Solution: Set "curbuf" to a valid value in win_close().
Files: src/window.c

Patch 5.6.031

Problem: Multi-byte: When a double-byte character ends in CSI, Vim waits for another character to be typed.
Solution: Recognize the CSI as the second byte of a character and don't wait for another one. (Yasuhiro Matsumoto)
Files: src/getchar.c

Patch 5.6.032

Problem: Functions with an argument that is a line number don't all accept ".", "\$", etc. (Ralf Arens)
Solution: Add get_art_lnum() and use it for setline(), line2byte() and synID().
Files: src/eval.c

Patch 5.6.033

Problem: Multi-byte: "f " sometimes skips to the second space. (Sung-Hyun Nam)
Solution: Change logic in searchc() to skip trailing byte of a double-byte character.
Also: Ask for second byte when searching for double-byte character. (Park Chong-Dae)
Files: src/search.c

Patch 5.6.034 (extra)

Problem: Compiling with Borland C++ 5.5 fails on tolower() and toupper().
Solution: Use TO_LOWER() and TO_UPPER() instead. Also adjust the Makefile to make using bcc 5.5 easier.
Files: src/edit.c, src/ex_docmd.c, src/misc1.c, src/Makefile.bor

Patch 5.6.035

Problem: Listing the "+comments" feature in the ":version" output depended on the wrong ID. (Stephen P. Wall)
Solution: Change "CRYPTV" to "COMMENTS".
Files: src/version.c

Patch 5.6.036

Problem: GTK GUI: Copy/paste text doesn't work between gvim and Eterm.
Solution: Support TEXT and COMPOUND_TEXT selection targets. (ChiDeok Hwang)
Files: src/gui_gtk_x11.c

Patch 5.6.037

Problem: Multi-byte: Can't use "f" command with multibyte character in GUI.
Solution: Enable XIM in Normal mode for the GUI. (Sung-Hyun Nam)
Files: src/gui_gtk_x11.c, src/multbyte.c

Patch 5.6.038

Problem: Multi-clicks in GUI are interpreted as a mouse wheel click. When **'ttymouse'** is "xterm" a mouse click is interpreted as a mouse wheel click.
Solution: Don't recognize the mouse wheel in check_termcode() in the GUI. Use 0x43 for a mouse drag in do_xterm_trace(), not 0x63.
Files: src/term.c, src/os_unix.c

Patch 5.6.039

Problem: Motif GUI under KDE: When trying to logout, Vim hangs up the system. (Hermann Rochholz)
Solution: When handling the WM_SAVE_YOURSELF event, set the WM_COMMAND property of the window to let the session manager know we finished saving ourselves.
Files: src/gui_x11.c

Patch 5.6.040

Problem: When using ":s" command, matching the regexp is done twice.
Solution: After copying the matched line, adjust the pointers instead of finding the match again. (Loic Grenie) Added vim_regnewptr().
Files: src/ex_cmds.c, src/regexp.c, src/proto/regexp.pro

Patch 5.6.041

Problem: GUI: Athena, Motif and GTK don't give more than 10 dialog buttons.
Solution: Remove the limit on the number of buttons.
Also support the 'v' flag in **'guioptions'**.
For GTK: Center the buttons.
Files: src/gui_athena.c, src/gui_gtk.c, src/gui_motif.c

Patch 5.6.042

Problem: When doing "vim -u vimrc" and vimrc contains ":q", the cursor in the terminal can remain off.
Solution: Call cursor_on() in mch_windexit().

Files: src/os_unix.c

Patch 5.6.043 (extra)

Problem: Win32 GUI: When selecting guifont with the dialog, 'guifont' doesn't include the bold or italic attributes.

Solution: Append ":i" and/or ":b" to 'guifont' in gui_mch_init_font().

Files: src/gui_w32.c

Patch 5.6.044 (extra)

Problem: MS-DOS and Windows: The line that dosinst.exe appends to autoexec.bat to set PATH is wrong when Vim is in a directory with an embedded space.

Solution: Use double quotes for the value when there is an embedded space.

Files: src/dosinst.c

Patch 5.6.045 (extra) (fixed version)

Problem: VMS: Various small problems.

Solution: Many small changes. (Zoltan Arpadffy)
File name modifier ":h" keeps the path separator.
File name modifier ":e" also removes version.
Compile with MAX_FEAT by default.
When checking for autocommands ignore version in file name.
Be aware of file names being case insensitive.
Added vt320 builtin termcap.
Be prepared for an empty default_vim_dir.

Files: runtime/gvimrc_example.vim, runtime/vimrc_example.vim,
runtime/doc/os_vms.txt, src/eval.c, src/feature.h, src/fileio.c,
src/gui_motif.c, src/gui_vms_conf.h, src/main.c, src/memline.c,
src/misc1.c, src/option.c, src/os_vms_conf.h, src/os_vms.c,
src/os_vms.h, src/os_vms.mms, src/tag.c, src/term.c, src/version.c

Patch 5.6.046

Problem: Systems with backslash in file name: With 'shellslash' set, "vim */*.c" only uses a slash for the first file name. (Har'El)

Solution: Fix slashes in file name arguments after reading the vimrc file.

Files: src/option.c

Patch 5.6.047

Problem: \$CPPFLAGS is not passed on to ctags configure.

Solution: Add it. (Walter Briscoe)

Files: src/config.mk.in, src/Makefile

Patch 5.6.048

Problem: **CTRL-R** in Command-line mode is documented to insert text as typed, but inserts text literally.

Solution: Make **CTRL-R** insert text as typed, use **CTRL-R CTRL-R** to insert literally. This is consistent with Insert mode. But characters that end Command-line mode are inserted literally.

Files: runtime/doc/index.txt, runtime/doc/cmdline.txt, src/ex_getln.c, src/ops.c, src/proto/ops.pro

Patch 5.6.049

Problem: Documentation for [!] after ":ijump" is wrong way around. (Benji Fisher)

Solution: Fix the documentation. Also improve the code to check for a match after a /* */ comment.
Files: runtime/doc/tagsearch.txt, src/search.c

Patch 5.6.050

Problem: Replacing is wrong when replacing a single-byte char with double-byte char or the other way around.
Solution: Shift the text after the character when it is replaced.
(Yasuhiro Matsumoto)
Files: src/normal.c, src/misc1.c

Patch 5.6.051

Problem: ":tprev" and ":tnext" don't give an error message when trying to go before the first or beyond the last tag. (Robert Webb)
Solution: Added error messages. Also: Delay a second when a file-read message is going to overwrite an error message, otherwise it won't be seen.
Files: src/fileio.c, src/tag.c

Patch 5.6.052

Problem: Multi-byte: When an Ex command has a '|' or '"' as a second byte, it terminates the command.
Solution: Skip second byte of multibyte char when checking for '|' and '".
(Asai Kenichi)
Files: src/ex_docmd.c

Patch 5.6.053

Problem: **CTRL-]** doesn't work on a tag that contains a '|'. (Cesar Crusius)
Solution: Escape '|', '"' and '\' in tag names when using **CTRL-]** and also for command-line completion.
Files: src/ex_getln.c, src/normal.c

Patch 5.6.054

Problem: When using ":e" and ":e #" the cursor is put in the first column when **'startofline'** is set. (Cordell)
Solution: Use the last known column when **'startofline'** is set. Also, use ECMD_LAST more often to simplify the code.
Files: src/buffer.c, src/ex_cmds.c, src/ex_docmd.c, src/proto/buffer.pro

Patch 5.6.055

Problem: When **'statusline'** only contains a text without "%" and doesn't fit in the window, Vim crashes. (Ron Aaron)
Solution: Don't use the pointer for the first item if there is no item.
Files: src/screen.c

Patch 5.6.056 (extra)

Problem: MS-DOS: F11 and F12 don't work when **'bioskey'** is set.
Solution: Use enhanced keyboard functions. (Vince Negri)
Detect presence of enhanced keyboard and set bioskey_read and bioskey_ready.
Files: src/os_msdos.c

Patch 5.6.057 (extra)

Problem: Win32 GUI: Multi-byte characters are wrong in dialogs and tear-off

menus.
Solution: Use system font instead of a fixed font. (Matsumoto, Muraoka)
Files: src/gui_w32.c

Patch 5.6.058

Problem: When the 'a' flag is not in '**guioptions**', non-Windows systems copy Visually selected text to the clipboard/selection on a yank or delete command anyway. On Windows it isn't done even when the 'a' flag is included.
Solution: Respect the 'a' flag in '**guioptions**' on all systems.
Files: src/normal.c

Patch 5.6.059 (extra)

Problem: When moving the cursor over italic text and the characters spill over to the cell on the right, that spill-over is deleted. Noticed in the Win32 GUI, can happen on other systems too.
Solution: Redraw italic text starting from a blank, like this is already done for bold text. (Vince Negri)
Files: src/gui.c, src/gui.h, src/gui_w32.c

Patch 5.6.060

Problem: Some bold characters spill over to the cell on the left, that spill-over can remain sometimes.
Solution: Redraw a character when the next character was bold and needs redrawing. (Robert Webb)
Files: src/screen.c

Patch 5.6.061

Problem: When xterm sends 8-bit controls, recognizing the version response doesn't work.
When using CSI instead of `<Esc>[` for the termcap color codes, using 16 colors doesn't work. (Neil Bird)
Solution: Also accept CSI in place of `<Esc>[` for the version string. Also check for CSI when handling colors 8-15 in `term_color()`. Use CSI for builtin xterm termcap entries when '**term**' contains "8bit".
Files: runtime/doc/term.txt, src/ex_cmds.c, src/option.c, src/term.c, src/os_unix.c, src/proto/option.pro, src/proto/term.pro

Patch 5.6.062

Problem: The documentation says that setting '**smartindent**' doesn't have an effect when '**cindent**' is set, but it does make a difference for lines starting with "#". (Neil Bird)
Solution: Really ignore '**smartindent**' when '**cindent**' is set.
Files: src/misc1.c, src/ops.c

Patch 5.6.063

Problem: Using "I" in Visual-block mode doesn't accept a count. (Johannes Zellner)
Solution: Pass the count on to `do_insert()` and `edit()`. (Allan Kelly)
Files: src/normal.c, src/ops.c, src/proto/ops.pro

Patch 5.6.064

Problem: MS-DOS and Win32 console: Mouse doesn't work correctly after

including patch 5.6.28. (Vince Negri)
Solution: Don't check for mouse scroll wheel when the mouse code contains the number of clicks.
Files: src/term.c

Patch 5.6.065

Problem: After moving the cursor around in Insert mode, typing a space can still trigger an abbreviation. (Benji Fisher)
Solution: Don't check for an abbreviation after moving around in Insert mode.
Files: src/edit.c

Patch 5.6.066

Problem: Still a few bold character spill-over remains after patch 60.
Solution: Clear character just in front of blanking out rest of the line. (Robert Webb)
Files: src/screen.c

Patch 5.6.067

Problem: When a file name contains a NL, the viminfo file is corrupted.
Solution: Use viminfo_writestring() to convert the NL to **CTRL-V** n. Also fix the Buffers menu and listing a menu name with a newline.
Files: runtime/menu.vim, src/buffer.c, src/mark.c, src/menu.c

Patch 5.6.068

Problem: Compiling the Perl interface doesn't work with Perl 5.6.0. (Bernhard Rosenkraenzer)
Solution: Also check xs_apiversion for the version number when prepending defines for PL_*.
Files: src/Makefile

Patch 5.6.069

Problem: "go" doesn't always end up at the right character when **'fileformat'** is "dos". (Bruce DeVisser)
Solution: Correct computations in ml_find_line_or_offset().
Files: src/memline.

Patch 5.6.070 (depends on 5.6.068)

Problem: Compiling the Perl interface doesn't work with Perl 5.6.0. (Bernhard Rosenkraenzer)
Solution: Simpler check instead of the one from patch 68.
Files: src/Makefile

Patch 5.6.071

Problem: "A" in Visual block mode on a Tab positions the cursor one char to the right. (Michael Haumann)
Solution: Correct the column computation in op_insert().
Files: src/ops.c

Patch 5.6.072

Problem: When starting Vim with "vim +startinsert", it enters Insert mode only after typing the first command. (Andrew Pimlott)
Solution: Insert a dummy command in the stuff buffer.
Files: src/main.c

Patch 5.6.073 (extra) (depends on 5.6.034)

Problem: Win32 GUI: When compiled with Bcc 5.5 menus don't work.
In dosinst.c toupper() and tolower() give an "internal compiler error" for Bcc 5.5.

Solution: Define WINVER to 4 to avoid compiling for Windows 2000. (Dan Sharp) Also cleaned up compilation arguments.
Use our own implementation of toupper() in dosinst.c. Use mytoupper() instead of tolower().

Files: src/Makefile.bor, src/dosinst.c

Patch 5.6.074 (extra)

Problem: Entering CSI directly doesn't always work, because it's recognized as the start of a special key. Mostly a problem with multibyte in the GUI.

Solution: Use K_CSI for a typed CSI character. Use <CSI> for a normal CSI, <xCSI> for a CSI typed in the GUI.

Files: runtime/doc/intro.txt, src/getchar.c, src/gui_amiga.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_riscos.c, src/gui_w32.c, src/keymap.h, src/misc2.c

Patch 5.6.075

Problem: When using "I" or "A" in Visual block mode while 'sts' is set may change spaces to a Tab the inserted text is not correct. (Mike Steed) And some other problems when using "A" to append after the end of the line.

Solution: Check for change in spaces/tabs after inserting the text. Append spaces to fill the gap between the end-of-line and the right edge of the block.

Files: src/ops.c

Patch 5.6.076

Problem: GTK GUI: Mapping <M-Space> doesn't work.

Solution: Don't use the "Alt" modifier twice in key_press_event().

Files: src/gui_gtk_x11.c

Patch 5.6.077

Problem: GUI: When interrupting an external program with **CTRL-C**, gvim might crash. (Benjamin Korvemaker)

Solution: Avoid using a NULL pointer in ui_inchar_undo().

Files: src/ui.c

Patch 5.6.078

Problem: Locale doesn't always work on FreeBSD. (David O'Brien)

Solution: Link with the "xpg4" library when available.

Files: src/configure.in, src/configure

Patch 5.6.079

Problem: Vim could crash when several Tcl interpreters are created and destroyed.

Solution: handle the "exit" command and nested ":tcl" commands better. (Ingo Wilken)

Files: runtime/doc/if_tcl.txt, src/if_tcl.c

Patch 5.6.080

Problem: When jumping to a tag, generating the tags file and jumping to the same tag again uses the old search pattern. (Sung-Hyun Nam)
Solution: Flush cached tag matches when executing an external command.
Files: src/misc2.c, src/proto/tag.pro, src/tag.c

Patch 5.6.081

Problem: ":syn include" uses a level for the included file, this confuses contained items included at the same level.
Solution: Use a unique tag for each included file. Changed sp_syn_inc_lvl to sp_syn_inc_tag. (Scott Bigham)
Files: src/syntax.c, src/structs.h

Patch 5.6.082

Problem: When using cscope, Vim can crash.
Solution: Initialize tag_fname in find_tags(). (Anton Blanchard)
Files: src/tag.c

Patch 5.6.083 (extra)

Problem: Win32: The visual beep can't be seen. (Eric Roesinger)
Solution: Flush the output before waiting with GdiFlush(). (Maurice S. Barnum)
Also: Allow specifying the delay in t_vb for the GUI.
Files: src/gui.c, src/gui_amiga.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_riscos.c, src/gui_w32.c, src/gui_x11.c, src/gui_beos.cc, src/proto/gui_amiga.pro, src/proto/gui_gtk_x11.pro, src/proto/gui_mac.pro, src/proto/gui_riscos.pro, src/proto/gui_w32.pro, src/proto/gui_x11.pro, src/proto/gui_beos.pro

Patch 5.6.084 (depends on 5.6.074)

Problem: GUI: Entering CSI doesn't always work for Athena and Motif.
Solution: Handle typed CSI as <xCSI> (forgot this bit in 5.6.074).
Files: src/gui_x11.c

Patch 5.6.085

Problem: Multi-byte: Using "r" to replace a double-byte char with a single-byte char moved the cursor one character. (Matsumoto)
Also, using a count when replacing a single-byte char with a double-byte char didn't work.
Solution: Don't use del_char() to delete the second byte.
Get "ptr" again after calling ins_char().
Files: src/normal.c

Patch 5.6.086 (extra)

Problem: Win32: When using libcall() and the returned value is not a valid pointer, Vim crashes.
Solution: Use IsBadStringPtr() to check if the pointer is valid.
Files: src/os_win32.c

Patch 5.6.087

Problem: Multi-byte: Commands and messages with multibyte characters are displayed wrong.
Solution: Detect double-byte characters. (Yasuhiro Matsumoto)
Files: src/ex_getln.c, src/message.c, src/misc2.c, src/screen.c

Patch 5.6.088

Problem: Multi-byte with Motif or Athena: The message "XIM requires fontset" is annoying when Vim was compiled with XIM support but it is not being used.
Solution: Remove that message.
Files: src/multbyte.c

Patch 5.6.089

Problem: On non-Unix systems it's possible to overwrite a read-only file without using "!".
Solution: Check if the file permissions allow overwriting before moving the file to become the backup file.
Files: src/fileio.c

Patch 5.6.090

Problem: When editing a file in "/home/dir/home/dir" this was replaced with "~". (Andreas Jellinghaus)
Solution: Replace the home directory only once in home_replace().
Files: src/misc1.c

Patch 5.6.091

Problem: When editing many "no file" files, can't create swap file, because .sw[a-p] have all been used. (Neil Bird)
Solution: Also use ".sv[a-z]", ".su[a-z]", etc.
Files: src/memline.c

Patch 5.6.092

Problem: FreeBSD: When setting \$TERM to a non-valid terminal name, Vim hangs in tputs().
Solution: After tgetent() returns an error code, call it again with the terminal name "dumb". This apparently creates an environment in which tputs() doesn't fail.
Files: src/term.c

Patch 5.6.093 (extra)

Problem: Win32 GUI: "ls | gvim -" will show a message box about reading stdin when Vim exits. (Donohue)
Solution: Don't write a message about the file read from stdin until the GUI has started.
Files: src/fileio.c

Patch 5.6.094

Problem: Problem with multibyte string for ":echo var".
Solution: Check for length in msg_outtrans_len_attr(). (Sung-Hyun Nam)
Also make do_echo() aware of multibyte characters.
Files: src/eval.c, src/message.c

Patch 5.6.095

Problem: With an Emacs TAGS file that include another a relative path doesn't always work.
Solution: Use expand_tag_fname() on the name of the included file. (Utz-Uwe Haus)
Files: src/tag.c

Patch 5.6.096

Problem: Unix: When editing many files, startup can be slow. (Paul Ackersviller)
Solution: Halve the number of stat() calls used to add a file to the buffer list.
Files: src/buffer.c

Patch 5.7a.001

Problem: GTK doesn't respond on drag&drop from ROX-Filer.
Solution: Add "text/uri-list" target. (Thomas Leonard)
Also: fix problem with checking for trash arguments.
Files: src/gui_gtk_x11.c

Patch 5.7a.002

Problem: Multi-byte: 'showmatch' is performed when second byte of an inserted double-byte char is a paren or brace.
Solution: Check IsTrailByte() before calling showmatch(). (Taro Muraoka)
Files: src/misc1.c

Patch 5.7a.003

Problem: Multi-byte: After using **CTRL-O** in Insert mode with the cursor at the end of the line on a multibyte character the cursor moves to the left.
Solution: Check for multibyte character at end-of-line. (Taro Muraoka)
Also: fix cls() to detect a double-byte character. (Chong-Dae Park)
Files: src/edit.c, src/search.c

Patch 5.7a.004

Problem: When reporting the search pattern offset, the string could be unterminated, which may cause a crash.
Solution: Terminate the string for the search offset. (Stephen P. Wall)
Files: src/search.c

Patch 5.7a.005

Problem: When ":s//~/\" doesn't find a match it reports "[NULL]" for the pattern.
Solution: Use get_search_pat() to obtain the actually used pattern.
Files: src/ex_cmds.c, src/proto/search.pro, src/search.c

Patch 5.7a.006 (extra)

Problem: VMS: Various problems, also with the VAXC compiler.
Solution: In many places use the Unix code for VMS too.
Added time, date and compiler version to version message.
(Zoltan Arpadffy)
Files: src/ex_cmds.c, src/ex_doccmd.c, src/globals.h, src/gui_vms_conf.h, src/main.c, src/message.c, src/misc1.c, src/os_vms.c, src/os_vms.h, src/os_vms.mms, src/os_vms_conf.h, src/proto/os_vms.pro, src/proto/version.pro, src/term.c, src/version.c, src/xxd/os_vms.mms, src/xxd/xxd.c

Patch 5.7a.007

Problem: Motif and Athena GUI: **CTRL-@** is interpreted as **CTRL-C**.
Solution: Only use "intr_char" when it has been set.
Files: src/gui_x11.c

Patch 5.7a.008

Problem: GTK GUI: When using **CTRL-L** the screen is redrawn twice, causing trouble for bold characters. Also happens when moving with the scrollbar. Best seen when **'writedelay'** is non-zero. When starting the GUI with **":gui"** the screen is redrawn once with the wrong colors.

Solution: Only set the geometry hints when the window size really changed. This avoids setting it each time the scrollbar is forcefully redrawn. Don't redraw in `expose_event()` when `gui.starting` is still set.

Files: `src/gui_gtk_x11.c`

VERSION 5.8

version-5.8

Version 5.8 is a bug-fix version of 5.7.

Changed

changed-5.8

Ctags is no longer included with Vim. It has grown into a project of its own. You can find it here: <http://ctags.sf.net>. It is highly recommended as a Vim companion when you are writing programs.

Added

added-5.8

New syntax files:

acedb	AceDB (Stewart Morris)
aflex	Aflex (Mathieu Clabaut)
antlr	Antlr (Mathieu Clabaut)
asm68k	68000 Assembly (Steve Wall)
automake	Automake (John Williams)
ayacc	Ayacc (Mathieu Clabaut)
b	B (Mathieu Clabaut)
bindzone	BIND zone (glory hump)
blank	Blank (Rafal Sulejman)
cfg	Configure files (Igor Prischepoff)
changelog	ChangeLog (Gediminas Paulauskas)
cl	Clever (Phil Uren)
crontab	Crontab (John Hoelzel)
csc	Essbase script (Raul Segura Acevedo)
cynlib	Cynlib(C++) (Phil Derrick)
cynpp	Cyn++ (Phil Derrick)
debchangelog	Debian Changelog (Wichert Akkerman)
debcontrol	Debian Control (Wichert Akkerman)
dns	DNS zone file (Jehsom)
dtml	Zope's DTML (Jean Jordaan)
dylan	Dylan, Dylan-intr and Dylan-lid (Brent Fulgham)
ecd	Embedix Component Description (John Beppu)

fgl	Informix 4GL (Rafal Sulejman)
foxpro	FoxPro (Powing Tse)
gsp	GNU Server Pages (Nathaniel Harward)
gtkrc	GTK rc (David Necas)
hercules	Hercules (Avant! Corporation) (Dana Edwards)
htmls	HTML/OS by Aestiva (Jason Rust)
inittab	SysV process control (David Necas)
iss	Inno Setup (Dominique Stephan)
jam	Jam (Ralf Lemke)
jess	Jess (Paul Baleme)
lprolog	LambdaProlog (Markus Mottl)
ia64	Intel Itanium (parth malwankar)
kix	Kixtart (Nigel Gibbs)
mgp	MaGic Point (Gerfried Fuchs)
mason	Mason (HTML with Perl) (Andrew Smith)
mma	Mathematica (Wolfgang Waltenberger)
nqc	Not Quite C (Stefan Scherer)
omnimark	Omnimark (Paul Terray)
openroad	OpenROAD (Luis Moreno Serrano)
named	BIND configuration (glory hump)
papp	PApp (Marc Lehmann)
pfmain	Postfix main config (Peter Kelemen)
pic	PIC assembly (Aleksandar Veselinovic)
ppwiz	PPWizard (Stefan Schwarzer)
progress	Progress (Phil Uren)
psf	Product Specification File (Rex Barzee)
r	R (Tom Payne)
registry	MS-Windows registry (Dominique Stephan)
robots	Robots.txt (Dominique Stephan)
rtf	Rich Text Format (Dominique Stephan)
setl	SETL (Alex Poylisher)
sgmldecl	SGML Declarations (Daniel A. Molina W.)
sinda	Sinda input (Adrian Nagle)
sindacmp	Sinda compare (Adrian Nagle)
sindaout	Sinda output (Adrian Nagle)
smith	SMITH (Rafal Sulejman)
snobol4	Snobol 4 (Rafal Sulejman)
strace	Strace (David Necas)
tak	TAK input (Adrian Nagle)
takcmp	TAK compare (Adrian Nagle)
takout	TAK output (Adrian Nagle)
tasm	Turbo assembly (FooLman)
texmf	TeX configuration (David Necas)
trasys	Trasys input (Adrian Nagle)
tssgm	TSS Geometry (Adrian Nagle)
tssop	TSS Optics (Adrian Nagle)
tsscl	TSS Command line (Adrian Nagle)
virata	Virata Configuration Script (Manuel M.H. Stol)
vsejcl	VSE JCL (David Ondrejko)
wdiff	Wordwise diff (Gerfried Fuchs)
wsh	Windows Scripting Host (Paul Moore)
xkb	X Keyboard Extension (David Necas)

Renamed php3 to php, it now also supports php4 (Lutz Eymers)

Patch 5.7.015

Problem: Syntax files for Vim 6.0 can't be used with 5.x.

Solution: Add the "default" argument to the ":highlight" command: Ignore the command if highlighting was already specified.

Files: src/syntax.c

Generate the Syntax menu with makemenu.vim, so that it doesn't have to be done when Vim is starting up. Reduces the startup time of the GUI.

Fixed

fixed-5.8

Conversion of docs to HTML didn't convert " tag s" to a hyperlink.

Fixed compiling under NeXT. (Jeroen C.M. Goudswaard)

optwin.vim gave an error when used in Vi compatible mode ('cpo' contains 'C').

Tcl interpreter: "buffer" command didn't check for presence of an argument.
(Dave Bodenstab)

dosinst.c: Added checks for too long file name.

Amiga: a file name starting with a colon was considered absolute but it isn't.
Amiga: ":pwd" added a slash when in the root of a drive.

Macintosh: Warnings for unused variables. (Bernhard Pruemmer)

Unix: When catching a deadly signal, handle it in such a way that it's unlikely that Vim will hang. Call _exit() instead of exit() in case of a severe problem.

Setting the window title from nothing to something didn't work after patch 29.

Check for ownership of .exrc and .vimrc was done with stat(). Use lstat() as well for extra security.

Win32 GUI: Printing a file with 'fileformat' "unix" didn't work. Set 'fileformat' to "dos" before writing the temp file.

Unix: Could start waiting for a character when checking for a CTRL-C typed when an X event is received.

Could not use Perl and Python at the same time on FreeBSD, because Perl used "-lc" and Python used the threaded C library.

Win32: The Mingw compiler gave a few warning messages.

When using "ZZ" and an autocommand for writing uses an abbreviation it didn't work. Don't stuff the ":x" command but execute it directly. (Mikael Berthe)

VMS doesn't always have lstat(), added an #ifdef around it.

Added a few corrections for the Macintosh. (Axel Kielhorn)

Win32: GvimExt could not edit more than a few files at once, the length of the argument was fixed.

Previously released patches for Vim 5.7:

Patch 5.7.001

Problem: When the current buffer is encrypted, and another modified buffer isn't, ":wall" will encrypt the other buffer.

Solution: In buf_write() use "buf" instead of "curbuf" to check for the crypt key.

Files: src/fileio.c

Patch 5.7.002

Problem: When '**showmode**' is set, using "**CTRL-O** :r file" waits three seconds before displaying the read text. (Wichert Akkerman)

Solution: Set "keep_msg" to the file message so that the screen is redrawn before the three seconds wait for displaying the mode message.

Files: src/fileio.c

Patch 5.7.003

Problem: Searching for "[[:cntrl:]]" doesn't work.

Solution: Exclude NUL from the matching characters, it terminates the list.

Files: src/regexp.c

Patch 5.7.004

Problem: GTK: When selecting a new font, Vim can crash.

Solution: In gui_mch_init_font() unreference the old font, not the new one.

Files: src/gui_gtk_x11.c

Patch 5.7.005

Problem: Multibyte: Inserting a wrapped line corrupts kterm screen. Pasting TEXT/COMPOUND_TEXT into Vim does not work. On Motif no XIM status line is displayed even though it is available.

Solution: Don't use xterm trick for wrapping lines for multibyte mode. Correct a missing "break", added TEXT/COMPOUND_TEXT selection request.

Add XIMStatusArea fallback code.

(Katsuhito Nagano)

Files: src/gui_gtk_x11.c, src/multbyte.c, src/screen.c, src/ui.c

Patch 5.7.006

Problem: GUI: redrawing the non-Visual selection is wrong when the window is unobscured. (Jean-Pierre Etienne)

Solution: Redraw the selection properly and don't clear it. Added "len" argument to clip_may_redraw_selection().

Files: src/gui.c, src/ui.c, src/proto/ui.pro

Patch 5.7.007

Problem: Python: Crash when using the current buffer twice.

Solution: Increase the reference count for buffer and window objects.
(Johannes Zellner)
Files: src/if_python.c

Patch 5.7.008

Problem: In Ex mode, backspacing over the first TAB doesn't work properly.
(Wichert Akkerman)
Solution: Switch the cursor on before printing the newline.
Files: src/ex_getln.c

Patch 5.7.009 (extra)

Problem: Mac: Crash when using a long file.
Solution: Don't redefine malloc() and free(), because it will break using
realloc().
Files: src/os_mac.h

Patch 5.7.010

Problem: When using **CTRL-A** on a very long number Vim can crash. (Michael
Naumann)
Solution: Truncate the length of the new number to avoid a buffer overflow.
Files: src/ops.c

Patch 5.7.011 (extra)

Problem: Win32 GUI on NT 5 and Win98: Displaying Hebrew is reversed.
Solution: Output each character separately, to avoid that Windows reverses
the text for some fonts. (Ron Aaron)
Files: src/gui_w32.c

Patch 5.7.012

Problem: When using "-complete=buffer" for ":command" the user command
fails.
Solution: In a user command don't replace the buffer name with a count for
the buffer number.
Files: src/ex_docmd.c

Patch 5.7.013

Problem: "gD" didn't always find a match in the first line, depending on
the column the search started at.
Solution: Reset the column to zero before starting to search.
Files: src/normal.c

Patch 5.7.014

Problem: Rot13 encoding was done on characters with accents, which is
wrong. (Sven Gottwald)
Solution: Only do rot13 encoding on ASCII characters.
Files: src/ops.c

Patch 5.7.016

Problem: When hitting 'n' for a ":s///c" command, the ignore-case flag was
not restored, some matches were skipped. (Daniel Blaustein)
Solution: Restore the reg_ic variable when 'n' was hit.
Files: src/ex_cmds.c

Patch 5.7.017

Problem: When using a Vim script for Vim 6.0 with <SID> before a function name, it produces an error message even when inside an "if version >= 600". (Charles Campbell)

Solution: Ignore errors in the function name when the function is not going to be defined.

Files: src/eval.c

Patch 5.7.018

Problem: When running "rvim" or "vim -Z" it was still possible to execute a shell command with system() and backtick-expansion. (Antonios A. Kavarnos)

Solution: Disallow executing a shell command in get_cmd_output() and mch_expand_wildcards().

Files: src/misc1.c, src/os_unix.c

Patch 5.7.019

Problem: Multibyte: In a substitute string, a multibyte character isn't skipped properly, can be a problem when the second byte is a backslash.

Solution: Skip an extra byte for a double-byte character. (Muraoka Taro)

Files: src/ex_cmds.c

Patch 5.7.020

Problem: Compilation doesn't work on MacOS-X.

Solution: Add a couple of #ifdefs. (Jamie Curmi)

Files: src/regexp.c, src/ctags/general.h

Patch 5.7.021

Problem: Vim sometimes produces a beep when started in an xterm. Only happens when compiled without mouse support.

Solution: Requesting the xterm version results in a K_IGNORE. This wasn't handled when mouse support is disabled. Accept K_IGNORE always.

Files: src/normal.c

Patch 5.7.022

Problem: %v in '**statusline**' is not displayed when it's equal to %c.

Solution: Check if %V or %v is used and handle them differently.

Files: src/screen.c

Patch 5.7.023

Problem: Crash when a WinLeave autocommand deletes the buffer in the other window.

Solution: Check that after executing the WinLeave autocommands there still is a window to be closed. Also update the test that was supposed to check for this problem.

Files: src/window.c, testdir/test13.in, testdir/test13.ok

Patch 5.7.024

Problem: Evaluating an expression for '**statusline**' can have side effects.

Solution: Evaluate the expression in a sandbox.

Files: src/edit.c, src/eval.c, src/proto/eval.pro, src/ex_cmds.c, src/ex_cmds.h, src/ex_docmd.c, src/globals.h, src/option.c, src/screen.c, src/undo.c

Patch 5.7.025 (fixed)

Problem: Creating a temp file has a race condition.
Solution: Create a private directory to write the temp files in.
Files: src/fileio.c, src/misc1.c, src/proto/misc1.pro,
src/proto/fileio.pro, src/memline.c, src/os_unix.h

Patch 5.7.026 (extra)

Problem: Creating a temp file has a race condition.
Solution: Create a private directory to write the temp files in.
This is the extra part of patch 5.7.025.
Files: src/os_msdos.h

Patch 5.7.027

Problem: Starting to edit a file can cause a crash. For example when in
Insert mode, using **CTRL-O** :help abbr<Tab> to scroll the screen and
then <CR>, which edits a help file. (Robert Bogomip)
Solution: Check if keep_msg is NULL before copying it.
Files: src/fileio.c

Patch 5.7.028

Problem: Creating a backup or swap file could fail in rare situations.
Solution: Use O_EXCL for open().
Files: src/fileio.c, src/memfile.c

Patch 5.7.029

Problem: Editing a file with an extremely long name crashed Vim.
Solution: Check for length of the name when setting the window title.
Files: src/buffer.c

Patch 5.7.030

Problem: A ":make" or ":grep" command with a very long argument could cause
a crash.
Solution: Allocate the buffer for the shell command.
Files: src/ex_docmd.c

Welcome to Vim Version 6.0! A large number of features has been added. This file mentions all the new items that have been added, changes to existing features and bug fixes compared to Vim 5.x.

See [vi_diff.txt](#) for an overview of differences between Vi and Vim 6.0.

See [version4.txt](#) for differences between Vim 3.0 and Vim 4.0.

See [version5.txt](#) for differences between Vim 4.0 and Vim 5.0.

INCOMPATIBLE CHANGES

incompatible-6

Cursor position in Visual mode
substitute command Vi compatible
global option values introduced
'fileencoding' changed
Digraphs changed
Filetype detection changed
Unlisted buffers introduced
CTRL-U in Command-line mode changed
Ctags gone
Documentation reorganized
Modeless selection and clipboard
Small incompatibilities

curpos-visual
substitute-CR
new-global-values
fileencoding-changed
digraphs-changed
filetypedetect-changed
new-unlisted-buffers
CTRL-U-changed
ctags-gone
documentation-6
modeless-and-clipboard
incomp-small-6

NEW FEATURES

new-6

Folding
Vertically split windows
Diff mode
Easy Vim: click-and-type
User manual
Flexible indenting
Extended search patterns
UTF-8 support
Multi-language support
Plugin support
Filetype plugins
File browser
Editing files over a network
Window for command-line editing
Debugging mode
Cursor in virtual position
Debugger interface
Communication between Vims
Buffer type options
Printing
Ports
Quickfix extended
Operator modifiers

new-folding
new-vertspl
new-diff-mode
new-evim
new-user-manual
new-indent-flex
new-searchpat
new-utf-8
new-multi-lang
new-plugins
new-filetype-plugins
new-file-browser
new-network-files
new-cmdwin
new-debug-mode
new-virtedit
new-debug-itf
new-vim-server
new-buftype
new-printing
ports-6
quickfix-6
new-operator-mod

Search Path	new-search-path
Writing files improved	new-file-writing
Argument list	new-argument-list
Restore a View	new-View
Color schemes	new-color-schemes
Various new items	new-items-6

IMPROVEMENTS	improvements-6
--------------	----------------

COMPILE TIME CHANGES	compile-changes-6
----------------------	-------------------

BUG FIXES	bug-fixes-6
-----------	-------------

VERSION 6.1	version-6.1
Changed	changed-6.1
Added	added-6.1
Fixed	fixed-6.1

VERSION 6.2	version-6.2
Changed	changed-6.2
Added	added-6.2
Fixed	fixed-6.2

VERSION 6.3	version-6.3
Changed	changed-6.3
Added	added-6.3
Fixed	fixed-6.3

VERSION 6.4	version-6.4
Changed	changed-6.4
Added	added-6.4
Fixed	fixed-6.4

INCOMPATIBLE CHANGES incompatible-6

These changes are incompatible with previous releases. Check this list if you run into a problem when upgrading from Vim 5.x to 6.0

Cursor position in Visual mode curpos-visual

When going from one window to another window on the same buffer while in Visual mode, the cursor position of the other window is adjusted to keep the same Visual area. This can be used to set the start of the Visual area in one window and the end in another. In vim 5.x the cursor position of the other window would be used, which could be anywhere and was not very useful.

Substitute command Vi compatible substitute-CR

The substitute string (the "to" part of the substitute command) has been made

Vi compatible. Previously a **CTRL-V** had a special meaning and could be used to prevent a `<CR>` to insert a line break. This made it impossible to insert a **CTRL-V** before a line break. Now a backslash is used to prevent a `<CR>` to cause a line break. Since the number of backslashes is halved, it is still possible to insert a line break at the end of the line. This now works just like Vi, but it's not compatible with Vim versions before 6.0.

When a `":s"` command doesn't make any substitutions, it no longer sets the '[' and ']' marks. This is not related to Vi, since it doesn't have these marks.

Global option values introduced

new-global-values

There are now global values for options which are local to a buffer or window. Previously the local options were copied from one buffer to another. When editing another file this could cause option values from a modeline to be used for the wrong file. Now the global values are used when entering a buffer that has not been used before. Also, when editing another buffer in a window, the local window options are reset to their global values. The `":set"` command sets both the local and global values, this is still compatible. But a modeline only sets the local value, this is not backwards compatible.

`":let &opt = val"` now sets the local and global values, like `":set"`. New commands have been added to set the global or local value:

```
:let &opt = val      like ":set"
:let &g:opt = val    like ":setglobal"
:let &l:opt = val    like ":setlocal"
```

'fileencoding' changed

fileencoding-changed

'fileencoding' was used in Vim 5.x to set the encoding used inside all of Vim. This was a bit strange, because it was local to a buffer and worked for all buffers. It could never be different between buffers, because it changed the way text in all buffers was interpreted.

It is now used for the encoding of the file related to the buffer. If you still set 'fileencoding' it is likely to be overwritten by the detected encoding from 'fileencodings', thus it is "mostly harmless".

The old FileEncoding autocommand now does the same as the new EncodingChanged event.

Digraphs changed

digraphs-changed

The default digraphs now correspond to RFC1345. This is very different from what was used in Vim 5.x. [digraphs](#)

Filetype detection changed

filetypedetect-changed

The filetype detection previously was using the "filetype" autocommand group. This caused confusion with the FileType event name (case is ignored). The group is now called "filetypedetect". It still works, but if the "filetype" group is used the autocommands will not be removed by ":filetype off".

The support for '**runtimepath**' has made the "myfiletypefile" and "mysyntaxfile" mechanism obsolete. They are still used for backwards compatibility.

The connection between the FileType event and setting the '**syntax**' option was previously in the "syntax" autocommand group. That caused confusion with the Syntax event name. The group is now called "syntaxset".

The distributed syntax files no longer contain "syntax clear". That makes it possible to include one in the other without tricks. The syntax is now cleared when the '**syntax**' option is set (by an autocommand added from synload.vim). This makes the syntax cleared when the value of '**syntax**' does not correspond to a syntax file. Previously the existing highlighting was kept.

Unlisted buffers introduced

new-unlisted-buffers

There is now a difference between buffers which don't appear in the buffer list and buffers which are really not in the buffer list. Commands like ":ls", ":bnext", ":blast" and the Buffers menu will skip buffers not in the buffer list. **unlisted-buffer**

The '**buflisted**' option can be used to make a buffer appear in the buffer list or not.

Several commands that previously added a buffer to the buffer list now create an unlisted buffer. This means that a ":bnext" and ":ball" will not find these files until they have actually been edited. For example, buffers used for the alternative file by ":write file" and ":read file".

Other commands previously completely deleted a buffer and now only remove the buffer from the buffer list. Commands relying on a buffer not to be present might fail. For example, a ":bdelete" command in an autocommand that relied on something following to fail (was used in the automatic tests).

:bwipeout can be used for the old meaning of ":bdelete".

The BufDelete autocommand event is now triggered when a buffer is removed from the buffer list. The BufCreate event is only triggered when a buffer is created that is added to the buffer list, or when an existing buffer is added to the buffer list. BufAdd is a new name for BufCreate.

The new BufNew event is for creating any buffer and BufWipeout for really deleting a buffer.

When doing Insert mode completion, only buffers in the buffer list are scanned. Added the 'U' flag to '**complete**' to do completion from unlisted buffers.

Unlisted buffers are not stored in a viminfo file.

CTRL-U in Command-line mode changed

CTRL-U-changed

Using **CTRL-U** when editing the command line cleared the whole line. Most shells only delete the characters before the cursor. Made it work like that. (Steve Wall)

You can get the old behavior with **CTRL-E CTRL-U**:
`:cnoremap <C-U> <C-E><C-U>`

Ctags gone

ctags-gone

Ctags is no longer part of the Vim distribution. It's now a grown-up program by itself, it deserves to be distributed separately. Ctags can be found here: <http://ctags.sf.net/>.

Documentation reorganized

documentation-6

The documentation has been reorganized, an item may not be where you found it in Vim 5.x.

- The user manual was added, some items have been moved to it from the reference manual.
- The quick reference is now in a separate file (so that it can be printed).

The examples in the documentation were previously marked with a ">" in the first column. This made it difficult to copy/paste them. There is now a single ">" before the example and it ends at a "<" or a non-blank in the first column. This also looks better without highlighting.

'**helpfile**' is no longer used to find the help tags file. This allows a user to add its own help files (e.g., for plugins).

Modeless selection and clipboard

modeless-and-clipboard

The modeless selection is used to select text when Visual mode can't be used, for example when editing the command line or at the more prompt.

In Vim 5.x the modeless selection was always used. On MS-Windows this caused the clipboard to be overwritten, with no way to avoid that. The modeless selection now obeys the 'a' and 'A' flags in '**guioptions**' and "autoselect" and "autoselectml" in '**clipboard**'. By default there is no automatic copy on MS-Windows. Use the `c_CTRL-Y` command to manually copy the selection.

To get the old behavior back, do this:

```
:set clipboard^=autoselectml guioptions+=A
```

Small incompatibilities

incomp-small-6

'**backupdir**', '**cdpath**', '**directory**', '**equalprg**', '**errorfile**', '**formatprg**', '**grepprg**', '**helpfile**', '**makeef**', '**makeprg**', '**keywordprg**', '**cscopeprg**', '**viminfo**' and '**runtimepath**' can no longer be set from a modeline, for better security.

Removed '_' from the '**breakat**' default: It's commonly used in keywords.

The default for '**mousehide**' is on, because this works well for most people.

The Amiga binary is now always compiled with "big" features. The "big" binary archive no longer exists.

The items "[RO]", "[+]", "[help]", "[Preview]" and "[filetype]" in '**statusline**' no longer have a leading space.

Non-Unix systems: When expanding wildcards for the Vim arguments, don't use '**suffixes**'. It now works as if the shell had expanded the arguments.

The '**lisp**', '**smartindent**' and '**cindent**' options are not switched off when '**paste**' is set. The auto-indenting is disabled when '**paste**' is set, but manual indenting with "=" still works.

When formatting with "=" uses '**cindent**' or '**indentexpr**' indenting, and there is no change in indent, this is not counted as a change ('modified' isn't set and there is nothing to undo).

Report '**modified**' as changed when '**fileencoding**' or '**fileformat**' was set. Thus it reflects the possibility to abandon the buffer without losing changes.

The "Save As" menu entry now edits the saved file. Most people expect it to work like this.

A buffer for a directory is no longer added to the Buffers menu.

Renamed <Return> to <Enter>, since that's what it's called on most keyboards. Thus it's now the hit-enter prompt instead of the hit-return prompt. Can map <Enter> just like <CR> or <Return>.

The default for the '**viminfo**' option is now '20,"50,h when '**compatible**' isn't set. Most people will want to use it, including beginners, but it required setting the option, which isn't that easy.

After using ":colder" the newer error lists are overwritten. This makes it possible to use ":grep" to browse in a tree-like way. Must use ":cnewer 99" to get the old behavior.

The patterns in '**errorformat**' would sometimes ignore case (MS-Windows) and sometimes not (Unix). Now case is always ignored. Add "\C" to the pattern to match case.

The 16 bit MS-DOS version is now compiled without the +listcmds feature (buffer list manipulation commands). They are not often needed and this

executable needs to be smaller.

'**sessionoptions**' now includes "curdir" by default. This means that restoring a session will result in the current directory being restored, instead of going to the directory where the session file is located.

A session deleted all buffers, deleting all marks. Now keep the buffer list, it shouldn't hurt for some existing buffers to remain present.
When the argument list is empty ":argdel *" caused an error message.

No longer put the search pattern from a tag jump in the history.

Use "SpecialKey" highlighting for unprintable characters instead of "NonText". The idea is that unprintable text or any text that's displayed differently from the characters in the file is using "SpecialKey", and "NonText" is used for text that doesn't really exist in the file.

Motif now uses the system default colors for the menu and scrollbar. Used to be grey. It's still possible to set the colors with ":highlight" commands and resources.

Formatting text with "gq" breaks a paragraph at a non-empty blank line. Previously the line would be removed, which wasn't very useful.

":normal" does no longer hang when the argument ends in half a command. Previously Vim would wait for more characters to be typed, without updating the screen. Now it pretends an <Esc> was typed.

Bitmaps for the toolbar are no longer searched for in "\$VIM/bitmaps" but in the "bitmaps" directories in '**runtimepath**'.

Now use the Cmdline-mode menus for the hit-enter prompt instead of the Normal mode menus. This generally works better and allows using the "Copy" menu to produce **CTRL-Y** to copy the modeless selection.

Moved the font selection from the Window to the Edit menu, together with the other settings.

The default values for '**isfname**' include more characters to make "gf" work better.

Changed the license for the documentation to the Open Publication License. This seemed fair, considering the inclusion of parts of the Vim book, which is also published under the OPL. The downside is that we can't force someone who would sell copies of the manual to contribute to Uganda.

After "ayy don't let ""yy or :let @" = val overwrite the "a register. Use the unnamed register instead.

MSDOS: A pattern "*.*)" previously also matched a file name without a dot. This was inconsistent with other versions.

In Insert mode, **CTRL-O CTRL-\ CTRL-N {cmd}** remains in Normal mode. Previously it would go back to Insert mode, thus confusing the meaning of **CTRL-\ CTRL-N**,

which is supposed to take us to Normal mode (especially in ":amenu").

Allow using ":" commands after an operator. Could be used to implement a new movement command. Thus it no longer aborts a pending operator.

For the Amiga the "-d {device}" argument was possible. When compiled with the diff feature, this no longer works. Use "-dev {device}" instead. [-dev](#)

Made the default mappings for <S-Insert> in Insert mode insert the text literally, avoids that special characters like BS cause side effects.

Using ":confirm" applied to the rest of the line. Now it applies only to the command right after it. Thus ":confirm if x | edit | endif" no longer works, use ":if x | confirm edit | endif". This was the original intention, that it worked differently was a bug.

NEW FEATURES

[new-6](#)

Folding

[new-folding](#)

Vim can now display a buffer with text folded. This allows overviewing the structure of a file quickly. It is also possible to yank, delete and put folded text, for example to move a function to another position.

There is a whole bunch of new commands and options related to folding. See [folding](#) .

Vertically split windows

[new-vertspl](#)

Windows can also be split vertically. This makes it possible to have windows side by side. One nice use for this is to compare two similar files (see [new-diff-mode](#)). The '[scrollbind](#)' option can be used to synchronize scrolling.

A vertical split can be created with the commands:

```
:vsplit or CTRL-W v or CTRL-W CTRL-V    :vsplit
:vnew                                     :vnew
:vertical {cmd}                           :vertical
```

The last one is a modifier, which has a meaning for any command that splits a window. For example:

```
:vertical stag main
```

Will vertically split the window and jump to the tag "main" in the new window.

Moving from window to window horizontally can be done with the [CTRL-W_h](#) and [CTRL-W_l](#) commands. The [CTRL-W_k](#) and [CTRL-W_j](#) commands have been changed to jump to the window above or below the cursor position.

The vertical and horizontal splits can be mixed as you like. Resizing windows is easy when using the mouse, just position the pointer on a status line or vertical separator and drag it. In the GUI a special mouse pointer shape

indicates where you can drag a status or separator line.

To resize vertically split windows use the `CTRL-W_<` and `CTRL-W_>` commands. To make a window the maximum width use the `CTRL-W |` command `CTRL-W_bar` .

To force a new window to use the full width or height of the Vim window, these two modifiers are available:

```
:topleft {cmd}      New window appears at the top with full
                    width or at the left with full height.
:botright {cmd}     New window appears at the bottom with full
                    width or at the right with full height.
```

This can be combined with `":vertical"` to force a vertical split:

```
:vert bot dsplit DEBUG
```

This will open a window at the far right, occupying the full height of the Vim window, with the cursor on the first definition of "DEBUG".

The help window is opened at the top, like `":topleft"` was used, if the current window is fewer than 80 characters wide.

A few options can be used to set the preferences for vertically split windows. They work similarly to their existing horizontal equivalents:

```
horizontal          vertical
'splitbelow'        'splitright'
'winheight'         'winwidth'
'winminheight'     'winminwidth'
```

It's possible to set `'winminwidth'` to zero, so that temporarily unused windows hardly take up space without closing them.

The new `'eadirection'` option tells where `'equalalways'` applies:

```
:set eadirection=both      both directions
:set eadirection=ver       equalize window heights
:set eadirection=hor       equalize windows widths
```

This can be used to avoid changing window sizes when you want to keep them.

Since windows can become quite narrow with vertical splits, text lines will often not fit. The `'sidescrolloff'` has been added to keep some context left and right of the cursor. The `'listchars'` option has been extended with the "precedes" item, to show a "<" for example, when there is text left off the screen. (Utz-Uwe Haus)

`"-0"` command line argument: Like `"-o"` but split windows vertically. (Scott Urban)

Added commands to move the current window to the very top (`CTRL-W K`), bottom (`CTRL-W J`), left (`CTRL-W H`) and right (`CTRL-W L`). In the new position the window uses the full width/height of the screen.

When there is not enough room in the status line for both the file name and the ruler, use up to half the width for the ruler. Useful for narrow windows.

Diff mode

`new-diff-mode`

In diff mode Vim shows the differences between two, three or four files.

Folding is used to hide the parts of the file that are equal.
Highlighting is used to show deleted and changed lines.
See [diff-mode](#) .

An easy way to start in diff mode is to start Vim as "vimdiff file1 file2".
Added the vimdiff manpage.

In a running Vim the `:diffsplit` command starts diff mode for the current file and another file. The `:diffpatch` command starts diff mode using the current file and a patch file. The `:diffthis` command starts diff mode for the current window.

Differences can be removed with the `:diffget` and `:diffput` commands.

- The `'diff'` option switches diff mode on in a window.
- The `:diffupdate` command refreshes the diffs.
- The `'diffopt'` option changes how diffs are displayed.
- The `'diffexpr'` option can be set how a diff is to be created.
- The `'patchexpr'` option can be set how patch is applied to a file.
- Added the "diff" folding method. When opening a window for diff-mode, set `'foldlevel'` to zero and `'foldenable'` on, to close the folds.
- Added the DiffAdd, DiffChange, DiffDelete and DiffText highlight groups to specify the highlighting for differences. The defaults are ugly...
- Unix: make a vimdiff symbolic link for "make install".
- Removed the now obsolete "vimdiff.vim" script from the distribution.
- Added the "[c" and "]c" commands to move to the next/previous change in diff mode.

Easy Vim: click-and-type

[new-evim](#)

eVim stands for "Easy Vim". This is a separate program, but can also be started as "vim -y".

This starts Vim with `'insertmode'` set to allow click-and-type editing. The `$(VIMRUNTIME)/evim.vim` script is used to add mappings and set options to be able to do most things like Notepad. This is only for people who can't stand two modes.

eView does the same but in readonly mode.

In the GUI a **CTRL-C** now only interrupts when busy with something, not when waiting for a character. Allows using **CTRL-C** to copy text to the clipboard.

User manual

[new-user-manual](#)

The user manual has been added. It is organised around editing tasks. It reads like a book, from start to end. It should allow beginners to start learning Vim. It helps everybody to learn using the most useful Vim features. It is much easier to read than the reference manual, but omits details. See [user-manual](#) .

The user manual includes parts of the Vim book by Steve Oualline [frombook](#) . It is published under the OPL [manual-copyright](#) .

When syntax highlighting is not enabled, the characters in the help file which mark examples ('>' and '<') and header lines ('~') are replaced with a space.

When closing the help window, the window layout is restored from before opening it, if the window layout didn't change since then.

When opening the help window, put it at the top of the Vim window if the current window is fewer than 80 characters and not full width.

Flexible indenting

[new-indent-flex](#)

Automatic indenting is now possible for any language. It works with a Vim script, which makes it very flexible to compute the indent.

The `":filetype indent on"` command enables using the provided indent scripts. This is explained in the user manual: [30.3](#) .

The `'indentexpr'` option is evaluated to get the indent for a line. The `'indentkeys'` option tells when to trigger re-indenting. Normally these options are set from an indent script. Like Syntax files, indent scripts will be created and maintained by many people.

Extended search patterns

[new-searchpat](#)

Added the possibility to match more than one line with a pattern. (partly by Loic Grenie)

New items in a search pattern for multi-line matches:

<code>\n</code>	match end-of-line, also in []
<code>_[]</code>	match characters in range and end-of-line
<code>_x</code>	match character class and end-of-line
<code>_.</code>	match any character or end-of-line
<code>_^</code>	match start-of-line, can be used anywhere in the regexp
<code>_\$\$</code>	match end-of-line, can be used anywhere in the regexp

Various other new items in search patterns:

<code>\c</code>	ignore case for the whole pattern
<code>\C</code>	match case for the whole pattern
<code>\m</code>	magic on for the following
<code>\M</code>	magic off for the following
<code>\v</code>	make following characters "very magic"
<code>\V</code>	make following characters "very nomagic"

<code>\@!</code>	don't match atom before this. Example: <code>"foo\(\bar\) \@!"</code> matches <code>"foo "</code> but not <code>"foobar"</code> .
<code>\@=</code>	match atom, resulting in zero-width match Example: <code>"foo\(\bar\) \@="</code> matches <code>"foo"</code> in <code>"foobar"</code> .
<code>\@<!</code>	don't match preceding atom before the current position

<code>\@<=</code>	match preceding atom before the current position
<code>\@></code>	match preceding atom as a subexpression
<code>\&</code>	match only when branch before and after it match
<code>\%[]</code>	optionally match a list of atoms; "end\%[if]" matches "end", "endi" and "endif"
<code>\%(\)</code>	like <code>\(\)</code> , but without creating a back-reference; there can be any number of these, overcomes the limit of nine <code>\(\)</code> pairs
<code>\%^</code>	match start-of-file (Chase Tingley)
<code>\%\$</code>	match end-of-file (Chase Tingley)
<code>\%#</code>	Match with the cursor position. (Chase Tingley)
<code>\?</code>	Just like <code>\=</code> but can't be used in a <code>"?"</code> command.
<code>\%23l</code>	match in line 23
<code>\%<23l</code>	match before line 23
<code>\%>23l</code>	match after line 23
<code>\%23c, \%<23c, \%>23c</code>	match in/before/after column 23
<code>\%23v, \%<23v, \%>23v</code>	match in/before/after virtual column 23

For syntax items:

<code>\z(...\)</code>	external reference match set (in region start pattern)
<code>\z1 - \z9</code>	external reference match use (in region skip or end pattern)
	(Scott Bigham)

<code>\zs</code>	use position as start of match
<code>\ze</code>	use position as end of match

Removed limit of matching only up to 32767 times with `*`, `\+`, etc.

Added support to match multibyte characters. (partly by Muraoka Taro)
 Made `"\<"` and `"\>"` work for UTF-8. (Muraoka Taro)

UTF-8 support

[new-utf-8](#)

Vim can now edit files in UTF-8 encoding. Up to 31 bit characters can be used, but only 16 bit characters are displayed. Up to two combining characters are supported, they overprint the preceding character. Double-wide characters are also supported. See [UTF-8](#).

UCS-2, UCS-4 and UTF-16 encodings are supported too, they are converted to UTF-8 internally. There is also support for editing Unicode files in a Latin1 environment. Other encodings are converted with `iconv()` or an external converter specified with `'charconvert'`.

Many new items for Multi-byte support:

- Added `'encoding'` option: specifies character encoding used inside Vim. It can be any 8-bit encoding, some double-byte encodings or Unicode. It is initialized from the environment when a supported value is found.
- Added `'fileencoding'` and `'fileencodings'`: specify character coding in a file, similar to `'fileformat'` and `'fileformats'`.

- When **'encoding'** is "utf-8" and **'fileencodings'** is "utf-8,latin1" this will automatically switch to latin1 if a file does not contain valid UTF-8.
- Added **'bomb'** option and detection of a BOM at the start of a file. Can be used with "ucs-bom" in **'fileencodings'** to automatically detect a Unicode file if it starts with a BOM. Especially useful on MS-Windows (NT and 2000), which uses ucs-2le files with a BOM (e.g., when exporting the registry).
 - Added the **'termencoding'** option: Specifies the encoding used for the terminal. Useful to put Vim in utf-8 mode while in a non-Unicode locale:


```
:let &termencoding = &encoding
:set encoding=utf-8
```
 - When **'viminfo'** contains the 'c' flag, the viminfo file is converted from the **'encoding'** it was written with to the current **'encoding'**.
 - Added **":scriptencoding"** command: convert lines in a sourced script to **'encoding'**. Useful for menu files.
 - Added **'guifontwide'** to specify a font for double-wide characters.
 - Added Korean support for character class detection. Also fix cls() in search.c. (Chong-Dae Park)
 - Win32: Typing multibyte characters without IME. (Alexander Smishlajev)
 - Win32 with Mingw: compile with iconv library. (Ron Aaron)
 - Win32 with MSVC: dynamically load iconv.dll library. (Muraoka Taro)
 - Make it possible to build a version with multibyte and iconv support with Borland 5.5. (Yasuhiro Matsumoto)
 - Added **'delcombine'** option: Delete combining character separately. (Ron Aaron)
 - The "xfontset" feature isn't required for "xim". These are now two independent features.
 - XIM: enable XIM when typing a language character (Insert mode, Search pattern, "f" or "r" command). Disable XIM when typing a Normal mode command.
 - When the XIM is active, show "XIM" in the **'showmode'** message. (Nam SungHyun)
 - Support "CursorIM" for XIM. (Nam SungHyun)
 - Added 'm' flag to **'formatoptions'**: When wrapping words, allow splitting at each multibyte character, not only at a space.
 - Made **":syntax keyword"** work with multibyte characters.
 - Added support for Unicode upper/lowercase flipping and comparing. (based on patch by Raphael Finkel)

Let "~" on multibyte characters that have a third case ("title case") switch between the three cases. (Raphael Finkel)

Allow defining digraphs for multibyte characters.

Added RFC1345 digraphs for Unicode.

Most Normal mode commands that accept a character argument, like "r", "t" and "f" now accept a digraph. The 'D' flag in **'cptions'** disables this to remain Vi compatible.

Added Language mapping and **'keymap'** to be able to type multibyte characters:

- Added the **":lmap"** command and friends: Define mappings that are used when typing characters in the language of the text. Also for "r", "t", etc. In Insert and Command-line mode **CTRL-^** switches the use of the mappings on/off. **CTRL-^** also toggles the use of an input method when no language mappings are present. Allows switching the IM back on halfway typing.
- **"<char-123>"** argument to **":map"**, allows to specify the decimal, octal or hexadecimal value of a character.

- Implemented the **'keymap'** option: Load a keymap file. Uses `":lnoremap"` to define mappings for the keymap. The new `":loadkeymap"` command is used in the keymap file.
- Added `'k'` flag in **'statusline'**: Value of `"b:keymap_name"` or **'keymap'** when it's being used. Uses `"<lang>"` when no keymap is loaded and `":lmap"`s are active. Show this text in the default statusline too.
- Added the **'iminsert'** and **'imsearch'** options: Specify use of langmap mappings and Input Method with an option. (Muraoka Taro)
Added **'imcmdline'** option: When set the input method is always enabled when starting to edit a command line. Useful for a XIM that uses dead keys to type accented characters.
Added **'imactivatekey'** option to better control XIM. (Muraoka Taro)
- When typing a mapping that's not finished yet, display the last character under the cursor in Insert mode and Command-line mode. Looks good for dead characters.
- Made the **'langmap'** option recognize multibyte characters. But mapping only works for 8-bit characters. Helps when using UTF-8.
- Use a different cursor for when `":lmap"` mappings are active. Can specify two highlight groups for an item in **'guicursor'**. By default `"lCursor"` and `"Cursor"` are equal, the user must set a color he likes.
Use the cursor color for hangul input as well. (Sung-Hyun Nam)
- Show `"(lang)"` for **'showmode'** when language mapping is enabled.
- UTF-8: Made `"r"` work with a `":lmap"` that includes a composing character. Also works for `"f"`, which now works to find a character that includes a composing character.

Other multibyte character additions:

- Support double-byte single-width characters for `eu-jp`: Characters starting with `0x8E`. Added `ScreenLines2[]` to store the second byte.

Multi-language support

[new-multi-lang](#)

The messages used in Vim can be translated. Several translations are available. This uses the `gettext` mechanism. It allows adding a translation without recompiling Vim. [multi-lang](#) (partly by Marcin Dalecki)

The translation files are in the `src/po` directory. The `src/po/README.txt` file explains a few things about doing a translation.

Menu translations are available as well. This uses the new `:menutranslate` command. The translations are found in the runtime directory `"lang"`. This allows a user to add a translation.

Added `:language` command to set the language (locale) for messages, time and character type. This allows switching languages in Vim without changing the locale outside of Vim.

Made it possible to have `vimtutor` use different languages. (Eduardo Fernandez) Spanish (Eduardo Fernandez), Italian (Antonio Colombo), Japanese (Yasuhiro Matsumoto) and French (Adrien Beau) translations are included.

Added `"vimtutor.bat"`: script to start Vim on a copy of the tutor file for MS-Windows. (Dan Sharp)

- Added v:lang variable to be able to get current language setting. (Marcin Dalecki) Also v:lc_time and v:ctype.
- Make it possible to translate the dialogs used by the menus. Uses global "menutrans_" variables. ":menutrans clear" deletes them.
- removed "broken locale" (Marcin Dalecki).
- Don't use color names in icons, use RGB values. The names could be translated.
- Win32: Added global IME support (Muraoka)
- Win32: Added dynamic loading of IME support.
- ":messages" prints a message about who maintains the messages or the translations. Useful to find out where to make a remark about a wrong translation.
- --disable-nls argument for configure: Disable use of gettext(). (Sung-Hyun Nam)
- Added NLS support for Win32 with the MingW compiler. (Eduardo Fernandez)
- When available, call bind_textdomain_codeset() to have gettext() translate messages to 'encoding'. This requires GNU gettext 0.10.36 or later.
- Added gettext support for Win32. This means messages will be translated when the locale is set and libintl.dll can be found. (Muraoka Taro) Also made it work with MingW compiler. (Eduardo Fernandez) Detect the language and set \$LANG to get the appropriate translated messages (if supported). Also use \$LANG to select a language, v:lang is a very different kind of name.
- Made gvimext.dll use translated messages, if possible. (Yasuhiro Matsumoto)

Plugin support

new-plugins

To make it really easy to load a Vim script when starting Vim, the "plugin" runtime directory can be used. All "*.vim" files in it will be automatically loaded. For Unix, the directory "~/vim/plugin" is used by default. The 'runtimepath' option can be set to look in other directories for plugins.

load-plugins add-plugin

The :runtime command has been added to load one or more files in 'runtimepath'.

Standard plugins:

netrw.vim - Edit files over a network new-network-files

gzip.vim - Edit compressed files

explorer.vim - Browse directories new-file-browser

Added support for local help files. add-local-help .

When searching for help tags, all "doc/tags" files in 'runtimepath' are used.

Added the ":helptags" command: Generate a tags file for a help directory.

The first line of each help file is automatically added to the "LOCAL ADDITIONS" section in doc/help.txt.

Added the <unique> argument to ":map": only add a mapping when it wasn't defined before.

When displaying an option value with 'verbose' set will give a message about

where the option was last set. Very useful to find out which script did set the value.

The new `:scriptnames` command displays a list of all scripts that have been sourced.

GUI: For Athena, Motif and GTK look for a toolbar bitmap in the "bitmaps" directories in `'runtimepath'`. Allows adding your own bitmaps.

Filetype plugins

new-filetype-plugins

A new group of files has been added to do settings for specific file types. These can be options and mappings which are specifically used for one value of `'filetype'`.

The files are located in `"$VIMRUNTIME/ftplugin"`. The `'runtimepath'` option makes it possible to use several sets of plugins: Your own, system-wide, included in the Vim distribution, etc.

To be able to make this work, several features were added:

- Added the `"s:"` variables, local to a script. Avoids name conflicts with global variables. They can be used in the script and in functions, autocommands and user commands defined in the script. They are kept between invocations of the same script. `s:var`
- Added the global value for local options. This value is used when opening a new buffer or editing another file. The option value specified in a modeline or filetype setting is not carried over to another buffer.
 - `":set"` sets both the local and the global value.
 - `":setlocal"` sets the local option value only.
 - `":setglobal"` sets or displays the global value for a local option.
 - `":setlocal name<"` sets a local option to its global value.
- Added the buffer-local value for some global options: `'equalprg'`, `'makeprg'`, `'errorformat'`, `'grepprg'`, `'path'`, `'dictionary'`, `'thesaurus'`, `'tags'`, `'include'` and `'define'`. This allows setting a local value for these global options, without making it incompatible.
- Added mappings and abbreviations local to a buffer: `":map <buffer>"`.
- In a mapping `"<Leader>"` can be used to get the value of the `"mapleader"` variable. This simplifies mappings that use `"mapleader"`. `"<Leader>"` defaults to `"\"`. `"<LocalLeader>"` does the same with `"maplocalleader"`. This is to be used for mappings local to a buffer.
- Added `<SID>` Script ID to define functions and mappings local to a script.
- Added `<script>` argument to `":noremap"` and `":noremenu"`: Only remap script-local mappings. Avoids that mappings from other scripts get in the way, but does allow using mappings defined in the script.
- User commands can be local to a buffer: `":command -buffer"`.

The new `":setfiletype"` command is used in the filetype detection autocommands, to avoid that `'filetype'` is set twice.

File browser

new-file-browser

When editing a directory, the explorer plugin will list the files in the directory. Pressing `<Enter>` on a file name edits that file. Pressing `<Enter>` on a directory moves the browser to that directory.

There are several other possibilities, such as opening a file in the preview window, renaming files and deleting files.

Editing files over a network

[new-network-files](#)

Files starting with `scp://`, `rcp://`, `ftp://` and `http://` are recognized as remote files. An attempt is made to access these files with the indicated method. For `http://` only reading is possible, for the others writing is also supported. Uses the `netrw.vim` script as a standard "plugin". [netrw](#)

Made "gf" work on a URL. It no longer assumes the file is local on the computer (mostly didn't work anyway, because the full path was required). Adjusted test2 for this.

Allow using a URL in '`path`'. Makes `":find index.html"` work.

GTK: Allow dropping a `http://` and `ftp://` URL on Vim. The `netrw` plugin takes care of downloading the file. (Mikael Berthe)

Window for command-line editing

[new-cmdwin](#)

The Command-line window can be used to edit a command-line with Normal and Insert mode commands. When it is opened it contains the history. This allows copying parts of previous command lines. [cmdwin](#)

The command-line window can be opened from the command-line with the key specified by the '`cedit`' option (like `Nvi`). It can also be opened directly from Normal mode with `"q:"`, `"q/"` and `"q?"`.

The '`cmdwinheight`' is used to specify the initial height of the window.

In Insert mode **CTRL-X CTRL-V** can be used to complete an Ex command line, like it's done on the command-line. This is also useful for writing Vim scripts!

Additionally, there is "improved Ex mode". Entered when Vim is started as `"exim"` or `"vim -E"`, and with the `"gQ"` command. Works like repeated use of `":"`, with full command-line editing and completion. (Ulf Carlsson)

Debugging mode

[new-debug-mode](#)

In debugging mode sourced scripts and user functions can be executed line by line. There are commands to step over a command or step into it. [debug-mode](#)

Breakpoints can be set to run until a certain line in a script or user function is executed. [:breakadd](#)

Debugging can be started with `":debug {cmd}"` to debug what happens when a command executes. The `-D` argument can be used to debug while starting up.

Cursor in virtual position

[new-virtedit](#)

Added the **'virtualedit'** option: Allow positioning the cursor where there is no actual character in Insert mode, Visual mode or always. (Matthias Kramm)
This is especially useful in Visual-block mode. It allows positioning a corner of the area where there is no text character. (Many improvements by Chase Tingley)

Debugger interface

[new-debug-utf](#)

This was originally made to work with Sun Visual Workshop. (Gordon Prieur)
See [debugger.txt](#) , [sign.txt](#) and [workshop.txt](#) .

Added the `":sign"` command to define and place signs. They can be displayed with two ASCII characters or an icon. The line after it can be highlighted. Useful to display breakpoints and the current PC position.

Added the `:wsverb` command to execute debugger commands.

Added balloon stuff: **'balloondelay'** and **'ballooneval'** options.

Added `"icon="` argument for `":menu"`. Allows defining a specific icon for a ToolBar item.

Communication between Vims

[new-vim-server](#)

Added communication between two Vims. Makes it possible to send commands from one Vim to another. Works for X-Windows and MS-Windows [clientserver](#) .

Use `"--remote"` to have files be edited in an already running Vim.

Use `"--remote-wait"` to do the same and wait for the editing to finish.

Use `"--remote-send"` to send commands from one Vim to another.

Use `"--remote-expr"` to have an expression evaluated in another Vim.

Use `"--serverlist"` to list the currently available Vim servers. (X only)

There are also functions to communicate between the server and the client.

[remote_send\(\)](#) [remote_expr\(\)](#)

(X-windows version implemented by Flemming Madsen, MS-Windows version by Paul Moore)

Added the command server name to the window title, so you can see which server name belongs to which Vim.

Removed the OleVim directory and SendToVim.exe and EditWithVim.exe from the distribution. Can now use "gvim --remote" and "gvim --remote-send", which is portable.

GTK+: Support running Vim inside another window. Uses the --socketid argument (Neil Bird)

Buffer type options

new-buftype

The **'buftype'** and **'bufhidden'** options have been added. They can be set to have different kinds of buffers. For example:

- **'buftype'** = "quickfix": buffer with error list
- **'buftype'** = "nofile" and **'bufhidden'** = "delete": scratch buffer that will be deleted as soon as there is no window displaying it.

'bufhidden' can be used to overrule the **'hidden'** option for one buffer.

In combination with **'buflisted'** and **'swapfile'** this offers the possibility to use various kinds of special buffers. See [special-buffers](#) .

Printing

new-printing

Included first implementation of the ":hardcopy" command for printing to paper. For MS-Windows any installed printer can be used. For other systems a PostScript file is generated, which can be printed with the **'printexpr'** option.

(MS-Windows part by Vince Negri, Vipin Aravind, PostScript by Vince Negri and Mike Williams)

Made ":hardcopy" work with multibyte characters. (Muraoka Taro, Yasuhiro Matsumoto)

Added options to tune the way printing works: (Vince Negri)

- **'printoptions'** defines various things.
- **'printhead'** specifies the header format. Added "N" field to **'statusline'** for the page number.
- **'printfont'** specifies the font name and attributes.
- **'printdevice'** defines the default printer for ":hardcopy!".

Ports

ports-6

Port to OS/390 Unix (Ralf Schandl)

- A lot of changes to handle EBCDIC encoding.
- Changed Ctrl('x') to Ctrl_x define.

Included jsbmouse support. (Darren Garth)

Support for dec mouse in Unix. (Steve Wall)

Port to 16-bit MS-Windows (Windows 3.1x) (Vince Negri)

Port to QNX. Supports the Photon GUI, mouse, etc. (Julian Kinraid)

Allow cross-compiling the Win32 version with Make_ming.mak. (Ron Aaron)

Added Python support for compiling with Mingw. (Ron Aaron)

Dos 32 bit: Added support the Windows clipboard. (David Kotchan)

Win32: Dynamically load Perl and Python. Allows compiling Vim with these interfaces and will try to find the DLLs at runtime. (Muraoka Taro)

Compiling the Win32 GUI with Cygwin. Also compile vimrun, dosinst and uninstall. (Gerfried)

Mac: Make Vim compile with the free MPW compiler supplied by Apple. And updates for CodeWarrior. (Axel Kielhorn)

Added typecasts and ifdefs as a start to make Vim work on Win64 (George Reilly)

Quickfix extended

quickfix-6

Added the "error window". It contains all the errors of the current error list. Pressing <Enter> in a line makes Vim jump to that line (in another window). This makes it easy to navigate through the error list.

[quickfix-window](#) .

- [:copen](#) opens the quickfix window.
- [:cclose](#) closes the quickfix window.
- [:cwindow](#) takes care that there is a quickfix window only when there are recognized errors. (Dan Sharp)
- Quickfix also knows "info", next to "warning" and "error" types. "%I" can be used for the start of a multi-line informational message. (Tony Leneis)
- The "%p" argument can be used in '[errorformat](#)' to get the column number from a line where "^" points to the column. (Stefan Roemer)
- When using "%f" in '[errorformat](#)' on a DOS/Windows system, also include "c:" in the filename, even when using "%f:".

Operator modifiers

new-operator-mod

Insert "v", "V" or **CTRL-V** between an operator and a motion command to force the operator to work characterwise, linewise or blockwise. [o_v](#)

Search Path

new-search-path

Vim can search in a directory tree not only in downwards but also upwards. Works for the **'path'**, **'cdpath'** and **'tags'** options. (Ralf Schandl)

Also use **"**"** for **'tags'** option. (Ralf Schandl)

Added **'includeexpr'**, can be used to modify file name found by **'include'** option.

Also use **'includeexpr'** for **"gf"** and **"<cfil>"** when the file can't be found without modification. Useful for doing **"gf"** on the name after an include or import statement.

Added the **'cdpath'** option: Locations to find a **":cd"** argument. (Raf)

Added the **'suffixesadd'** option: Suffixes to be added to a file name when searching for a file for the **"gf"**, **"[I"**, etc. commands.

Writing files improved

new-file-writing

Added the **'backupcopy'** option: Select whether a file is to be copied or renamed to make a backup file. Useful on Unix to speed up writing an ordinary file. Useful on other systems to preserve file attributes and when editing a file on a Unix filesystem.

Added the **'autowriteall'** option. Works like **'autowrite'** but for more commands.

Added the **'backupskip'** option: A list of file patterns to skip making a backup file when it matches. The default for Unix includes **"/tmp/*"**, this makes **"crontab -e"** work.

Added support for Access Control Lists (ACL) for FreeBSD and Win32. The ACL is copied from the original file to the new file (or the backup if it's copied).

ACL is also supported for AIX, Solaris and generic POSIX. (Tomas Ogren)
And on SGI.

Argument list

new-argument-list

The support for the argument list has been extended. It can now be manipulated to contain the files you want it to contain.

The argument list can now be local to a window. It is created with the **:arglocal** command. The **:argglobal** command can be used to go back to the global argument list.

The **:argdo** command executes a command on all files in the argument list.

File names can be added to the argument list with **:argadd** . File names can be removed with **:argdelete** .

"##" can be used like "#", it is replaced by all the names in the argument list concatenated. Useful for ":grep foo ##".

The `:argedit` adds a file to the argument list and edits it. Like ":argadd" and then ":edit".

Restore a View

new-View

The ":mkview" command writes a Vim script with the settings and mappings for one window. When the created file is sourced, the view of the window is restored. It's like ":mksession" for one window.

The View also contains the local argument list and manually created, opened and closed folds.

Added the ":loadview" command and the `'viewdir'` option: Allows for saving and restoring views of a file with simple commands. ":mkview 1" saves view 1 for the current file, ":loadview 1" loads it again. Also allows quickly switching between two views on one file. And saving and restoring manual folds and the folding state.

Added `'viewoptions'` to specify how ":mkview" works.

":mksession" now also works fine with vertical splits. It has been further improved and restores the view of each window. It also works properly with preview and quickfix windows.

`'sessionoptions'` is used for ":mkview" as well.

Added "curdir" and "sesdir" to `'sessionoptions'`. Allows selection of what the current directory will be restored to.

The session file now also contains the argument list(s).

Color schemes

new-color-schemes

Support for loading a color scheme. Added the ":colorscheme" command.

Automatically add menu entries for available schemes.

Should now properly reset the colors when `'background'` or `'t_Co'` is changed.

":highlight clear" sets the default colors again.

":syntax reset" sets the syntax highlight colors back to the defaults.

For ":set bg&" guess the value. This allows a color scheme to switch back to the default colors.

When syntax highlighting is switched on and a color scheme was defined, reload the color scheme to define the colors.

Various new items

new-items-6

Normal mode commands:

"gi" Jump to the ^ mark and start Insert mode. Also works when the mark is just after the line. `gi`

"g'm" and "g`m" Jump to a mark without changing the jumplist. Now you can use `g`` to jump to the last known position in a file without side effects. Also useful in mappings.

[', [`,]' and]` move the cursor to the next/previous lowercase mark.

`g_` Go to last non-blank in line. (Steve Wall)

Options:

'autoread' When detected that a file changed outside of Vim, automatically read a buffer again when it's not changed. It has a global and a local value. Use `":setlocal autoread<"` to go back to using the global value for **'autoread'**.

'debug' When set to "msg" it will print error messages that would otherwise be omitted. Useful for debugging **'indentexpr'** and **'foldexpr'**.

'lispwords' List of words used for lisp indenting. It was previously hard coded. Added a number of Lisp names to the default.

'fold...' Many new options for folding.

'modifiable' When off, it is impossible to make changes to a buffer. The %m and %M items in **'statusline'** show a '-'.
'previewwindow' Set in the preview window. Used in a session file to mark a window as the preview window.

'printfont'
'printexpr'
'printhead'
'printdevice'
'printoptions' for `":hardcopy"`.

'buflisted' Makes a buffer appear in the buffer list or not.

Use `"vim{version}:"` for modelines, only to be executed when the version is `>= {version}`. Also `"vim>{version}"`, `"vim<{version}"` and `"vim={version}"`.

Ex commands:

`:sav[eas][!] {file}`
Works like `":w file"` and `":e #"`, but without loading the file again and avoiding other side effects. `:saveas`

`:silent[!] {cmd}` Execute a command silently. Also don't use a delay that would come after the message. And don't do `'showmatch'`.
RISCOS: Removed that `"!~cmd"` didn't output anything, and didn't wait for `<Enter>` afterwards. Can use `":silent !cmd"` now.

`:menu <silent>` Add a menu that won't echo Ex commands.

`:map <silent>` Add a mapping that won't echo Ex commands.

`:checktime` Check for changed buffers.

`:verbose {cmd}` Set `'verbose'` for one command.

`:echormsg {expr}`
`:echoerr {expr}` Like `":echo"` but store the message in the history. (Mark Waggoner)

`:grepadd` Works just like `":grep"` but adds to the current error list instead of defining a new list. `:grepadd`

`:finish` Finish sourcing a file. Can be used to skip the rest of a Vim script. `:finish`

`:leftabove`
`:aboveleft` Split left/above current window.

`:rightbelow`
`:belowright` Split right/below current window.

`:first, :bfirst, :ptfirst, etc.`
Alias for `":rewind"`. It's more logical compared to `":last"`.

`:enew` Edit a new, unnamed buffer. This is needed, because `":edit"` re-edits the same file. (Wall)

`:quitall` Same as `":qall"`.

`:match` Define match highlighting local to a window. Allows highlighting an item in the current window without interfering with syntax highlighting.

`:menu enable`
`:menu disable` Commands to enable/disable menu entries without removing them. (Monish Shah)

`:windo` Execute a command in all windows.

`:bufdo` Execute a command in all buffers.

`:wincmd` Window (CTRL-W) command. Useful when a Normal mode command can't be used (e.g., for a CursorHold autocommand). See [CursorHold-example](#) for a nice application with it.

`:lcd` and `:lchdir`
Set local directory for a window. (Benjie Chen)

`:hide {command}` Execute `{command}` with `'hidden'` set.

`:emenu` in Visual mode to execute a `":vmenu"` entry.

`:popup` Pop up a popup menu.

`:redraw` Redraw the screen even when busy with a script or function.

`:hardcopy` Print to paper.

`:compiler` Load a Vim script to do settings for a specific compiler.

`:z#` List numbered lines. (Bohdan Vlasyuk)

New marks:

`'(and ')` Begin or end of current sentence. Useful in Ex commands.

`'{ and '}` Begin or end of current paragraph. Useful in Ex commands.

`'.` Position of the last change in the current buffer.

`'^` Position where Insert mode was stopped.

Store the `^` and `.` marks in the viminfo file. Makes it possible to jump to the last insert position or changed text.

New functions:

`argidx()` Current index in argument list.

`buflisted()` Checks if the buffer exists and has `'buflisted'` set.

`cindent()` Get indent according to `'cindent'`.

`eventhandler()` Returns 1 when inside an event handler and interactive commands can't be used.

`executable()` Checks if a program or batch script can be executed.

`filewritable()` Checks if a file can be written. (Ron Aaron)

`foldclosed()` Find out if there is a closed fold. (Johannes Zellner).

`foldclosedend()` Find the end of a closed fold.

`foldlevel()` Find out the foldlevel. (Johannes Zellner)

`foreground()` Move the GUI window to the foreground.

`getchar()` Get one character from the user. Can be used to define a mapping that takes an argument.

`getcharmod()` Get last used key modifier.

`getbufvar()` gets the value of an option or local variable in a buffer (Ron Aaron)

`getfsize()` Return the size of a file.

`getwinvar()` gets the value of an option or local variable in a window (Ron Aaron)

`globpath()` Find matching files in a list of directories.

`hasmapto()` Detect if a mapping to a string is already present.

`iconv()` Convert a string from one encoding to another.

`indent()` gets the indent of a line (Ron Aaron)

`inputdialog()` Like `input()` but use a GUI dialog when possible. Currently only works for Win32, Motif, Athena and GTK.

Use `inputdialog()` for the Edit/Settings/Text Width menu. Also for the Help/Find.. and Toolbar FindHelp items.
 (Win32 support by Thore B. Karlsen)
 (Win16 support by Vince Negri)

`inputsecret()` Ask the user to type a string without showing the typed keys.
 (Charles Campbell)

`libcall()` for Unix (Neil Bird, Johannes Zellner, Stephen Wall)
`libcallnr()` for Win32 and Unix

`lispindent()` Get indent according to **'lisp'**.

`mode()` Return a string that indicates the current mode.

`nextnonblank()` Skip blank lines forwards.

`prevnonblank()` Skip blank lines backwards. Useful to for indent scripts.

`resolve()` MS-Windows: resolve a shortcut to the file it points to.
 Unix: resolve a symbolic link.

`search()` Search for a pattern.

`searchpair()` Search for matching pair. Can be used in indent files to find the "if" matching an endif.

`setbufvar()` sets an option or variable local to a buffer (Ron Aaron)
`setwinvar()` sets an option or variable local to a window (Ron Aaron)

`stridx()` Search for first occurrence of one string in another.
`strridx()` Search for last occurrence of one string in another.

`tolower()` Convert string to all-lowercase.
`toupper()` Convert string to all-uppercase.

`type()` Check the type of an expression.

`wincol()` window column of the cursor
`winwidth()` Width of a window. (Johannes Zellner)
`winline()` window line of the cursor

Added expansion of curly braces in variable and function names. This can be used for variable names that include the value of an option. Or a primitive form of arrays. (Vince Negri)

New autocommand events:

`BufWinEnter` Triggered when a buffer is displayed in a window, after using the modelines. Can be used to load a view.

`BufWinLeave` Triggered when a buffer is no longer in a window. Also triggered when exiting Vim. Can be used to save views.

`FileChangedRO` Triggered before making the first change to a read-only file. Can be used to check-out the file. (Scott Graham)

`TermResponse` Triggered when the terminal replies to the version-request. The `v:termresponse` internal variable holds the result. Can be used to react to the version of the terminal. (Ronald Schild)

`FileReadCmd` Triggered before reading a file.
`BufReadCmd` Triggered before reading a file into a buffer.
`FileWriteCmd` Triggered before writing a file.
`BufWriteCmd` Triggered before writing a buffer into a file.
`FileAppendCmd` Triggered before appending to a file.
`FuncUndefined` Triggered when a user function is not defined. (Ron Aaron)

The autocommands for the *Cmd events read or write the file instead of normal file read/write. Use this in `netrw.vim` to be able to edit files on a remote system. (Charles Campbell)

New Syntax files:

bdf	BDF font definition (Nikolai Weibull)
catalog	SGML catalog (Johannes Zellner)
debchangelog	Debian Changelog (Wichert Akkerman)
debcontrol	Debian Control (Wichert Akkerman)
dot	dot (Markus Mottl)
dsl	DSSSL syntax (Johannes Zellner)
etern	Eterm configuration (Nikolai Weibull)
indent	Indent profile (Nikolai Weibull)
lftp	LFTP (Nikolai Weibull)
lynx	Lynx config (Doug Kearns)
mush	mush sourcecode (Bek Oberin)
natural	Natural (Marko Leipert)
pilrc	Pal resource compiler (Brian Schau)
plm	PL/M (Philippe Coulonges)
povini	Povray configuration (David Necas)
ratpoison	Ratpoison config/command (Doug Kearns)
readline	readline config (Nikolai Weibull)
screen	Screen RC (Nikolai Weibull)
specman	Specman (Or Freund)
sqlforms	SQL*Forms (Austin Ziegler)
terminfo	terminfo (Nikolai Weibull)
tidy	Tidy configuration (Doug Kearns)
wget	Wget configuration (Doug Kearns)

Updated many syntax files to work both with Vim 5.7 and 6.0.

Interface to Ruby. (Shugo Maeda)

Support dynamic loading of the Ruby interface on MS-Windows. (Muraoka Taro)

Support this for Mingw too. (Benoit Cerrina)

Win32: Added possibility to load TCL dynamically. (Muraoka Taro)

Also for Borland 5.5. (Dan Sharp)

Win32: When editing a file that is a shortcut (*.lnk file), edit the file it links to. Unless **'binary'** is set, then edit the shortcut file itself.

(Yasuhiro Matsumoto)

The ":command" command now accepts a "-bar" argument. This allows the user command to be followed by "| command".

The preview window is now also used by these commands:

- **:pedit** edits the specified file in the preview window
- **:psearch** searches for a word in included files, like **:ijump**, and displays the found text in the preview window.

Added the **CTRL-W P** command: go to preview window.

MS-DOS and MS-Windows also read the system-wide vimrc file \$VIM/vimrc. Mostly for NT systems with multiple users.

A double-click of the mouse on a character that has a "%" match selects from that character to the match. Similar to "v%".

"-S session.vim" argument: Source a script file when starting up. Convenient way to start Vim with a session file.

Added "--cmd {command}" Vim argument to execute a command before a vimrc file is loaded. (Vince Negri)

Added the "-M" Vim argument: reset 'modifiable' and 'write', thus disallow making changes and writing files.

Added runtime/delmenu.vim. Source this to remove all menus and prepare for loading new menus. Useful when changing 'langmenu'.

Perl script to filter Perl error messages to quickfix usable format. (Joerg Ziefle)

Added runtime/macros/less.vim: Vim script to simulate less, but with syntax highlighting.

MS-Windows install program: (Jon Merz)

- The Win32 program can now create shortcuts on the desktop and install Vim in the Start menu.
- Possibly remove old "Edit with Vim" entries.
- The Vim executable is never moved or \$PATH changed. A small batch file is created in a directory in \$PATH. Fewer choices to be made.
- Detect already installed Vim versions and offer to uninstall them first.

Improved the MS-Windows uninstal program. It now also deletes the entries in the Start menu, icons from the desktop and the created batch files. (Jon Merz) Also made it possible to delete only some of these. Also unregister gvim for OLE.

Generate a self-installing Vim package for MS-Windows. This uses NSIS. (Jon Merz et al.)

Added ":filetype detect". Try detecting the filetype again. Helps when writing a new shell script, after adding "#!/bin/csh".

Added ":augroup! name" to delete an autocommand group. Needed for the client-server "--remote-wait".

Add the Vim version number to the viminfo file, useful for debugging.

=====

IMPROVEMENTS

improvements-6

Added the 'n' flag in 'coptions': When omitted text of wrapped lines is not put between line numbers from 'number' option. Makes it a lot easier to read wrapped lines.

When there is a format error in a tags file, the byte position is reported so that the error can be located.

"gf" works in Visual mode: Use the selected text as the file name. (Chase Tingley)

Allow ambiguous mappings. Thus "aa" and "aaa" can both be mapped, the longest matching one is used. Especially useful for ":lmap" and 'keymap'.

Encryption: Ask the key to be typed twice when crypting the first time. Otherwise a typo might cause the text to be lost forever. (Chase Tingley)

The window title now has "VIM" on the end. The file name comes first, useful in the taskbar. A "+" is added when the file is modified. "=" is added for a read-only file. "-" is added for a file with 'modifiable' off.

In Visual mode, mention the size of the selected area in the 'showcmd' position.

Added the "b:changedtick" variable. Incremented at each change, also for undo. Can be used to take action only if the buffer has been changed.

In the replacement string of a ":s" command "\=" can be used to replace with the result of an expression. From this expression the submatch() function can be used to access submatches.

When doing ":qall" and there is a change in a buffer that is being edited in another window, jump to that window, instead of editing that buffer in the current window.

Added the "++enc=" and "++ff=" arguments to file read/write commands to force using the given 'encoding' or 'fileformat'. And added the "v:cmdarg" variable, to be used for FileReadCmd autocommands that read/write the file themselves.

When reading stdin, first read the text in binary mode and then re-read it with automatic selection of 'fileformat' and 'fileencoding'. This avoids problems with not being able to rewind the file (e.g., when a line near the end of the file ends in LF instead of CR-LF).

When reading text from stdin and the buffer is empty, don't mark it changed. Allows exiting without trouble.

Added an ID to many error messages. This will make it easier to find help for a message.

Insert mode:

- "CTRL-G j" and "CTRL-G k" can be used to insert in another line in the same column. Useful for editing a table.
- Added Thesaurus completion with CTRL-X CTRL-T. (Vince Negri)
- Added the 'thesaurus' option, to use instead of 'dictionary' for thesaurus completion. Added the 's' flag in 'complete'.
- Made CTRL-X CTRL-L in Insert mode use the 'complete' option. It now also scans other loaded buffers for matching lines.
- CTRL-R now also works in Insert mode while doing completion with CTRL-X or CTRL-N. (Neil Bird)
- When doing Insert mode completion, when completion is finished check for a

match with words from **'cinkeys'** or **'indentkeys'**.

Performance:

- Made display updating more efficient. Insert/delete lines may be used for all changes, also for undo/redo.
- The display is not redrawn when there is typeahead in Insert mode. Speeds up **CTRL-R** a lot.
- Improved speed of screen output for 32 bit DOS version. (Vince Negri)
- When dragging with the mouse, there is a lookahead to skip mouse codes when there is another one next. Makes dragging with the mouse a lot faster.
- Also a memory usage improvement: When calling `u_save` with a single line, don't save it if the line was recently saved for the same undo already.
- When using a script that appends one character at a time, the amount of allocated memory was growing steadily. Also when **'undolevels'** is -1. Caused by the line saved for "U" never to be freed. Now free an undo block when it becomes empty.
- GUI and Dos32: Use a vertical scroll region, to make scrolling in a vertically split window faster. No need to redraw the whole window.
- When scrolling isn't possible with terminal codes (e.g., for a vertically split window) redraw from `ScreenLines[]`. That should be faster than going through the lines with `win_line()`, especially when using syntax highlighting.
- The Syntax menu is now pre-generated by a separate script. Makes loading the menu 70% faster. This can halve the startup time of `gvim`.
- When doing `:"help tag"`, don't open `help.txt` first, jump directly to the help tag. It's faster and avoids an extra message.
- Win32: When a file name doesn't end in `".lnk"` don't try resolving a shortcut, it takes quite a bit of time.
- Don't update the mouse pointer shape while there are typeahead characters.
- Change `META[]` from a string into an array, avoids using `strchr()` on it.
- Don't clear the command line when adding characters, avoids that `screen_fill` is called but doesn't do anything.

Robustness:

- Unix: Check for running out of stack space when executing a regexp. Avoids a nasty crash. Only works when the system supports running the signal function on another stack.
- Disallow `:"source <dirname>"`. On unix it's possible to read a directory, does not make sense to use it as Vim commands.

Security:

- When reading from or writing to a temporary file, check that it isn't a symbolic link. Gives some protection against symlink attacks.
- When creating a backup file copy or a swap file, check for it already existing to avoid a symlink attack. (Colin Phipps)
- Evaluating options which are an expression is done in a `sandbox`. If the option was set by a modeline, it cannot cause damage.
- Use a secure way to generate temp file names: Create a private directory for temp files. Used for Unix, MS-DOS and OS/2.
- **'makeef'** can be empty, which means that an internally generated file name is used. The old default was `"/tmp/file"`, which is a security risk. Writing **'makeef'** in the current directory fails in a read-only directory and causes trouble when using `:"grep"` on all files. Made the default empty for all systems, so that a temp file is used.

- The command from a tags file is executed in the sandbox for better security.
- The Ruby, Tcl and Python interfaces cannot be used from the sandbox. They might do dangerous things. Perl is still possible, but limited to the Safe environment. (Donnie Smith)

Syntax highlighting:

- Optimized the speed by caching the state stack all over the file, not just the part being displayed. Required for folding.
- Added `":syntax sync fromstart"`: Always parse from the start of the file.
- Added the `"display"` argument for syntax items: use the item only when displaying the result. Can make parsing faster for text that isn't going to be displayed.
- When using **CTRL-L**, the cached states are deleted, to force parsing the text again.
- Use elfhash algorithm for table of keywords. This should give a better distribution and speedup keyword lookup. (Campbell)
- Also allow the `"lc"` leading context for skip and end patterns. (Scott Bigham)
- Syntax items can have the `"extend"` argument to undo the effect of a `"keepend"` argument of an item it is contained in. Makes it possible to have some contained items extend a region while others don't.
- `":syntax clear"` now deletes the `b:current_syntax` variable. That's logical, since no syntax is defined after this command.
- Added `":syntax enable"`: switch on syntax highlighting without changing the colors. This allows specifying the colors in the `.vimrc` file without the need for a `mysyntaxfile`.
- Added `":syntax reset"`: reset the colors to their defaults.
- Added the `"contains=TOP"` and `"contains=CONTAINED"` arguments. Makes it possible to define a transparent item that doesn't contain itself.
- Added a `"containedin"` argument to syntax items. Allows adding a contained item to an existing item (e.g., to highlight a name in a comment).

Modeless selection:

- When in the command-line window, use modeless selection in the other windows. Makes it possible to copy visible text to the command-line window.
- Support modeless selection on the cmdline in a terminal. Previously it was only possible for the GUI.
- Make double-right-click in modeless selection select a whole word. Single right click doesn't use the word selection started by a double-left-click. Makes it work like in Visual mode.
- The modeless selection no longer has an implied automatic copy to the clipboard. It now obeys the `'a'` and `'A'` flags in `'guioptions'` or `"autoselect"` and `"autoselectml"` in `'clipboard'`.
- Added the **CTRL-Y** command in Cmdline-mode to copy the modeless selection to the clipboard. Also works at the hit-enter prompt and the more prompt. Removed the mappings in `runtime/mswin.vim` for **CTRL-Y** and **CTRL-Z** in cmdline-mode to be able to use **CTRL-Y** in the new way.

Reduced the amount of stack space used by `regmatch()` to allow it to handle complicated patterns on a longer text.

`'isfname'` now includes `'%'` and `'#'`. Makes `"vim dir\#file"` work for MS-DOS.

Added keypad special keys `<kEnter>`, `<k0>` - `<k9>`. When not mapped they behave

like the ASCII equivalent. (Ivan Wellesz and Vince Negri)

Recognize a few more xterm keys: <C-Right>, <C-Left>, <C-End>, <C-Home>

Also trigger the BufUnload event when Vim is going to exit. Perhaps a script needs to do some cleaning up.

Expand expression in backticks: ``={expr}``. Can be used where backtick expansion is done. (Vince Negri)

GUI:

- Added 'L' and 'R' flags in **'guioptions'**: Add a left or right scrollbar only when there is a vertically split window.
- X11: When a color can't be allocated, use the nearest match from the colormap. This avoids that black is used for many things. (Monish Shah)
Also do this for the menu and scrollbar, to avoid that they become black.
- Win32 and X11: Added **'mousethshape'** option: Adjust the mouse pointer shape to the current mode. (Vince Negri)
- Added the **'linespace'** option: Insert a pixel line between lines. (Nam)
- Allow modeless selection (without moving the cursor) by keeping CTRL and SHIFT pressed. (Ivan Wellesz)
- Motif: added toolbar. (Gordon Prieur) Also added tooltips.
- Athena: added toolbar and tooltips. (David Harrison -- based on Gordon Prieur's work)
- Made the **'toolbar'** option work for Athena and Motif. Can now switch between text and icons on the fly. (David Harrison)
- Support menu separator lines for Athena. (David Harrison)
- Athena: Adjust the arrow pixmap used in a pullright menu to the size of the font. (David Harrison)
- Win32: Added "c" flag to **'guifont'** to be able to specify the charset. (Artem Khodush)
- When no --enable-xim argument is given, automatically enable it when a X GUI is used. Required for dead key support (and multibyte input).
- After a file selection dialog, check that the edited files were not changed or deleted. The Win32 dialog allows deleting and renaming files.
- Motif and Athena: Added support for "editres". (Marcin Dalecki)
- Motif and Athena: Added "menuFont" to be able to specify a font or fontset for the menus. Can also be set with the "Menu" highlight group. Useful when the locale is different from **'encoding'**. (David Harrison)
When FONTSET_ALWAYS is defined, always use a fontset for the menus. Should avoid trouble with changing from a font to a fontset. (David Harrison)
- Highlighting and font for the tooltips can be specified with the "Tooltip" highlight group. (David Harrison)
- The Cmdline-mode menus can be used at the more-prompt. This mostly works fine, because they start with a **CTRL-C**. The "Copy" menu works to copy the modeless selection. Allows copying the output of ":set all" or ":intro" without auto-selection.
- When starting the GUI when there is no terminal connected to stdout and stderr, display error messages in a dialog. Previously they wouldn't be displayed at all.
- Allow setting **'browse_dir'** to the name of a directory, to be used for the file dialog. (Dan Sharp)
- b:browsefilter and g:browsefilter can be set to the filters used for the file dialog. Supported for Win32 and Motif GUI. (Dan Sharp)

X11:

- Support for the clipboard selection as register "+. When exiting or suspending copy the selection to cut buffer 0. Should allow copy/paste with more applications in a X11-standard way. (Neil Bird)
- Use the X clipboard in any terminal, not just in an xterm.
Added "exclude:" in '**clipboard**': Specify a pattern to match against terminal names for which no connection should be made to the X server. The default currently work for FreeBSD and Linux consoles.
- Added a few messages for when '**verbose**' is non-zero to show what happens when trying to connect to the X server. Should help when trying to find out why startup is slow.

GTK GUI: (partly by Marcin Dalecki)

- With some fonts the characters can be taller than ascent + descent. E.g., "-misc-fixed-x-x-x-x-18-x-x-x-x-iso10646-1". Add one to the character cell height.
- Implement "no" value for '**winaltkeys**': don't use Alt-Key as a menu shortcut, when '**wak**' changed after creating the menus.
- Setting '**wak**' after the GUI started works.
- recycle text GC's to reduce communication.
- Adjust icon size to window manager.
- Cleanup in font handling.
- Replace XQueryColor with GDK calls.
- Gnome support. Detects Gnome in configure and uses different widgets. Otherwise it's much like GTK. (Andy Kahn)
It is disabled by default, because it causes a few problems.
- Removed the special code to fork first and then start the GUI. Now use _exit() instead of exit(), this works fine without special tricks.
- Dialogs sometimes appeared a bit far away. Position the dialogs inside the gvim window. (Brent Verner)
- When dropping a file on Vim, remove extra slashes from the start of the path. Also shorten the file name if possible.

Motif: (Marcin Dalecki)

- Made the dialog layout better.
- Added find and find/replace dialogs.
- For the menus, change "iso-8859" to "iso_8859", Linux appears to need this.
- Added icon to dialogs, like for GTK.
- Use XPM bitmaps for the icon when possible. Use the Solaris XpmP.h include file when it's available.
- Change the shadow of the toolbar items to get a visual feedback of it being pressed on non-LessTif.
- Use gadgets instead of windows for some items for speed.

Command line completion:

- Complete environment variable names. (Mike Steed)
- For ":command", added a few completion methods: "mapping", "function", "expression" and "environment".
- When a function doesn't take arguments, let completion add () instead of (.

For MS-DOS, MS-Windows and OS/2: Expand %VAR% environment variables like \$VAR. (Walter Briscoe)

Redirect messages to the clipboard ":redir @*" and to the unnamed register

`":redir @"`. (Wall)

`":let @/ = ''` clears the search pattern, instead of setting it to an empty string.

Expression evaluation:

- `"? :` can be used like in C.
- `col("$")` returns the length of the cursor line plus one. (Stephen P. Wall)
- Optional extra argument for `match()`, `matchend()` and `matchstr()`: Offset to start looking for a match.
- Made third argument to `strpart()` optional. (Paul Moore, Zdenek Sekera)
- `exists()` can also be used to check for Ex commands and defined autocommands.
- Added extra argument to `input()`: Default text.
- Also set `"v:errmsg"` when using `":silent! cmd"`.
- Added the `v:prevcount` variable: `v:count` for the previous command.
- Added `"v:progname"`, name with which Vim was started. (Vince Negri)
- In the verbose message about returning from a function, also show the return value.

Cscope:

- Added the `cscope_connection()` function. (Andy Kahn)
- `":cscope kill -1` kills all cscope connections. (Andy Kahn)
- Added the `'cscopepathcomp'` option. (Scott Hauck)
- Added `":scscope"` command, split window and execute Cscope command. (Jason Duell)

VMS:

- Command line arguments are always uppercase. Interpret a `"-X"` argument as `"-x"` and `"-/X"` as `"-X"`.
- Set `'makeprg'` and `'grepprg'` to meaningful defaults. (Zoltan Arpadffy)
- Use the X-clipboard feature and the X command server. (Zoltan Arpadffy)

Macintosh: (Dany St-Amant)

- Allow a tags file to have CR, CR-LF or LF line separator. (Axel Kielhorn)
- Carbonized (while keeping non Carbon code)
(Some work "stolen" from Ammon Skidmore)
- Improved the menu item index handling (should be faster)
- Runtime commands now handle `/` in file name (MacOS 9 version)
- Added `":winpos"` support.
- Support using `"~"` in file names for home directory.

Options:

- When using `set +=` or `^=`, check for items used twice. Duplicates are removed. (Vince Negri)
- When setting an option that is a list of flags, remove duplicate flags.
- If possible, use `getrlimit()` to set `'maxmemtot'` and `'maxmem'`. (Pina)
- Added `"alpha"` to `'nrformats'`: increment or decrement an alphabetic character with **CTRL-A** and **CTRL-X**.
- `":set opt&vi"` sets an option to its Vi default, `":set opt&vim"` to its Vim default. Useful to set `'cpo'` to its Vim default without knowing what flags that includes.
- `'scrolloff'` now also applies to a long, wrapped line that doesn't fit in the window.
- Added more option settings to the default menus.

- Updated the option window with new options. Made it a bit easier to read.

Internal changes:

- Split line pointers in text part and attributes part. Allows for future change to make attribute more than one byte.
- Provide a qsort() function for systems that don't have it.
- Changed the big switch for Normal mode commands into a table. This cleans up the code considerably and avoids trouble for some optimizing compilers.
- Assigned a negative value to special keys, to avoid them being mixed up with Unicode characters.
- Global variables expand_context and expand_pattern were not supposed to be global. Pass them to ExpandOne() and all functions called by it.
- No longer use the global reg_ic flag. It caused trouble and in a few places it was not set.
- Removed the use of the stuff buffer for "*", "K", **CTRL-]**, etc. Avoids problem with autocommands.
- Moved some code from ex_docmd.c to ex_cmds2.c. The file was getting too big. Also moved some code from screen.c to move.c.
- Don't include the CRC table for encryption, generate it. Saves quite a bit of space in the source code. (Matthias Kramm)
- Renamed multibyte.c to mbyte.c to avoid a problem with 8.3 filesystems.
- Removed the GTK implementation of ":findhelp", it now uses the ToolBar.FindHelp menu entry.
- Renamed mch_windexit() to mch_exit(), mch_init() to mch_early_init() and mch_shellinit() to mch_init().

Highlighting:

- In a ":highlight" listing, show "xxx" with the highlight color.
- Added support for xterm with 88 or 256 colors. The right color numbers will be used for the name used in a ":highlight" command. (Steve Wall)
- Added "default" argument for ":highlight". When included, the command is ignored if highlighting for the group was already defined. All syntax files now use ":hi default ..." to allow the user to specify colors in his vimrc file. Also, the "if did_xxx_syntax_inits" is not needed anymore. This greatly simplifies using non-default colors for a specific language.
- Adjusted colortest.vim: Included colors on normal background and reduced the size by using a while loop. (Rafael Garcia-Suarez)
- Added the "DarkYellow" color name. Just to make the list of standard colors consistent, it's not really a nice color to use.

When an xterm is in 8-bit mode this is detected by the code returned for `t_RV`. All key codes are automatically converted to their 8-bit versions.

The `OPT_TCAP_QUERY` in xterm patch level 141 and later is used to obtain the actual key codes used and the number of colors for `t_Co`. Only when `t_RV` is also used.

":browse set" now also works in the console mode. ":browse edit" will give an error message.

":bdelete" and ":bunload" only report the number of deleted/unloaded buffers when more than **'report'**. The message was annoying when deleting a buffer in a script.

Jump list:

- The number of marks kept in the jumplist has been increased from 50 to 100.
- The jumplist is now stored in the viminfo file. **CTRL-O** can be used to jump to positions from a previous edit session.
- When doing `:"split` copy the jumplist to the new window.

Also set the '[' and ']' marks for the "~" and "r" commands. These marks are now always set when making a change with a Normal mode command.

Python interface: Allow setting the width of a vertically split window. (John Cook)

Added "=word" and "=~word" to **'cinkeys'** (also used in **'indentkeys'**).

Added "j1" argument in **'cinoptions'**: indent {} inside () for Java. (Johannes Zellner)

Added the "l" flag in **'cinoptions'**. (Anduin Withers)

Added 'C', 'U', 'w' and 'm' flags to **'cinoptions'**. (Servatius Brandt)

When doing `:"wall` or `:"wqall` and a modified buffer doesn't have a name, mention its buffer number in the error message.

`:"function Name` lists the function with line numbers. Makes it easier to find out where an error happened.

In non-blockwise Visual mode, "r" replaces all selected characters with the typed one, like in blockwise Visual mode.

When editing the last file in the argument list in any way, allow exiting. Previously this was only possible when getting to that file with `:"next` or `:"last`.

Added the 'l' flag to **'formatoptions'**. (Vit Stradal)

Added 'n' flag in **'formatoptions'**: format a numbered list.

Swap file:

- When a swap file already exists, and the user selects "Delete" at the ATTENTION prompt, use the same ".swp" swapfile, to avoid creating a ".swo" file which won't always be found.
- When giving the ATTENTION message and the date of the file is newer than the date of swap file, give a warning about this.
- Made the info for an existing swap file a bit shorter, so that it still fits on a 24 line screen.
- It was possible to make a symlink with the name of a swap file, linking to a file that doesn't exist. Vim would then silently use another file (if open with O_EXCL refuses a symlink). Now check for a symlink to exist. Also do another check for an existing swap file just before creating it to catch a symlink attack.

The g **CTRL-G** command also works in Visual mode and counts the number of words. (Chase Tingley)

Give an error message when using **'shell'** and it's empty.

Added the possibility to include "%s" in 'shellpipe'.

Added "uhex" value for 'display': show non-printable characters as <xx>. Show unprintable characters with NonText highlighting, also in the command line.

When asked to display the value of a hidden option, tell it's not supported.

Win32:

- When dropping a shortcut on gvim (.lnk file) edit the target, not the shortcut itself. (Yasuhiro Matsumoto)
- Added C versions of the OpenWithVim and SendToVim programs. (Walter Briscoe)
- When 'shell' is "cmd" or "cmd.exe", set 'shellredir' to redirect stderr too. Also check for the Unix shell names.
- When \$HOMEDRIVE and \$HOMEPATH are defined, use them to define \$HOME. (Craig Barkhouse)

Win32 console version:

- Includes the user and system name in the ":version" message, when available. It generates a pathdef.c file for this. (Jon Miner)
- Set the window icon to Vim's icon (only for Windows 2000). While executing a shell command, modify the window title to show this. When exiting, restore the cursor position too. (Craig Barkhouse)
- The Win32 console version can be compiled with OLE support. It can only function as a client, not as an OLE server.

Errorformat:

- Let "%p" in 'errorformat' (column of error indicated by a row of characters) also accept a line of dots.
- Added "%v" item in 'errorformat': Virtual column number. (Dan Sharp)
- Added a default 'errorformat' value for VMS. (Jim Bush)

The "p" command can now be used in Visual mode. It overwrites the selected text with the contents of a register.

Highlight the <> items in the intro message to make clear they are special.

When using the "c" flag for ":substitute", allow typing "l" for replacing this item and then stop: "last".

When printing a verbose message about sourcing another file, print the line number.

When resizing the Vim window, don't use 'equalalways'. Avoids that making the Vim window smaller makes split windows bigger. And it's what the docs say.

When typing **CTRL-D** in Insert mode, just after an autoindent, then hitting CR kept the remaining white space. Now made it work like BS: delete the autoindent to avoid a blank non-empty line results.

Added a GetHwnd() call to the OLE interface. (Vince Negri)

Made ":normal" work in an event handler. Useful when dropping a file on Vim

and for CursorHold autocommands.

For the MS-Windows version, don't change to the directory of the file when a slash is used instead of a backslash. Explorer should always use a backslash, the user can use a slash when typing the command.

Timestamps:

- When a buffer was changed outside of Vim and regaining focus, give a dialog to allow the user to reload the file. Now also for other GUIs than MS-Windows. And also used in the console, when compiled with dialog support.
- Inspect the file contents to find out if it really changed, ignore situations where only the time stamp changed (e.g., checking the file out from CVS).
- When checking the timestamp, first check if the file size changed, to avoid a file compare then. Makes it quicker for large (log) files that are appended to.
- Don't give a warning for a changed or deleted file when **'buftype'** is set.
- No longer warn for a changed directory. This avoids that the file explorer produces warnings.
- Checking timestamps is only done for buffers that are not hidden. These will be checked when they become unhidden.
- When checking for a file being changed outside of Vim, also check if the file permissions changed. When the file contents didn't change but the permissions did, give a warning.
- Avoid checking too often, otherwise the dialog keeps popping up for a log file that steadily grows.

Mapping **<M-A>** when **'encoding'** is "latin1" and then setting **'encoding'** to "utf-8" causes the first byte of a multibyte to be mapped. Can cause very hard to find problems. Disallow mapping part of a multibyte character.

For ":python" and ":tcl" accept an in-line script. (Johannes Zellner)
Also for ":ruby" and ":perl". (Benoit Cerrina)

Made ":syn include" use **'runtimepath'** when the file name is not a full path.

When **'switchbuf'** contains "split" and the current window is empty, don't split the window.

Unix: Catch SIGPWR to preserve files when the power is about to go down.

Sniff interface: (Anton Leherbauer)

- fixed windows code, esp. the event handling stuff
- adaptations for sniff 4.x (\$SNIFF_DIR4)
- support for adding sniff requests at runtime

Support the notation **<A-x>** as an alias for **<M-x>**. This logical, since the Alt key is used.

":find" accepts a count, which means that the count'th match in **'path'** is used.

":ls" and ":buffers" output shows modified/readonly/modifiable flag. When a

buffer is active show "a" instead of nothing. When a buffer isn't loaded show nothing instead of "-".

Unix install:

- When installing the tools, set absolute paths in tools scripts efm_perl.pl and mve.awk. Avoids that the user has to edit these files.
- Install Icons for KDE when the directories exist and the icons do not exist yet.

Added has("win95"), to be able to distinguish between MS-Windows 95/98/ME and NT/2000/XP in a Vim script.

When a ":cd" command was typed, echo the new current directory. (Dan Sharp)

When using ":winpos" before the GUI window has been opened, remember the values until it is opened.

In the ":version" output, add "/dyn" for features that are dynamically loaded. This indicates the feature may not always work.

On Windows NT it is possible that a directory is read-only, but a file can be deleted. When making a backup by renaming the file and 'backupdir' doesn't use the current directory, this causes the original file to be deleted, without the possibility to create a new file. Give an extra error message then to warn to user about this.

Made **CTRL-R CTRL-O** at the command line work like **CTRL-R CTRL-R**, so that it's consistent with Insert mode.

COMPILE TIME CHANGES

compile-changes-6

All generated files have been moved out of the "src" directory. This makes it easy to see which files are not edited by hand. The files generated by configure are now in the "src/auto" directory. For Unix, compiled object files go in the objects directory.

The source archive was over the 1.4M floppy limit. The archives are now split up into two runtime and two source archives. Also provide a bzip2 compressed archive that contains all the sources and runtime files.

Added "reconfig" as a target for make. Useful when changing some of the arguments that require flushing the cache, such as switching from GTK to Motif. Adjusted the meaning of GUI_INC_LOC and GUI_LIB_LOC to be consistent over different GUIs.

Added src/README.txt to give an overview of the main parts of the source code.

The Unix Makefile now fully supports using \$(DESTDIR) to install to a specific location. Replaces the manual setting of *ENDLOC variables.

Added the possibility for a maintainer of a binary version to include his e-mail address with the --with-compiledby configure argument.

Included features are now grouped in "tiny", "small", "normal", "big" and "huge". This replaces "min-features" and "max-features". Using "tiny" disables multiple windows for a really small Vim.

For the tiny version or when FEAT_WINDOWS is not defined: Firstwin and lastwin are equal to curwin and don't use w_next and w_prev.

Added the +listcmds feature. Can be used to compile without the Vim commands that manipulate the buffer list and argument list (the buffer list itself is still there, can't do without it).

Added the +vreplace feature. It is disabled in the "small" version to avoid that the 16 bit DOS version runs out of memory.

Removed GTK+ support for versions older than 1.1.16.

The configure checks for using PTYs have been improved. Code taken from a recent version of screen.

Added configure options to install Vim, Ex and View under another name (e.g., vim6, ex6 and view6).

Added "--with-global-runtime" configure argument. Allows specifying the global directory used in the 'runtimepath' default.

Made enabling the SNIFF+ interface possible with a configure argument.

Configure now always checks /usr/local/lib for libraries and /usr/local/include for include files. Helps finding the stuff for iconv() and gettext().

Moved the command line history stuff into the +cmdline_hist feature, to exclude the command line history from the tiny version.

MS-Windows: Moved common functions from Win16 and Win32 to os_mswin.c. Avoids having to change two files for one problem. (Vince Negri)

Moved common code from gui_w16.c and gui_w32.c to gui_w48.c (Vince Negri)

The jumplist is now a separate feature. It is disabled for the "small" version (16 bit MS-DOS).

Renamed all types ending in _t to end in _T. Avoids potential problems with system types.

Added a configure check for X11 header files that implicitly define the return type to int. (Steve Wall)

"make doslang" in the top directory makes an archive with the menu and .mo files for Windows. This uses the files generated on Unix, these should work on MS-Windows as well.

Merged a large part of os_vms.c with os_unix.c. The code was duplicated in the past which made maintenance more work. (Zoltan Arpadffy)

Updated the Borland C version 5 Makefile: (Dan Sharp)

- Fixed the Perl build
- Added python and tcl builds
- Added dynamic perl and dynamic python builds
- Added uninstal.exe build
- Use "yes" and "no" for the options, like in Make_mvc.mak.

Win32: Merged Make_gvc.mak and Make_ovc.mak into one file: Make_ivc.mak. It's much smaller, many unnecessary text has been removed. (Walter Briscoe)
Added Make_dvc.mak to be able to debug exe generated with Make_mvc.mak in MS-Devstudio. (Walter Briscoe)

MS-Windows: The big gvim.exe, which includes OLE, now also includes dynamically loaded Tcl, Perl and Python. This uses ActivePerl 5.6.1, ActivePython 2.1.1 and ActiveTCL 8.3.3

Added AC_EXEEXT to configure.in, to check if the executable needs ".exe" for Cygwin or MingW. Renamed SUFFIX to EXEEXT in Makefile.

Win32: Load comdlg32.dll delayed for faster startup. Only when using VC 6. (Vipin Aravind)

Win32: When compiling with Borland, allow using IME. (Yasuhiro Matsumoto)

Win32: Added Makefile for Borland 5 to compile gvimext.dll. (Yasuhiro Matsumoto)

BUG FIXES

bug-fixes-6

When checking the command name for "gvim", "ex", etc. ignore case. Required for systems where case is ignored in command names.

Search pattern "[a-c-e]" also matched a 'd' and didn't match a '-'.
[Note: This text is partially obscured by a large black redaction box in the original image.]

When double-clicking in another window, wasn't recognized as double click, because topline is different. Added set_mouse_topline().

The BROKEN_LOCALE check was broken. (Marcin Dalecki)

When "t_Co" is set, the default colors remain the same, thus wrong. Reset the colors after changing "t_Co". (Steve Wall)

When exiting with ":wqall" the messages about writing files could overwrite each other and be lost forever.

When starting Vim with an extremely long file name (around 1024 characters) it would crash. Added a few checks to avoid buffer overflows.

CTRL-E could get stuck in a file with very long lines.

":au syntax<Tab>" expanded event names while it should expand groups starting with "syntax".

When expanding a file name caused an error (e.g., for `<amatch>`) it was produced even when inside an "if 0".

'**cindent**' formatted C comments differently from what the '**comments**' option specified. (Steve Wall)

Default for '**grepprg**' didn't include the file name when only grepping in one file. Now /dev/null has been added for Unix.

Opening the option window twice caused trouble. Now the cursor goes to the existing option window.

":sview" and ":view" didn't set '**readonly**' for an existing buffer. Now do set '**readonly**', unless the buffer is also edited in another window.

GTK GUI: When '**guioptions**' excluded 'g', the more prompt caused the toolbar and menubar to disappear and resize the window (which clears the text). Now always grey-out the toplevel menus to avoid that the menubar changes size or disappears.

When re-using the current buffer for a new buffer, buffer-local variables were not deleted.

GUI: when '**scrolloff**' is 0 dragging the mouse above the window didn't cause a down scroll. Now pass on a mouse event with mouse_row set to -1.

Win32: Console version didn't work on telnet, because of switching between two console screens. Now use one console screen and save/restore the contents when needed. (Craig Barkhouse)

When reading a file the magic number for encryption was included in the file length. (Antonio Colombo)

The quickfix window contained leading whitespace and NULs for multi-line messages. (David Harrison)

When using cscope, redundant tags were removed. This caused a numbering problem, because they were all listed. Don't remove redundant cscope tags. (David Bustos).

Cscope: Test for which matches are in the current buffer sometimes failed, causing a jump to another match than selected. (David Bustos)

Win32: Buffer overflow when adding a charset name in a font.

'**titlestring**' and '**iconstring**' were evaluating an expression in the current context, which could be a user function, which is a problem for local variables vs global variables.

Win32 GUI: Mapping `<M-F>` didn't work. Now handle SHIFT and CTRL in `_OnSysChar()`.

Win32 GUI: (on no file), `:vs<CR>:q<CR>` left a trail of pixels down the middle.

Could also happen for the ruler. `screen_puts()` didn't clear the right char in `ScreenLines[]` for the bold trick.

Win32: `":%!sort|uniq"` didn't work, because the input file name touches the `"|"`. Insert a space before the `"|"`.

OS/2: Expanding wildcards included non-existing files. Caused `":runtime"` to fail, which caused syntax highlighting to fail.

Pasting a register containing **CTRL-R** on the command line could cause an endless loop that can't be interrupted. Now it can be stopped with **CTRL-C**.

When **'verbose'** is set, a message for file read/write could overwrite the previous message.

When **'verbose'** is set, the header from `":select"` was put after the last message. Now start a new line.

The hit-enter prompt reacted to the response of the `t_RV` string, causing messages at startup to disappear.

When `t_Co` was set to 1, colors were still used. Now only use color when `t_Co > 1`.

Listing functions with `":function"` didn't quit when `'q'` or `':'` was typed at the more prompt.

Use `mkstemp()` instead of `mktemp()` when it's available, avoids a warning for linking on FreeBSD.

When doing Insert mode completion it's possible that `b_sfname` is NULL. Don't give it to `printf()` for the "Scanning" message.

`":set runtimepath-= $VIMRUNTIME"` didn't work, because expansion of wildcards was done after trying to remove the string. Now for `":set opt+=val"` and `":set opt-=val"` the expansion of wildcards is done before adding or removing "val".

Using **CTRL-V** with the `"r"` command with a blockwise Visual selection inserted a **CTRL-V** instead of getting a special character.

Unix: Changed the order of libraries: Put `-lXdmcp` after `-lX11` and `-lSM -lICE` after `-lXdmcp`. Should fix link problem on HP-UX 10.20.

Don't remove the last `"-lm"` from the link line. Vim may link but fail later when the GUI starts.

When the shell returns with an error when trying to expand wildcards, do include the pattern when the `"EW_NOTFOUND"` flag was set.

When expanding wildcards with the shell fails, give a clear error message instead of just `"1 returned"`.

Selecting a Visual block, with the start partly on a Tab, deleting it leaves the cursor too far to the left. Causes `"s"` to work in the wrong position.

Pound sign in `normal.c` caused trouble on some compilers. Use `0xA3` instead.

Warning for changing a read-only file wasn't given when **'insertmode'** was set.

Win32: When **'shellxquote'** is set to a double quote (e.g., using csh), "!:start notepad file" doesn't work. Remove the double quotes added by **'shellxquote'** when using "!:start". (Pavol Juhas)

The "**<f-args>**" argument of ":command" didn't accept Tabs for white space. Also, don't add an empty argument when there are trailing blanks.

":e test\\je" edited "test\je", but ":next test\\je" edited "testje". Backslashes were removed one time too many for ":next".

VMS: "gf" didn't work properly. Use vms_fixfilename() to translate the file name. (Zoltan Arpadffy)

After ":hi Normal ctermbg=black ctermfg=white" and suspending Vim not all characters are redrawn with the right background.

When doing "make test" without +eval or +windows feature, many tests failed. Now have test1 generate a script to copy the correct output, so that a test that doesn't work is skipped.

On FreeBSD the Perl interface added "-lc" to the link command and Python added "-pthread". These two don't work together, because the libc_r library should be used. Removed "-lc" from Perl, it should not be needed. Also: Add "-pthread" to \$LIBS, so that the checks for functions is done with libc_r. Sigaltstack() appears to be missing from libc_r.

The Syntax sub-menus were getting too long, reorganized them and added another level for some languages.

Visual block "r"eplace didn't work well when a Tab is partly included. (Matthias Kramm)

When yanking a Visual block, where some lines end halfway the block, putting the text somewhere else doesn't insert a block. Padd with spaces for missing characters. Added "y_width" to struct yankreg. (Matthias Kramm)

If a substitute string has a multibyte character after a backslash only the first byte of it was skipped. (Muraoka Taro)

Win32: Numeric keypad keys were missing from the builtin termcap entry.

When a file was read-only ":wa!" didn't force it to be written. (Vince Negri)

Amiga: A file name starting with a colon was considered absolute but it isn't. Amiga: ":pwd" added a slash when in the root of a drive.

Don't let **'tymouse'** default to "dec" when compiled with dec mouse support. It breaks the gpm mouse (Linux console).

The prototypes for the Perl interface didn't work for threaded Perl. Added a sed command to remove the prototypes from proto/if_perl.pro and added them

manually to `if_perl.xs`.

When `":w!"` resets the `'readonly'` option the title and status lines were not updated.

`":args"` showed the current file when the argument list was empty. Made this work like Vi: display nothing.

`"99:<C-U>echo v:count"` echoed "99" in Normal mode, but 0 in Visual mode. Don't set `v:count` when executing a stuffed command.

Amiga: Got a requester for `"home:"` because it's in the default runtime path. Don't bring up a requester when searching for a file in `'path'`, sourcing the `.vimrc` file or using `":runtime"`.

Win16 and Win32: Considered a file `"\path\file"` absolute. Can cause the same file to appear as two different buffers.

Win32: Renaming a file to an empty string crashed Vim. Happened when using `explorer.vim` and hitting ESC at the rename prompt.

Win32: `strftime()` crashed when called with a `"-1"` value for the time.

Win32 with Borland compiler: `mch_FullName()` didn't work, caused tag file not to be found.

Cscope sometimes jumped to the wrong tag. (David Bustos)

OS/2: Could not find the tags file. `mch_expand_wildcards()` added another slash to a directory name.

When using `">>"` the ``]` mark was not in the last column.

When Vim was compiled without menu support, `filetype.vim` was still trying to source the `menu.vim` script. (Rafael Garcia-Suarez)

`":ptag"` added an item to the tag stack.

Win32 IME: `"gr"` didn't use IME mode.

In the `"vim --help"` message the term "options" was used for arguments. That's confusing, call them "arguments".

When there are two windows, and a `BufUnload` autocommand for closing window #1 closed window #2, Vim would crash.

When there is a preview window and only one other window, `":q"` wouldn't exit.

In Insert mode, when cancelling a digraph with ESC, the `'?` wasn't removed.

On Unix `glob(".*")` returned `."` and `.."`, on Windows it didn't. On Windows `glob("*")` also returned files starting with a dot. Made this work like Unix on all systems.

Win32: Removed old code to open a console. Vimrun is now used and works fine.

Compute the room needed by the intro message accurately, so that it also fits on a 25 line console. (Craig Barkhouse)

":ptnext" was broken. Now remember the last tag used in the preview window separately from the tagstack.

Didn't check for "-display" being the last argument. (Wichert Akkerman)

GTK GUI: When starting "gvim" under some conditions there would be an X error. Don't replace the error handler when creating the xterm clipboard. (Wichert Akkerman)

Adding a space after a help tag caused the tag not to be found. E.g., ":he autoindent ".

Was trying to expand a URL into a full path name. On Windows this resulted in the current directory to be prepended to the URL. Added vim_isAbsName() and vim_FullName() to avoid that various machine specific functions do it differently.

":n *.c" ":cd .." ":n" didn't use the original directory of the file. Vi only does it for the current file (looks like a bug). Now remember the buffer used for the entry in the argument list and use its name (adjusted when doing ":cd"), unless it's deleted.

When inserting a special key as its name ("" as four characters) after moving around in Insert mode, undo didn't work properly.

Motif GUI: When using the right mouse button, for some people gvim froze for a couple of seconds (Motif 1.2?). This doesn't happen when there is no Popup menu. Solved by only creating a popup menu when 'mousemodel' is "popup" or "popup_setpos". (David Harrison)

Motif: When adding many menu items, the "Help" menu disappeared but the menubar didn't wrap. Now manually set the menubar height.

When using <BS> in Insert mode to remove a line break, or using "J" to join lines, the cursor could end up halfway a multibyte character. (Muraoka Taro)

Removed defining SVR4 in configure. It causes problems for some X header files and doesn't appear to be used anywhere.

When 'wildignore' is used, 'ignorecase' for a tag match was not working.

When 'wildignore' contains "*~" it was impossible to edit a file ending in a "~". Now don't recognize a file ending in "~" as containing wildcards.

Disabled the mouse code for OS/2. It was not really used.

":mksession" always used the full path name for a buffer, also when the short name could be used.

":mkvimrc" and ":mksession" didn't save 'wildchar' and 'pastetoggle' in such a

way that they would be restored. Now use the key name if possible, this is portable.

After recovering a file and abandoning it, an ":edit" command didn't give the ATTENTION prompt again. Would be useful to be able to delete the file in an easy way. Reset the BF_RECOVERED flag when unloading the buffer.

histdel() could match or ignore case, depending on what happened before it. Now always match case.

When a window size was specified when splitting a window, it would still get the size from 'winheight' or 'winwidth' if it's larger.

When using "append" or "insert" inside a function definition, a line starting with "function" or "endfunction" caused confusion. Now recognize the commands and skip lines until a ".".

At the end of any function or sourced file need_wait_return could be reset, causing messages to disappear when redrawing.

When in a while loop the line number for error messages stayed fixed. Now the line number is remembered in the while loop.

"cd c:/" didn't work on MS-DOS. mch_isdir() removed a trailing slash.

MS-Windows: getftime() didn't work when a directory had a trailing slash or backslash. Didn't show the time in the explorer because of this.

When doing wildcard completion, a directory "a/" sorted after "a-b". Now recognize path separators when sorting files.

Non-Unix systems: When editing "c:/dir/../file" and "c:/file" they were created as different buffers, although it's the same file. Expand to a full file name also when an absolute name contains "..".

"g&" didn't repeat the last substitute properly.

When 'clipboard' was set to "unnamed", a "Y" command would not write to "0. Now make a copy of register 0 to the clipboard register.

When the search pattern matches in many ways, it could not always be interrupted with a **CTRL-C**. And **CTRL-C** would have to be hit once for every line when 'hlsearch' is on.

When 'incsearch' is on and interrupting the search for a match, don't abandon the command line.

When turning a directory name into a full path, e.g., with fnamemodify(), sometimes a slash was added. Make this consistent: Don't add a slash.

When a file name contains a "!", using it in a shell command will cause trouble: "!:cat %". Escape the "!" to avoid that. Escape it another time when 'shell' contains "sh".

Completing a file name that has a tail that starts with a "~" didn't work:

":e view/~<Tab>".

Using a ":command" argument that contains `< and >` but not for a special argument was not skipped properly.

The DOS install program: On Win2000 the check for a vim.exe or gvim.exe in \$PATH didn't work, it always found it in the current directory. Rename the vim.exe in the current dir to avoid this. (Walter Briscoe)

In the MS-DOS/Windows install program, use %VIM% instead of an absolute path, so that moving Vim requires only one change in the batch file.

Mac: mch_FullName() changed the "fname" argument and didn't always initialize the buffer.

MS-DOS: mch_FullName() didn't fix forward/backward slashes in an absolute file name.

"echo expand("%:~p:h)" with an empty file name removed one directory name on MS-DOS. For Unix, when the file name is a directory, the directory name was removed. Now make it consistent: "%:~p" adds a path separator for all systems, but no path separator is added in other situations.

Unix: When checking for a **CTRL-C** (could happen any time) and there is an X event (e.g., clipboard updated) and there is typeahead, Vim would hang until a character was typed.

MS-DOS, MS-Windows and Amiga: expanding "\$ENV/foo" when \$ENV ends in a colon, had the slash removed.

":he \^=" gave an error for using _. ":he ^==" didn't find tag :set^=. Even "he :set^=" didn't find it.

A tags file name "D:/tags" was used as file "tags" in "D:". That doesn't work when the current path for D: isn't the root of the drive.

Removed calls to XtInitializeWidgetClass(), they shouldn't be necessary.

When using a dtterm or various other color terminals, and the Normal group has been set to use a different background color, the background wouldn't always be displayed with that color. Added check for "ut" termcap entry: If it's missing, clearing the screen won't give us the current background color. Need to draw each character instead. Vim now also works when the "cl" (clear screen) termcap entry is missing.

When repeating a "/" search command with a line offset, the "n" did use the offset but didn't make the motion linewise. Made "d/pat/+2" and "dn" do the same.

Win32: Trying to use ":tearoff" for a menu that doesn't exist caused a crash.

OpenPTY() didn't work on Sequent. Add a configure check for getpseudotty().

C-indenting: Indented a line starting with ")" with the matching "(", but not

a line starting with "x)" looks strange. Also compute the indent for aligning with items inside the () and use the lowest indent.

MS-DOS and Windows: ":n *.vim" also matched files ending in "~".
Moved mch_expandpath() from os_win16.c and os_msdos.c to misc1.c, they are equal.

Macintosh: (Dany St-Amant)

- In Vi-compatible mode didn't read files with CR line separators.
- Fixed a bug in the handling of Activate/Deactivate Event
- Fixed a bug in gui_mch_dialog (using wrong pointer)

Multibyte GDK XIM: While composing a multibyte-word, if user presses a mouse button, then the word is removed. It should remain and composing end.
(Sung-Hyun Nam)

MS-DOS, MS-Windows and OS/2: When reading from stdin, automatic CR-LF conversion by the C library got in the way of detecting a "dos" **'fileformat'**.

When **'smartcase'** is set, patterns with "\S" would also make **'ignorecase'** reset.

When clicking the mouse in a column larger than 222, it moved to the first column. Can't encode a larger number in a character. Now limit the number to 222, don't jump back to the first column.

GUI: In some versions CSI would cause trouble, either when typed directly or when part of a multibyte sequence.

When using multibyte characters in a ":normal" command, a trailing byte that is CSI or K_SPECIAL caused problems.

Wildmenu didn't handle multibyte characters.

":sleep 10" could not be interrupted on Windows, while "gs" could. Made them both work the same.

Unix: When waiting for a character is interrupted by an X-windows event (e.g., to obtain the contents of the selection), the wait time would not be honored. A message could be overwritten quickly. Now compute the remaining waiting time.

Windows: Completing "\\share\c\$\S" inserted a backslash before the \$ and then the name is invalid. Don't insert the backslash.

When doing an auto-write before ":make", IObuff was overwritten and the wrong text displayed later.

On the Mac the directories "c:/tmp" and "c:/temp" were used in the defaults for **'backupdir'** and **'directory'**, they don't exist.

The check for a new file not to be on an MS-DOS filesystem created the file temporarily, which can be slow. Don't do this if there is another check for the swap file being on an MS-DOS filesystem.

Don't give the "Changing a readonly file" warning when reading from stdin.

When using the "Save As" menu entry and not entering a file name, would get an error message for the trailing ":edit #". Now only do that when the alternate file name was changed.

When Vim owns the X11 selection and is being suspended, an application that tries to use the selection hangs. When Vim continues it could no longer obtain the selection. Now give up the selection when suspending.

option.h and globals.h were included in some files, while they were already included in vim.h. Moved the definition of EXTERN to vim.h to avoid doing it twice.

When repeating an operator that used a search pattern and the search pattern contained characters that have a special meaning on the cmdline (e.g., **CTRL-U**) it didn't work.

Fixed various problems with using K_SPECIAL (0x80) and CSI (0x9b) as a byte in a (multibyte) character. For example, the "r" command could not be repeated.

The DOS/Windows install program didn't always work from a directory with a long filename, because \$VIM and the executable name would not have the same path.

Multi-byte:

- Using an any-but character range [^x] in a regexp didn't work for UTF-8. (Muraoka Taro)
- When backspacing over inserted characters in Replace mode multibyte characters were not handled correctly. (Muraoka Taro)
- Search commands "#" and "*" didn't work with multibyte characters. (Muraoka Taro)
- Word completion in Insert mode didn't work with multibyte characters. (Muraoka Taro)
- Athena/Motif GUI: when 'linespace' is non-zero the cursor would be drawn too wide (number of bytes instead of cell width).
- When changing 'encoding' to "euc-jp" and inserting a character Vim would crash.
- For euc-jp characters positioning the cursor would sometimes be wrong. Also, with two characters with 0x8e leading byte only the first one would be displayed.
- When using DYNAMIC_ICONV on Win32 conversion might fail because of using the wrong error number. (Muraoka Taro)
- Using Alt-x in the GUI while 'encoding' was set to "utf-8" didn't produce the right character.
- When using Visual block selection and only the left half of a double-wide character is selected, the highlighting continued to the end of the line.
- Visual-block delete didn't work properly when deleting the right half of a double-wide character.
- Overstrike mode for the cmdline replaced only the first byte of a multibyte character.
- The cursor in Replace mode (also in the cmdline) was too small on a double-wide character.

- When a multibyte character contained a 0x80 byte, it didn't work (was using a CSI byte instead). (Muraoka Taro)
- Wordwise selection with the mouse didn't work.
- Yanking a modeless selection of multibyte characters didn't work.
- When **'selection'** is "exclusive", selecting a word that ends in a multibyte character used wrong highlighting for the following character.

Win32 with Make_mvc.mak: Didn't compile for debugging. (Craig Barkhouse)

Win32 GUI: When "vimrun.exe" is used to execute an external command, don't give a message box with the return value, it was already printed by vimrun. Also avoid printing the return value of the shell when ":silent!" is used.

Win32: selecting a lot of text and using the "find/replace" dialog caused a crash.

X11 GUI: When typing a character with the 8th bit set and the Meta/Alt modifier, the modifier was removed without changing the character.

Truncating a message to make it fit on the command line, using "..." for the middle, didn't always compute the space correctly.

Could not imap <C-@>. Now it works like <Nul>.

VMS:

- Fixed a few things for VAXC. os_vms_fix.com had some strange **CTRL-M** characters. (Zoltan Arpadffy and John W. Hamill)
- Added VMS-specific defaults for the **'isfname'** and **'isprint'** options. (Zoltan Arpadffy)
- Removed os_vms_osdef.h, it's no longer used.

The gzip plugin used a ":normal" command, this doesn't work when dropping a compressed file on Vim.

In very rare situations a binary search for a tag would fail, because an uninitialized value happens to be half the size of the tag file. (Narendran)

When using BufEnter and BufLeave autocommands to enable/disable a menu, it wasn't updated right away.

When doing a replace with the "c"onfirm flag, the cursor was positioned after the ruler, instead of after the question. With a long replacement string the screen could scroll up and cause a "more" prompt. Now the message is truncated to make it fit.

Motif: The autoconf check for the Xp library didn't work.

When **'verbose'** is set to list lines of a sourced file, defining a function would reset the counter used for the "more" prompt.

In the Win32 find/replace dialog, a '/' character caused problems. Escape it with a backslash.

Starting a shell with ":sh" was different from starting a shell for **CTRL-Z**

when suspending doesn't work. They now work the same way.

Jumping to a file mark while in a changed buffer gave a "mark not set" error.

":execute histget("cmd")" causes an endless loop and crashed Vim. Now catch all commands that cause too much recursiveness.

Removed "Failed to open input method" error message, too many people got this when they didn't want to use a XIM.

GUI: When compiled without the +windows feature, the scrollbar would start below line one.

Removed the trick with redefining character class functions from regexp.c.

Win32 GUI: Find dialog gives focus back to main window, when typing a character mouse pointer is blanked, it didn't reappear when moving it in the dialog window. (Vince Negri)

When recording and typing a **CTRL-C**, no character was recorded. When in Insert mode or cancelling half a command, playing back the recorded sequence wouldn't work. Now record the **CTRL-C**.

When the GUI was started, mouse codes for DEC and netterm were still checked for.

GUI: When scrolling and **'writedelay'** is non-zero, the character under the cursor was displayed in the wrong position (one line above/below with **CTRL-E/CTRL-Y**).

A ":normal" command would reset the **'scrollbind'** info. Causes problems when using a ":normal" command in an autocommand for opening a file.

Windows GUI: a point size with a dot, like "7.5", wasn't recognized. (Muraoka Taro)

When **'scrollbind'** wasn't set would still remember the current position, wasting time.

GTK: Crash when **'shell'** doesn't exist and doing":!ls". Use `_exit()` instead of `exit()` when the child couldn't execute the shell.

Multi-byte:

- GUI with double-byte encoding: a mouse click in left half of double-wide character put the cursor in previous char.
- Using double-byte encoding and **'selection'** is "exclusive": "vey" and "^Vey" included the character after the word.
- When using a double-byte encoding and there is a lead byte at the end of the line, the preceding line would be displayed. "ga" also showed wrong info.
- "gf" didn't include multibyte characters before the cursor properly.
(Muraoka Taro)

GUI: The cursor was sometimes not removed when scrolling. Changed the policy from redrawing the cursor after each call to `gui_write()` to only update it at

the end of `update_screen()` or when setting the cursor position. Also only update the scrollbars at the end of `update_screen()`, that's the only place where the window text may have been scrolled.

Formatting `/*<Tab>long text`", produced `* <Tab>`" in the next line. Now remove the space before the Tab.

Formatting `/*<Tab> long text`", produced `* <Tab> long text`" in the next line. Now keep the space after the Tab.

In some places non-ASCII alphabetical characters were accepted, which could cause problems. For example, `:"X`" (X being such a character).

When a pattern matches the end of the line, the last character in the line was highlighted for `'hlsearch'`. That looks wrong for `"/\%3c"`. Now highlight the character just after the line.

Motif: If a dialog was closed by clicking on the "X" of the window frame Vim would no longer respond.

When using **CTRL-X** or **CTRL-A** on a number with many leading zeros, Vim would crash. (Matsumoto)

When `'insertmode'` is set, the mapping in `mshwin.vim` for **CTRL-V** didn't work in Select mode. Insert mode wasn't restarted after overwriting the text. Now allow nesting Insert mode with insert and change commands. **CTRL-O** `cwfoo<Esc>` now also works.

Clicking with the right mouse button in another window started Visual mode, but used the start position of the current window. Caused `ml_get` errors when the line number was invalid. Now stay in the same window.

When `'selection'` is "exclusive", `"gv"` sometimes selected one character fewer.

When `'comments'` contains more than one start/middle/end triplet, the optional flags could be mixed up. Also didn't align the end with the middle part.

Double-right-click in Visual mode didn't update the shown mode.

When the Normal group has a font name, it was never used when starting up. Now use it when `'guifont'` and `'guifontset'` are empty. Setting a font name to a highlight group before the GUI was started didn't work.

`"make test"` didn't use the name of the generated Vim executable.

`'cindent'` problems:

- Aligned with an "else" inside a do-while loop for a line below that loop. (Meikel Brandmeyer)
- A line before a function would be indented even when terminated with a semicolon. (Meikel Brandmeyer)
- `'cindent'` gave too much indent to a line after a `"};"` that ends an array init.
- Support declaration lines ending in `","` and `"\"`. (Meikel Brandmeyer)
- A case statement inside a do-while loop was used for indenting a line after

- the do-while loop. (Meikel Brandmeyer)
- When skipping a string in a line with one double quote it could continue in the previous line. (Meikel Brandmeyer)

When **'list'** is set, **'hlsearch'** didn't highlight a match at the end of the line. Now highlight the '\$'.

The Paste menu item in the menu bar, the popup menu and the toolbar were all different. Now made them all equal to how it was done in mswin.vim.

st_dev can be smaller than "unsigned". The compiler may give an overflow warning. Added a configure check for dev_t.

Athena: closing a confirm() dialog killed Vim.

Various typos in the documentation. (Matt Dunford)

Python interface: The definition of _DEBUG could cause trouble, undefine it. The error message for not being able to load the shared library wasn't translated. (Muraoka Taro)

Mac: (Dany St-Amant and Axel Kielhorn)

- Several fixes.
- Vim was eating 80% of the CPU time.
- The project os_mac.pbxproj didn't work, Moved it to a subdirectory.
- Made the menu priority work for the menubar.
- Fixed a problem with dragging the scrollbar.
- Cleaned up the various #ifdefs.

Unix: When catching a deadly signal and we keep getting one use _exit() to exit in a quick and dirty way.

Athena menu ordering didn't work correctly. (David Harrison)

A ":make" or ":grep" command with a long argument could cause a crash.

Doing ":new file" and using "Quit" for the ATTENTION dialog still opened a new window.

GTK: When starting the GUI and there is an error in the .vimrc file, don't present the wait-return prompt, since the message was given in the terminal.

When there was an error in a .vimrc file the terminal where gvim was started could be cleared. Set msg_row in main.c before writing any messages.

GTK and X11 GUI: When trying to read characters from the user (e.g. with input()) before the Vim window was opened caused Vim to hang when it was started from the desktop.

OS/390 uses 31 bit pointers. That broke some computations with MAX_COL. Reduce MAX_COL by one bit for OS/390. (Ralf Schandl)

When defining a function and it already exists, Vim didn't say it existed until after typing it. Now do this right away when typing it.

The message remembered for displaying later (`keep_msg`) was sometimes pointing into a generic buffer, which might be changed by the time the message is displayed. Now make a copy of the message.

When using multibyte characters in a menu and a trailing byte is a backslash, the menu would not be created correctly. (Muraoka Taro)
Using a multibyte character in the substitute string where a trail byte is a backslash didn't work. (Muraoka Taro)

When setting `"t_Co"` in a vimrc file, then setting it automatically from an xterm termresponse and then setting it again manually caused a crash.

When getting the value of a string option that is not supported, the number zero was returned. This breaks a check like `"&enc == "asdf"`. Now an empty string is returned for string options.

Crashed when starting the GTK GUI while using `'notitle'` in the vimrc, setting `'title'` in the gvimrc and starting the GUI with `":gui"`. Closed the connection to the X server accidentally.

Had to hit return after selecting an entry for `":ts"`.

The message from `":cn"` message was sometimes cleared. Now display it after redrawing if it doesn't cause a scroll (truncated when necessary).

`hangulin.c` didn't compile when the GUI was disabled. Disable it when it won't work.

When setting a termcap option like `"t_CO"`, the value could be displayed as being for a normal key with a modifier, like `"<M-=>"`.

When expanding the argument list, entries which are a directory name did not get included. This stopped `"vim c:/"` from opening the file explorer.

`":syn match sd "^" nextgroup=asdf"` skipped the first column and matched the nextgroup in the second column.

GUI: When `'lazyredraw'` is set, `'showmatch'` didn't work. Required flushing the output.

Don't define the `<NetMouse>` termcode in an xterm, reduces the problem when someone types `<Esc>` } in Insert mode.

Made `slash_adjust()` work correctly for multibyte characters. (Yasuhiro Matsumoto)

Using a filename in Big5 encoding for autocommands didn't work (backslash in trailbyte). (Yasuhiro Matsumoto)

DOS and Windows: Expanding `*.vim` also matched `file.vimfoo`. Expand path like Unix to avoid problems with Windows dir functions. Merged the DOS and Win32 functions.

Win32: `GvimExt` could not edit more than a few files at once, the length of the

argument was fixed.

"ls -l * | xargs vim" worked, but the input was in cooked mode. Now switch to raw mode when needed. Use dup() to copy the stderr file descriptor to stdin to make shell commands work. No longer requires an external program to do this.

When using ":filetype off", ftplugin and indent usage would be switched off at the same time. Don't do this, setting '**filetype**' manually can still use them.

GUI: When writing a double-byte character, it could be split up in two calls to gui_write(), which doesn't work. Now flush before the output buffer becomes full.

When '**laststatus**' is set and '**cmdheight**' is two or bigger, the intro message would be written over the status line.

The ":intro" command didn't work when there wasn't enough room.

Configuring for Ruby failed with a recent version of Ruby. (Akinori Musha)

Athena: When deleting the directory in which Vim was started, using the file browser made Vim exit. Removed the use of XtAppError().

When using autoconf 2.50, UNIX was not defined. Moved the comment for "#undef UNIX" to a separate line.

Win32: Disabled _OnWindowPosChanging() to make maximize work better.

Win32: Compiling with VC 4.0 didn't work. (Walter Briscoe)

Athena:

- Finally fixed the problems with deleting a menu. (David Harrison)
- Athena: When closing the confirm() dialog, worked like OK was pressed, instead of Cancel.

The file explorer didn't work in compatible mode, because of line continuation.

Didn't give an error message for ":digraph a".

When using Ex mode in the GUI and typing a special key, <BS> didn't delete it correctly. Now display '?' for a special key.

When an operator is pending, clicking in another window made it apply to that window, even though the line numbers could be beyond the end of the buffer.

When a function call doesn't have a terminating ")" Vim could crash.

Perl interface: could crash on exit with perl 5.6.1. (Anduin Withers)

Using %P in '**errorformat**' wasn't handled correctly. (Tomas Zellerin)

Using a syntax cluster that includes itself made Vim crash.

GUI: With **'ls'** set to 2, dragging the status line all the way up, then making the Vim window smaller: Could not the drag status line anymore.

"vim -c startinsert! file" placed cursor on last char of a line, instead of after it. A `":set"` command in the buffer menu set `w_set_curswant`. Now don't do this when `w_curswant` is `MAXCOL`.

Win32: When the gvim window was maximized and selecting another font, the window would no longer fill the screen.

The line with **'pastetoggle'** in `":options"` didn't show the right value when it is a special key. Hitting `<CR>` didn't work either.

Formatting text, resulting in a % landing in the first line, repeated the % in the following lines, like it's the start of a comment.

GTK: When adding a toolbar item while gvim is already running, it wasn't possible to use the tooltip. Now it works by adding the tooltip first.

The output of `"g CTRL-G"` mentioned "Char" but it's actually bytes.

Searching for the end of a oneline region didn't work correctly when there is an offset for the highlighting.

Syntax highlighting: When synchronizing on C-comments, `/**/` was seen as the start of a comment.

Win32: Without scrollbars present, the MS mouse scroll wheel didn't work. Also handle the scrollbars when they are not visible.

Motif: When there is no right scrollbar, the bottom scrollbar would still leave room for it. (Marcin Dalecki)

When changing **'guicursor'** and the value is invalid, some of the effects would still take place. Now first check for errors and only make the new value effective when it's OK.

Using "A" In Visual block mode, appending to lines that don't extend into the block, padding was wrong.

When pasting a block of text, a character that occupies more than one screen column could be deleted and spaces inserted instead. Now only do that with a tab.

Fixed conversion of documentation to HTML using Perl. (Dan Sharp)

Give an error message when a menu name starts with a dot.

Avoid a hang when executing a shell from the GUI on HP-UX by pushing "ptem" even when `sys/ptem.h` isn't present.

When creating the temp directory, make sure `umask` is `077`, otherwise the directory is not accessible when it was set to `0177`.

Unix: When resizing the window and a redraw is a bit slow, could get a window resize event while redrawing, resulting in a messed up window. Any input (e.g., a mouse click) would redraw.

The "%B" item in the status line became zero in Insert mode (that's normal) for another than the current window.

The menu entries to convert to xxd and back didn't work in Insert mode.

When ":vglobal" didn't find a line where the pattern doesn't match, the error message would be the wrong way around.

When ignoring a multi-line error message with "%-A", the continuation lines would be used anyway. (Servatius Brandt)

"grx" on a double-wide character inserted "x", instead of replacing the character with "x ". "gR" on <xx> ('display' set the "uhex") didn't replace at all. When doing "gRxx" on a control character the first "x" would be inserted, breaking the alignment.

Added "0)" to 'cinkeys', so that when typing a) it is put in the same place as where "==" would put it.

Win32: When maximized, adding/removing toolbar didn't resize the text area.

When using <C-RightMouse> a count was discarded.

When typing CTRL-V and <RightMouse> in the command line, would insert <LeftMouse>.

Using "vis" or "vas" when 'selection' is exclusive didn't include the last character.

When adding to an option like 'grepprg', leading space would be lost. Don't expand environment variables when there is no comma separating the items.

GUI: When using a bold-italic font, would still use the bold trick and underlining.

Motif: The default button didn't work in dialogs, the first one was always used. Had to give input focus to the default button.

When using CTRL-T to jump within the same file, the ' mark wasn't set.

Undo wasn't Vi compatible when using the 'c' flag for ":s". Now it undoes the whole ":s" command instead of each confirmed replacement.

The Buffers menu, when torn-off, disappeared when being refreshed. Add a dummy item to avoid this.

Removed calling msg_start() in main(), it should not be needed.

vim_strpbrk() did not support multibyte characters. (Muraoka Taro)

The Amiga version didn't compile, the code was too big for relative jumps.
Moved a few files from ex_docmd.c to ex_cmds2.c

When evaluating the "=" register resulted in the "=" register being changed, Vim would crash.

When doing ":view file" and it fails, the current buffer was made read-only.

Motif: For some people the separators in the toolbar disappeared when resizing the Vim window. (Marcin Dalecki)

Win32 GUI: when setting '**lines**' to a huge number, would not compute the available space correctly. Was counting the menu height twice.

Conversion of the docs to HTML didn't handle the line with the +quickfix tag correctly. (Antonio Colombo)

Win32: fname_case() didn't handle multibyte characters correctly. (Yasuhiro Matsumoto)

The Cygwin version had trouble with fchdir(). Don't use that function for Cygwin.

The generic check in scripts.vim for "conf" syntax was done before some checks in filetype.vim, resulting in "conf" syntax too often.

Dos32: Typing lagged behind. Would wait for one biostick when checking if a character is available.

GTK: When setting '**columns**' while starting up "gvim", would set the width of the terminal it was started in.

When using ESC in Insert mode, an autoindent that wraps to the next line caused the cursor to move to the end of the line temporarily. When the character before the cursor was a double-wide multibyte character the cursor would be on the right half, which causes problems with some terminals.

Didn't handle multibyte characters correctly when expanding a file name. (Yasuhiro Matsumoto)

Win32 GUI: Errors generated before the GUI is decided to start were not reported.

globpath() didn't reserve enough room for concatenated results. (Anduin Withers)

When expanding an option that is very long already, don't do the expansion, it would be truncated to MAXPATHL. (Anduin Withers)

When '**selection**' is "exclusive", using "Fx" in Visual mode only moved until just after the character.

When using IME on the console to enter a file name, the screen may scroll up. Redraw the screen then. (Yasuhiro Matsumoto)

Motif: In the find/replace dialog the "Replace" button didn't work first time, second time it replaced all matches. Removed the use of ":s///c".
GTK: Similar problems with the find/replace dialog, moved the code to a common function.

X11: Use shared GC's for text. (Marcin Dalecki)

"]i" found the match under the cursor, instead of the first one below it. Same for "]I", "] **CTRL-I**", "]d", "]D" and "] **CTRL-D**".

Win16: When maximized and the font is changed, don't change the window size. (Vince Negri)

When '**lbr**' is set, deleting a block of text could leave the cursor in the wrong position.

Win32: When opening a file with the "Edit with Vim" popup menu entry, wildcards would cause trouble. Added the "--literal" argument to avoid expanding file names.

When using "gv", it didn't restore that "\$" was used in Visual block mode.

Win32 GUI: While waiting for a shell command to finish, the window wasn't redrawn at all. (Yasuhiro Matsumoto)

Syntax highlighting: A match that continues on a next line because of a contained region didn't end when that region ended.

The ":s" command didn't allow flags like 'e' and 'i' right after it.

When using ":s" to split a line, marks were moved to the next line. Vi keeps them in the first line.

When using ":n" ":rew", the previous context mark was at the top of the file, while Vi puts it in the same place as the cursor. Made it Vi compatible.

Fixed Vi incompatibility: Text was not put in register 1 when using "c" and "d" with a motion character, when deleting within one line with one of the commands: % () `

Win32 GUI: The tooltip for tear-off items remained when the tear-off item was no longer selected.

GUI: When typing ":" at the more prompt, would return to Normal mode and not redraw the screen.

When starting Vim with an argument "-c g/at/p" the printed lines would overwrite each other.

BeOS: Didn't compile. Configure didn't add the os_beos files, the QNX check removed them. Various changes to os_beos.cc. (Joshua Haberman)
Removed the check for the hardware platform, the BeBox has not been produced for a long time now.

Win32 GUI: don't use a message box when the shell returns an error code, display the message in the Vim window.

Make_mvc.mak always included "/debug" for linking. "GUI=no" argument didn't work. Use "DEBUG=yes" instead of "DEBUG=1" to make it consistent. (Dan Sharp)

When a line in the tags file ended in ";" (no TAB following) the command would not be recognized as a search command.

X11: The inputMethod resource never worked. Don't use the "none" input method for SGI, it apparently makes the first character in Input method dropped.

Fixed incorrect tests in os_mac.h. (Axel Kielhorn)

Win32 console: When the console where Vim runs in is closed, Vim could hang in trying to restore the window icon. (Yasuhiro Matsumoto)

When using ":3call func()" or ":3,3call func()" the line number was ignored.

When '**showbreak**' and '**linebreak**' were both set, Visual highlighting sometimes continued until the end of the line.

GTK GUI: Tearoff items were added even when '**guioptions**' didn't contain 't' when starting up.

MS-Windows: When the current directory includes a "~", searching files with "gf" or ":find" didn't work. A "\$" in the directory had the same problem. Added mch_has_exp_wildcard() functions.

When reducing the Vim window height while starting up, would get an out-of-memory error message.

When editing a very long search pattern, '**incsearch**' caused the redraw of the command line to fail.

Motif GUI: On some systems the "Help" menu would not be on the far right, as it should be. On some other systems (esp. IRIX) the command line would not completely show. Solution is to only resize the menubar for Lesstif.

Using "%" in a line that contains "\\\" twice didn't take care of the quotes properly. Now make a difference between \" and \\\".

For non-Unix systems a dummy file is created when finding a swap name to detect a 8.3 filesystem. When there is an existing swap file, would get a warning for the file being created outside of Vim. Also, when closing the Vim window the file would remain.

Motif: The menu height was always computed, using a "-menuheight" argument was setting the room for the command line. Now make clear the argument is not supported.

For some (EBCDIC) systems, POUND was equal to '#'. Added an #if for that to avoid a duplicate case in a switch.

The GUI may have problems when forking. Always call `_exit()` instead of `exit()` in the parent, the child will call `exit()`.

Win32 GUI: Accented characters were often wrong in dialogs and tearoff menus. Now use `CP_ACP` instead of `CP_OEMCP`. (Vince Negri)

When displaying text with syntax highlighting causes an error (e.g., running out of stack) the syntax highlighting is disabled to avoid further messages.

When a command in a `.vimrc` or `.gvimrc` causes an ATTENTION prompt, and Vim was started from the desktop (no place to display messages) it would hang. Now open the GUI window early to be able to display the messages and pop up the dialog.

`"r<CR>`" on a multibyte character deleted only the first byte of the character. `"3r<CR>`" deleted three bytes instead of three characters.

When interrupting reading a file, Vi considers the buffer modified. Added the `'i'` flag in `'coptions'` flag for this (we don't want it modified to be able to do `":q"`).

When using an item in `'guicursor'` that starts with a colon, Vim would get stuck or crash.

When putting a file mark in a help file and later jumping back to it, the options would not be set. Extended the modeline in all help files to make this work better.

When a modeline contained `:::` the local option values would be printed. Now ignore it.

Some help files did not use a 8.3 names, which causes problems when using MS-DOS unzip. Renamed `"multibyte.txt"` to `"mbyte.txt"`, `"rightleft.txt"` to `"rileft.txt"`, `"tagsearch.txt"` to `"tagsrch.txt"`, `"os_riscos.txt"` to `"os_risc.txt"`.

When Visual mode is blockwise, using `"iw"` or `"aw"` made it characterwise. That doesn't seem right, only do this when in linewise mode. But then do it always, not only when start and end of Visual mode are equal.

When using `"viw"` on a single-letter word and `'selection'` is exclusive, would not include the word.

When formatting text from Insert mode, using `CTRL-O`, could mess up undo information.

While writing a file (also for the backup file) there was no check for an interrupt (hitting `CTRL-C`). Vim could hang when writing a large file over a slow network, and moving the mouse didn't make it appear (when `'mousehide'` is set) and the screen wasn't updated in the GUI. Also allow interrupting when syncing the swap file, it can take a long time.

When using `":mksession"` while there is help window, it would later be restored

to the right file but not marked as a help buffer. ":help" would then open another window. Now use the value "help" for **'buftype'** to mark a help buffer.

The session file contained absolute path names in option values, that doesn't work when the home directory depends on the situation. Replace the home directory with ~/ when possible.

When using **'showbreak'** a TAB just after the shown break would not be counted correctly, the cursor would be positioned wrong.

With **'showbreak'** set to "--->" or "----->" and **'sts'** set to 4, inserting tabs did not work right. Could cause a crash. Backspacing was also wrong, could get stuck at a line break.

Win32: crashed when tearing off a menu with over 300 items.

GUI: A menu or toolbar item would appear when only a tooltip was defined for it.

When **'scrolloff'** is non-zero and "\$" is in **'coptions'**, using "s" while the last line of the file is the first line on screen, the text wasn't displayed.

When running "autoconf", delete the configure cache to force starting cleanly when configure is run again.

When changing the Normal colors for cterm, the value of **'background'** was changed even when the GUI was used.

The warning for a missing vimrun.exe was always given on startup, but some people just editing a file don't need to be bothered by it. Only show it when vimrun would be used.

When using "%" in a multibyte text it could get confused by trailbytes that match. (Muraoka Taro)

Termcap entry for RiscOS was wrong, using 7 and 8 in octal codes.

Athena: The title of a dialog window and the file selector window were not set. (David Harrison)

The "htmlLink" highlight group specified colors, which gives problems when using a color scheme. Added the "Underlined" highlight group for this.

After using ":insert" or ":change" the '[' mark would be one line too low.

When looking for the file name after a match with **'include'** one character was skipped. Same for **'define'**.

Win32 and DJGPP: When editing a file with a short name in a directory, and editing the same file but using the long name, would end up with two buffers on the same file.

"gf" on a filename that starts with "../" only worked when the file being edited is in the current directory. An include file search didn't work

properly for files starting with "../" or ".". Now search both relative to the file and to the current directory.

When **'printhead'**, **'titlestring'**, **'iconstring'**, **'rulerformat'** or **'statusline'** contained "%{" but no following "}" memory was corrupted and a crash could happen.

":@append" and then inserting two lines did not redraw the blank lines that were scrolled back down.

When using insert mode completion in a narrow window, the message caused a scroll up. Now shorten the message if it doesn't fit and avoid writing the ruler over the message.

XIM still didn't work correctly on some systems, especially SGI/IRIX. Added the **'imdisable'** option, which is set by default for that system.

Patch 6.0aw.008

Problem: When the first character of a file name is over 127, the Buffers menu entry would get a negative priority and cause problems.

Solution: Reduce the multiplier for the first character when computing the hash value for a Buffers menu entry.

Files: runtime/menu.vim

Patch 6.0aw.010

Problem: Win32: ":browse edit dir/dir" didn't work. (Vikas)

Solution: Change slashes to backslashes in the directory passed to the file browser.

Files: src/gui_w48.c

Athena file browser: On some systems wcstombs() can't be used to get the length of a multibyte string. Use the maximum length then. (Yasuhiro Matsumoto)

Patch 6.0ax.001

Problem: When **'patchmode'** is set, appending to a file gives an empty original file. (Ed Ralston)

Solution: Also make a backup copy when appending and **'patchmode'** is set.

Files: src/fileio.c

Patch 6.0ax.002

Problem: When **'patchmode'** is set, appending to a compressed file gives an uncompressed original file. (Ed Ralston)

Solution: Create the original file before decompressing.

Files: runtime/plugin/gzip.vim

Patch 6.0ax.005

Problem: Athena file selector keeps the title of the first invocation.

Solution: Set the title each time the file selector is opened. (David Harrison)

Files: src/gui_at_fs.c

Patch 6.0ax.007

Problem: When using GPM (mouse driver in a Linux console) a double click is

interpreted as a scroll wheel click.
Solution: Check if GPM is being used when deciding if a mouse event is for the scroll wheel.
Files: src/term.c

Patch 6.0ax.010

Problem: The Edit.Save menu and the Save toolbar button didn't work when the buffer has no file name.

Solution: Use a file browser to ask for a file name. Also fix the toolbar Find item in Visual mode.

Files: runtime/menu.vim

Patch 6.0ax.012

Problem: When **'cptions'** contains "\$", breaking a line for **'textwidth'** doesn't redraw properly. (Stefan Schulze)

Solution: Remove the dollar before breaking the line.

Files: src/edit.c

Patch 6.0ax.014

Problem: Win32: On Windows 98 ":make -f file" doesn't work when **'shell'** is "command.com" and **'makeprg'** is "nmake". The environment isn't passed on to "nmake".

Solution: Also use vimrun.exe when redirecting the output of a command.

Files: src/os_win32.c

Patch 6.0ax.016

Problem: The version number was reported wrong in the intro screen.

Solution: Check for a version number with two additional letters.

Files: src/version.c

Patch 6.0ax.019

Problem: When scrolling a window with folds upwards, switching to another vertically split window and back may not update the scrollbar.

Solution: Limit w_botline to the number of lines in the buffer plus one.

Files: src/move.c

VERSION 6.1 version-6.1

This section is about improvements made between version 6.0 and 6.1.

This is a bug-fix release, there are not really any new features.

Changed changed-6.1

'iminsert' and **'imsearch'** are no longer set as a side effect of defining a language-mapping using ":lmap".

Added added-6.1

Syntax files:

ampl	AMPL (David Krief)
ant	Ant (Johannes Zellner)
baan	Baan (Her van de Vliert)
cs	C# (Johannes Zellner)
lifelines	Lifelines (Patrick Texier)
lscript	LotusScript (Taryn East)
moo	MOO (Timo Frenay)
nsis	NSIS (Alex Jakushev)
ppd	Postscript Printer Description (Bjoern Jacke)
rpl	RPL/2 (Joel Bertrand)
scilab	Scilab (Benoit Hamelin)
splint	Splint (Ralf Wildenhues)
sqlj	SQLJ (Andreas Fischbach)
wvdial	WvDial (Prahlad Vaidyanathan)
xf86conf	XFree86 config (Nikolai Weibull)
xmodmap	Xmodmap (Nikolai Weibull)
xslt	Xslt (Johannes Zellner)
monk	Monk (Mike Litherland)
xsd	Xsd (Johannes Zellner)
cdl	CDL (Raul Segura Acevedo)
sendpr	Send-pr (Hendrik Scholz)

Added indent file for Scheme. (Dorai Sitaram)
Added indent file for Prolog. (Kontra Gergely)
Added indent file for Povray (David Necas)
Added indent file for IDL (Aleksandar Jelenak)
Added C# indent and ftplugin scripts.

Added Ukrainian menu translations. (Bohdan Vlasyuk)
Added ASCII version of the Czech menus. (Jiri Brezina)

Added Simplified Chinese translation of the tutor. (Mendel L Chan)

Added Russian keymap for yawerty keyboard.

Added an explanation of using the vimrc file in the tutor.
Changed tutor.vim to get the right encoding for the Taiwanese tutor.

Added Russian tutor. (Andrey Kiselev)
Added Polish tutor. (Mikolaj Machowski)

Added darkblue color scheme. (Bohdan Vlasyuk)

When packing the dos language archive automatically generate the .mo files that are required.

Improved NSIS script to support NSIS 180. Added icons for the enabled/disabled status. (Mirek Pruchnik)

cp1250 version of the Slovak message translations.

Compiler plugins for IRIX compilers. (David Harrison)

Fixed

fixed-6.1

The license text was updated to make the meaning clearer and make it compatible with the GNU GPL. Otherwise distributors have a problem when linking Vim with a GPL'ed library.

When installing the "less.sh" script it was not made executable. (Chuck Berg)

Win32: The "9" key on the numpad wasn't working. (Julian Kinraid)

The NSIS install script didn't work with NSIS 1.80 or later. Also add Vim-specific icons. (Pruchnik)

The script for conversion to HTML contained an "if" in the wrong place. (Michael Geddes)

Allow using ":ascii" in the sandbox, it's harmless.

Removed creat() from osdef2.h.in, it wasn't used and may cause a problem when it's redefined to creat64().

The text files in the VisVim directory were in "dos" format. This caused problems when applying a patch. Now keep them in "unix" format and convert them to "dos" format only for the PC archives.

Add ruby files to the dos source archive, they can be used by Make_mvc.mak. (Mirek Pruchnik)

"cp -f" doesn't work on all systems. Change "cp -f" in the Makefile to "rm -f" and "cp".

Didn't compile on a Compaq Tandem Himalaya OSS. (Michael A. Benzinger)

The GTK file selection dialog didn't include the "Create Dir", "Delete File" and "Rename File" buttons.

When doing ":browse source" the dialog has the title "Run Macro". Better would be "Source Vim script". (Yegappan Lakshmanan)

Win32: Don't use the printer font as default for the font dialog.

"make doslang" didn't work when configure didn't run (yet). Set \$MAKEMO to "yes". (Mirek Pruchnik)

The ToolBar TagJump item used "g]", which prompts for a selection even when there is only one matching tag. Use "g<C-]>" instead.

The ming makefile for message translations didn't have the right list of files.

The MS-Windows 3.1 version complains about LIBINTL.DLL not found. Compile

this version without message translations.

The Borland 5 makefile contained a check for Ruby which is no longer needed. The URLs for the TCL library was outdated. (Dan Sharp)

The eviso.ps file was missing from the DOS runtime archive, it's needed for printing PostScript in the 32bit DOS version.

In menu files ":scriptencoding" was used in a wrong way after patch 6.1a.032 Now use ":scriptencoding" in the file where the translations are given. Do the same for all menus in latin1 encoding.

Included a lot of fixes for the Macintosh, mostly to make it work with Carbon. (Dany StAmant, Axel Kielhorn, Benji Fisher)

Improved the vimtutor shell script to use \$TMPDIR when it exists, and delete the copied file when exiting in an abnormal way. (Max Ischenko)

When "iconv.dll" can't be found, try using "libiconv.dll".

When encryption is used, filtering with a shell command wasn't possible.

DJGPP: ":cd c:" always failed, can't get permissions for "c:".
Win32: ":cd c:/" failed if the previous current directory on c: had become invalid.

DJGPP: Shift-Del and Del both produce \316\123. Default mapping for Del is wrong. Disabled it.

Dependencies on header files in MingW makefile was wrong.

Win32: Don't use ACL stuff for MSVC 4.2, it's not supported. (Walter Briscoe)

Win32 with Borland: bcc.cfg was caching the value for \$(BOR), but providing a different argument to make didn't regenerate it.

Win32 with MSVC: Make_ivc.mak generates a new if_ole.h in a different directory, the if_ole.h in the src directory may be used instead. Delete the distributed file.

When a window is vertically split and then ":ball" is used, the window layout is messed up, can cause a crash. (Muraoka Taro)

When '**insertmode**' is set, using File/New menu and then double clicking, "i" is soon inserted. (Merlin Hansen)

When Select mode is active and using the Buffers menu to switch to another buffer, an old selection comes back. Reset VISual_reselect for a ":buffer" command.

When Select mode is active and '**insertmode**' is set, using the Buffers menu to switch to another buffer, did not return to Insert mode. Make sure "restart_edit" is set.

When double clicking on the first character of a word while **'selection'** is "exclusive" didn't select that word.

Patch 6.0.001

Problem: Loading the sh.vim syntax file causes error messages. (Corinna Vinschen)
Solution: Add an "if". (Charles Campbell)
Files: runtime/syntax/sh.vim

Patch 6.0.002

Problem: Using a '@' item in **'viminfo'** doesn't work. (Marko Leipert)
Solution: Add '@' to the list of accepted items.
Files: src/option.c

Patch 6.0.003

Problem: The configure check for ACLs on AIX doesn't work.
Solution: Fix the test program so that it compiles. (Tomas Ogren)
Files: src/configure.in, src/auto/configure

Patch 6.0.004

Problem: The find/replace dialog doesn't reuse a previous argument properly.
Solution: After removing a "\V" terminate the string. (Zwane Mwaikambo)
Files: src/gui.c

Patch 6.0.005

Problem: In Insert mode, **"CTRL-O :ls"** has a delay before redrawing.
Solution: Don't delay just after wait_return() was called. Added the did_wait_return flag.
Files: src/globals.h, src/message.c, src/normal.c, src/screen.c

Patch 6.0.006

Problem: With a vertical split, **'number'** set and **'scrolloff'** non-zero, making the window width very small causes a crash. (Niklas Lindstrom)
Solution: Check for a zero width.
Files: src/move.c

Patch 6.0.007

Problem: When setting **'filetype'** while there is no FileType autocommand, a following **":setfiletype"** would set **'filetype'** again. (Kobus Retief)
Solution: Set did_filetype always when **'filetype'** has been set.
Files: src/option.c

Patch 6.0.008

Problem: **'imdisable'** is missing from the options window. (Michael Naumann)
Solution: Add an entry for it.
Files: runtime/optwin.vim

Patch 6.0.009

Problem: Nextstep doesn't have S_ISBLK. (John Beppu)
Solution: Define S_ISBLK using S_IFBLK.

Files: src/os_unix.h

Patch 6.0.010

Problem: Using "gf" on a file name starting with "./" or "../" in a buffer without a name causes a crash. (Roy Lewis)

Solution: Check for a NULL file name.

Files: src/misc2.c

Patch 6.0.011

Problem: Python: After replacing or deleting lines get an ml_get error. (Leo Lipelis)

Solution: Adjust the cursor position for deleted or added lines.

Files: src/if_python.c

Patch 6.0.012

Problem: Polish translations contain printf format errors, this can result in a crash when using one of them.

Solution: Fix for translated messages. (Michal Politowski)

Files: src/po/pl.po

Patch 6.0.013

Problem: Using ":silent! cmd" still gives some error messages, like for an invalid range. (Salman Halim)

Solution: Reset emsg_silent after calling emsg() in do_one_cmd().

Files: src/ex_docmd.c

Patch 6.0.014

Problem: When '**modifiable**' is off and '**virtualedit**' is "all", "rx" on a TAB still changes the buffer. (Muraoka Taro)

Solution: Check if saving the line for undo fails.

Files: src/normal.c

Patch 6.0.015

Problem: When '**cpoptions**' includes "S" and "filetype plugin on" has been used, can get an error for deleting the b:did_ftplugin variable. (Ralph Henderson)

Solution: Only delete the variable when it exists.

Files: runtime/ftplugin.vim

Patch 6.0.016

Problem: bufnr(), bufname() and bufwinnr() don't find unlisted buffers when the argument is a string. (Hari Krishna Dara)
Also for setbufvar() and getbufvar().

Solution: Also find unlisted buffers.

Files: src/eval.c

Patch 6.0.017

Problem: When '**ttybuiltin**' is set and a builtin termcap entry defines t_Co and the external one doesn't, it gets reset to empty. (David Harrison)

Solution: Only set t_Co when it wasn't set yet.

Files: src/term.c

Patch 6.0.018

Problem: Initializing **'encoding'** may cause a crash when `setlocale()` is not used. (Dany St-Amant)
Solution: Check for a NULL pointer.
Files: `src/mbyte.c`

Patch 6.0.019

Problem: Converting a string with multibyte characters to a printable string, e.g., with `strtrans()`, may cause a crash. (Tomas Zellerin)
Solution: Correctly compute the length of the result in `transstr()`.
Files: `src/charset.c`

Patch 6.0.020

Problem: When obtaining the value of a global variable internally, could get the function-local value instead. Applies to using `<Leader>` and `<LocalLeader>` and resetting highlighting in a function.
Solution: Prepend "g:" to the variable name. (Aric Blumer)
Files: `src/syntax.c`, `src/term.c`

Patch 6.0.021

Problem: The **'cscopepathcomp'** option didn't work.
Solution: Change `USE_CSCOPE` to `FEAT_CSCOPE`. (Mark Feng)
Files: `src/option.c`

Patch 6.0.022

Problem: When using the **'langmap'** option, the second character of a command starting with "g" isn't adjusted.
Solution: Apply **'langmap'** to the second character. (Alex Kapranoff)
Files: `src/normal.c`

Patch 6.0.023

Problem: Loading the `lhaskell` syntax doesn't work. (Thore B. Karlsen)
Solution: Use `":runtime"` instead of `"source"` to load `haskell.vim`.
Files: `runtime/syntax/lhaskell.vim`

Patch 6.0.024

Problem: Using **"CTRL-V u 9900"** in Insert mode may cause a crash. (Noah Levitt)
Solution: Don't insert a NUL byte in the text, use a newline.
Files: `src/misc1.c`

Patch 6.0.025

Problem: The pattern `"\vx(.|$)"` doesn't match "x" at the end of a line. (Preben Peppe Guldborg)
Solution: Always see a "\$" as end-of-line after "\v". Do the same for "^".
Files: `src/regexp.c`

Patch 6.0.026

Problem: GTK: When using arrow keys to navigate through the menus, the separators are selected.
Solution: Set the separators "insensitive". (Pavel Kankovsky)
Files: `src/gui_gtk.c`, `src/gui_gtk_x11.c`

Patch 6.0.027

Problem: VMS: Printing doesn't work, the file is deleted too quickly.

No longer need the VMS specific printing menu.
gethostname() is not available with VAXC.

Solution: Adjust the '**printexpr**' option default. Fix the other problems and update the documentation. (Zoltan Arpadffy)
Files: runtime/doc/os_vms.txt, runtime/menu.vim, src/INSTALLvms.txt, src/Make_vms.mms, src/option.c, src/os_unix.c, src/os_vms_conf.h

Patch 6.0.028

Problem: Can't compile without +virtualedit and with +visualextra. (Geza Lakner)
Solution: Add an #ifdef for +virtualedit.
Files: src/ops.c

Patch 6.0.029

Problem: When making a change in line 1, then in line 2 and then deleting line 1, undo info could be wrong. Only when the changes are undone at once. (Gerhard Hochholzer)
Solution: When not saving a line for undo because it was already done before, remember for which entry the last line must be computed. Added ue_getbot_entry pointer for this. When the number of lines changes, adjust the position of newer undo entries.
Files: src/structs.h, src/undo.c

Patch 6.0.030

Problem: Using ":source! file" doesn't work inside a loop or after ":argdo". (Pavol Juhas)
Solution: Execute the commands in the file right away, do not let the main loop do it.
Files: src/ex_cmds2.c, src/ex_docmd.c, src/getchar.c, src/globals.h, src/proto/ex_docmd.pro, src/proto/getchar.pro

Patch 6.0.031

Problem: Nextstep doesn't have setenv() or putenv(). (John Beppu)
Solution: Move putenv() from pty.c to misc2.c
Files: src/misc2.c, src/pty.c

Patch 6.0.032

Problem: When changing a setting that affects all folds, they are not displayed immediately.
Solution: Set the redraw flag in foldUpdateAll().
Files: src/fold.c

Patch 6.0.033

Problem: Using '**wildmenu**' on MS-Windows, file names that include a space are only displayed starting with that space. (Xie Yuheng)
Solution: Don't recognize a backslash before a space as a path separator.
Files: src/screen.c

Patch 6.0.034

Problem: Calling searchpair() with three arguments could result in a crash or strange error message. (Kalle Bjorklid)
Solution: Don't use the fifth argument when there is no fourth argument.
Files: src/eval.c

Patch 6.0.035

Problem: The menu item Edit/Global_Settings/Toggle_Toolbar doesn't work when **'ignorecase'** is set. (Allen Castaban)
Solution: Always match case when checking if a flag is already present in **'guioptions'**.
Files: runtime/menu.vim

Patch 6.0.036

Problem: OS/2, MS-DOS and MS-Windows: Using a path that starts with a slash in **'tags'** doesn't work as expected. (Mathias Koehrer)
Solution: Only use the drive, not the whole path to the current directory. Also make it work for "c:dir/file".
Files: src/misc2.c

Patch 6.0.037

Problem: When the user has set "did_install_syntax_menu" to avoid the default Syntax menu it still appears. (Virgilio)
Solution: Don't add the three default items when "did_install_syntax_menu" is set.
Files: runtime/menu.vim

Patch 6.0.038

Problem: When **'selection'** is "exclusive", deleting a block of text at the end of a line can leave the cursor beyond the end of the line.
Solution: Correct the cursor position.
Files: src/ops.c

Patch 6.0.039

Problem: "gP" leaves the cursor in the wrong position when **'virtualedit'** is used. Using "c" in blockwise Visual mode leaves the cursor in a strange position.
Solution: For "gP" reset the "coladd" field for the ']' mark. For "c" leave the cursor on the last inserted character.
Files: src/ops.c

Patch 6.0.040

Problem: When **'fileencoding'** is invalid and writing fails because of this, the original file is gone. (Eric Carlier)
Solution: Restore the original file from the backup.
Files: src/fileio.c

Patch 6.0.041

Problem: Using ":language messages en" when LC_MESSAGES is undefined results in setting LC_CTYPE. (Eric Carlier)
Solution: Set \$LC_MESSAGES instead.
Files: src/ex_cmds2.c

Patch 6.0.042

Problem: ":mksession" can't handle file names with a space.
Solution: Escape special characters in file names with a backslash.
Files: src/ex_docmd.c

Patch 6.0.043

Problem: Patch 6.0.041 was wrong.
Solution: Use `mch_getenv()` instead of `vim_getenv()`.
Files: `src/ex_cmds2.c`

Patch 6.0.044

Problem: Using a "containedin" list for a syntax item doesn't work for an item that doesn't have a "contains" argument. Also, "containedin" doesn't ignore a transparent item. (Timo Frenay)
Solution: When there is a "containedin" argument somewhere, always check for contained items. Don't check for the transparent item but the item it's contained in.
Files: `src/structs.h`, `src/syntax.c`

Patch 6.0.045

Problem: After creating a fold with a Visual selection, another window with the same buffer still has inverted text. (Sami Salonen)
Solution: Redraw the inverted text.
Files: `src/normal.c`

Patch 6.0.046

Problem: When `getrlimit()` returns an 8 byte number the check for running out of stack may fail. (Anthony Meijer)
Solution: Skip the stack check if the limit doesn't fit in a long.
Files: `src/auto/configure`, `src/config.h.in`, `src/configure.in`, `src/os_unix.c`

Patch 6.0.047

Problem: Using a regexp with `"\(\\" inside a "\%[" item causes a crash. (Samuel Lacas)
Solution: Don't allow nested atoms inside "\%[".
Files: src/regexp.c`

Patch 6.0.048

Problem: Win32: In the console the mouse doesn't always work correctly. Sometimes after getting focus a mouse movement is interpreted like a button click.
Solution: Use a different function to obtain the number of mouse buttons. Avoid recognizing a button press from undefined bits. (Vince Negri)
Files: `src/os_win32.c`

Patch 6.0.049

Problem: When using `evim` the intro screen is misleading. (Adrian Nagle)
Solution: Mention whether `'insertmode'` is set and the menus to be used.
Files: `runtime/menu.vim`, `src/version.c`

Patch 6.0.050

Problem: UTF-8: "viw" doesn't include non-ASCII characters before the cursor. (Bertilo Wennergren)
Solution: Use `dec_cursor()` instead of decrementing the column number.
Files: `src/search.c`

Patch 6.0.051

Problem: UTF-8: Using **CTRL-R** on the command line doesn't insert composing characters. (Ron Aaron)

Solution: Also include the composing characters and fix redrawing them.
Files: src/ex_getln.c, src/ops.c

Patch 6.0.052

Problem: The check for rlim_t in patch 6.0.046 does not work on some systems. (Zdenek Sekera)

Solution: Also look in sys/resource.h for rlim_t.

Files: src/auto/configure, src/configure.in

Patch 6.0.053 (extra)

Problem: Various problems with QNX.

Solution: Minor fix for configure. Switch on terminal clipboard support in main.c. Fix "pterm" mouse support. os_qnx.c didn't build without photon. (Julian Kinraid)

Files: src/auto/configure, src/configure.in, src/gui_photon.c, src/main.c, src/misc2.c, src/option.h, src/os_qnx.c, src/os_qnx.h, src/syntax.c

Patch 6.0.054

Problem: When using mswin.vim, **CTRL-V** pastes a block of text like it is normal text. Using **CTRL-V** in blockwise Visual mode leaves "x" characters behind.

Solution: Make **CTRL-V** work as it should. Do the same for the Paste menu entries.

Files: runtime/menu.vim, runtime/mswin.vim

Patch 6.0.055

Problem: GTK: The selection isn't copied the first time.

Solution: Own the selection at the right moment.

Files: src/gui_gtk_x11.c

Patch 6.0.056

Problem: Using "**CTRL-O** cw" in Insert mode results in a nested Insert mode. **<Esc>** doesn't leave Insert mode then.

Solution: Only use nested Insert mode when '**insertmode**' is set or when a mapping is used.

Files: src/normal.c

Patch 6.0.057

Problem: Using ":wincmd g}" in a function doesn't work. (Gary Holloway)

Solution: Execute the command directly, instead of putting it in the typeahead buffer.

Files: src/normal.c, src/proto/normal.pro, src/window.c

Patch 6.0.058

Problem: When a Cursorhold autocommand moved the cursor, the ruler wasn't updated. (Bohdan Vlasyuk)

Solution: Update the ruler after executing the autocommands.

Files: src/gui.c

Patch 6.0.059

Problem: Highlighting for '**hlsearch**' isn't visible in lines that are highlighted for diff highlighting. (Gary Holloway)

Solution: Let '**hlsearch**' highlighting overrule diff highlighting.

Files: src/screen.c

Patch 6.0.060

Problem: Motif: When the tooltip is to be popped up, Vim crashes.
(Gary Holloway)

Solution: Check for a NULL return value from gui_motif_fontset2fontlist().

Files: src/gui_beval.c

Patch 6.0.061

Problem: The toolbar buttons to load and save a session do not correctly use v:this_session.

Solution: Check for v:this_session to be empty instead of existing.

Files: runtime/menu.vim

Patch 6.0.062

Problem: Crash when **'verbose'** is > 3 and using ":shell". (Yegappan Lakshmanan)

Solution: Avoid giving a NULL pointer to printf(). Also output a newline and switch the cursor on.

Files: src/misc2.c

Patch 6.0.063

Problem: When **'coptions'** includes "\$", using "cw" to type a ')' on top of the "\$" doesn't update syntax highlighting after it.

Solution: Stop displaying the "\$" when typing a ')' in its position.

Files: src/search.c

Patch 6.0.064 (extra)

Problem: The NSIS install script doesn't work with newer versions of NSIS. The diff feature doesn't work when there isn't a good diff.exe on the system.

Solution: Replace the GetParentDir instruction by a user function. Fix a few cosmetic problems. Use defined constants for the version number, so that it's defined in one place only. Only accept the install directory when it ends in "vim".
(Eduardo Fernandez)

Add a diff.exe and use it from the default _vimrc.

Files: nsis/gvim.nsi, nsis/README.txt, src/dosinst.c

Patch 6.0.065

Problem: When using ":normal" in **'indentexpr'** it may use redo characters before its argument. (Neil Bird)

Solution: Save and restore the stuff buffer in ex_normal().

Files: src/ex_docmd.c, src/getchar.c, src/globals.h, src/structs.h

Patch 6.0.066

Problem: Sometimes undo for one command is split into two undo actions.
(Halim Salman)

Solution: Don't set the undo-synced flag when reusing a line that was already saved for undo.

Files: src/undo.c

Patch 6.0.067

Problem: if_xcmdsrv.c doesn't compile on systems where fd_set isn't defined

in the usual header file (e.g., AIX). (Mark Waggoner)
Solution: Include sys/select.h in if_xcmdsrv.c for systems that have it.
Files: src/if_xcmdsrv.c

Patch 6.0.068

Problem: When formatting a Visually selected area with "gq" and the number of lines increases the last line may not be redrawn correctly. (Yegappan Lakshmanan)
Solution: Correct the area to be redrawn for inserted/deleted lines.
Files: src/ops.c

Patch 6.0.069

Problem: Using "K" on a word that includes a "!" causes a "No previous command" error, because the "!" is expanded. (Craig Jeffries)
Solution: Put a backslash before the "!".
Files: src/normal.c

Patch 6.0.070

Problem: Win32: The error message for a failed dynamic linking of a Perl, Ruby, Tcl and Python library is unclear about what went wrong.
Solution: Give the name of the library or function that could not be loaded. Also for the iconv and gettext libraries when '**verbose**' is set.
Files: src/eval.c, src/if_perl.xs, src/if_python.c, src/if_ruby.c, src/if_tcl.c, src/mbyte.c, src/os_win32.c, src/proto/if_perl.pro, src/proto/if_python.pro, src/proto/if_ruby.pro, src/proto/if_tcl.pro, src/proto/mbyte.pro

Patch 6.0.071

Problem: The "iris-ansi" builtin termcap isn't very good.
Solution: Fix the wrong entries. (David Harrison)
Files: src/term.c

Patch 6.0.072

Problem: When '**lazyredraw**' is set, a mapping that stops Visual mode, moves the cursor and starts Visual mode again causes a redraw problem. (Brian Silverman)
Solution: Redraw both the old and the new Visual area when necessary.
Files: src/normal.c, src/screen.c

Patch 6.0.073 (extra)

Problem: DJGPP: When using **CTRL-Z** to start a shell, the prompt is halfway the text. (Volker Kiefel)
Solution: Position the system cursor before starting the shell.
Files: src/os_msdos.c

Patch 6.0.074

Problem: When using "&" in a substitute string a multibyte character with a trailbyte 0x5c is not handled correctly.
Solution: Recognize multibyte characters inside the "&" part. (Muraoka Taro)
Files: src/regexp.c

Patch 6.0.075

Problem: When closing a horizontally split window while '**eadirection**' is "hor" another horizontally split window is still resized. (Aron

Griffis)
Solution: Only resize windows in the same top frame as the window that is split or closed.
Files: src/main.c, src/proto/window.pro, src/window.c

Patch 6.0.076

Problem: Warning for wrong pointer type when compiling.
Solution: Use char instead of char_u pointer.
Files: src/version.c

Patch 6.0.077

Problem: Patch 6.0.075 was incomplete.
Solution: Fix another call to win_equal().
Files: src/option.c

Patch 6.0.078

Problem: Using "daw" at the end of a line on a single-character word didn't include the white space before it. At the end of the file it didn't work at all. (Gavin Sinclair)
Solution: Include the white space before the word.
Files: src/search.c

Patch 6.0.079

Problem: When "W" is in '**coptions**' and '**backupcopy**' is "no" or "auto", can still overwrite a read-only file, because it's renamed. (Gary Holloway)
Solution: Add a check for a read-only file before renaming the file to become the backup.
Files: src/fileio.c

Patch 6.0.080

Problem: When using a session file that has the same file in two windows, the fileinfo() call in do_ecmd() causes a scroll and a hit-enter prompt. (Robert Webb)
Solution: Don't scroll this message when '**shortmess**' contains '0'.
Files: src/ex_cmds.c

Patch 6.0.081

Problem: After using ":saveas" the new buffer name is added to the Buffers menu with a wrong number. (Chauk-Mean Proum)
Solution: Trigger BufFilePre and BufFilePost events for the renamed buffer and BufAdd for the old name (which is with a new buffer).
Files: src/ex_cmds.c

Patch 6.0.082

Problem: When swapping screens in an xterm and there is an (error) message from the vimrc script, the shell prompt is after the message.
Solution: Output a newline when there was output on the alternate screen. Also when starting the GUI.
Files: src/main.c

Patch 6.0.083

Problem: GTK: When compiled without menu support the buttons in a dialog don't have any text. (Erik Edelman)

Solution: Add the text also when GTK_USE_ACCEL isn't defined. And define GTK_USE_ACCEL also when not using menus.
Files: src/gui_gtk.c

Patch 6.0.084

Problem: UTF-8: a "r" command with an argument that is a keymap for a character with a composing character can't be repeated with ".".
(Raphael Finkel)
Solution: Add the composing characters to the redo buffer.
Files: src/normal.c

Patch 6.0.085

Problem: When '**mousefocus**' is set, using "s" to go to Insert mode and then moving the mouse pointer to another window stops Insert mode, while this doesn't happen with "a" or "i". (Robert Webb)
Solution: Reset finish_op before calling edit().
Files: src/normal.c

Patch 6.0.086

Problem: When using "gu" the message says "~ed".
Solution: Make the message say "changed".
Files: src/ops.c

Patch 6.0.087 (lang)

Problem: Message translations are incorrect, which may cause a crash.
(Peter Figura)
The Turkish translations needed more work and the maintainer didn't have time.
Solution: Fix order of printf arguments. Remove %2\$d constructs.
Add "-v" to msgfmt to get a warning for wrong translations.
Don't install the Turkish translations for now.
Update a few more translations.
Files: src/po/Makefile, src/po/af.po, src/po/cs.po, src/po/cs.cp1250.po, src/po/de.po, src/po/es.po, src/po/fr.po, src/po/it.po, src/po/ja.po, src/po/ja.sjis.po, src/po/ko.po, src/po/pl.po, src/po/sk.po, src/po/uk.po, src/po/zh_CN.UTF-8.po, src/po/zh_CN.cp936.po, src/po/zh_CN.po, src/po/zh_TW.po

Patch 6.0.088

Problem: "." doesn't work after using "rx" in Visual mode. (Charles Campbell)
Solution: Also store the replacement character in the redo buffer.
Files: src/normal.c

Patch 6.0.089

Problem: In a C file, using "==" to align a line starting with "* " after a line with "* -" indents one space too few. (Piet Delpert)
Solution: Align with the previous line if the comment-start-string matches there.
Files: src/misc1.c

Patch 6.0.090

Problem: When a wrapping line does not fit in a window and '**scrolloff**' is bigger than half the window height, moving the cursor left or

right causes the screen to flash badly. (Lubomir Host)
Solution: When there is not enough room to show **'scrolloff'** screen lines and near the end of the line, show the end of the line.
Files: src/move.c

Patch 6.0.091

Problem: Using **CTRL-O** in Insert mode, while **'virtualedit'** is "all" and the cursor is after the end-of-line, moves the cursor left. (Yegappan Lakshmanan)
Solution: Keep the cursor in the same position.
Files: src/edit.c

Patch 6.0.092

Problem: The explorer plugin doesn't ignore case of **'suffixes'** on MS-Windows. (Mike Williams)
Solution: Match or ignore case as appropriate for the OS.
Files: runtime/plugin/explorer.vim

Patch 6.0.093

Problem: When the Tcl library couldn't be loaded dynamically, get an error message when closing a buffer or window. (Muraoka Taro)
Solution: Only free structures if already using the Tcl interpreter.
Files: src/if_tcl.c

Patch 6.0.094

Problem: Athena: When clicking in the horizontal scrollbar Vim crashes. (Paul Ackersviller)
Solution: Use the thumb size instead of the window pointer of the scrollbar (which is NULL). (David Harrison)
Also avoid that scrolling goes the wrong way in a narrow window.
Files: src/gui_athena.c

Patch 6.0.095

Problem: Perl: Deleting lines may leave the cursor beyond the end of the file.
Solution: Check the cursor position after deleting a line. (Serguei)
Files: src/if_perl.xs

Patch 6.0.096

Problem: When `":saveas fname"` fails because the file already exists, the file name is changed anyway and a following `":w"` will overwrite the file. (Eric Carlier)
Solution: Don't change the file name if the file already exists.
Files: src/ex_cmds.c

Patch 6.0.097

Problem: Re-indenting in Insert mode with **CTRL-F** may cause a crash with a multibyte encoding.
Solution: Avoid using a character before the start of a line. (Sergey Vlasov)
Files: src/edit.c

Patch 6.0.098

Problem: GTK: When using Gnome the "Search" and "Search and Replace" dialog

boxes are not translated.
Solution: Define ENABLE_NLS before including gnome.h. (Eduardo Fernandez)
Files: src/gui_gtk.c, src/gui_gtk_x11.c

Patch 6.0.099

Problem: Cygwin: When running Vi compatible MS-DOS line endings cause trouble.
Solution: Make the default for **'fileformats'** "unix,dos" in Vi compatible mode. (Michael Schaap)
Files: src/option.h

Patch 6.0.100

Problem: ":badd +0 test%file" causes a crash.
Solution: Take into account that the "+0" is NUL terminated when allocating room for replacing the "%".
Files: src/ex_docmd.c

Patch 6.0.101

Problem: ":mksession" doesn't restore editing a file that has a '#' or '%' in its name. (Wolfgang Blankenburg)
Solution: Put a backslash before the '#' and '%'.
Files: src/ex_docmd.c

Patch 6.0.102

Problem: When changing folds the cursor may appear halfway a closed fold. (Nam SungHyun)
Solution: Set w_cline_folded correctly. (Yasuhiro Matsumoto)
Files: src/move.c

Patch 6.0.103

Problem: When using **'scrollbind'** a large value of **'scrolloff'** will make the scroll binding stop near the end of the file. (Coen Engelbarts)
Solution: Don't use **'scrolloff'** when limiting the topline for scroll binding. (Dany StAmant)
Files: src/normal.c

Patch 6.0.104

Problem: Multi-byte: When '\$' is in **'coptions'**, typing a double-wide character that overwrites the left half of an old double-wide character causes a redraw problem and the cursor stops blinking.
Solution: Clear the right half of the old character. (Yasuhiro Matsumoto)
Files: src/edit.c, src/screen.c

Patch 6.0.105

Problem: Multi-byte: In a window of one column wide, with syntax highlighting enabled a crash might happen.
Solution: Skip getting the syntax attribute when the character doesn't fit anyway. (Yasuhiro Matsumoto)
Files: src/screen.c

Patch 6.0.106 (extra)

Problem: Win32: When the printer font is wrong, there is no error message.
Solution: Give an appropriate error message. (Yasuhiro Matsumoto)
Files: src/os_mswin.c

Patch 6.0.107 (extra)

Problem: VisVim: When editing another file, a modified file may be written unexpectedly and without warning.
Solution: Split the window if a file was modified.
Files: VisVim/Commands.cpp

Patch 6.0.108

Problem: When using folding could try displaying line zero, resulting in an error for a NULL pointer.
Solution: Stop decrementing w_topline when the first line of a window is in a closed fold.
Files: src/window.c

Patch 6.0.109

Problem: XIM: When the input method is enabled, repeating an insertion with "." disables it. (Marcel Svitalsky)
Solution: Don't store the input method status when a command comes from the stuff buffer.
Files: src/ui.c

Patch 6.0.110

Problem: Using undo after executing "Ox?jAx?kdd" from a register in an empty buffer gives an error message. (Gerhard Hochholzer)
Solution: Don't adjust the bottom line number of an undo block when it's zero. Add a test for this problem.
Files: src/undo.c, src/testdir/test20.in, src/testdir/test20.ok

Patch 6.0.111

Problem: The virtcol() function doesn't take care of **'virtualedit'**.
Solution: Add the column offset when needed. (Yegappan Lakshmanan)
Files: src/eval.c

Patch 6.0.112

Problem: The explorer plugin doesn't sort directories with a space or special character after a directory with a shorter name.
Solution: Ignore the trailing slash when comparing directory names. (Mike Williams)
Files: runtime/plugin/explorer.vim

Patch 6.0.113

Problem: ":edit ~/fname" doesn't work if \$HOME includes a space. Also, expanding wildcards with the shell may fail. (John Daniel)
Solution: Escape spaces with a backslash when needed.
Files: src/ex_docmd.c, src/misc1.c, src/proto/misc1.pro, src/os_unix.c

Patch 6.0.114

Problem: Using ":p" with fnamemodify() didn't expand "~/\" or "~/user/" to a full path. For Win32 the current directory was prepended. (Michael Geddes)
Solution: Expand the home directory.
Files: src/eval.c

Patch 6.0.115 (extra)

Problem: Win32: When using a dialog with a textfield it cannot scroll the text.
Solution: Add ES_AUTOHSCROLL to the textfield style. (Pedro Gomes)
Files: src/gui_w32.c

Patch 6.0.116 (extra)

Problem: MS-Windows NT/2000/XP: filewritable() doesn't work correctly for filesystems that use ACLs.
Solution: Use ACL functions to check if a file is writable. (Mike Williams)
Files: src/eval.c, src/macros.h, src/os_win32.c, src/proto/os_win32.pro

Patch 6.0.117 (extra)

Problem: Win32: when disabling the menu, "set lines=999" doesn't use all the available screen space.
Solution: Don't subtract the fixed caption height but the real menu height from the available screen space. Also: Avoid recursion in gui_mswin_get_menu_height().
Files: src/gui_w32.c, src/gui_w48.c

Patch 6.0.118

Problem: When \$TMPDIR is a relative path, the temp directory is missing a trailing slash and isn't deleted when Vim exits. (Peter Holm)
Solution: Add the slash after expanding the directory to an absolute path.
Files: src/fileio.c

Patch 6.0.119 (depends on patch 6.0.116)

Problem: VMS: filewritable() doesn't work properly.
Solution: Use the same method as for Unix. (Zoltan Arpadffy)
Files: src/eval.c

Patch 6.0.120

Problem: The conversion to html isn't compatible with XHTML.
Solution: Quote the values. (Jess Thrysoee)
Files: runtime/syntax/2html.vim

Patch 6.0.121 (extra) (depends on patch 6.0.116)

Problem: Win32: After patch 6.0.116 Vim doesn't compile with mingw32.
Solution: Add an #ifdef HAVE_ACL.
Files: src/os_win32.c

Patch 6.0.122 (extra)

Problem: Win16: Same resize problems as patch 6.0.117 fixed for Win32. And dialog textfield problem from patch 6.0.115.
Solution: Set old_menu_height only when used. Add ES_AUTOHSCROLL flag. (Vince Negri)
Files: src/gui_w16.c

Patch 6.0.123 (depends on patch 6.0.119)

Problem: Win16: Compilation problems.
Solution: Move "&&" to other lines. (Vince Negri)
Files: src/eval.c

Patch 6.0.124

Problem: When using a ":substitute" command that starts with "\="

(evaluated as an expression), "~" was still replaced with the previous substitute string.

Solution: Skip the replacement when the substitute string starts with "\=".

Files: src/ex_cmds.c, runtime/doc/change.txt

Patch 6.0.125 (extra)

Problem: Win32: When using the multi_byte_ime feature pressing the shift key would be handled as if a character was entered, thus mappings with a shifted key didn't work. (Charles Campbell)

Solution: Ignore pressing the shift, control and alt keys.

Files: src/os_win32.c

Patch 6.0.126

Problem: The python library was always statically linked.

Solution: Link the python library dynamically. (Matthias Klose)

Files: src/auto/configure, src/configure.in

Patch 6.0.127

Problem: When using a terminal that swaps screens and the Normal background color has a different background, using an external command may cause the color of the wrong screen to be changed. (Mark Waggoner)

Solution: Don't call screen_stop_highlight() in stoptermcap().

Files: src/term.c

Patch 6.0.128

Problem: When moving a vertically split window to the far left or right, the scrollbars are not adjusted. (Scott E Lee) When 'mousefocus' is set the mouse pointer wasn't adjusted.

Solution: Adjust the scrollbars and the mouse pointer.

Files: src/window.c

Patch 6.0.129

Problem: When using a very long file name, ":ls" (repeated a few times) causes a crash. Test with "vim `perl -e 'print "A"x1000'`". (Tejeda)

Solution: Terminate a string before getting its length in buflist_list().

Files: src/buffer.c

Patch 6.0.130

Problem: When using ":cprev" while the error window is open, and the new line at the top wraps, the window isn't correctly drawn. (Yegappan Lakshmanan)

Solution: When redrawing the topline don't scroll twice.

Files: src/screen.c

Patch 6.0.131

Problem: When using bufname() and there are two matches for listed buffers and one match for an unlisted buffer, the unlisted buffer is used. (Aric Blumer)

Solution: When there is a match with a listed buffer, don't check for unlisted buffers.

Files: src/buffer.c

Patch 6.0.132

Problem: When setting **'iminsert'** in the vimrc and using an xterm with two screens the ruler is drawn in the wrong screen. (Igor Goldenberg)

Solution: Only draw the ruler when using the right screen.

Files: src/option.c

Patch 6.0.133

Problem: When opening another buffer while **'keymap'** is set and **'iminsert'** is zero, **'iminsert'** is set to one unexpectedly. (Igor Goldenberg)

Solution: Don't set **'iminsert'** as a side effect of defining a **":lmap"** mapping. Only do that when **'keymap'** is set.

Files: src/getchar.c, src/option.c

Patch 6.0.134

Problem: When completing **":set tags="** a path with an embedded space causes the completion to stop. (Sektor van Skijlen)

Solution: Escape spaces with backslashes, like for **":set path="**. Also take backslashes into account when searching for the start of the path to complete (e.g., for **'backupdir'** and **'cscopeprg'**).

Files: src/ex_docmd.c, src/ex_getln.c, src/option.c, src/structs.h

Patch 6.0.135

Problem: Menus that are not supposed to do anything used **"<Nul>"**, which still produced an error beep.
When **CTRL-O** is mapped for Insert mode, **":amenu"** commands didn't work in Insert mode.
Menu language falls back to English when \$LANG ends in **"@euro"**.

Solution: Use **"<Nop>"** for a menu item that doesn't do anything, just like mappings.
Use **":anoremenu"** instead of **":amenu"**.
Ignore **"@euro"** in the locale name.

Files: runtime/makemenu.vim, runtime/menu.vim, src/menu.c

Patch 6.0.136

Problem: When completing in Insert mode, a mapping could be unexpectedly applied.

Solution: Don't use mappings when checking for a typed character.

Files: src/edit.c

Patch 6.0.137

Problem: GUI: When using the find or find/replace dialog from Insert mode, the input mode is stopped.

Solution: Don't use the input method status when the main window doesn't have focus.

Files: src/ui.c

Patch 6.0.138

Problem: GUI: When using the find or find/replace dialog from Insert mode, the text is inserted when **CTRL-O** is mapped. (Andre Pang)
When opening the dialog again, a whole word search isn't recognized.
When doing "replace all" a whole word search was never done.

Solution: Don't put a search or replace command in the input buffer, execute it directly.

Recognize "\<" and "\>" after removing "\V".
Add "\<" and "\>" also for "replace all".

Files: src/gui.c

Patch 6.0.139

Problem: When stopping **'wildmenu'** completion, the statusline of the bottom-left vertically split window isn't redrawn. (Yegappan Lakshmanan)

Solution: Redraw all the bottom statuslines.

Files: src/ex_getln.c, src/proto/screen.pro, src/screen.c

Patch 6.0.140

Problem: Memory allocated for local mappings and abbreviations is leaked when the buffer is wiped out.

Solution: Clear the local mappings when deleting a buffer.

Files: src/buffer.c, src/getchar.c, src/proto/getchar.pro, src/vim.h

Patch 6.0.141

Problem: When using ":enew" in an empty buffer, some buffer-local things are not cleared. b:keymap_name is not set.

Solution: Clear user commands and mappings local to the buffer when re-using the current buffer. Reload the keymap.

Files: src/buffer.c

Patch 6.0.142

Problem: When Python is linked statically, loading dynamic extensions might fail.

Solution: Add an extra linking flag when needed. (Andrew Rodionoff)

Files: src/configure.in, src/auto/configure

Patch 6.0.143

Problem: When a syntax item includes a line break in a pattern, the syntax may not be updated properly when making a change.

Solution: Add the "linebreaks" argument to ":syn sync".

Files: runtime/doc/syntax.txt, src/screen.c, src/structs.h, src/syntax.c

Patch 6.0.144

Problem: After patch 6.0.088 redoing "veU" doesn't work.

Solution: Don't add the "U" to the redo buffer, it will be used as an undo command.

Files: src/normal.c

Patch 6.0.145

Problem: When Vim can't read any input it might get stuck. When redirecting stdin and stderr Vim would not read commands from a file. (Servatius Brandt)

Solution: When repeatedly trying to read a character when it's not possible, exit Vim. When stdin and stderr are not a tty, still try reading from them, but don't do a blocking wait.

Files: src/ui.c

Patch 6.0.146

Problem: When **'statusline'** contains "%{'-'}" this results in a zero. (Milan Vancura)

Solution: Don't handle numbers with a minus as a number, they were not displayed anyway.
Files: src/buffer.c

Patch 6.0.147

Problem: It's not easy to mark a [Vim version as](#) being modified. The new license requires this.
Solution: Add the --modified-by argument to configure and the MODIFIED_BY define. It's used in the intro screen and the ":version" output.
Files: src/auto/configure, src/configure.in, src/config.h.in, src/feature.h, src/version.c

Patch 6.0.148

Problem: After "p" in an empty line, `[goes to the second character. (Kontra Gergely)
Solution: Don't increment the column number in an empty line.
Files: src/ops.c

Patch 6.0.149

Problem: The pattern "\(.{\-}\)*" causes a hang. When using a search pattern that causes a stack overflow to be detected Vim could still hang.
Solution: Correctly report "operand could be empty" when using "{-}". Check for "out_of_stack" inside loops to avoid a hang.
Files: src/regexp.c

Patch 6.0.150

Problem: When using a multibyte encoding, patch 6.0.148 causes "p" to work like "P". (Sung-Hyun Nam)
Solution: Compute the byte length of a multibyte character.
Files: src/ops.c

Patch 6.0.151

Problem: Redrawing the status line and ruler can be wrong when it contains multibyte characters.
Solution: Use character width and byte length correctly. (Yasuhiro Matsumoto)
Files: src/screen.c

Patch 6.0.152

Problem: strstrans() could hang on an illegal UTF-8 byte sequence.
Solution: Skip over illegal bytes. (Yasuhiro Matsumoto)
Files: src/charset.c

Patch 6.0.153

Problem: When using (illegal) double-byte characters and Vim syntax highlighting Vim can crash. (Yasuhiro Matsumoto)
Solution: Increase a pointer over a character instead of a byte.
Files: src/regexp.c

Patch 6.0.154

Problem: MS-DOS and MS-Windows: The menu entries for xxd don't work when there is no xxd in the path.
When converting back from Hex the filetype may remain "xxd" if it is not detected.

Solution: When xxd is not in the path use the one in the runtime directory, where the install program has put it.
Clear the **'filetype'** option before detecting the new value.
Files: runtime/menu.vim

Patch 6.0.155

Problem: Mac: compilation problems in ui.c after patch 6.0.145. (Axel Kielhorn)
Solution: Don't call mch_inchar() when NO_CONSOLE is defined.
Files: src/ui.c

Patch 6.0.156

Problem: Starting Vim with the -b argument and two files, ":next" doesn't set **'binary'** in the second file, like Vim 5.7. (Norman Diamond)
Solution: Set the global value for **'binary'**.
Files: src/option.c

Patch 6.0.157

Problem: When defining a user command with "--complete=dir" files will also be expanded. Also, "--complete=mapping" doesn't appear to work. (Michael Naumann)
Solution: Use the expansion flags defined with the user command.
Handle expanding mappings specifically.
Files: src/ex_docmd.c

Patch 6.0.158

Problem: When getting the warning for a file being changed outside of Vim and reloading the file, the **'readonly'** option is reset, even when the permissions didn't change. (Marcel Svitalsky)
Solution: Keep **'readonly'** set when reloading a file and the permissions didn't change.
Files: src/fileio.c

Patch 6.0.159

Problem: Wildcard expansion for ":emenu" also shows separators.
Solution: Skip menu separators for ":emenu", ":popup" and ":tearoff". Also, don't handle ":tmenu" as if it was ":tearoff". And leave out the alternatives with "&" included.
Files: src/menu.c

Patch 6.0.160

Problem: When compiling with GCC 3.0.2 and using the "-O2" argument, the optimizer causes a problem that makes Vim crash.
Solution: Add a configure check to avoid "-O2" for this version of gcc.
Files: src/configure.in, src/auto/configure

Patch 6.0.161 (extra)

Problem: Win32: Bitmaps don't work with signs.
Solution: Make it possible to use bitmaps with signs. (Muraoka Taro)
Files: src/ex_cmds.c, src/feature.h, src/gui_w32.c, src/gui_x11.c, src/proto/gui_w32.pro, src/proto/gui_x11.pro

Patch 6.0.162

Problem: Client-server: An error message for a wrong expression appears in

the server instead of the client.
Solution: Pass the error message from the server to the client. Also adjust the example code. (Flemming Madsen)
Files: src/globals.h, src/if_xcmdsrv.c, src/main.c, src/os_mswin.c, src/proto/if_xcmdsrv.pro, src/proto/os_mswin.pro, runtime/doc/eval.txt, runtime/tools/xcmdsrv_client.c

Patch 6.0.163

Problem: When using a GUI dialog, a file name is sometimes used like it was a directory.
Solution: Separate path and file name properly. For GTK, Motif and Athena concatenate directory and file name for the default selection.
Files: src/diff.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/gui_athena.c, src/gui_gtk.c, src/gui_motif.c, src/message.c

Patch 6.0.164

Problem: After patch 6.0.135 the menu entries for pasting don't work in Insert and Visual mode. (Muraoka Taro)
Solution: Add `<script>` to allow script-local mappings.
Files: runtime/menu.vim

Patch 6.0.165

Problem: Using `--remote` and executing locally gives unavoidable error messages.
Solution: Add `--remote-silent` and `--remote-wait-silent` to silently execute locally. For Win32 there was no error message when a server didn't exist.
Files: src/eval.c, src/if_xcmdsrv.c, src/main.c, src/os_mswin.c, src/proto/if_xcmdsrv.pro, src/proto/os_mswin.pro

Patch 6.0.166

Problem: GUI: There is no way to avoid dialogs to pop up.
Solution: Add the 'c' flag to `'guioptions'`: Use console dialogs. (Yegappan Lakshmanan)
Files: runtime/doc/options.txt, src/option.h, src/message.c

Patch 6.0.167

Problem: When `'fileencodings'` is "latin2" some characters in the help files are displayed wrong.
Solution: Force the `'fileencoding'` for the help files to be "latin1".
Files: src/fileio.c

Patch 6.0.168

Problem: `":%s/\n/#/"` doesn't replace at an empty line. (Bruce DeVisser)
Solution: Don't skip matches after joining two lines.
Files: src/ex_cmds.c

Patch 6.0.169

Problem: When run as `evim` and the GUI can't be started we get stuck in a terminal without menus in Insert mode.
Solution: Exit when using `"evim"` and `"gvim -y"` when the GUI can't be started.
Files: src/main.c

Patch 6.0.170

Problem: When printing double-width characters the size of tabs after them is wrong. (Muraoka Taro)
Solution: Correctly compute the column after a double-width character.
Files: src/ex_cmds2.c

Patch 6.0.171

Problem: With **'keymodel'** including "startsel", in Insert mode after the end of a line, shift-Left does not move the cursor. (Steve Hall)
Solution: **CTRL-O** doesn't move the cursor left, need to do that explicitly.
Files: src/edit.c

Patch 6.0.172

Problem: **CTRL-Q** doesn't replace **CTRL-V** after **CTRL-X** in Insert mode while it does in most other situations.
Solution: Make **CTRL-X CTRL-Q** work like **CTRL-X CTRL-V** in Insert mode.
Files: src/edit.c

Patch 6.0.173

Problem: When using "P" to insert a line break the cursor remains past the end of the line.
Solution: Check for the cursor being beyond the end of the line.
Files: src/ops.c

Patch 6.0.174

Problem: After using "gd" or "gD" the search direction for "n" may still be backwards. (Servatius Brandt)
Solution: Reset the search direction to forward.
Files: src/normal.c, src/search.c, src/proto/search.pro

Patch 6.0.175

Problem: ":help /\z(\)" doesn't work. (Thomas Koehler)
Solution: Double the backslashes.
Files: src/ex_cmds.c

Patch 6.0.176

Problem: When killed by a signal autocommands are still triggered as if nothing happened.
Solution: Add the v:dying variable to allow autocommands to work differently when a deadly signal has been trapped.
Files: src/eval.c, src/os_unix.c, src/vim.h

Patch 6.0.177

Problem: When **'commentstring'** is empty and **'foldmethod'** is "marker", "zf" doesn't work. (Thomas S. Urban)
Solution: Add the marker even when **'commentstring'** is empty.
Files: src/fold.c, src/normal.c

Patch 6.0.178

Problem: Uninitialized memory read from xp_backslash field.
Solution: Initialize xp_backslash field properly.
Files: src/eval.c, src/ex_docmd.c, src/ex_getln.c, src/misc1.c, src/tag.c

Patch 6.0.179

Problem: Win32: When displaying UTF-8 characters may read uninitialized memory.
Solution: Add `utfc_ptr2len_check_len()` to avoid reading past the end of a string.
Files: `src/mbyte.c`, `src/proto/mbyte.pro`, `src/gui_w32.c`

Patch 6.0.180

Problem: Expanding environment variables in a string that ends in a backslash could go past the end of the string.
Solution: Detect the trailing backslash.
Files: `src/misc1.c`

Patch 6.0.181

Problem: When using `":cd dir"` memory was leaked.
Solution: Free the allocated memory. Also avoid an uninitialized memory read.
Files: `src/misc2.c`

Patch 6.0.182

Problem: When using a regexp on multibyte characters, could try to read a character before the start of the line.
Solution: Don't decrement a pointer to before the start of the line.
Files: `src/regexp.c`

Patch 6.0.183

Problem: Leaking memory when `":func!"` redefines a function.
Solution: Free the function name when it's not used.
Files: `src/eval.c`

Patch 6.0.184

Problem: Leaking memory when expanding option values.
Solution: Don't always copy the expanded option into allocated memory.
Files: `src/option.c`

Patch 6.0.185

Problem: Crash in Vim when pasting a selection in another application, on a 64 bit machine.
Solution: Fix the format for an Atom to 32 bits. (Peter Derr)
Files: `src/ui.c`

Patch 6.0.186

Problem: X11: Three warnings when compiling the client-server code.
Solution: Add a typecast to unsigned char.
Files: `src/if_xcmdsrv.c`

Patch 6.0.187

Problem: "I" in Visual mode and then "u" reports too many changes. (Andrew Stryker)
"I" in Visual linewise mode adjusts the indent for no apparent reason.
Solution: Only save those lines for undo that are changed.
Don't change the indent after inserting in Visual linewise mode.
Files: `src/ops.c`

Patch 6.0.188

Problem: Win32: After patch 6.0.161 signs defined in the vimrc file don't work.

Solution: Initialize the sign icons after initializing the GUI. (Vince Negri)

Files: src/gui.c, src/gui_x11.c

Patch 6.0.189

Problem: The size of the Visual area isn't always displayed when scrolling ('ruler' off, 'showcmd' on). Also not when using a search command. (Sylvain Hitier)

Solution: Redisplay the size of the selection after showing the mode.

Files: src/screen.c

Patch 6.0.190

Problem: GUI: when 'mouse' is empty a click with the middle button still moves the cursor.

Solution: Paste at the cursor position instead of the mouse position.

Files: src/normal.c

Patch 6.0.191

Problem: When no servers are available serverlist() gives an error instead of returning an empty string. (Hari Krishna)

Solution: Don't give an error message.

Files: src/eval.c

Patch 6.0.192

Problem: When 'virtualedit' is set, "ylj" goes to the wrong column. (Andrew Nikitin)

Solution: Reset the flag that w_virtcol is valid when moving the cursor back to the start of the operated area.

Files: src/normal.c

Patch 6.0.193

Problem: When 'virtualedit' is set, col(".") after the end of the line should return one extra.

Solution: Add one to the column.

Files: src/eval.c

Patch 6.0.194

Problem: "--remote-silent" tries to send a reply to the client, like it was "--remote-wait".

Solution: Properly check for the argument.

Files: src/main.c

Patch 6.0.195

Problem: When 'virtualedit' is set and a search starts in virtual space ":call search('x')" goes to the wrong position. (Eric Long)

Solution: Reset coladd when finding a match.

Files: src/search.c

Patch 6.0.196

Problem: When 'virtualedit' is set, 'selection' is "exclusive" and visually

selecting part of a tab at the start of a line, "x" joins it with the previous line. Also, when the selection spans more than one line the whole tab is deleted.

Solution: Take coladd into account when adjusting for **'selection'** being "exclusive". Also expand a tab into spaces when deleting more than one line.

Files: src/normal.c, src/ops.c

Patch 6.0.197

Problem: When **'virtualedit'** is set and **'selection'** is "exclusive", "v\$x" doesn't delete the last character in the line. (Eric Long)

Solution: Don't reset the inclusive flag. (Helmut Stiegler)

Files: src/normal.c

Patch 6.0.198

Problem: When **'virtualedit'** is set and **'showbreak'** is not empty, moving the cursor over the line break doesn't work properly. (Eric Long)

Solution: Make getviscol() and getviscol2() use getvvcoll() to obtain the virtual cursor position. Adjust coladvance() and oneleft() to skip over the **'showbreak'** characters.

Files: src/edit.c, src/misc2.c

Patch 6.0.199

Problem: Multi-byte: could use iconv() after calling iconv_end(). (Yasuhiro Matsumoto)

Solution: Stop converting input and output stream after calling iconv_end().

Files: src/mbyte.c

Patch 6.0.200

Problem: A script that starts with "#!perl" isn't recognized as a Perl filetype.

Solution: Ignore a missing path in a script header. Also, speed up recognizing scripts by simplifying the patterns used.

Files: runtime/scripts.vim

Patch 6.0.201

Problem: When scrollbinding and doing a long jump, switching windows jumps to another position in the file. Scrolling a few lines at a time is OK. (Johannes Zellner)

Solution: When setting w_topline reset the flag that indicates w_botline is valid.

Files: src/diff.c

Patch 6.0.202

Problem: The "icon=" argument for the menu command to define a toolbar icon with a file didn't work for GTK. (Christian J. Robinson)
For Motif and Athena a full path was required.

Solution: Search the icon file using the specified path. Expand environment variables in the file name.

Files: src/gui_gtk.c, src/gui_x11.c

Patch 6.0.203

Problem: Can change **'fileformat'** even though **'modifiable'** is off. (Servatius Brandt)

Solution: Correct check for kind of set command.
Files: src/option.c

Patch 6.0.204

Problem: ":unlet" doesn't work for variables with curly braces. (Thomas Scott Urban)
Solution: Handle variable names with curly braces properly. (Vince Negri)
Files: src/eval.c

Patch 6.0.205 (extra)

Problem: "gvim -f" still forks when using the batch script to start Vim.
Solution: Add an argument to "start" to use a foreground session (Michael Geddes)
Files: src/dosinst.c

Patch 6.0.206

Problem: Unix: if expanding a wildcard in a file name results in a wildcard character and there are more parts in the path with a wildcard, it is expanded again.
Windows: ":edit \[abc]" could never edit the file "[abc]".
Solution: Don't expand wildcards in already expanded parts.
Don't remove backslashes used to escape the special meaning of a wildcard; can edit "[abc]" if '[' is removed from '**isfname**'.
Files: src/misc1.c, src/os_unix.c

Patch 6.0.207 (extra)

Problem: Win32: The shortcuts and start menu entries let Vim startup in the desktop directory, which is not very useful.
Solution: Let shortcuts start Vim in \$HOME or \$HOMEDIR\$HOMEPATH.
Files: src/dosinst.c

Patch 6.0.208

Problem: GUI: When using a keymap and the cursor is not blinking, **CTRL-^** in Insert mode doesn't directly change the cursor color. (Alex Solow)
Solution: Force a redraw of the cursor after **CTRL-^**.
Files: src/edit.c

Patch 6.0.209

Problem: GUI GTK: After selecting a '**guifont**' with the font dialog there are redraw problems for multibyte characters.
Solution: Separate the font dialog from setting the new font name to avoid that "*" is used to find wide and bold fonts.
When redrawing extra characters for the bold trick, take care of UTF-8 characters.
Files: src/gui.c, src/gui_gtk_x11.c, src/option.c, src/proto/gui.pro, src/proto/gui_gtk_x11.pro

Patch 6.0.210

Problem: After patch 6.0.167 it's no longer possible to edit a help file in another encoding than latin1.
Solution: Let the "++enc=" argument overrule the encoding.
Files: src/fileio.c

Patch 6.0.211

Problem: When reading a file fails, the buffer is empty, but it might still be possible to write it with ":w" later. The original file is lost then. (Steve Amerige)
Solution: Set the **'readonly'** option for the buffer.
Files: src/fileio.c

Patch 6.0.212

Problem: GUI GTK: confirm("foo", "") causes a crash.
Solution: Don't make a non-existing button the default. Add a default "OK" button if none is specified.
Files: src/eval.c, src/gui_gtk.c

Patch 6.0.213

Problem: When a file name contains unprintable characters, **CTRL-G** and other commands don't work well.
Solution: Turn unprintable into printable characters. (Yasuhiro Matsumoto)
Files: src/buffer.c, src/charset.c

Patch 6.0.214

Problem: When there is a buffer without a name, empty entries appear in the jumplist saved in the viminfo file.
Solution: Don't write jumplist entries without a file name.
Files: src/mark.c

Patch 6.0.215

Problem: After using "/" from Visual mode the Paste menu and Toolbar entries don't work. Pasting with the middle mouse doesn't work and modeless selection doesn't work.
Solution: Use the command line mode menus and use the mouse like in the command line.
Files: src/gui.c, src/menu.c, src/ui.c

Patch 6.0.216

Problem: After reloading a file, displayed in another window than the current one, which was changed outside of Vim the part of the file around the cursor set by autocommands may be displayed, but jumping back to the original cursor position when entering the window again.
Solution: Restore the topline of the window.
Files: src/fileio.c

Patch 6.0.217

Problem: When getting help from a help file that was used before, an empty unlisted buffer remains in the buffer list. (Eric Long)
Solution: Wipe out the buffer used to do the tag jump from.
Files: src/buffer.c, src/ex_cmds.c, src/proto/buffer.pro

Patch 6.0.218

Problem: With explorer plugin: "vim -o filename dirname" doesn't load the explorer window until entering the window.
Solution: Call s:EditDir() for each window after starting up.
Files: runtime/plugin/explorer.vim

Patch 6.0.219

Problem: ":setlocal" and ":setglobal", without arguments, display terminal options. (Zdenek Sekera)
Solution: Skip terminal options for these two commands.
Files: src/option.c

Patch 6.0.220

Problem: After patch 6.0.218 get a beep on startup. (Muraoka Taro)
Solution: Don't try going to another window when there isn't one.
Files: runtime/plugin/explorer.vim

Patch 6.0.221

Problem: When using ":bdel" and all other buffers are unloaded the lowest numbered buffer is jumped to instead of the most recent one. (Dave Cecil)
Solution: Prefer an unloaded buffer from the jumplist.
Files: src/buffer.c

Patch 6.0.222

Problem: When '**virtualedit**' is set and using autoindent, pressing Esc after starting a new line leaves behind part of the autoindent. (Helmut Stiegler)
Solution: After deleting the last char in the line adjust the cursor position in del_bytes().
Files: src/misc1.c, src/ops.c

Patch 6.0.223

Problem: When splitting a window that contains the explorer, hitting CR on a file name gives error messages.
Solution: Set the window variables after splitting the window.
Files: runtime/plugin/explorer.vim

Patch 6.0.224

Problem: When '**sidescroll**' and '**sidescrolloff**' are set in a narrow window the text may jump left-right and the cursor is displayed in the wrong position. (Aric Blumer)
Solution: When there is not enough room, compute the left column for the window to put the cursor in the middle.
Files: src/move.c

Patch 6.0.225

Problem: In Visual mode "gk" gets stuck in a closed fold. (Srinath Avadhanula)
Solution: Behave differently in a closed fold.
Files: src/normal.c

Patch 6.0.226

Problem: When doing ":recover file" get the ATTENTION prompt. After recovering the same file five times get a read error or a crash. (Alex Davis)
Solution: Set the recoverymode flag before setting the file name. Correct the amount of used memory for the size of block zero.
Files: src/ex_docmd.c

Patch 6.0.227 (extra)

Problem: The RISC OS port has several problems.
Solution: Update the makefile and fix some of the problems. (Andy Wingate)
Files: src/Make_ro.mak, src/os_riscos.c, src/os_riscos.h,
src/proto/os_riscos.pro, src/search.c

Patch 6.0.228

Problem: After putting text in Visual mode the ']' mark is not at the end of the put text.
Undo doesn't work properly when putting a word into a Visual selection that spans more than one line.
Solution: Correct the ']' mark for the deleting the Visually selected text. #ifdef code that depends on FEAT_VISUAL properly.
Also fix that "d" crossing line boundary puts '[' just before deleted text.
Fix undo by saving all deleted lines at once.
Files: src/ex_docmd.c, src/globals.h, src/normal.c, src/ops.c,
src/structs.h, src/vim.h

Patch 6.0.229

Problem: Multi-byte: With 'm' in '**formatoptions**', formatting doesn't break at a multibyte char followed by an ASCII char, and the other way around. (Muraoka Taro)
When joining lines a space is inserted between multibyte characters, which is not always wanted.
Solution: Check for multibyte character before and after the breakpoint. Don't insert a space before or after a multibyte character when joining lines and the 'M' flag is in '**formatoptions**'. Don't insert a space between multibyte characters when the 'B' flag is in '**formatoptions**'.
Files: src/edit.c, src/ops.c, src/option.h

Patch 6.0.230

Problem: The ":" used as a motion after an operator is exclusive, but sometimes it should be inclusive.
Solution: Make the "v" in between an operator and motion toggle inclusive/exclusive. (Servatius Brandt)
Files: runtime/doc/motion.txt, src/normal.c

Patch 6.0.231

Problem: "gd" and "gD" don't work when the variable matches in a comment just above the match to be found. (Servatius Brandt)
Solution: Continue searching in the first column below the comment.
Files: src/normal.c

Patch 6.0.232

Problem: "vim --version" prints on stderr while "vim --help" prints on stdout.
Solution: Make "vim --version" use stdout.
Files: runtime/doc/starting.txt, src/globals.h, src/main.c, src/message.c

Patch 6.0.233

Problem: "\\1{,8}" in a regexp is not allowed, but it should work, because there is an upper limit. (Jim Battle)

Solution: Allow using "{min,max}" after an atom that can be empty if there is an upper limit.
Files: src/regexp.c

Patch 6.0.234

Problem: It's not easy to set the cursor position without modifying marks.
Solution: Add the cursor() function. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/eval.c

Patch 6.0.235

Problem: When writing a file and renaming the original file to make the backup, permissions could change when setting the owner.
Solution: Only set the owner when it's needed and set the permissions again afterwards.
When '**backupcopy**' is "auto" check that the owner and permissions of a newly created file can be set properly.
Files: src/fileio.c

Patch 6.0.236

Problem: ":edit" without argument should move cursor to line 1 in Vi compatible mode.
Solution: Add 'g' flag to '**cptions**'.
Files: runtime/doc/options.txt, src/ex_docmd.c, src/option.h

Patch 6.0.237

Problem: In a C file, using the filetype plugin, re-indenting a comment with two spaces after the middle "*" doesn't align properly.
Solution: Don't use a middle entry from a start/middle/end to line up with the start of the comment when the start part doesn't match with the actual comment start.
Files: src/misc1.c

Patch 6.0.238

Problem: Using a ":substitute" command with a substitute() call in the substitution expression causes errors. (Srinath Avadhanula)
Solution: Save and restore pointers when doing substitution recursively.
Files: src/regexp.c

Patch 6.0.239

Problem: Using "A" to append after a Visually selected block which is after the end of the line, spaces are inserted in the wrong line and other unexpected effects. (Michael Naumann)
Solution: Don't advance the cursor to the next line.
Files: src/ops.c

Patch 6.0.240

Problem: Win32: building with Python 2.2 doesn't work.
Solution: Add support for Python 2.2 with dynamic linking. (Paul Moore)
Files: src/if_python.c

Patch 6.0.241

Problem: Win32: Expanding the old value of an option that is a path that starts with a backslash, an extra backslash is inserted.
Solution: Only insert backslashes where needed.

Also handle multibyte characters properly when removing backslashes.

Files: src/option.c

Patch 6.0.242

Problem: GUI: On a system with an Exceed X server sometimes get a "Bad Window" error. (Tommi Maekitalo)

Solution: When forking, use a pipe to wait in the parent for the child to have done the setsid() call.

Files: src/gui.c

Patch 6.0.243

Problem: Unix: "vim --version" outputs a NL before the last line instead of after it. (Charles Campbell)

Solution: Send the NL to the same output stream as the text.

Files: src/message.c, src/os_unix.c, src/proto/message.pro

Patch 6.0.244

Problem: Multi-byte: Problems with (illegal) UTF-8 characters in menu and file name (e.g., icon text, status line).

Solution: Correctly handle unprintable characters. Catch illegal UTF-8 characters and replace them with <xx>. Truncating the status line wasn't done correctly at a multibyte character. (Yasuhiro Matsumoto)

Added correct_cmdspos() and transchar_byte().

Files: src/buffer.c, src/charset.c, src/ex_getln.c, src/gui.c, src/message.c, src/screen.c, src/vim.h

Patch 6.0.245

Problem: After using a color scheme, setting the **'background'** option might not work. (Peter Horst)

Solution: Disable the color scheme if it switches **'background'** back to the wrong value.

Files: src/option.c

Patch 6.0.246

Problem: ":echomsg" didn't use the highlighting set by ":echohl". (Gary Holloway)

Solution: Use the specified attributes for the message. (Yegappan Lakshmanan)

Files: src/eval.c

Patch 6.0.247

Problem: GTK GUI: Can't use gvim in a kpart widget.

Solution: Add the "--echo-wid" argument to let Vim echo the window ID on stdout. (Philippe Fremy)

Files: runtime/doc/starting.txt, src/globals.h, src/gui_gtk_x11.c, src/main.c

Patch 6.0.248

Problem: When using compressed help files and **'encoding'** isn't "latin1", Vim converts the help file before decompressing. (David Reviejo)

Solution: Don't convert a help file when **'binary'** is set.

Files: src/fileio.c

Patch 6.0.249

Problem: "vim -t edit -c 'sta ex_help'" doesn't move cursor to edit().
Solution: Don't set the cursor on the first line for "-c" arguments when there also is a "-t" argument.
Files: src/main.c

Patch 6.0.250 (extra)

Problem: Macintosh: Various problems when compiling.
Solution: Various fixes, mostly #ifdefs. (Dany St. Amant)
Files: src/gui_mac.c, src/main.c, src/misc2.c, src/os_mac.h, src/os_mac.pbproj/project.pbxproj, src/os_unix.c

Patch 6.0.251 (extra)

Problem: Macintosh: menu shortcuts are not very clear.
Solution: Show the shortcut with the Mac clover symbol. (raindog)
Files: src/gui_mac.c

Patch 6.0.252

Problem: When a user function was defined with "abort", an error that is not inside if/endif or while/ endwhile doesn't abort the function. (Servatius Brandt)
Solution: Don't reset did_emsg when the function is to be aborted.
Files: src/ex_docmd.c

Patch 6.0.253

Problem: When '**insertmode**' is set, after "<C-O>:edit file" the next <C-O> doesn't work. (Benji Fisher) <C-L> has the same problem.
Solution: Reset need_start_insertmode once in edit().
Files: src/edit.c

Patch 6.0.254 (extra)

Problem: Borland C++ 5.5: Checking for stack overflow doesn't work correctly. Matters when using a complicated regexp.
Solution: Remove -N- from Make_bc5.mak. (Yasuhiro Matsumoto)
Files: src/Make_bc5.mak

Patch 6.0.255 (extra) (depends on patch 6.0.116 and 6.0.121)

Problem: Win32: ACL support doesn't work well on Samba drives.
Solution: Add a check for working ACL support. (Mike Williams)
Files: src/os_win32.c

Patch 6.0.256 (extra)

Problem: Win32: ":highlight Comment guifg=asdf" does not give an error message. (Randall W. Morris) Also for other systems.
Solution: Add gui_get_color() to give one error message for all systems.
Files: src/gui.c, src/gui_amiga.c, src/gui_athena.c, src/gui_motif.c, src/gui_riscos.c, src/gui_x11.c, src/gui_gtk_x11.c, src/proto/gui.pro, src/syntax.c

Patch 6.0.257

Problem: Win32: When '**mousefocus**' is set and there is a BufRead autocommand, after the dialog for permissions changed outside of Vim: '**mousefocus**' stops working. (Robert Webb)

Solution: Reset need_mouse_correct after checking timestamps.
Files: src/fileio.c

Patch 6.0.258

Problem: When '**scrolloff**' is 999 and there are folds, the text can jump up and down when moving the cursor down near the end of the file.
(Lubomir Host)

Solution: When putting the cursor halfway the window start counting lines at the end of a fold.

Files: src/move.c

Patch 6.0.259

Problem: MS-DOS: after editing the command line the cursor shape may remain like in Insert mode. (Volker Kiefel)

Solution: Reset the cursor shape after editing the command line.

Files: src/ex_getln.c

Patch 6.0.260

Problem: GUI: May crash while starting up when giving an error message for missing color. (Servatius Brandt)

Solution: Don't call gui_write() when still starting up. Don't give error message for empty color name. Don't use '**t_vb**' while the GUI is still starting up.

Files: src/fileio.c, src/gui.c, src/misc1.c, src/ui.c

Patch 6.0.261

Problem: nr2char() and char2nr() don't work with multibyte characters.

Solution: Use '**encoding**' for these functions. (Yasuhiro Matsumoto)

Files: runtime/doc/eval.txt, src/eval.c

Patch 6.0.262 (extra)

Problem: Win32: IME doesn't work properly. OnImeComposition() isn't used at all.

Solution: Adjust various things for IME.

Files: src/globals.h, src/gui_w32.c, src/mbyte.c, src/proto/ui.pro, src/structs.h, src/ui.c

Patch 6.0.263

Problem: GTK: When a dialog is closed by the window manager, Vim hangs.
(Christian J. Robinson)

Solution: Use GTK_WIDGET_DRAWABLE() instead of GTK_WIDGET_VISIBLE().

Files: src/gui_gtk.c, src/gui_gtk_x11.c

Patch 6.0.264

Problem: The amount of virtual memory is used to initialize '**maxmemtot**', which may be much more than the amount of physical memory, resulting in a lot of swapping.

Solution: Get the amount of physical memory with sysctl(), sysconf() or sysinfo() when possible.

Files: src/auto/configure, src/configure.in, src/config.h.in, src/os_unix.c, src/os_unix.h

Patch 6.0.265

Problem: Win32: Using backspace while '**fkmap**' is set causes a crash.

(Jamshid Oasjmoha)
Solution: Don't try mapping special keys.
Files: src/farsi.c

Patch 6.0.266

Problem: The rename() function deletes the file if the old and the new name are the same. (Volker Kiefel)
Solution: Don't do anything if the names are equal.
Files: src/fileio.c

Patch 6.0.267

Problem: UTF-8: Although **'isprint'** says a character is printable, utf_char2cells() still considers it unprintable.
Solution: Use vim_isprintc() for characters up to 0x100. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 6.0.268 (extra) (depends on patch 6.0.255)

Problem: Win32: ACL check crashes when using forward slash in file name.
Solution: Improve the check for the path in the file name.
Files: src/os_win32.c

Patch 6.0.269

Problem: Unprintable characters in a file name may cause problems when using the **'statusline'** option or when **'buftype'** is "nofile".
Solution: call trans_characters() for the resulting statusline. (Yasuhiro Matsumoto)
Files: src/buffer.c, src/screen.c, src/charset.c

Patch 6.0.270 (depends on patch 6.0.267)

Problem: A tab causes UTF-8 text to be displayed in the wrong position. (Ron Aaron)
Solution: Correct utf_char2cells() again.
Files: src/mbyte.c

Patch 6.1a.001 (extra)

Problem: 32bit DOS: copying text to the clipboard may cause a crash. (Jonathan D Johnston)
Solution: Don't copy one byte too much in SetClipboardData().
Files: src/os_msdos.c

Patch 6.1a.002

Problem: GTK: On some configurations, when closing a dialog from the window manager, Vim hangs.
Solution: Catch the "destroy" signal. (Aric Blumer)
Files: src/gui_gtk.c

Patch 6.1a.003

Problem: Multi-byte: With UTF-8 double-wide char and **'virtualedit'** set: yanking in Visual mode doesn't include the last byte. (Eric Long)
Solution: Don't add a space for a double-wide character.
Files: src/ops.c

Patch 6.1a.004 (extra)

Problem: MINGW: undefined type. (Ron Aaron)

Solution: Make `GetCompositionString_inUCS2()` static.
Files: `src/gui_w32.c`, `src/gui_w48.c`, `src/proto/gui_w32.pro`

Patch 6.1a.005 (extra)

Problem: Win32: `":hardcopy"` doesn't work after `":hardcopy!"`. (Jonathan Johnston)

Solution: Don't keep the driver context when using `":hardcopy!"`. (Vince Negri)

Files: `src/os_mswin.c`

Patch 6.1a.006

Problem: multibyte: after setting **'encoding'** the window title might be wrong.

Solution: Force resetting the title. (Yasuhiro Matsumoto)

Files: `src/option.c`

Patch 6.1a.007

Problem: Filetype detection for `"*.inc"` doesn't work.

Solution: Use a `":let"` command. (David Schweikert)

Files: `runtime/filetype.vim`

Patch 6.1a.008 (extra)

Problem: Win32: ACL detection for network shares doesn't work.

Solution: Include the trailing (back)slash in the root path. (Mike Williams)

Files: `src/os_win32.c`

Patch 6.1a.009

Problem: When using `"\@<="` or `"\@<!"` in a pattern, a `"\1"` may refer to a () part that follows, but it generates an error message.

Solution: Allow a forward reference when there is a following `"\@<="` or `"\@<!"`.

Files: `runtime/doc/pattern.txt`, `src/regexp.c`

Patch 6.1a.010

Problem: When using `":help"` and opening a new window, the alternate file isn't set.

Solution: Set the alternate file to the previously edited file.

Files: `src/ex_cmds.c`

Patch 6.1a.011

Problem: GTK: `":set co=77"`, change width with the mouse, `":set co=77"` doesn't resize the window. (Darren Hiebert)

Solution: Set the form size after handling a resize event.

Files: `src/gui_gtk_x11.c`

Patch 6.1a.012

Problem: GTK: The file browser always returns a full path. (Lohner)

Solution: Shorten the file name if possible.

Files: `src/gui_gtk.c`

Patch 6.1a.013

Problem: When using `"=~word"` in **'cinkeys'** or **'indentkeys'**, the case of the last character of the word isn't ignored. (Raul Segura Acevedo)

Solution: Ignore case when checking the last typed character.

Files: src/edit.c

Patch 6.1a.014

Problem: After patch 6.1a.006 can't compile without the title feature.

Solution: Add an #ifdef.

Files: src/option.c

Patch 6.1a.015

Problem: MS-Windows: When expanding a file name that contains a '[' or '{' an extra backslash is inserted. (Raul Segura Acevedo)

Solution: Avoid adding the backslash.

Files: src/ex_getln.c

Patch 6.1a.016

Problem: Completion after ":language" doesn't include "time". (Raul Segura Acevedo)

Solution: Add the alternative to the completions.

Files: src/ex_cmds2.c

Patch 6.1a.017

Problem: Clicking the mouse in the top row of a window where the first line doesn't fit moves the cursor to the wrong column.

Solution: Add the skipcol also for the top row of a window.

Files: src/ui.c

Patch 6.1a.018

Problem: When '**scrolloff**' is one and the window height is one, "gj" can put the cursor above the window. (Raul Segura Acevedo)

Solution: Don't let skipcol become bigger than the cursor column.

Files: src/move.c

Patch 6.1a.019

Problem: When using a composing character on top of an ASCII character, the "l" command clears the composing character. Only when '**ruler**' and '**showcmd**' are off. (Raphael Finkel)

Solution: Don't move the cursor by displaying characters when there are composing characters.

Files: src/screen.c

Patch 6.1a.020

Problem: GTK: after patch 6.1a.011 resizing with the mouse doesn't always work well for small sizes. (Adrien Beau)

Solution: Use another way to avoid the problem with ":set co=77".

Files: src/gui_gtk_x11.c

Patch 6.1a.021

Problem: Several Syntax menu entries are wrong or confusing.

Solution: Rephrase and correct the menu entries. (Adrien Beau)

Files: runtime/makemenu.vim, runtime/menu.vim

Patch 6.1a.022

Problem: A tags file might be used twice on case insensitive systems. (Rick Swanton)

Solution: Don't use the same file name twice in the default for the '**tags**'

option. Ignore case when comparing names of already visited files.
Files: src/misc2.c, src/option.c

Patch 6.1a.023
Problem: When starting the GUI get "C" characters echoed in the terminal.
Solution: Don't try sending a clear-screen command while the GUI is starting up.
Files: src/screen.c

Patch 6.1a.024
Problem: In other editors **CTRL-F** is often used for a find dialog.
Solution: In evim use **CTRL-F** for the find dialog.
Files: runtime/evim.vim

Patch 6.1a.025
Problem: The choices for the fileformat dialog can't be translated.
Solution: Add g:menutrans_fileformat_choices. (Adrien Beau)
Files: runtime/menu.vim

Patch 6.1a.026
Problem: Indenting Java files is wrong with "throws", "extends" and "implements" clauses.
Solution: Update the Java indent script.
Files: runtime/indent/java.vim

Patch 6.1a.027
Problem: A few Syntax menu entries missing or incorrect.
Solution: Add and correct the menu entries. (Adrien Beau)
Shorten a few menus to avoid they become too long.
Files: runtime/makemenu.vim, runtime/menu.vim

Patch 6.1a.028
Problem: XIM: problems with feedback and some input methods.
Solution: Use iconv for calculating the cells. Remove the queue for key_press_event only when text was changed. (Yasuhiro Matsumoto)
Files: src/globals.h, src/mbyte.c, src/screen.c

Patch 6.1a.029
Problem: After patch 6.1a.028 can't compile GTK version with XIM but without multibyte chars.
Solution: Add an #ifdef. (Aschwin Marsman)
Files: src/mbyte.c

Patch 6.1a.030
Problem: With double-byte encodings toupper() and tolower() may have wrong results.
Solution: Skip double-byte characters. (Eric Long)
Files: src/eval.c

Patch 6.1a.031
Problem: Accessing the **'balloondelay'** variable may cause a crash.
Solution: Make the variable for **'balloondelay'** a long. (Olaf Seibert)
Files: src/option.h

Patch 6.1a.032 (extra)

Problem: Some menu files used a wrong encoding name for "scriptencoding".

Solution: Move the translations to a separate file, which is sourced after setting "scriptencoding".

Also add Czech menu translations in ASCII and update the other encodings.

Files: runtime/lang/menu_cs_cz.iso_8859-1.vim,
runtime/lang/menu_cs_cz.iso_8859-2.vim,
runtime/lang/menu_czech_czech_republic.1250.vim,
runtime/lang/menu_czech_czech_republic.1252.vim,
runtime/lang/menu_czech_czech_republic.ascii.vim,
runtime/lang/menu_de_de.iso_8859-1.vim,
runtime/lang/menu_de_de.latin1.vim,
runtime/lang/menu_fr_fr.iso_8859-1.vim,
runtime/lang/menu_fr_fr.latin1.vim,
runtime/lang/menu_french_france.1252.vim,
runtime/lang/menu_german_germany.1252.vim,
runtime/lang/menu_ja_jp.euc-jp.vim,
runtime/lang/menu_ja_jp.utf-8.vim,
runtime/lang/menu_japanese_japan.932.vim

Patch 6.1a.033

Problem: XIM: doesn't reset input context.

Solution: call xim_reset() with im_set_active(FALSE). (Takuhiro Nishioka)

Files: src/mbyte.c

Patch 6.1a.034 (extra)

Problem: Win32: The ACL checks for a readonly file still don't work well.

Solution: Remove the ACL checks, go back to how it worked in Vim 6.0.

Files: src/os_win32.c

Patch 6.1a.035

Problem: multibyte: When using ":sh" in the GUI, typed and displayed multibyte characters are not handled correctly.

Solution: Deal with multibyte characters to and from the shell. (Yasuhiro Matsumoto) Also handle UTF-8 composing characters.

Files: src/os_unix.c

Patch 6.1a.036

Problem: GTK: the save-yourself event was not handled.

Solution: Catch the save-yourself event and preserve swap files. (Neil Bird)

Files: src/gui_gtk_x11.c

Patch 6.1a.037

Problem: The MS-Windows key mapping doesn't include CTRL-S for saving. (Vlad Sandrini)

Solution: Map CTRL-S to ":update".

Files: runtime/mswin.vim

Patch 6.1a.038

Problem: Solaris: Including both sys/sysctl.h and sys/sysinfo.h doesn't work. (Antonio Colombo)

Solution: Don't include sys/sysinfo.h when not calling sysinfo().

Files: src/os_unix.c

Patch 6.1a.039

Problem: Not all visual basic files are recognized.

Solution: Add checks to catch *.ctl files. (Raul Segura Acevedo)

Files: runtime/filetype.vim

Patch 6.1a.040

Problem: A *.pl file is recognized as Perl, but it could be a prolog file.

Solution: Check the first non-empty line. (Kontra Gergely)

Files: runtime/filetype.vim

Patch 6.1a.041

Problem: When pressing the left mouse button in the command line and then moving the mouse upwards, nearly all the text is selected.

Solution: Don't try extending a modeless selection when there isn't one.

Files: src/ui.c

Patch 6.1a.042

Problem: When merging files, ":diffput" and ":diffget" are used a lot, but they require a lot of typing.

Solution: Add "dp" for ":diffput" and "do" for ":diffget".

Files: runtime/doc/diff.txt, src/diff.c, src/normal.c, src/proto/diff.pro

Patch 6.1b.001 (extra)

Problem: Checking for wildcards in a path does not handle multibyte characters with a trail byte which is a wildcard.

Solution: Handle multibyte characters correctly. (Muraoka Taro)

Files: src/os_amiga.c, src/os_mac.c, src/os_msdos.c, src/os_mswin.c, src/os_unix.c

Patch 6.1b.002

Problem: A regexp that ends in "\{" is not flagged as an error. May cause a stack overflow when 'incsearch' is set. (Gerhard Hochholzer)

Solution: Handle a missing "}" as an error.

Files: src/regexp.c

Patch 6.1b.003 (extra)

Problem: The RISC OS GUI doesn't compile.

Solution: Include changes since Vim 5.7. (Andy Wingate)

Files: src/Make_ro.mak, src/gui_riscos.c, src/os_riscos.c, src/os_riscos.h, src/proto/gui_riscos.pro

Patch 6.1b.004

Problem: col(">") returns a negative number for linewise selection. (Neil Bird)

Solution: Don't add one to MAXCOL.

Files: src/eval.c

Patch 6.1b.005

Problem: Using a search pattern that causes an out-of-stack error while 'hlsearch' is set keeps giving the hit-Enter prompt.

A search pattern that takes a long time delays typing when

'incsearch' is set.
Solution: Stop **'hlsearch'** highlighting when the regexp causes an error.
Stop searching for **'incsearch'** when a character is typed.
Files: src/globals.h, src/message.c, src/screen.c, src/search.c,
src/vim.h

Patch 6.1b.006

Problem: When entering a composing character on the command line with
CTRL-V, the text isn't redrawn correctly.
Solution: Redraw the text under and after the cursor.
Files: src/ex_getln.c

Patch 6.1b.007

Problem: When the cursor is in the white space between two sentences, "dis"
deletes the first character of the following sentence, "das"
deletes a space after the sentence.
Solution: Backup the cursor one character in these situations.
Files: src/search.c

Patch 6.1b.008

Problem: *.xsl files are not recognized as xslt but xml.
Monk files are not recognized.
Solution: Delete the duplicate line for *.xsl. (Johannes Zellner)
Recognize monk files.
Files: runtime/filetype.vim

Patch 6.1b.009

Problem: Can't always compile small features and then adding eval feature,
"sandbox" is undefined. (Axel Kielhorn)
Solution: Always define "sandbox" when the eval feature is used.
Files: src/globals.h

Patch 6.1b.010 (extra)

Problem: When compiling gvimext.cpp with MSVC 4.2 get a number of warnings.
Solution: Change "true" to "TRUE". (Walter Briscoe)
Files: GvimExt/gvimext.cpp

Patch 6.1b.011

Problem: When using a very long string for confirm(), can't quit the
displaying at the more prompt. (Hari Krishna Dara)
Solution: Jump to the end of the message to show the choices.
Files: src/message.c

Patch 6.1b.012

Problem: Multi-byte: When **'showbreak'** is set and a double-wide character
doesn't fit at the right window edge the cursor gets stuck there.
Using cursor-left gets stuck when **'virtualedit'** is set. (Eric
Long)
Solution: Fix the way the extra ">" character is counted when **'showbreak'** is
set. Don't correct cursor for virtual editing on a double-wide
character.
Files: src/charset.c, src/edit.c

Patch 6.1b.013

Problem: A user command that partly matches with a buffer-local user command and matches full with a global user command unnecessarily gives an 'ambiguous command' error.
Solution: Find the full global match even after a partly local match.
Files: src/ex_docmd.c

Patch 6.1b.014

Problem: EBCDIC: switching mouse events off causes garbage on screen. Positioning the cursor in the GUI causes garbage.
Solution: Insert an ESC in the terminal code. (Ralf Schandl)
Use "\b" instead of "\010" for KS_LE.
Files: src/os_unix.c, src/term.c

Patch 6.1b.015

Problem: Vimtutor has a typo. Get a warning for "tempfile" if it doesn't exist.
Solution: Move a quote to the end of a line. (Max Ischenko)
Use "mktemp" first, more systems have it.
Files: src/vimtutor

Patch 6.1b.016

Problem: GTK: loading a fontset that works partly, Vim might hang or crash.
Solution: Avoid that char_width becomes zero. (Yasuhiro Matsumoto)
Files: src/gui_gtk_x11.c

Patch 6.1b.017

Problem: GUI: When using ":shell" and there is a beep, nothing happens.
Solution: Call vim_beep() to produce the beep from the shell. (Yasuhiro Matsumoto)
Files: src/message.c

Patch 6.1b.018 (depends on 6.1b.006)

Problem: When entering the encryption key, special keys may still reveal the typed characters.
Solution: Make sure stars are used or nothing is shown in all cases.
Files: src/digraph.c, src/getchar.c, src/ex_getln.c

Patch 6.1b.019 (depends on 6.1b.005)

Problem: A search pattern that takes a long time slows down typing when **'incsearch'** is set.
Solution: Pass SEARCH_PEEK to dosearch().
Files: src/ex_getln.c

Patch 6.1b.020

Problem: When using the matchit plugin, "%" finds a match on the "end" of a ":syntax region" command in Vim scripts.
Solution: Skip over ":syntax region" commands by setting b:match_skip.
Files: runtime/ftplugin/vim.vim

Patch 6.1b.021

Problem: when **'mousefocus'** is set, **CTRL-W CTRL-]** sometimes doesn't warp the pointer to the new window. (Robert Webb)
Solution: Don't reset need_mouse_correct when checking the timestamp of a file.

Files: src/fileio.c

Patch 6.1b.022

Problem: With lots of folds "j" does not obey '**scrolloff**' properly.
(Srinath Avadhanula)

Solution: Go to end of the fold before counting context lines.

Files: src/move.c

Patch 6.1b.023

Problem: On MS-Windows system() may cause checking timestamps, because Vim loses and gains input focus, while this doesn't happen on Unix.

Solution: Don't check timestamps while system() is busy.

Files: src/ex_cmds2.c, src/fileio.c, src/globals.h, src/misc1.c

Patch 6.1b.024 (extra)

Problem: Gettext 0.11 complains that "sjis" is not a standard name.

Solution: Use "cp932" instead.

Files: src/po/sjiscorr.c

Patch 6.1b.025 (extra)

Problem: Win32: When closing gvim while it is minimized and has a changed file, the file-changed dialog pops up in a corner of the screen.

Solution: Put the dialog in the middle of the screen.

Files: src/gui_w48.c

Patch 6.1b.026

Problem: When '**diffopt**' contains '**iwhite**' but not '**icase**': differences in case are not highlighted properly. (Gerhard Hochholzer)

Solution: Don't ignore case when ignoring white space differences.

Files: src/diff.c

Patch 6.1b.027

Problem: "vim --remote +" may cause a crash.

Solution: Check for missing file name argument. (Martin Kahlert)

Files: src/main.c

Patch 6.1b.028 (extra)

Problem: Win16: Can't compile after patch 6.1b.025.

Solution: Add code specifically for Win16. (Vince Negri)

Files: src/gui_w48.c

Patch 6.1b.029

Problem: Win32: When a directory on an NTFS partition is read/execute (no delete,modify,write) and the file has modify rights, trying to write the file deletes it. Making the file read/write/execute (not delete) solves it. (Mark Canup)

Solution: Use the Unix code to check for a writable directory. If not, then make a backup copy and overwrite the file.

Files: src/fileio.c

Patch 6.1b.030 (extra)

Problem: Mac: small mistake in the build script and prototypes.

Solution: Fix the build script and add the prototypes. (Axel Kielhorn)

Files: src/os_mac.build, src/gui_mac.c

Patch 6.1b.031 (extra)

Problem: Win32 GUI: ":set guifont=*" doesn't set **'guifont'** to the resulting font name. (Vlad Sandrini)

Solution: Put the code back in gui_mch_init_font() to form the font name out of the logfont.

Files: src/gui_w48.c

Patch 6.1b.032

Problem: Athena: Setting a color scheme before the GUI has started causes a crash. (Todd Blumer)

Solution: Don't try using color names that haven't been set yet.

Files: src/gui_athena.c

Patch 6.1b.033

Problem: When using a count after a ":s" command may get ml_get errors. (Dietmar Lang)

Solution: Check that the resulting range does not go past the end of the buffer.

Files: src/ex_cmds.c

Patch 6.1b.034

Problem: After sourcing mswin.vim, when using <C-S-Right> after auto-indenting and then , get warning for allocating ridiculous amount of memory. (Dave Delgreco)

Solution: Adjust the start of the Visual area when deleting the auto-indent.

Files: src/edit.c

Patch 6.1b.035

Problem: When using evim, dropping a file on Vim and then double clicking on a word, it is changed to "i". (Merlin Hansen)

Solution: Reset need_start_insertmode after editing the file.

Files: src/ex_docmd.c

VERSION 6.2

version-6.2

This section is about improvements made between version 6.1 and 6.2.

This is mainly a bug-fix release. There are also a few new features.

Main new features:

- Support for GTK 2. (Daniel Elstner)
- Support for editing Arabic text. (Nadim Shaikli & Isam Bayazidi)
- ":try" command and exception handling. (Servatius Brandt)
- Support for the neXtaw GUI toolkit (mostly like Athena). (Alexey Froloff)
- Cscope support for Win32. (Khorev Sergey)
- Support for PostScript printing in various 8-bit encodings. (Mike Williams)

Changed

changed-6.2

Removed the scheme indent file, the internal Lisp indenting works well now.

Moved the GvimEXt, OleVim and VisVim directories into the "src" directory. This is more consistent with how xxd is handled.

The VisVim.dll file is installed in the top directory, next to gvimext.dll, instead of in a subdirectory "VisVim". Fixes that NSIS was uninstalling it from the wrong directory.

Removed the art indent file, it didn't do anything.

submatch() returned line breaks with CR instead of LF.

Changed the Win32 Makefiles to become more uniform and compile gvimext.dll. (Dan Sharp)

'cindent': Align a "/*" comment with a "/*" comment in a previous line. (Helmut Stiegler)

Previously only for xterm-like terminals parent widgets were followed to find the title and icon label. Now do this for all terminal emulators.

Made it possible to recognize backslashes for "%" matching. The 'M' flag in 'coptions' disables it. (Haakon Riiser)

Removed the Make_tcc.mak makefile for Turbo C. It didn't work and we probably can't make it work (the compiler runs out of memory).

Even though the documentation refers to keywords, "[CTRL-D" was using 'isident' to find matches. Changed it to use 'iskeyword'. Also applies to other commands that search for defined words in included files such as ":dsearch", "[D" and "[d".

Made 'keywordprg' global-local. (Christian Robinson)

Enabled the Netbeans interface by default. Reversed the configure argument from "--enable-netbeans" to "--disable-netbeans".

Added

added-6.2

New options:

- 'arabic'
- 'arabicshape'
- 'ambiwidth'
- 'autochdir'
- 'casemap'
- 'copyindent'
- 'cscopequickfix'
- 'preserveindent'
- 'printencoding'
- 'rightleftcmd'
- 'termbidi'

'toolbariconsizesize'
'winfixheight'

New keymaps:

Serbian (Aleksandar Veselinovic)
Chinese Pinyin (Fredrik Roubert)
Esperanto (Antoine J. Mechelynck)

New syntax files:

Valgrind (Roger Luethi)
Smarty template (Manfred Stienstra)
MySQL (Kenneth Pronovici)
RockLinux package description (Piotr Esden-Tempski)
MMIX (Dirk Huesken)
gkrellmrc (David Necas)
Tilde (Tobias Rundtrom)
Logtalk (Paulo Moura)
PLP (Juerd Waalboer)
fvwm2m4 (David Necas)
IPfilter (Hendrik Scholz)
fstab (Radu Dineiu)
Quake (Nikolai Weibull)
Occam (Mario Schweigler)
lpc (Shizhu Pan)
Exim conf (David Necas)
EDIF (Artem Zankovich)
.cvsrc (Nikolai Weibull)
.fetchmailrc (Nikolai Weibull)
GNU gpg (Nikolai Weibull)
Grub (Nikolai Weibull)
Modconf (Nikolai Weibull)
RCS (Dmitry Vasiliev)
Art (Dorai Sitaram)
Renderman Interface Bytestream (Andrew J Bromage)
Mailcap (Doug Kearns)
Subversion commit file (Dmitry Vasiliev)
Microsoft IDL (Vadim Zeitlin)
WildPackets EtherPeek Decoder (Christopher Shinn)
Spyce (Rimon Barr)
Resolv.conf (Radu Dineiu)
A65 (Clemens Kirchgatterer)
sshconfig and sshdconfig (David Necas)
Cheetah and HTMLCheetah (Max Ischenko)
Packet filter (Camiel Dobbelaar)

New indent files:

Eiffel (David Clarke)
Tilde (Tobias Rundtrom)
Occam (Mario Schweigler)
Art (Dorai Sitaram)
PHP (Miles Lott)
Dylan (Brent Fulgham)

New tutor translations:

Slovak (Lubos Celko)
Greek (Christos Kontas)
German (Joachim Hofmann)
Norwegian (Øyvind Holm)

New filetype plugins:

Occam (Mario Schweigler)
Art (Dorai Sitaram)
ant.vim, aspvbs.vim, config.vim, csc.vim, csh.vim, dtd.vim, html.vim,
jsp.vim, pascal.vim, php.vim, sgml.vim, sh.vim, svg.vim, tcsh.vim,
xhtml.vim, xml.vim, xsd.vim. (Dan Sharp)

New compiler plugins:

Checkstyle (Doug Kearns)
g77 (Ralf Wildenhues)
fortran (Johann-Guenter Simon)
Xmllint (Doug Kearns)
Ruby (Tim Hammerquist)
Modelsim vcom (Paul Baleme)

New menu translations:

Brazilian (José de Paula)
British (Mike Williams)
Korean in UTF-8. (Nam SungHyun)
Norwegian (Øyvind Holm)
Serbian (Aleksandar Jelenak)

New message translation for Norwegian. (Øyvind Holm)

New color scheme:

desert (Hans Fugal)

Arabic specific features. **'arabicshape'**, **'termbidi'**, **'arabic'** and **'rightleftcmd'** options. (Nadim Shaikli & Isam Bayazidi)

Support for neXtaw GUI toolkit, mostly like Athena. (Alexey Froloff)

Win32: cscope support. (Khorev Sergey)

VMS: various improvements to documentation and makefiles. (Zoltan Arpadffy)

Added "x" key to the explorer plugin: execute the default action. (Yasuhiro Matsumoto)

Compile gvimext.dll with MingW. (Rene de Zwart)

Add the "tohtml.vim" plugin. It defines the ":TOhtml" user command, an easy way to convert text to HTML.

Added ":try" / ":catch" / ":finally" / ":endtry" commands. Add E999 numbers to all error messages, so that they can be caught by the number. (Servatius Brandt)

Moved part of ex_docmd.c to the new ex_eval.c source file.

Include support for GTK+ 2.2.x (Daniel Elstner)

Adds the "~" register: drag & drop text.

Adds the **'toolbariconsize'** option.

Add -Dalloca when running lint to work around a problem with alloca() prototype.

When selecting an item in the error window to jump to, take some effort to find an ordinary window to show the file in (not a preview window).

Support for PostScript printing of various 8-bit encodings. (Mike Williams)

inputdialog() accepts a third argument that is used when the dialog is cancelled. Makes it possible to see a difference between cancelling and entering nothing.

Included Aap recipes. Can be used to update Vim to the latest version, building and installing.

"/" option in **'cinoptions'**: extra indent for comment lines. (Helmut Stiegler)

Vim variable "v:register" and functions setreg(), getreg() and getregtype(). (Michael Geddes)

"v" flag in **'cptions'**: Leave text on screen with backspace in Insert mode. (Phillip Vandry)

Dosinst.exe also finds gvimext.dll in the "GvimExt" directory. Useful when running install in the "src" directory for testing.

Support tag files that were sorted with case ignored. (Flemming Madsen)

When completing a wildcard in a leading path element, as in "../*/Makefile", only the last part ("Makefile") was listed. Support custom defined command line completion. (Flemming Madsen)

Also recognize "rxvt" as an xterm-like terminal. (Tomas Styblo)

Proper X11 session management. Fixes that the WM_SAVE_YOURSELF event was not used by popular desktops. (Neil Bird)

Not used for Gnome 2, it has its own handling.

Support BOR, DEBUG and SPAWNO arguments for the Borland 3 Makefile. (Walter Briscoe)

Support page breaks for printing. Adds the "formfeed" field in **'printoptions'**. (Mike Williams)

Mac OSX: multi-language support: iconv and gettext. (Muraoka Taro, Axel Kielhorn)

"\Z" flag in patterns: ignore differences in combining characters. (Ron Aaron)

Added **'preserveindent'** and **'copyindent'** options. They use existing white space characters instead of using Tabs as much as possible. (Chris Leishman)

Updated Unicode tables to Unicode 4.0. (Raphael Finkel)

Support for the mouse wheel in rxvt. (AIDA Shinra)

Win32: Added ":8" file modifier to get short filename. Test50 tests the ":8" expansion on Win32 systems. (Michael Geddes)

'**cscopequickfix**' option: Open quickfix window for Cscope commands. Also cleanup the code for giving messages. (Khorev Sergey)

GUI: Support more than 222 columns for mouse positions.

":stopinsert" command: Don't return to Insert mode.

"interrupt" command for debug mode. Useful for simulating **CTRL-C**. (Servatius Brandt)

Fixed

fixed-6.2

Removed a few unused #defines from config.h.in, os_os2_cfg.h and os_vms_conf.h.

The Vim icons in PNG format didn't have a transparent background. (Greg Roelofs)

Fixed a large number of spelling mistakes in the docs. (Adri Verhoef)

The #defines for prototype generation were causing trouble. Changed them to typedefs.

A new version of libintl.h uses __asm__, which confuses cproto. Define a dummy __asm__ macro.

When '**virtualedit**' is set can't move to halfway an unprintable character. Cripples **CTRL-V** selection. (Taro Muraoka)

Allow moving to halfway an unprintable character. Don't let getvcol() change the pos->coladd argument.

When a tab wraps to the next line, '**listchars**' is set and '**foldcolumn**' is non-zero, only one character of the foldcolumn is highlighted. (Muraoka Taro)

When using ":catch" without an argument Vim crashes. (Yasuhiro Matsumoto)

When no argument given use the ".*" pattern.

Win32: When gvim.exe is started from a shortcut with the window style property set to maximize Vim doesn't start with a maximized window. (Yasuhiro Matsumoto) Open the window with the default size and don't call ShowWindow() again when it's already visible. (Helmut Stiegler)

gui_gtk.c used MAX, but it's undefined to avoid a conflict with system header files.

Win32: When closing a window from a mapping some pixels remain on the statusline. (Yasuhiro Matsumoto)

A column number in an errorformat that goes beyond the end of the line may cause a crash.

`":throw 'test'"` crashes Vim. (Yasuhiro Matsumoto)

The file selector's scrollbar colors are not set after doing a `":hi Scrollbar guifg=color"`. And the file selector's colors are not changed by the `colorscheme` command. (David Harrison)

Motif: When compiling with `FEAT_FOOTER` defined, the text area gets a few pixels extra space on the right. Remove the special case in `gui_get_base_width()`. (David Harrison)

Using **CTRL-R CTRL-P** in Insert mode puts the `']` mark in the wrong position. (Helmut Stiegler)

When `'formatoptions'` includes `"awct"` a non-comment wasn't auto-formatted.

Using a `"--cmd"` argument more than 10 times caused a crash.

DEC style mouse support didn't work if the page field is not empty. (Uribarri)

`"vim -l one two"` did only set `'lisp'` in the first file. Vi does it for every file.

`":set tw<"` didn't work. Was checking for `'^'` instead of `'<'`.

In `":hardcopy > %.ps"` the `"%"` was not expanded to the current filename.

Made `":redraw"` also update the Visual area.

When a not implemented command, such as `":perl"`, has wrong arguments the less important error was reported, giving the user the idea the command could work.

On non-Unix systems autocommands for writing did not attempt a match with the short file name, causing a pattern like `"a/b"` to fail.

VMS: `e_screenmode` was not defined and a few other fixes for VMS. (Zoltan Arpadffy)

`redraw_msg()` depended on `FEAT_ARABIC` instead of `FEAT_RIGHTLEFT`. (Walter Briscoe)

Various changes for the PC Makefiles. (Walter Briscoe)

Use `_truname()` instead of our own code to expand a file name into a full path. (Walter Briscoe)

Error in filetype check for `/etc/modutils`. (Lubomir Host)

Cscope interface: allocated a buffer too small.

Win16: remove a trailing backslash from a path when obtaining the permission flags. (Vince Negri)

When searching for tags with case ignored Vim could hang.

When searching directories with a stopdir could get a crash. Did not re-allocate enough memory. (Vince Negri)

A user command may cause a crash. Don't use the command index when it's negative. (Vince Negri)

putenv() didn't work for MingW and Cygwin. (Dan Sharp)

Many functions were common between os_msdos.c and os_win16.c. Use os_msdos.c for compiling the Win16 version and remove the functions from os_win16.c. (Vince Negri)

For terminals that behave like an xterm but didn't have a name that is recognized, the window title would not always be set.

When syntax highlighting is off ":hardcopy" could still attempt printing colors.

Crash when using ":catch" without an argument. (Servatius Brandt)

Win32: ":n #" doubled the backslashes.

Fixed Arabic shaping for the command line. (Nadim Shaikli)

Avoid splitting up a string displayed on the command line into individual characters, it breaks Arabic shapping.

Updated Cygwin and MingW makefiles to use more dependencies. (Dan Sharp)

2html.vim didn't work with **'nomagic'** set.

When a local argument list is used and doing ":only" Vim could crash later. (Muraoka Taro)

When using "%P" in **'statusline'** and the fillchar is "-", a percentage of 3% could result in "-3%". Also avoid changing a space inside a filename to the fill character.

MSwin: Handling of backslashes and double quotes for command line arguments was not like what other applications do. (Walter Briscoe)

Test32 sometimes didn't work, because test11.out was written as TEST11.OUT.

Avoid pointer conversions warnings for Borland C 5.5 in dosinst.c and uninstal.c.

More improvements for Make_bc3.mak file. (Walter Briscoe)

When `":syn sync linebreaks=1"` is used, editing the first line caused a redraw of the whole screen.

Making translated messages didn't work, `if_perl.xs` wasn't found. (Vlad Sandrini)

Motif and Athena: moving Vim to the foreground didn't uniconify it. Use `XMapRaised()` instead of `XRaiseWindow()`. (Srikanth Sankaran)

When using `":ptag"` in a window where `'scrollbind'` is set the preview window would also have `'scrollbind'` set. Also reset `'foldcolumn'` and `'diff'`.

Various commands that split a window took over `'scrollbind'`, which is hardly ever desired. Esp. for `"q:"` and `":copen"`. Mostly reset `'scrollbind'` when splitting a window.

When `'shellslash'` is set in the vimrc file the first entry of `":scriptnames"` would still have backslashes. Entries in the quickfix list could also have wrong (back)slashes.

Win32: printer dialog texts were not translated. (Yasuhiro Matsumoto)

When using a multibyte character with a `K_SPECIAL` byte or a special key code with `--remote-send` the received byte sequence was mangled. Put it in the typeahead buffer instead of the input buffer.

Win32: The cursor position was incorrect after changing cursor shape. (Yasuhiro Matsumoto).

Win32: When `'encoding'` is not the current codepage the title could not be set to non-ascii characters.

`"vim -d scp://machine/file1 scp://machine/file2"` did not work, there was only one window. Fixed the netrw plugin not to wipe out the buffer if it is displayed in other windows.

`"/$"` caused `"e"` in last column of screen to disappear, a highlighted blank was displayed instead.

`":s/ *\ze\n//e"` removed the line break and introduced arbitrary text. Was using the line count including what matched after the `"\ze"`.

Using the `"c"` flag with `":s"` changed the behavior when a line break is replaced and `"\@<="` is used. Without `"c"` a following match was not found.

`":%s/\vA@<=\nB@=//gce"` got stuck on `"A\nB"` when entering `"n"`.

VMS: add `HAVE_STRFTIME` in the config file. (Zoltan Arpadffy)

When a delete prompts if a delete should continue when yanking is not possible, restore `msg_silent` afterwards.

`":sign"` did not complain about a missing argument.

When adding or deleting a sign '**hlsearch**' highlighting could disappear. Use the generic functions for updating signs.

On MS-Windows NT, 2K and XP don't use command.com but cmd.exe for testing. Makes the tests work on more systems.

In the DOS tests don't create "/tmp" to avoid an error.

Mac classic: Problems with reading files with CR vs CR/LF. Rely on the library version of fgets() to work correctly for Metrowerks 2.2. (Axel Kielhorn)

When typing a password a "*" was shown for each byte instead of for each character. Added multibyte handling to displaying the stars. (Yasuhiro Matsumoto)

When using Perl 5.6 accessing \$curbuf doesn't work. Add an #ifdef to use different code for 5.6 and 5.8. (Dan Sharp)

MingW and Cygwin: Don't strip the debug executable. (Dan Sharp)

An assignment to a variable with curlies that includes "==" doesn't work. Skip over the curlies before searching for an "=". (Vince Negri)

When cancelling the selection of alternate matching tags the tag stack index could be advanced too far, resulting in an error message when using **CTRL-T**.

Patch 6.1.001

Problem: When formatting UTF-8 text it might be wrapped at a space that is followed by a composing character. (Raphael Finkel)
Also correct a display error for removing a composing char on top of a space.

Solution: Check for a composing character on a space.

Files: src/edit.c, src/misc1.c, src/screen.c

Patch 6.1.002 (extra)

Problem: Win32: after a ":popup" command the mouse pointer stays hidden.

Solution: Unhide the mouse pointer before showing the menu.

Files: src/gui_w48.c

Patch 6.1.003

Problem: When '**laststatus**' is zero and there is a vertical split, the vertical separator is drawn in the command line. (Srikant Sankaran)

Solution: Don't draw the vertical separator where there is no statusline.

Files: src/screen.c

Patch 6.1.004

Problem: Unicode 3.2 changes width and composing of a few characters. (Markus Kuhn)

Solution: Adjust the Unicode functions for the character width and composing characters.

Files: src/mbyte.c

Patch 6.1.005

Problem: When using more than 50 items in **'statusline'** Vim might crash. (Steve Hall)

Solution: Increment itemcnt in check_stl_option(). (Flemming Madsen)

Files: src/option.c

Patch 6.1.006

Problem: When using "P" in Visual mode to put linewise selected text, the wrong text is deleted. (Jakub Turski)

Solution: Put the text before the Visual area and correct the text to be deleted for the inserted lines.
Also fix that "p" of linewise text in Visual block mode doesn't work correctly.

Files: src/normal.c, src/ops.c

Patch 6.1.007

Problem: Using ":filetype plugin off" when filetype plugins were never enabled causes an error message. (Yiu Wing)

Solution: Use ":silent!" to avoid the error message.

Files: runtime/ftplugof.vim

Patch 6.1.008

Problem: The "%" command doesn't ignore "\" inside a string, it's seen as the end of the string. (Ken Clark)

Solution: Skip a double quote preceded by an odd number of backslashes.

Files: src/search.c

Patch 6.1.009

Problem: Vim crashes when using a huge number for the maxwid value in a statusline. (Robert M. Nowotniak)

Solution: Check for an overflow that makes maxwid negative.

Files: src/buffer.c

Patch 6.1.010

Problem: Searching backwards for a question mark with "?\?" doesn't work. (Alan Isaac) Same problem in ":s?\?" and ":g?\?".

Solution: Change the "\?" in a pattern to "?" when using "?" as delimiter.

Files: src/ex_cmds.c, src/ex_docmd.c, src/proto/regexp.pro, src/regexp.c, src/search.c, src/syntax.c, src/tag.c

Patch 6.1.011

Problem: XIM: doesn't work correctly when **'number'** is set. Also, a focus problem when selecting candidates.

Solution: Fix the XIM problems. (Yasuhiro Matsumoto)

Files: src/mbyte.c, src/screen.c

Patch 6.1.012

Problem: A system() call might fail if fread() does CR-LF to LF translation.

Solution: Open the output file in binary mode. (Pavol Huhás)

Files: src/misc1.c

Patch 6.1.013

Problem: Win32: The default for **'printexpr'** doesn't work when there are special characters in **'printdevice'**.
Solution: Add double quotes around the device name. (Mike Williams)
Files: runtime/doc/option.txt, src/option.c

Patch 6.1.014

Problem: An operator like "r" used in Visual block mode doesn't use **'virtualedit'** when it's set to "block".
Solution: Check for **'virtualedit'** being active in Visual block mode when the operator was started.
Files: src/ex_docmd.c, src/globals.h, src/misc2.c, src/normal.c, src/ops.c, src/undo.c

Patch 6.1.015

Problem: After patch 6.1.014 can't compile with tiny features. (Christian J. Robinson)
Solution: Add the missing define of virtual_op.
Files: src/vim.h

Patch 6.1.016 (extra)

Problem: Win32: Outputting Hebrew or Arabic text might have a problem with reversing.
Solution: Replace the RevOut() function with ETO_IGNORELANGUAGE. (Ron Aaron)
Files: src/gui_w32.c

Patch 6.1.017

Problem: Cygwin: After patch 6.1.012 Still doesn't do binary file I/O. (Pavol Juhas)
Solution: Define BINARY_FILE_IO for Cygwin.
Files: src/os_unix.h

Patch 6.1.018

Problem: Error message when using cterm highlighting. (Leonardo Di Lella)
Solution: Remove a backslash before a question mark.
Files: runtime/syntax/cterm.vim

Patch 6.1.019 (extra)

Problem: Win32: File name is messed up when editing just a drive name. (Walter Briscoe)
Solution: Append a NUL after the drive name. (Vince Negri)
Files: src/os_win32.c

Patch 6.1.020

Problem: col(">") returns a huge number after using Visual line mode.
Solution: Return the length of the line instead.
Files: src/eval.c

Patch 6.1.021 (depends on patch 6.1.009)

Problem: Vim crashes when using a huge number for the minwid value in a statusline. (Robert M. Nowotniak)
Solution: Check for an overflow that makes minwid negative.
Files: src/buffer.c

Patch 6.1.022

Problem: Grabbing the status line above the command-line window works like the bottom status line was grabbed. (Jim Battle)
Solution: Make it possible to grab the status line above the command-line window, so that it can be resized.
Files: src/ui.c

Patch 6.1.023 (extra)

Problem: VMS: running tests doesn't work properly.
Solution: Adjust the makefile. (Zoltan Arpadffy)
Files: src/testdir/Make_vms.mms

Patch 6.1.024

Problem: When header files use a new syntax for declaring functions, Vim can't figure out missing prototypes properly.
Solution: Accept braces around a function name. (M. Warner Losh)
Files: src/osdef.sh

Patch 6.1.025

Problem: Five messages for "vim --help" don't start with a capital. (Vlad Sandrini)
Solution: Make the messages consistent.
Files: src/main.c

Patch 6.1.026

Problem: *.patch files are not recognized as diff files. In a script a "VAR=val" argument after "env" isn't ignored. PHP scripts are not recognized.
Solution: Add *.patch for diff filetypes. Ignore "VAR=val". Recognize PHP scripts. (Roman Neuhauser)
Files: runtime/filetype.vim, runtime/scripts.vim

Patch 6.1.027

Problem: When '**foldcolumn**' is non-zero, a special character that wraps to the next line disturbs the foldcolumn highlighting. (Yasuhiro Matsumoto)
Solution: Only use the special highlighting when drawing text characters.
Files: src/screen.c

Patch 6.1.028

Problem: Client-server: When a --remote-expr fails, Vim still exits with status zero.
Solution: Exit Vim with a non-zero status to indicate the --remote-expr failed. (Thomas Scott Urban)
Files: src/main.c

Patch 6.1.029

Problem: When '**encoding**' is an 8-bit encoding other than "latin1", editing a utf-8 or other Unicode file uses the wrong conversion. (Jan Fedak)
Solution: Don't use Unicode to latin1 conversion for 8-bit encodings other than "latin1".
Files: src/fileio.c

Patch 6.1.030

Problem: When **CTRL-N** is mapped in Insert mode, it is also mapped after **CTRL-X CTRL-N**, while it is not mapped after **CTRL-X CTRL-F**.
(Kontra Gergely)

Solution: Don't map **CTRL-N** after **CTRL-X CTRL-N**. Same for **CTRL-P**.

Files: src/getchar.c

Patch 6.1.031

Problem: Cygwin: Xxd could read a file in text mode instead of binary mode.

Solution: Use "rb" or "rt" when needed. (Pavol Juhas)

Files: src/xxd/xxd.c

Patch 6.1.032

Problem: Can't specify a quickfix file without jumping to the first error.

Solution: Add the ":cgetfile" command. (Yegappan Lakshmanan)

Files: runtime/doc/index.txt, runtime/doc/quickfix.txt, src/ex_cmds.h,
src/quickfix.c

Patch 6.1.033

Problem: GUI: When the selection is lost and the Visual highlighting is changed to underlining, the cursor is left in a different position. (Christian Michon)

Solution: Update the cursor position after redrawing the selection.

Files: src/ui.c

Patch 6.1.034

Problem: A CVS diff file isn't recognized as diff filetype.

Solution: Skip lines starting with "? " before checking for an "Index:" line.

Files: runtime/scripts.vim

Patch 6.1.035 (extra, depends on 6.1.016)

Problem: Win32: Outputting Hebrew or Arabic text might have a problem with reversing on MS-Windows 95/98/ME.

Solution: Restore the RevOut() function and use it in specific situations only. (Ron Aaron)

Files: src/gui_w32.c

Patch 6.1.036

Problem: This command may cause a crash: ":v/./,/-j". (Ralf Arens)

Solution: Compute the right length of the regexp when it's empty.

Files: src/search.c

Patch 6.1.037

Problem: When '**lazyredraw**' is set, pressing "q" at the hit-enter prompt causes an incomplete redraw and the cursor isn't positioned.
(Lubomir Host)

Solution: Overrule '**lazyredraw**' when do_redraw is set.

Files: src/main.c, src/screen.c

Patch 6.1.038

Problem: Multi-byte: When a ":s" command contains a multibyte character where the trail byte is '~' the text is messed up.

Solution: Properly skip multibyte characters in regtilde() (Muraoka Taro)

Files: src/regexp.c

Patch 6.1.039

Problem: When folds are defined and the file is changed outside of Vim, reloading the file doesn't update the folds. (Anders Schack-Nielsen)

Solution: Recompute the folds after reloading the file.

Files: src/fileio.c

Patch 6.1.040

Problem: When changing directory for expanding a file name fails there is no error message.

Solution: Give an error message for this situation. Don't change directory if we can't return to the original directory.

Files: src/diff.c, src/ex_docmd.c, src/globals.h, src/misc1.c, src/os_unix.c

Patch 6.1.041

Problem: ":mkvimrc" doesn't handle a mapping that has a leading space in the rhs. (Davyd Ondrejko)

Solution: Insert a **CTRL-V** before the leading space. Also display leading and trailing white space in <> form.

Files: src/getchar.c, src/message.c

Patch 6.1.042

Problem: "vim -r" doesn't show all matches when 'wildignore' removes swap files. (Steve Talley)

Solution: Keep all matching swap file names.

Files: src/memline.c

Patch 6.1.043

Problem: After patch 6.1.040 a few warnings are produced.

Solution: Add a type cast to "char *" for mch_chdir(). (Axel Kielhorn)

Files: src/diff.c, src/ex_docmd.c, src/misc1.c, src/os_unix.c

Patch 6.1.044 (extra)

Problem: GUI: When using the find/replace dialog with text that contains a slash, an invalid substitute command is generated.

On Win32 a find doesn't work when 'insertmode' is set.

Solution: Escape slashes with a backslash.

Make the Win32, Motif and GTK gui use common code for the find/replace dialog.

Add the "match case" option for Motif and GTK.

Files: src/feature.h, src/proto/gui.pro, src/gui.c, src/gui.h, src/gui_motif.c, src/gui_gtk.c, src/gui_w48.c

Patch 6.1.045

Problem: In Visual mode, with lots of folds and 'scrolloff' set to 999, moving the cursor down near the end of the file causes the text to jump up and down. (Lubomir Host)

Solution: Take into account that the cursor may be on the last line of a closed fold.

Files: src/move.c

Patch 6.1.046

Problem: X11 GUI: ":set lsp=2 gcr=n-v-i:hor1-blinkon0" draws a black rectangle. ":set lsp=2 gcr=n-v-i:hor10-blinkon0" makes the cursor disappear. (Nam SungHyun)
Solution: Correctly compute the height of the horizontal cursor.
Files: src/gui_gtk_x11.c, src/gui_x11.c

Patch 6.1.047

Problem: When skipping commands after an error was encountered, expressions for ":if", ";elseif" and ":while" are still evaluated.
Solution: Skip the expression after an error. (Servatius Brandt)
Files: src/ex_docmd.c

Patch 6.1.048

Problem: Unicode 3.2 changes were missing a few Hangul Jamo characters.
Solution: Recognize more characters as composing characters. (Jungshik Shin)
Files: src/mbyte.c

Patch 6.1.049 (extra)

Problem: On a 32 bit display a valid color may cause an error message, because its pixel value is negative. (Chris Paulson-Ellis)
Solution: Check for -11111 instead of the color being negative. Don't add one to the pixel value, -1 may be used for white.
Files: src/globals.h, src/gui.c, src/gui.h, src/gui_amiga.c, src/gui_athena.c, src/gui_beos.cc, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_riscos.c, src/gui_w16.c, src/gui_w32.c, src/gui_w48.c, src/gui_x11.c, src/mbyte.c, src/syntax.c

Patch 6.1.050 (depends on 6.1.049)

Problem: After patch 6.1.049 the non-GUI version doesn't compile.
Solution: Add an #ifdef FEAT_GUI. (Robert Stanton)
Files: src/syntax.c

Patch 6.1.051 (depends on 6.1.044)

Problem: Doesn't compile with GUI and small features.
Solution: Adjust the #if for ga_append().
Files: src/misc2.c

Patch 6.1.052

Problem: Unix: The executable() function doesn't work when the "which" command isn't available.
Solution: Go through \$PATH manually. Also makes it work for VMS.
Files: src/os_unix.c

Patch 6.1.053

Problem: When '**sessionoptions**' contains "globals", or "localoptions" and an option value contains a line break, the resulting script is wrong.
Solution: Use "\n" and "\r" for a line break. (Srinath Avadhanula)
Files: src/eval.c

Patch 6.1.054

Problem: GUI: A mouse click is not recognized at the more prompt, even when '**mouse**' includes 'r'.
Solution: Recognize a mouse click at the more prompt.

Also accept a mouse click in the last line in the GUI.
Add "ml" entry in **'mouseshape'**.

Files: src/gui.c, src/message.c, src/misc1.c, src/misc2.c, src/option.c,
src/structs.h

Patch 6.1.055

Problem: When editing a compressed file, Vim will inspect the contents to guess the filetype.

Solution: Don't source scripts.vim for .Z, .gz, .bz2, .zip and .tgz files.

Files: runtime/filetype.vim, runtime/plugin/gzip.vim

Patch 6.1.056

Problem: Loading the Syntax menu can take quite a bit of time.

Solution: Add the "skip_syntax_sel_menu" variable. When it's defined the available syntax files are not in the Syntax menu.

Files: runtime/doc/gui.txt, runtime/menu.vim

Patch 6.1.057

Problem: An ESC inside a mapping doesn't work as documented when **'insertmode'** is set, it does go from Visual or Normal mode to Insert mode. (Benji Fisher)

Solution: Make it work as documented.

Files: src/normal.c

Patch 6.1.058

Problem: When there is a closed fold just above the first line in the window, using **CTRL-X CTRL-Y** in Insert mode will show only one line of the fold. (Alexey Marinichev)

Solution: Correct the topline by putting it at the start of the fold.

Files: src/move.c

Patch 6.1.059

Problem: ":redir > ~/file" doesn't work. (Stephen Rasku)

Solution: Expand environment variables in the ":redir >" argument.

Files: src/ex_docmd.c

Patch 6.1.060

Problem: When **'virtualedit'** is set and **'selection'** is "exclusive", deleting a character just before a tab changes the tab into spaces. Undo doesn't restore the tab. (Helmut Stiegler)

Solution: Don't replace the tab by spaces when it's not needed. Correctly save the line before it's changed.

Files: src/ops.c

Patch 6.1.061

Problem: When **'virtualedit'** is set and **'selection'** is "exclusive", a Visual selection that ends just after a tab doesn't include that tab in the highlighting. (Helmut Stiegler)

Solution: Use a different way to exclude the character under the cursor.

Files: src/screen.c

Patch 6.1.062

Problem: The "man" filetype plugin doesn't work properly on Solaris 5.

Solution: Use a different way to detect that "man -s" should be used. (Hugh

Sasse)
Files: runtime/ftplugin/man.vim

Patch 6.1.063

Problem: Java indenting doesn't work properly.
Solution: Ignore comments when checking if the indent doesn't increase after a "}".
Files: runtime/indent/java.vim

Patch 6.1.064

Problem: The URLs that the netrw plugin recognized for ftp and rcp did not conform to the standard method://[user@]host[:port]/path.
Solution: Use `ftp://[user@]host[[:#]port]/path`, which supports both the new and the previous style. Also added a bit of dav/cadaver support. (Charles Campbell)
Files: runtime/plugin/netrw.vim

Patch 6.1.065

Problem: VMS: The colorscheme, keymap and compiler menus are not filled in.
Solution: Ignore case when looking for ".vim" files. (Coen Engelbarts)
Files: runtime/menu.vim

Patch 6.1.066 (extra)

Problem: When calling system() in a plugin reading stdin hangs.
Solution: Don't set the terminal to RAW mode when it wasn't in RAW mode before the system() call.
Files: src/os_amiga.c, src/os_msdos.c, src/os_riscos.c, src/os_unix.c, src/os_win16.c, src/os_win32.c

Patch 6.1.067

Problem: ":set viminfo+=f0" is not working. (Benji Fisher)
Solution: Check the "f" flag instead of "" in '**viminfo**'.
Files: src/mark.c

Patch 6.1.068

Problem: When a file is reloaded after it was changed outside of Vim, diff mode isn't updated. (Michael Naumann)
Solution: Invalidate the diff info so that it's updated when needed.
Files: src/fileio.c

Patch 6.1.069

Problem: When '**showmatch**' is set and "\$" is in '**cptions**', using "C}<Esc>" may forget to remove the "\$". (Preben Guldborg)
Solution: Restore dollar_vcol after displaying the matching cursor position.
Files: src/search.c

Patch 6.1.070 (depends on 6.1.060)

Problem: Compiler warning for signed/unsigned mismatch. (Mike Williams)
Solution: Add a typecast to int.
Files: src/ops.c

Patch 6.1.071

Problem: When '**selection**' is exclusive, g **CTRL-G** in Visual mode counts one character too much. (David Necas)

Solution: Subtract one from the end position.
Files: src/ops.c

Patch 6.1.072

Problem: When a file name in a tags file starts with http:// or something else for which there is a BufReadCmd autocommand, the file isn't opened anyway.

Solution: Check if there is a matching BufReadCmd autocommand and try to open the file.

Files: src/fileio.c, src/proto/fileio.pro, src/tag.c

Patch 6.1.073 (extra)

Problem: BC5: Can't easily specify a tiny, small, normal, big or huge version.

Solution: Allow selecting the version with the FEATURES variable. (Ajit Thakkar)

Files: src/Make_bc5.mak

Patch 6.1.074

Problem: When '**cdpath**' includes "../..", changing to a directory in which we currently already are doesn't work. ff_check_visited() adds the directory both when using it as the root for searching and for the actual matches. (Stephen Rasku)

Solution: Use a separate list for the already searched directories.

Files: src/misc2.c

Patch 6.1.075 (depends on 6.1.072)

Problem: Can't compile fileio.c on MS-Windows.

Solution: Add a declaration for the "p" pointer. (Madoka Machitani)

Files: src/fileio.c

Patch 6.1.076 (extra)

Problem: Macintosh: explorer plugin doesn't work on Mac Classic. IME doesn't work. Dialog boxes don't work on Mac OS X

Solution: Fix explorer plugin and key modifiers. (Axel Kielhorn)
Fix IME support. (Muraoka Taro)

Disable dialog boxes. (Benji Fisher)

Files: src/edit.c, src/feature.h, src/gui_mac.c, src/os_mac.c

Patch 6.1.077

Problem: On a Debian system with ACL linking fails. (Lubomir Host)

Solution: When the "acl" library is used, check if the "attr" library is present and use it.

Files: src/auto/configure, src/configure.in, src/link.sh

Patch 6.1.078

Problem: When using '**foldmethod**' "marker" and the end marker appears before the start marker in the file, no fold is found. (Nazri Ramliy)

Solution: Don't let the fold depth go negative.

Files: src/fold.c

Patch 6.1.079

Problem: When using "s" in Visual block mode with '**virtualedit**' set, when the selected block is after the end of some lines the wrong text

is inserted and some lines are skipped. (Servatius Brandt)
Solution: Insert the right text and extend short lines.
Files: src/ops.c

Patch 6.1.080

Problem: When using gcc with /usr/local already in the search path, adding it again causes problems.
Solution: Adjust configure.in to avoid adding /usr/local/include and /usr/local/lib when using GCC and they are already used. (Johannes Zellner)
Files: src/auto/configure, src/configure.in

Patch 6.1.081

Problem: ":help CTRL-_CTRL-N" doesn't work. (Christian J. Robinson)
Solution: Double the backslash to avoid the special meaning of "_".
Files: src/ex_cmds.c

Patch 6.1.082

Problem: On MS-Windows the vimrc_example.vim script is sourced and then mswin.vim. This enables using select mode, but since "p" is mapped it doesn't replace the selection.
Solution: Remove the mapping of "p" from vimrc_example.vim, it's obsolete. (Vlad Sandrini)
Files: runtime/vimrc_example.vim

Patch 6.1.083

Problem: When \$LANG is "sk" or "sk_sk", the Slovak menu file isn't found. (Martin Lacko)
Solution: Guess the right menu file based on the system.
Files: runtime/lang/menu_sk_sk.vim

Patch 6.1.084 (depends on 6.1.080)

Problem: "include" and "lib" are mixed up when checking the directories gcc already searches.
Solution: Swap the variable names. (SunHo Kim)
Files: src/auto/configure, src/configure.in

Patch 6.1.085

Problem: When using CTRL-O CTRL-\ CTRL-N from Insert mode, the displayed mode "(insert)" isn't removed. (Benji Fisher)
Solution: Clear the command line.
Files: src/normal.c

Patch 6.1.086 (depends on 6.1.049)

Problem: The guifg color for CursorIM doesn't take effect.
Solution: Use the foreground color when it's defined. (Muraoka Taro)
Files: src/gui.c

Patch 6.1.087

Problem: A thesaurus with Japanese characters has problems with characters in different word classes.
Solution: Only separate words with single-byte non-word characters. (Muraoka Taro)
Files: src/edit.c

Patch 6.1.088 (extra)

Problem: Win32: no debugging info is generated. Tags file excludes .cpp files.

Solution: Add "/map" to compiler flags. Add "*.cpp" to ctags command.
(Muraoka Taro)

Files: src/Make_mvc.mak

Patch 6.1.089

Problem: On BSDI systems there is no ss_sp field in stack_t. (Robert Jan)

Solution: Use ss_base instead.

Files: src/auto/configure, src/configure.in, src/config.h.in,
src/os_unix.c

Patch 6.1.090

Problem: **CTRL-F** gets stuck when '**scrolloff**' is non-zero and there is a mix of long wrapping lines and a non-wrapping line.

Solution: Check that **CTRL-F** scrolls at least one line.

Files: src/move.c

Patch 6.1.091

Problem: GTK: Can't change preeditstate without setting '**imactivatekey**'.

Solution: Add some code to change preeditstate for OnTheSpot. (Yasuhiro Matsumoto)

Files: src/mbyte.c

Patch 6.1.092

Problem: ":mapclear <buffer>" doesn't work. (Srikanth Adayapalam)

Solution: Allow an argument for ":mapclear".

Files: src/ex_cmds.h

Patch 6.1.093 (extra)

Problem: Mac and MS-Windows GUI: when scrolling while ":s" is working the results can be messed up, because the cursor is moved.

Solution: Disallow direct scrolling when not waiting for a character.

Files: src/gui_mac.c, src/gui_w16.c, src/gui_w32.c, src/gui_w48.c

Patch 6.1.094

Problem: Cygwin: Passing a file name that has backslashes isn't handled very well.

Solution: Convert file name arguments to Posix. (Chris Metcalf)

Files: src/main.c

Patch 6.1.095

Problem: When using signs can free an item on the stack.

Overruling sign colors doesn't work. (Srikanth Sankaran)

Solution: Don't free the item on the stack. Use NULL instead of "none" for the value of the color.

Files: src/gui_x11.c

Patch 6.1.096

Problem: When erasing the right half of a double-byte character, it may cause further characters to be erased. (Yasuhiro Matsumoto)

Solution: Make sure only one character is erased.

Files: src/screen.c

Patch 6.1.097 (depends on 6.1.090)

Problem: When **'scrolloff'** is set to a huge value, **CTRL-F** at the end of the file scrolls one line. (Lubomir Host)

Solution: Don't scroll when **CTRL-F** detects the end-of-file.

Files: src/move.c

Patch 6.1.098

Problem: MS-Windows: When the xxd program is under "c:\program files" the "Convert to Hex" menu doesn't work. (Brian Mathis)

Solution: Put the path to xxd in double quotes.

Files: runtime/menu.vim

Patch 6.1.099

Problem: Memory corrupted when closing a fold with more than 99999 lines.

Solution: Allocate more space for the fold text. (Walter Briscoe)

Files: src/eval.c

Patch 6.1.100 (extra, depends on 6.1.088)

Problem: Win32: VC5 and earlier don't support the /mapinfo option.

Solution: Add "/mapinfo" only when "MAP=lines" is specified. (Muraoka Taro)

Files: src/Make_mvc.mak

Patch 6.1.101

Problem: After using ":options" the tabstop of a new window is 15. Entry in ":options" window for **'autowriteall'** is wrong. (Antoine J Mechelynck) Can't insert a space in an option value.

Solution: Use ":setlocal" instead of ":set". Change "aw" to "awa". Don't map space in Insert mode.

Files: runtime/optwin.vim

Patch 6.1.102

Problem: Unprintable and multibyte characters in a statusline item are not truncated correctly. (Yasuhiro Matsumoto)

Solution: Count the width of characters instead of the number of bytes.

Files: src/buffer.c

Patch 6.1.103

Problem: A function returning from a while loop, with **'verbose'** set to 12 or higher, doesn't mention the return value. A function with the **'abort'** attribute may return -1 while the verbose message says something else.

Solution: Move the verbose message about returning from a function to call_func(). (Servatius Brandt)

Files: src/eval.c

Patch 6.1.104

Problem: GCC 3.1 appears to have an optimizer problem that makes test 3 crash.

Solution: For GCC 3.1 add -fno-strength-reduce to avoid the optimizer bug. Filter out extra info from "gcc --version".

Files: src/auto/configure, src/configure.in

Patch 6.1.105

Problem: Win32: The default for **'shellpipe'** doesn't redirect stderr. (Dion Nicolaas)
Solution: Redirect stderr, depending on the shell (like for **'shellredir'**).
Files: src/option.c

Patch 6.1.106

Problem: The maze program crashes.
Solution: Change "11" to "27" and it works. (Greg Roelofs)
Files: runtime/macros/maze/mazeansi.c

Patch 6.1.107

Problem: When **'list'** is set the current line in the error window may be displayed wrong. (Muraoka Taro)
Solution: Don't continue the line after the \$ has been displayed and the rightmost column is reached.
Files: src/screen.c

Patch 6.1.108

Problem: When interrupting a filter command such as "!!sleep 20" the file becomes read-only. (Mark Brader)
Solution: Only set the read-only flag when opening a buffer is interrupted. When the shell command was interrupted, read the output that was produced so far.
Files: src/ex_cmds.c, src/fileio.c

Patch 6.1.109

Problem: When **'eadirection'** is "hor", using **CTRL-W** = doesn't equalize the window heights. (Roman Neuhauser)
Solution: Ignore **'eadirection'** for **CTRL-W** =
Files: src/window.c

Patch 6.1.110

Problem: When using ":badd file" when "file" is already present but not listed, it stays unlisted. (David Frey)
Solution: Set **'buflisted'**.
Files: src/buffer.c

Patch 6.1.111

Problem: It's not possible to detect using the Unix sources on Win32 or Mac.
Solution: Add has("macunix") and has("win32unix").
Files: runtime/doc/eval.txt, src/eval.c

Patch 6.1.112

Problem: When using ":argdo", ":bufdo" or ":windo", **CTRL-O** doesn't go to the cursor position from before this command but every position where the argument was executed.
Solution: Only remember the cursor position from before the ":argdo", ":bufdo" and ":windo".
Files: src/ex_cmds2.c, src/mark.c

Patch 6.1.113

Problem: ":bufdo bwipe" only wipes out half the buffers. (Roman Neuhauser)
Solution: Decide what buffer to go to next before executing the command.

Files: src/ex_cmds2.c

Patch 6.1.114

Problem: ":python import vim", ":python vim.current.buffer[0:0] = []" gives a lalloc(0) error. (Chris Southern)

Solution: Don't allocate an array when it's size is zero.

Files: src/if_python.c

Patch 6.1.115

Problem: "das" on the white space at the end of a paragraph does not delete the "." the sentence ends with.

Solution: Don't exclude the last character when it is not white space.

Files: src/search.c

Patch 6.1.116

Problem: When 'endoffline' is changed while 'binary' is set a file should be considered modified. (Olaf Buddenhagen)

Solution: Remember the 'eol' value when editing started and consider the file changed when the current value is different and 'binary' is set. Also fix that the window title isn't updated when 'ff' or 'bin' changes.

Files: src/option.c, src/structs.h

Patch 6.1.117

Problem: Small problem with editing a file over ftp: and with Cygwin.

Solution: Remove a dot from a ":normal" command. Use "cygdrive" where appropriate. (Charles Campbell)

Files: runtime/plugin/netrw.vim

Patch 6.1.118

Problem: When a file in diff mode is reloaded because it changed outside of Vim, other windows in diff mode are not always updated. (Michael Naumann)

Solution: After reloading a file in diff mode mark all windows in diff mode for redraw.

Files: src/diff.c

Patch 6.1.119 (extra)

Problem: With the Sniff interface, using Sniff 4.0.X on HP-UX, there may be a crash when connecting to Sniff.

Solution: Initialize sniff_rq_sep such that its value can be changed. (Martin Egloff)

Files: src/if_sniff.c

Patch 6.1.120 (depends on 6.1.097)

Problem: When 'scrolloff' is non-zero and there are folds, CTRL-F at the end of the file scrolls part of a closed fold. (Lubomir Host)

Solution: Adjust the first line to the start of a fold.

Files: src/move.c

Patch 6.1.121 (depends on 6.1.098)

Problem: When starting Select mode from Insert mode, then using the Paste menu entry, the cursor is left before the last pasted character. (Mario Schweigler)

Solution: Set the cursor for Insert mode one character to the right.
Files: runtime/menu.vim

Patch 6.1.122

Problem: `":file name"` creates a new buffer to hold the old buffer name, which becomes the alternate file. This buffer is unexpectedly listed.

Solution: Create the buffer for the alternate name unlisted.
Files: src/ex_cmds.c

Patch 6.1.123

Problem: A `":match"` command with more than one argument doesn't report an error.

Solution: Check for extra characters. (Servatius Brandt)
Files: src/ex_docmd.c

Patch 6.1.124

Problem: When trying to exit and there is a hidden buffer that had `'eol'` off and `'bin'` set exiting isn't possible. (John McGowan)

Solution: Set `b_start_eol` when clearing the buffer.
Files: src/buffer.c

Patch 6.1.125

Problem: Explorer plugin asks for saving a modified buffer even when it's open in another window as well.

Solution: Count the number of windows using the buffer.
Files: runtime/plugin/explorer.vim

Patch 6.1.126

Problem: Adding the choices in the syntax menu is consuming much of the startup time of the GUI while it's not often used.

Solution: Only add the choices when the user wants to use them.
Files: Makefile, runtime/makemenu.vim, runtime/menu.vim, runtime/synmenu.vim, src/Makefile

Patch 6.1.127

Problem: When using `"--remote file"` and the server has `'insertmode'` set, commands are inserted instead of being executed. (Niklas Volbers)

Solution: Go to Normal mode again after the `":drop"` command.
Files: src/main.c

Patch 6.1.128

Problem: The expression `"input('very long prompt')"` puts the cursor in the wrong line (column is OK).

Solution: Add the wrapped lines to the indent. (Yasuhiro Matsumoto)
Files: src/ex_getln.c

Patch 6.1.129

Problem: On Solaris editing `"file/"` and then `"file"` results in using the same buffer. (Jim Battle)

Solution: Before using `stat()`, check that there is no illegal trailing slash.

Files: src/auto/configure, src/config.h.in, src/configure.in, src/macros.h src/misc2.c, src/proto/misc2.pro

Patch 6.1.130

Problem: The documentation for some of the **'errorformat'** items is unclear.
Solution: Add more examples and explain hard to understand items. (Stefan Roemer)
Files: runtime/doc/quickfix.txt

Patch 6.1.131

Problem: X11 GUI: when expanding a CSI byte in the input stream to K_CSI, the CSI byte itself isn't copied.
Solution: Copy the CSI byte.
Files: src/gui_x11.c

Patch 6.1.132

Problem: Executing a register in Ex mode may cause commands to be skipped. (John McGowan)
Solution: In Ex mode use an extra check if the register contents was consumed, to avoid input goes into the typeahead buffer.
Files: src/ex_docmd.c

Patch 6.1.133

Problem: When drawing double-wide characters in the statusline, may clear half of a character. (Yasuhiro Matsumoto)
Solution: Force redraw of the next character by setting the attributes instead of putting a NUL in ScreenLines[]. Do put a NUL in ScreenLines[] when overwriting half of a double-wide character.
Files: src/screen.c

Patch 6.1.134

Problem: An error for a trailing argument of ":match" should not be given after ":if 0". (Servatius Brandt)
Solution: Only do the check when executing commands.
Files: src/ex_docmd.c

Patch 6.1.135

Problem: Passing a command to the shell that includes a newline always has a backslash before the newline.
Solution: Remove one backslash before the newline. (Servatius Brandt)
Files: src/ex_docmd.c

Patch 6.1.136

Problem: When \$TERM is "linux" the default for **'background'** is "dark", even though the GUI uses a light background. (Hugh Allen)
Solution: Don't mark the option as set when defaulting to "dark" for the linux console. Also reset **'background'** to "light" when the GUI has a light background.
Files: src/option.c

Patch 6.1.137

Problem: Converting to HTML has a clumsy way of dealing with tabs which may change the highlighting.
Solution: Replace tabs with spaces after converting a line to HTML. (Preben Guldborg)
Files: runtime/syntax/2html.vim

Patch 6.1.138 (depends on 6.1.126)

Problem: Adding extra items to the Syntax menu can't be done when the "Show individual choices" menu is used.

Solution: Use ":runtime!" instead of ":source", so that all synmenu.vim files in the runtime path are loaded. (Servatius Brandt)
Also fix that a translated menu can't be removed.

Files: runtime/menu.vim

Patch 6.1.139

Problem: Cygwin: PATH_MAX is not defined.

Solution: Include limits.h. (Dan Sharp)

Files: src/main.c

Patch 6.1.140

Problem: Cygwin: ":args `ls *.c`" does not work if the shell command produces CR NL line separators.

Solution: Remove the CR characters ourselves. (Pavol Juhas)

Files: src/os_unix.c

Patch 6.1.141

Problem: ":wincmd gx" may cause problems when mixed with other commands. ":wincmd c" doesn't close the window immediately. (Benji Fisher)

Solution: Pass the extra command character directly instead of using the stuff buffer and call ex_close() directly.

Files: src/ex_docmd.c, src/normal.c, src/proto/normal.pro, src/proto/window.pro, src/window.c

Patch 6.1.142

Problem: Defining paragraphs without a separating blank line isn't possible. Paragraphs can't be formatted automatically.

Solution: Allow defining paragraphs with lines that end in white space. Added the 'w' and 'a' flags in '**formatoptions**'.

Files: runtime/doc/change.txt, src/edit.c, src/misc1.c, src/normal.c, src/option.h, src/ops.c, src/proto/edit.pro, src/proto/ops.pro, src/vim.h

Patch 6.1.143 (depends on 6.1.142)

Problem: Auto formatting near the end of the file moves the cursor to a wrong position. In Insert mode some lines are made one char too narrow. When deleting a line undo might not always work properly.

Solution: Don't always move to the end of the line in the last line. Don't position the cursor past the end of the line in Insert mode. After deleting a line save the cursor line for undo.

Files: src/edit.c, src/ops.c, src/normal.c

Patch 6.1.144

Problem: Obtaining the size of a line in screen characters can be wrong. A pointer may wrap around zero.

Solution: In win_linetabsize() check for a MAXCOL length argument. (Jim Dunleavy)

Files: src/charset.c

Patch 6.1.145

Problem: GTK: Drag&drop with more than 3 files may cause a crash. (Mickael Marchand)
Solution: Rewrite the code that parses the received list of files to be more robust.
Files: src/charset.c, src/gui_gtk_x11.c

Patch 6.1.146

Problem: MS-Windows: When \$HOME is constructed from \$HOMEDRIVE and \$HOMEPATH, it is not used for storing the _viminfo file. (Normal Diamond)
Solution: Set \$HOME with the value obtained from \$HOMEDRIVE and \$HOMEPATH.
Files: src/misc1.c

Patch 6.1.147 (extra)

Problem: MS-Windows: When a dialog has no default button, pressing Enter ends it anyway and all buttons are selected.
Solution: Don't end a dialog when there is no default button. Don't select all button when there is no default. (Vince Negri)
Files: src/gui_w32.c

Patch 6.1.148 (extra)

Problem: MS-Windows: ACL is not properly supported.
Solution: Add an access() replacement that also works for ACL. (Mike Williams)
Files: runtime/doc/editing.txt, src/os_win32.c

Patch 6.1.149 (extra)

Problem: MS-Windows: Can't use diff mode from the file explorer.
Solution: Add a "diff with Vim" context menu entry. (Dan Sharp)
Files: GvimExt/gvimext.cpp, GvimExt/gvimext.h

Patch 6.1.150

Problem: OS/2, MS-Windows and MS-DOS: When '**shellslash**' is set getcwd() still uses backslash. (Yegappan Lakshmanan)
Solution: Adjust slashes in getcwd().
Files: src/eval.c

Patch 6.1.151 (extra)

Problem: Win32: The NTFS substream isn't copied.
Solution: Copy the substream when making a backup copy. (Muraoka Taro)
Files: src/fileio.c, src/os_win32.c, src/proto/os_win32.pro

Patch 6.1.152

Problem: When \$LANG is iso8859-1 translated menus are not used.
Solution: Change iso8859 to iso_8859.
Files: runtime/menu.vim

Patch 6.1.153

Problem: Searching in included files may search recursively when the path starts with "../". (Sven Berkvens-Matthijsse)
Solution: Compare full file names, use inode/device when possible.
Files: src/search.c

Patch 6.1.154 (extra)

Problem: DJGPP: "vim -h" leaves the cursor in a wrong position.
Solution: Don't position the cursor using uninitialized variables. (Jim Dunleavy)
Files: src/os_msdos.c

Patch 6.1.155

Problem: Win32: Cursor may sometimes disappear in Insert mode.
Solution: Change "hor10" in '**guicursor**' to "hor15". (Walter Briscoe)
Files: src/option.c

Patch 6.1.156

Problem: Conversion between DBCS and UCS-2 isn't implemented cleanly.
Solution: Clean up a few things.
Files: src/mbyte.c, src/structs.h

Patch 6.1.157

Problem: '**hlsearch**' highlights only the second comma in ",,,," with "/, \@<=[^,]*". (Preben Guldberg)
Solution: Also check for an empty match to start just after a previous match.
Files: src/screen.c

Patch 6.1.158

Problem: "zs" and "ze" don't work correctly with ":set nowrap siso=1". (Preben Guldberg)
Solution: Take '**siso**' into account when computing the horizontal scroll position for "zs" and "ze".
Files: src/normal.c

Patch 6.1.159

Problem: When expanding an abbreviation that includes a multibyte character too many characters are deleted. (Andrey Urazov)
Solution: Delete the abbreviation counting characters instead of bytes.
Files: src/getchar.c

Patch 6.1.160

Problem: ":%read file.gz" doesn't work. (Preben Guldberg)
Solution: Don't use the '[' mark after it has become invalid.
Files: runtime/plugin/gzip.vim

Patch 6.1.161 (depends on 6.1.158)

Problem: Warning for signed/unsigned compare. Can set '**siso**' to a negative value. (Mike Williams)
Solution: Add a typecast. Add a check for '**siso**' being negative.
Files: src/normal.c, src/option.c

Patch 6.1.162

Problem: Python interface: Didn't initialize threads properly.
Solution: Call PyEval_InitThreads() when starting up.
Files: src/if_python.c

Patch 6.1.163

Problem: Win32: Can't compile with Python after 6.1.162.
Solution: Dynamically load PyEval_InitThreads(). (Dan Sharp)

Files: src/if_python.c

Patch 6.1.164

Problem: If **'modifiable'** is off, converting to xxd fails and **'filetype'** is changed to "xxd" anyway.

Solution: Don't change **'filetype'** when conversion failed.

Files: runtime/menu.vim

Patch 6.1.165

Problem: Making changes in several lines and then a change in one of these lines that splits it in two or more lines, undo information was corrupted. May cause a crash. (Dave Fishburn)

Solution: When skipping to save a line for undo because it was already saved, move it to become the last saved line, so that when the command changes the line count other saved lines are not involved.

Files: src/undo.c

Patch 6.1.166

Problem: When **'autoindent'** is set and mswin.vim has been sourced, pasting with **CTRL-V** just after auto-indenting removes the indent. (Shlomi Fish)

Solution: First insert an "x" and delete it again, so that the auto-indent remains.

Files: runtime/mswin.vim

Patch 6.1.167

Problem: When giving a negative argument to ":retab" strange things start happening. (Hans Ginzel)

Solution: Check for a negative value.

Files: src/ex_cmds.c

Patch 6.1.168

Problem: Pressing **CTRL-C** at the hit-enter prompt doesn't end the prompt.

Solution: Make **CTRL-C** stop the hit-enter prompt.

Files: src/message.c

Patch 6.1.169

Problem: bufexists() finds a buffer by using the name of a symbolic link to it, but bufnr() doesn't. (Yegappan Lakshmanan)

Solution: When bufnr() can't find a buffer, try using the same method as bufexists().

Files: src/eval.c

Patch 6.1.170

Problem: Using ":mksession" uses the default session file name, but "vim -S" doesn't. (Hans Ginzel)

Solution: Use the default session file name if "-S" is the last command line argument or another option follows.

Files: runtime/doc/starting.txt, src/main.c

Patch 6.1.171

Problem: When opening a line just above a closed fold with "O" and the comment leader is automatically inserted, the cursor is displayed in the first column. (Sung-Hyun Nam)

Solution: Update the flag that indicates the cursor is in a closed fold.
Files: src/misc1.c

Patch 6.1.172

Problem: Command line completion of ":tag /pat" does not show the same results as the tags the command actually finds. (Gilles Roy)

Solution: Don't modify the pattern to make it a regexp.

Files: src/ex_getln.c, src/tag.c

Patch 6.1.173

Problem: When using remote control to edit a position in a file and this file is the current buffer and it's modified, the window is split and the ":drop" command fails.

Solution: Don't split the window, keep editing the same buffer. Use the ":drop" command in VisVim to avoid the problem there.

Files: src/ex_cmds.c, src/ex_cmds2.c, src/proto/ex_cmds2.pro, VisVim/Commands.cpp

Patch 6.1.174

Problem: It is difficult to know in a script whether an option not only exists but really works.

Solution: Add "exists('+option')".

Files: runtime/doc/eval.txt, src/eval.c

Patch 6.1.175

Problem: When reading commands from a pipe and a **CTRL-C** is pressed, Vim will hang. (Piet Delpont)

Solution: Don't keep reading characters to clear typeahead when an interrupt was detected, stop when a single **CTRL-C** is read.

Files: src/getchar.c, src/ui.c

Patch 6.1.176

Problem: When the stack limit is very big a false out-of-stack error may be detected.

Solution: Add a check for overflow of the stack limit computation. (Jim Dunleavy)

Files: src/os_unix.c

Patch 6.1.177 (depends on 6.1.141)

Problem: ":wincmd" does not allow a following command. (Gary Johnson)

Solution: Check for a following " | cmd". Also give an error for trailing characters.

Files: src/ex_docmd.c

Patch 6.1.178

Problem: When '**expandtab**' is set "r<C-V><Tab>" still expands the Tab. (Bruce deVisser)

Solution: Replace with a literal Tab.

Files: src/normal.c

Patch 6.1.179 (depends on 6.1.091)

Problem: When using X11R5 XIMPreserveState is undefined. (Albert Chin)

Solution: Include the missing definitions.

Files: src/mbyte.c

Patch 6.1.180

Problem: Use of the GUI code for forking is inconsistent.
Solution: Define MAY_FORK and use it for later #ifdefs. (Ben Fowler)
Files: src/gui.c

Patch 6.1.181

Problem: If the terminal doesn't wrap from the last char in a line to the next line, the last column is blanked out. (Peter Karp)
Solution: Don't output a space to mark the wrap, but the same character again.
Files: src/screen.c

Patch 6.1.182 (depends on 6.1.142)

Problem: It is not possible to auto-format comments only. (Moshe Kaminsky)
Solution: When the 'a' and 'c' flags are in '**formatoptions**' only auto-format comments.
Files: runtime/doc/change.txt, src/edit.c

Patch 6.1.183

Problem: When '**fencs**' is empty and '**enc**' is utf-8, reading a file with illegal bytes gives "CONVERSION ERROR" even though no conversion is done. '**readonly**' is set, even though writing the file results in an unmodified file.
Solution: For this specific error use "ILLEGAL BYTE" and don't set '**readonly**'.
Files: src/fileio.c

Patch 6.1.184 (extra)

Problem: The extra mouse buttons found on some mice don't work.
Solution: Support two extra buttons for MS-Windows. (Michael Geddes)
Files: runtime/doc/term.txt, src/edit.c, src/ex_getln.c, src/gui.c, src/gui_w32.c, src/gui_w48.c, src/keymap.h, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/vim.h

Patch 6.1.185 (depends on 6.1.182)

Problem: Can't compile without +comments feature.
Solution: Add #ifdef FEAT_COMMENTS. (Christian J. Robinson)
Files: src/edit.c

Patch 6.1.186 (depends on 6.1.177)

Problem: ":wincmd" does not allow a following comment. (Aric Blumer)
Solution: Check for a following double quote.
Files: src/ex_docmd.c

Patch 6.1.187

Problem: Using ":doarg" with '**hidden**' set and the current file is the only argument and was modified gives an error message. (Preben Guldberg)
Solution: Don't try re-editing the same file.
Files: src/ex_cmds2.c

Patch 6.1.188 (depends on 6.1.173)

Problem: Unused variable in the small version.

Solution: Move the declaration for "p" inside #ifdef FEAT_LISTCMDS.
Files: src/ex_cmds2.c

Patch 6.1.189

Problem: inputdialog() doesn't work when 'c' is in '**guioptions**'. (Aric Blumer)

Solution: Fall back to the input() function in this situation.

Files: src/eval.c

Patch 6.1.190 (extra)

Problem: VMS: doesn't build with GTK GUI. Various other problems.

Solution: Fix building for GTK. Improved Perl, Python and TCL support.

Improved VMS documentation. (Zoltan Arpadffy)

Added Vimtutor for VMS (T. R. Wyant)

Files: runtime/doc/os_vms.txt, src/INSTALLvms.txt, src/gui_gtk_f.h,
src/if_tcl.c, src/main.c, src/gui_gtk_vms.h, src/Make_vms.mms,
src/os_vms.opt, src/proto/if_tcl.pro, vimtutor.com,
src/testdir/Make_vms.mms

Patch 6.1.191

Problem: When using "vim -s script" and redirecting the output, the delay for the "Output is not to a terminal" warning slows Vim down too much.

Solution: Don't delay when reading commands from a script.

Files: src/main.c

Patch 6.1.192

Problem: ":diffsplit" doesn't add "hor" to '**scrollopt**'. (Gary Johnson)

Solution: Add "hor" to '**scrollopt**' each time ":diffsplit" is used.

Files: src/diff.c, src/main.c

Patch 6.1.193

Problem: Crash in in_id_list() for an item with a "containedin" list. (Dave Fishburn)

Solution: Check for a negative syntax id, used for keywords.

Files: src/syntax.c

Patch 6.1.194

Problem: When "t_ti" is set but it doesn't cause swapping terminal pages, "ZZ" may cause the shell prompt to appear on top of the file-write message.

Solution: Scroll the text up in the Vim page before swapping to the terminal page. (Michael Schroeder)

Files: src/os_unix.c

Patch 6.1.195

Problem: The quickfix and preview windows always keep their height, while other windows can't fix their height.

Solution: Add the '**winfixheight**' option, so that a fixed height can be specified for any window. Also fix that the wildmenu may resize a one-line window to a two-line window if '**ls**' is zero.

Files: runtime/doc/options.txt, runtime/optwin.vim, src/ex_cmds.c,
src/ex_getln.c, src/globals.h, src/option.c, src/quickfix.c,
src/screen.c, src/structs.h, src/window.c

Patch 6.1.196 (depends on 6.1.084)

Problem: On Mac OS X 10.2 generating osdef.h fails.

Solution: Add `-no-cpp-precomp` to avoid using precompiled header files, which disables printing the search path. (Ben Fowler)

Files: `src/auto/configure`, `src/configure.in`

Patch 6.1.197

Problem: `":help <C-V><C-\><C-V><C-N>"` (resulting in `<1c><0e>`) gives an error message. (Servatius Brandt)

Solution: Double the backslash in `"CTRL-\"`.

Files: `src/ex_cmds.c`

Patch 6.1.198 (extra) (depends on 6.1.076)

Problem: Mac OS X: Dialogues don't work.

Solution: Fix a crashing problem for some GUI dialogues. Fix a problem when saving to a new file from the GUI. (Peter Cucka)

Files: `src/feature.h`, `src/gui_mac.c`

Patch 6.1.199

Problem: `'guifontwide'` doesn't work on Win32.

Solution: Output each wide character separately. (Michael Geddes)

Files: `src/gui.c`

Patch 6.1.200

Problem: `":syn sync fromstart"` is not skipped after `":if 0"`. This can make syntax highlighting very slow.

Solution: Check `"eap->skip"` appropriately. (Rob West)

Files: `src/syntax.c`

Patch 6.1.201 (depends on 6.1.192)

Problem: Warning for illegal pointer combination. (Zoltan Arpadffy)

Solution: Add a typecast.

Files: `src/diff.c`

Patch 6.1.202 (extra)(depends on 6.1.148)

Problem: Win32: `filewritable()` doesn't work properly on directories.

Solution: fix `filewritable()`. (Mike Williams)

Files: `src/os_win32.c`

Patch 6.1.203

Problem: `":%s/~/"` causes a crash after `":%s/x/"`. (Gary Holloway)

Solution: Avoid reading past the end of a line when `"~"` is empty.

Files: `src/regexp.c`

Patch 6.1.204 (depends on 6.1.129)

Problem: Warning for an illegal pointer on Solaris.

Solution: Add a typecast. (Derek Wyatt)

Files: `src/misc2.c`

Patch 6.1.205

Problem: The gzip plugin changes the alternate file when editing a compressed file. (Oliver Fuchs)

Solution: Temporarily remove the `'a'` and `'A'` flags from `'cpo'`.

Files: runtime/plugin/gzip.vim

Patch 6.1.206

Problem: The script generated with ":mksession" doesn't work properly when some commands are mapped.

Solution: Use ":normal!" instead of ":normal". And use ":wincmd" where possible. (Muraoka Taro)

Files: src/ex_docmd.c, src/fold.c

Patch 6.1.207

Problem: Indenting a Java file hangs below a line with a comment after a command.

Solution: Break out of a loop. (Andre Pang)
Also line up } with matching {.

Files: runtime/indent/java.vim

Patch 6.1.208

Problem: Can't use the buffer number from the Python interface.

Solution: Add buffer.number. (Michal Vitecek)

Files: src/if_python.c

Patch 6.1.209

Problem: Printing doesn't work on Mac OS classic.

Solution: Use a ":" for path separator when opening the resource file. (Axel Kielhorn)

Files: src/ex_cmds2.c

Patch 6.1.210

Problem: When there is an iconv() conversion error when reading a file there can be an error the next time iconv() is used.

Solution: Reset the state of the iconv() descriptor. (Yasuhiro Matsumoto)

Files: src/fileio.c

Patch 6.1.211

Problem: The message "use ! to override" is confusing.

Solution: Make it "add ! to override".

Files: src/buffer.c, src/eval.c, src/ex_docmd.c, src/fileio.c,
src/globals.h

Patch 6.1.212

Problem: When Vim was started with "-R" ":new" creates a buffer '**noreadonly**' while ":enew" has '**readonly**' set. (Preben Guldberg)

Solution: Don't set '**readonly**' in a new empty buffer for ":enew".

Files: src/ex_docmd.c

Patch 6.1.213

Problem: Using **CTRL-W** H may cause a big gap to appear below the last window. (Aric Blumer)

Solution: Don't set the window height when there is a vertical split. (Yasuhiro Matsumoto)

Files: src/window.c

Patch 6.1.214

Problem: When installing Vim and the runtime files were checked out from

CVS the CVS directories will also be installed.
Solution: Avoid installing the CVS dirs and their contents.
Files: src/Makefile

Patch 6.1.215

Problem: Win32: ":pwd" uses backslashes even when '**shellslash**' is set.
(Xiangjiang Ma)
Solution: Adjust backslashes before printing the message.
Files: src/ex_docmd.c

Patch 6.1.216

Problem: When dynamically loading the iconv library, the error codes may be confused.
Solution: Use specific error codes for iconv and redefine them for dynamic loading. (Yasuhiro Matsumoto)
Files: src/fileio.c, src/mbyte.c, src/vim.h

Patch 6.1.217

Problem: When sourcing the same Vim script using a different name (symbolic link or MS-Windows 8.3 name) it is listed twice with ":scriptnames". (Tony Mechelynck)
Solution: Turn the script name into a full path before using it. On Unix compare inode/device numbers.
Files: src/ex_cmds2.c

Patch 6.1.218

Problem: No error message for using the function argument "5+". (Servatius Brandt)
Solution: Give an error message if a function or variable is expected but is not found.
Files: src/eval.c

Patch 6.1.219

Problem: When using ":amenu :b 1<CR>" with a Visual selection and '**insertmode**' is set, Vim does not return to Insert mode. (Mickael Marchand)
Solution: Add the command CTRL-\ **CTRL-G** that goes to Insert mode if '**insertmode**' is set and to Normal mode otherwise. Append this to menus defined with ":amenu".
Files: src/edit.c, src/ex_getln.c, src/normal.c

Patch 6.1.220

Problem: When using a BufReadPost autocommand that changes the line count, e.g., "\$-ljoin", reloading a file that was changed outside Vim does not work properly. (Alan G Isaac)
Solution: Make the buffer empty before reading the new version of the file. Save the lines in a dummy buffer, so that they can be put back when reading the file fails.
Files: src/buffer.c, src/ex_cmds.c, src/fileio.c, src/globals.h, src/proto/buffer.pro

Patch 6.1.221

Problem: Changing case may not work properly, depending on the current locale.

Solution: Add the **'casemap'** option to let the user choose how changing case is to be done.
Also fix lowering case when an UTF-8 character doesn't keep the same byte length.

Files: runtime/doc/options.txt, src/ascii.h, src/auto/configure, src/buffer.c, src/charset.c, src/config.h.in, src/configure.in, src/diff.c, src/edit.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/gui_amiga.c, src/gui_mac.c, src/gui_photon.c, src/gui_w48.c, src/gui_beos.cc, src/macros.h, src/main.c, src/mbyte.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/option.c, src/os_msdos.c, src/os_mswin.c, src/proto/charset.pro, src/regexp.c, src/option.h, src/syntax.c

Patch 6.1.222 (depends on 6.1.219)

Problem: Patch 6.1.219 was incomplete.

Solution: Add the changes for ":amenu".

Files: src/menu.c

Patch 6.1.223 (extra)

Problem: Win32: When IME is activated **'iminsert'** is set, but it might never be reset when IME is disabled. (Muraoka Taro)
All systems: **'iminsert'** is set to 2 when leaving Insert mode, even when langmap is being used. (Peter Valach)

Solution: Don't set "b_p_iminsert" in _OnImeNotify(). (Muraoka Taro)
Don't store the status of the input method in **'iminsert'** when **'iminsert'** is one. Also for editing the command line and for arguments to Normal mode commands.

Files: src/edit.c, src/ex_getln.c, src/gui_w32.c, src/normal.c

Patch 6.1.224

Problem: "expand('\$VAR')" returns an empty string when the expanded \$VAR is not an existing file. (Aric Blumer)

Solution: Included non-existing files, as documented.

Files: src/eval.c

Patch 6.1.225

Problem: Using <C-O><C-^> in Insert mode has a delay when starting "vim -u NONE" and ":set nosp hidden". (Emmanuel) do_ecmd() uses fileinfo(), the redraw is done after a delay to give the user time to read the message.

Solution: Put the message from fileio() in "keep_msg", so that the redraw is done before the delay (still needed to avoid the mode message overwrites the fileinfo() message).

Files: src/buffer.c

Patch 6.1.226

Problem: Using ":debug" with a ":normal" command may cause a hang. (Colin Keith)

Solution: Save the typeahead buffer when obtaining a debug command.

Files: src/ex_cmds2.c, src/getchar.c, src/proto/getchar.pro

Patch 6.1.227

Problem: It is possible to use a variable name "asdf:asdf" and ":let j:asdf

= 5" does not give an error message. (Mikolaj Machowski)
Solution: Check for a ":" inside the variable name.
Files: src/eval.c

Patch 6.1.228 (extra)

Problem: Win32: The special output function for Hangul is used too often, causing special handling for other situations to be skipped. bInComposition is always FALSE, causing ImeGetTempComposition() always to return NULL.
Solution: Remove HanExtTextOut(). Delete the dead code around bInComposition and ImeGetTempComposition().
Files: src/gui_w16.c, src/gui_w32.c, src/gui_w48.c

Patch 6.1.229

Problem: Win32: Conversion to/from often used codepages requires the iconv library, which is not always available.
Solution: Use standard MS-Windows functions for the conversion when possible. (mostly by Glenn Maynard)
Also fixes missing declaration for patch 6.1.220.
Files: src/fileio.c

Patch 6.1.230 (extra)

Problem: Win16: building doesn't work.
Solution: Exclude the XBUTTON handling. (Vince Negri)
Files: src/gui_w48.c

Patch 6.1.231

Problem: Double clicking with the mouse to select a word does not work for multibyte characters.
Solution: Use vim_iswordc() instead of vim_isIDc(). This means **'iskeyword'** is used instead of **'isident'**. Also fix that mixing ASCII with multibyte word characters doesn't work, the mouse class for punctuation and word characters was mixed up.
Files: src/normal.c

Patch 6.1.232 (depends on 6.1.226)

Problem: Using ex_normal_busy while it might not be available. (Axel Kielhorn)
Solution: Only use ex_normal_busy when FEAT_EX_EXTRA is defined.
Files: src/ex_cmds2.c

Patch 6.1.233

Problem: ":help expr-|" does not work.
Solution: Don't use the '|' as a command separator
Files: src/ex_cmds.c

Patch 6.1.234 (depends on 6.1.217)

Problem: Get a warning for using a negative value for st_dev.
Solution: Don't assign a negative value to st_dev.
Files: src/ex_cmds2.c

Patch 6.1.235 (depends on 6.1.223)

Problem: **'iminsert'** is changed from 1 to 2 when leaving Insert mode. (Peter Valach)

Solution: Check "State" before resetting it to NORMAL.
Files: src/edit.c

Patch 6.1.236

Problem: Memory leaks when appending lines for ":diffget" or ":diffput" and when reloading a changed buffer.

Solution: Free a line after calling ml_append().
Files: src/diff.c, src/fileio.c

Patch 6.1.237

Problem: Putting in Visual block mode does not work correctly when "\$" was used or when the first line is short. (Christian Michon)

Solution: First delete the selected text and then put the new text. Save and restore registers as necessary.

Files: src/globals.h, src/normal.c, src/ops.c, src/proto/ops.pro, src/vim.h

Patch 6.1.238 (extra)

Problem: Win32: The "icon=" argument for the ":menu" command does not search for the bitmap file.

Solution: Expand environment variables and search for the bitmap file. (Vince Negri)

Make it consistent, use the same mechanism for X11 and GTK.

Files: src/gui.c src/gui_gtk.c, src/gui_w32.c, src/gui_x11.c, src/proto/gui.pro

Patch 6.1.239

Problem: Giving an error for missing :endif or :endwhile when being interrupted.

Solution: Don't give these messages when interrupted.

Files: src/ex_docmd.c, src/os_unix.c

Patch 6.1.240 (extra)

Problem: Win32 with BCC 5: CPU may be defined in the environment, which causes a wrong argument for the compiler. (Walter Briscoe)

Solution: Use CPUNR instead of CPU.

Files: src/Make_bc5.mak

Patch 6.1.241

Problem: Something goes wrong when drawing or undrawing the cursor.

Solution: Remember when the cursor invalid in a better way.

Files: src/gui.c

Patch 6.1.242

Problem: When pasting a large number of lines on the command line it is not possible to interrupt. (Jean Jordaan)

Solution: Check for an interrupt after each pasted line.

Files: src/ops.c

Patch 6.1.243 (extra)

Problem: Win32: When the OLE version is started and wasn't registered, a message pops up to suggest registering, even when this isn't possible (when the registry is not writable).

Solution: Check if registering is possible before asking whether it should

be done. (Walter Briscoe)
Also avoid restarting Vim after registering.

Files: src/if_ole.cpp

Patch 6.1.244

Problem: Patch 6.1.237 was missing the diff for vim.h. (Igor Goldenberg)

Solution: Include it here.

Files: src/vim.h

Patch 6.1.245

Problem: Comparing with ignored case does not work properly for Unicode with a locale where case folding an ASCII character results in a multibyte character. (Glenn Maynard)

Solution: Handle ignore-case compare for Unicode differently.

Files: src/mbyte.c

Patch 6.1.246

Problem: ":blast" goes to the first buffer if the last one is unlisted. (Andrew Stryker)

Solution: From the last buffer search backwards for the first listed buffer instead of forwards.

Files: src/ex_docmd.c

Patch 6.1.247

Problem: ACL support doesn't always work properly.

Solution: Add a configure argument to disable ACL "--disable-acl". (Thierry Vignaud)

Files: src/auto/configure, src/configure.in

Patch 6.1.248

Problem: Typing 'q' at the more-prompt for ":let" does not quit the listing. (Hari Krishna Dara)

Solution: Quit the listing when got_int is set.

Files: src/eval.c

Patch 6.1.249

Problem: Can't expand a path on the command line if it includes a "|" as a trail byte of a multibyte character.

Solution: Check for multibyte characters. (Yasuhiro Matsumoto)

Files: src/ex_docmd.c

Patch 6.1.250

Problem: When changing the value of 'lines' inside the expression set with 'diffexpr' Vim might crash. (Dave Fishburn)

Solution: Don't allow changing the screen size while updating the screen.

Files: src/globals.h, src/option.c, src/screen.c

Patch 6.1.251

Problem: Can't use completion for ":lcd" and ":lchdir" like ":cd".

Solution: Expand directory names for these commands. (Servatius Brandt)

Files: src/ex_docmd.c

Patch 6.1.252

Problem: "vi}" does not include a line break when the "}" is at the start

of a following line. (Kamil Burzynski)
Solution: Include the line break.
Files: src/search.c

Patch 6.1.253 (extra)

Problem: Win32 with Cygwin: Changes the path of arguments in a wrong way.
(Xiangjiang Ma)
Solution: Don't use cygwin_conv_to_posix_path() for the Win32 version.
Update the Cygwin makefile to support more features. (Dan Sharp)
Files: src/Make_cyg.mak, src/if_ole.cpp, src/main.c

Patch 6.1.254

Problem: exists("foo{bar}") does not work. ':unlet v{"a"}r' does not work.
":let v{a}r1 v{a}r2" does not work. ":func F{(1)}" does not work.
":delfunc F{"}" does not give an error message. ':delfunc F{"F"}'
does not work.
Solution: Support magic braces for the exists() argument. (Vince Negri)
Check for trailing comments explicitly for ":unlet". Add support
for magic braces in further arguments of ":let". Look for a
parenthesis only after the function name. (Servatius Brandt)
Also expand magic braces for "exists('*expr')". Give an error
message for an invalid ":delfunc" argument. Allow quotes in the
":delfunc" argument.
Files: src/eval.c, src/ex_cmds.h, src/ex_docmd.c

Patch 6.1.255 (depends on 6.1.254)

Problem: Crash when loading menu.vim a second time. (Christian Robinson)
":unlet garbage foo" tries unletting "foo" after an error message.
(Servatius Brandt)
Very long function arguments cause very long messages when
'**verbose**' is 14 or higher.
Solution: Avoid reading from uninitialized memory.
Break out of a loop after an invalid argument for ":unlet".
Truncate long function arguments to 80 characters.
Files: src/eval.c

Patch 6.1.256 (depends on 6.1.255)

Problem: Defining a function after ":if 0" could still cause an error
message for an existing function.
Leaking memory when there are trailing characters for ":delfunc".
Solution: Check the "skip" flag. Free the memory. (Servatius Brandt)
Files: src/eval.c

Patch 6.1.257

Problem: ":cwindow" always sets the previous window to the last but one
window. (Benji Fisher)
Solution: Set the previous window properly.
Files: src/globals.c, src/quickfix.c, src/window.c

Patch 6.1.258

Problem: Buffers menu doesn't work properly for multibyte buffer names.
Solution: Use a pattern to get the left and right part of the name.
(Yasuhiro Matsumoto)
Files: runtime/menu.vim

Patch 6.1.259 (extra)

Problem: Mac: with **'patchmode'** is used filenames are truncated.

Solution: Increase the BASENAMELEN for Mac OS X. (Ed Ralston)

Files: src/os_mac.h

Patch 6.1.260 (depends on 6.1.104)

Problem: GCC 3.2 still seems to have an optimizer problem. (Zvi Har'El)

Solution: Use the same configure check as used for GCC 3.1.

Files: src/auto/configure, src/configure.in

Patch 6.1.261

Problem: When deleting a line in a buffer which is not the current buffer, using the Perl interface Delete(), the cursor in the current window may move. (Chris Houser)

Solution: Don't adjust the cursor position when changing another buffer.

Files: src/if_perl.xs

Patch 6.1.262

Problem: When jumping over folds with "z[" , "zj" and "zk" the previous position is not remembered. (Hari Krishna Dara)

Solution: Set the previous context mark before jumping.

Files: src/fold.c

Patch 6.1.263

Problem: When typing a multibyte character that triggers an abbreviation it is not inserted properly.

Solution: Handle adding the typed multibyte character. (Yasuhiro Matsumoto)

Files: src/getchar.c

Patch 6.1.264 (depends on patch 6.1.254)

Problem: exists() does not work for built-in functions. (Steve Wall)

Solution: Don't check for the function name to start with a capital.

Files: src/eval.c

Patch 6.1.265

Problem: libcall() can be used in **'foldexpr'** to call any system function. rename(), delete() and remote_send() can also be used in **'foldexpr'**. These are security problems. (Georgi Guninski)

Solution: Don't allow using libcall(), rename(), delete(), remote_send() and similar functions in the sandbox.

Files: src/eval.c

Patch 6.1.266 (depends on 6.1.265)

Problem: Win32: compile error in eval.c. (Bill McCarthy)

Solution: Move a variable declaration.

Files: src/eval.c

Patch 6.1.267

Problem: Using "p" to paste into a Visual selected area may cause a crash.

Solution: Allocate enough memory for saving the register contents. (Muraoka Taro)

Files: src/ops.c

Patch 6.1.268

Problem: When triggering an abbreviation with a multibyte character, this character is not correctly inserted after expanding the abbreviation. (Taro Muraoka)
Solution: Add ABBR_OFF to all characters above 0xff.
Files: src/edit.c, src/ex_getln.c, src/getchar.c

Patch 6.1.269

Problem: After using input() text written with ":redir" gets extra indent. (David Fishburn)
Solution: Restore msg_col after using input().
Files: src/ex_getln.c

Patch 6.1.270 (depends on 6.1.260)

Problem: GCC 3.2.1 still seems to have an optimizer problem.
Solution: Use the same configure check as used for GCC 3.1.
Files: src/auto/configure, src/configure.in

Patch 6.1.271

Problem: When compiling without the +syntax feature there are errors.
Solution: Don't use some code for syntax highlighting. (Roger Cornelius)
Make test 45 work without syntax highlighting.
Also fix an error in a pattern matching: "\%(" was not supported.
Files: src/ex_cmds2.c, src/regexp.c, src/testdir/test45.in

Patch 6.1.272

Problem: After using ":set define<" a crash may happen. (Christian Robinson)
Solution: Make a copy of the option value in allocated memory.
Files: src/option.c

Patch 6.1.273

Problem: When the cursor doesn't blink, redrawing an exposed area may hide the cursor.
Solution: Always draw the cursor, also when it didn't move. (Muraoka Taro)
Files: src/gui.c

Patch 6.1.274 (depends on 6.1.210)

Problem: Resetting the iconv() state after each error is wrong for an incomplete sequence.
Solution: Don't reset the iconv() state.
Files: src/fileio.c

Patch 6.1.275

Problem: When using "v" in a startup script, get warning message that terminal cannot highlight. (Charles Campbell)
Solution: Only give the message after the terminal has been initialized.
Files: src/normal.c

Patch 6.1.276

Problem: "gvim --remote file" doesn't prompt for an encryption key.
Solution: The further characters the client sends to the server are used. Added inputsave() and inputrestore() to allow prompting the user directly and not using typeahead.
Also fix possible memory leak for ":normal".

Files: src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/getchar.c,
src/main.c, src/proto/getchar.pro, src/proto/ui.pro,
src/runtime/doc/eval.txt, src/structs.h, src/ui.c, src/vim.h

Patch 6.1.277 (depends on 6.1.276)

Problem: Compilation error when building with small features.

Solution: Define trash_input_buf() when needed. (Kelvin Lee)

Files: src/ui.c

Patch 6.1.278

Problem: When using signs the line number of a closed fold doesn't line up with the other line numbers. (Kamil Burzynski)

Solution: Insert two spaces for the sign column.

Files: src/screen.c

Patch 6.1.279

Problem: The prototype for msg() and msg_attr() do not match the function definition. This may cause trouble for some compilers. (Nix)

Solution: Use va_list for systems that have stdarg.h. Use "int" instead of "void" for the return type.

Files: src/auto/configure, src/config.h.in, src/configure.in,
src/proto.h, src/message.c

Patch 6.1.280

Problem: It's possible to use an argument "firstline" or "lastline" for a function but using "a:firstline" or "a:lastline" in the function won't work. (Benji Fisher)

Solution: Give an error message for these arguments.

Also avoid that the following function body causes a whole row of errors, skip over it after an error in the first line.

Files: src/eval.c

Patch 6.1.281

Problem: In Insert mode **CTRL-X CTRL-G** leaves the cursor after the ruler.

Solution: Set the cursor position before waiting for the argument of **CTRL-G**. (Yasuhiro Matsumoto)

Files: src/edit.c

Patch 6.1.282

Problem: Elvis uses "se" in a modeline, Vim doesn't recognize this.

Solution: Also accept "se " where "set " is accepted in a modeline. (Yasuhiro Matsumoto)

Files: src/buffer.c

Patch 6.1.283

Problem: For ":sign" the icon file name cannot contain a space.

Solution: Handle backslashes in the file name. (Yasuhiro Matsumoto)

Files: src/ex_cmds.c

Patch 6.1.284

Problem: On Solaris there is a warning for "struct utimbuf".

Solution: Move including "utime.h" to outside the function. (Derek Wyatt)

Files: src/fileio.c

Patch 6.1.285

Problem: Can't wipe out a buffer with **'bufhide'** option.
Solution: Add "wipe" value to **'bufhide'**. (Yegappan Lakshmanan)
Files: runtime/doc/options.txt, src/buffer.c, src/option.c, src/quickfix.c

Patch 6.1.286

Problem: **'showbreak'** cannot contain multibyte characters.
Solution: Allow using all printable characters for **'showbreak'**.
Files: src/charset.c, src/move.c, src/option.c

Patch 6.1.287 (depends on 6.1.285)

Problem: Effect of "delete" and "wipe" in **'bufhide'** were mixed up.
Solution: Wipe out when wiping out is asked for.
Files: src/buffer.c

Patch 6.1.288

Problem: ":silent function F" hangs. (Hari Krishna Dara)
Solution: Don't use msg_col, it is not incremented when using ":silent".
Also made the function output look a bit better. Don't translate "function".
Files: src/eval.c

Patch 6.1.289 (depends on 6.1.278)

Problem: Compiler warning for pointer. (Axel Kielhorn)
Solution: Add a typecast for " ".
Files: src/screen.c

Patch 6.1.290 (extra)

Problem: Truncating long text for message box may break multibyte character.
Solution: Adjust to start of multibyte character. (Yasuhiro Matsumoto)
Files: src/os_mswin.c

Patch 6.1.291 (extra)

Problem: Win32: CTRL-@ doesn't work. Don't even get a message for it.
Solution: Recognize the keycode for CTRL-@. (Yasuhiro Matsumoto)
Files: src/gui_w48.c

Patch 6.1.292 (extra, depends on 6.1.253)

Problem: Win32: Can't compile with new MingW compiler.
Borland 5 makefile doesn't generate pathdef.c.
Solution: Remove -wwide-multiply argument. (Rene de Zwart)
Various fixes for other problems in Win32 makefiles. (Dan Sharp)
Files: src/Make_bc5.mak, src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak

Patch 6.1.293

Problem: byte2line() returns a wrong result for some values.
Solution: Change ">=" to ">" in ml_find_line_or_offset(). (Bradford C Smith)
Add one to the line number when at the end of a block.
Files: src/memline.c

Patch 6.1.294

Problem: Can't include a multibyte character in a string by its hex value.
(Benji Fisher)
Solution: Add "\u...": a character specified with up to four hex numbers
and stored according to the value of 'encoding'.
Files: src/eval.c

Patch 6.1.295 (extra)

Problem: Processing the cs.po file generates an error. (Rahul Agrawal)
Solution: Fix the printf format characters in the translation.
Files: src/po/cs.po

Patch 6.1.296

Problem: Win32: When cancelling the font dialog 'guifont' remains set to
"*.
Solution: Restore the old value of 'guifont' (Yasuhiro Matsumoto)
Files: src/option.c

Patch 6.1.297

Problem: "make test" fails in test6 in an UTF-8 environment. (Benji Fisher)
Solution: Before executing the BufReadPost autocommands save the current
fileencoding, so that the file isn't marked changed.
Files: src/fileio.c

Patch 6.1.298

Problem: When using signs and the first line of a closed fold has a sign
it can be redrawn as if the fold was open. (Kamil Burzynski)
Solution: Don't redraw a sign inside a closed fold.
Files: src/screen.c

Patch 6.1.299

Problem: ":edit +set\ ro file" doesn't work.
Solution: Halve the number of backslashes in the "+cmd" argument.
Files: src/ex_docmd.c

Patch 6.1.300 (extra)

Problem: Handling of ETO_IGNORELANGUAGE is confusing.
Solution: Clean up the handling of ETO_IGNORELANGUAGE. (Glenn Maynard)
Files: src/gui_w32.c

Patch 6.1.301 (extra)

Problem: French translation of file-save dialog doesn't show file name.
Solution: Insert a star in the printf string. (Francois Terrot)
Files: src/po/fr.po

Patch 6.1.302

Problem: Counting lines of the Visual area is incorrect for closed folds.
(Mikolaj Machowski)
Solution: Correct the start and end for the closed fold.
Files: src/normal.c

Patch 6.1.303 (extra)

Problem: The Top/Bottom/All text does not always fit in the ruler when
translated to Japanese. Problem with a character being wider when
in a bold font.

Solution: Use ETO_PDY to specify the width of each character. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 6.1.304 (extra, depends on 6.1.292)

Problem: Win32: Postscript is always enabled in the MingW Makefile. Pathdef.c isn't generated properly with Make_bc5.mak. (Yasuhiro Matsumoto)

Solution: Change an ifdef to an ifeq. (Madoka Machitani)
Use the Borland make redirection to generate pathdef.c. (Maurice Barnum)

Files: src/Make_bc5.mak, src/Make_ming.mak

Patch 6.1.305

Problem: When '**verbose**' is 14 or higher, a function call may cause reading uninitialized data. (Walter Briscoe)

Solution: Check for end-of-string in trunc_string().

Files: src/message.c

Patch 6.1.306

Problem: The AIX VisualAge cc compiler doesn't define __STDC__.

Solution: Use __EXTENDED__ like __STDC__. (Jess Thrysoee)

Files: src/os_unix.h

Patch 6.1.307

Problem: When a double-byte character has an illegal tail byte the display is messed up. (Yasuhiro Matsumoto)

Solution: Draw "XX" instead of the wrong character.

Files: src/screen.c

Patch 6.1.308

Problem: Can't reset the Visual mode returned by visualmode().

Solution: Use an optional argument to visualmode(). (Charles Campbell)

Files: runtime/doc/eval.txt, src/eval.c, src/normal.c, src/structs.h

Patch 6.1.309

Problem: The tutor doesn't select German if the locale name is "German_Germany.1252". (Joachim Hofmann)

Solution: Check for "German" in the locale name. Also check for ".ge". And include the German and Greek tutors.

Files: runtime/tutor/tutor.de, runtime/tutor/tutor.vim, runtime/tutor/tutor.gr, runtime/tutor/tutor.gr.cp737

Patch 6.1.310 (depends on 6.1.307)

Problem: All double-byte characters are displayed as "XX".

Solution: Use ">= 32" instead of "< 32". (Yasuhiro Matsumoto)

Files: src/screen.c

Patch 6.1.311 (extra)

Problem: VMS: path in window title doesn't include necessary separator. file version doesn't always work properly with Unix.

Crashes because of memory overwrite in GUI.

Didn't always handle files with lowercase and correct path.

Solution: Fix the problems. Remove unnecessary file name translations.
(Zoltan Arpadffy)
Files: src/buffer.c, src/ex_cmds2.c, src/fileio.c, src/memline.c,
src/misc1.c, src/misc2.c, src/os_unix.c, src/os_vms.c, src/tag.c

Patch 6.1.312

Problem: When using ":silent" debugging is also done silently.
Solution: Disable silence while at the debug prompt.
Files: src/ex_cmds2.c

Patch 6.1.313

Problem: When a ":drop fname" command is used and "fname" is open in
another window, it is also opened in the current window.
Solution: Change to the window with "fname" instead.
Don't redefine the argument list when dropping only one file.
Files: runtime/doc/windows.txt, src/ex_cmds2.c, src/ex_cmds.c,
src/ex_docmd.c, src/proto/ex_cmds2.pro, src/proto/ex_docmd.pro

Patch 6.1.314 (depends on 6.1.126)

Problem: Missing backslash in "Generic Config file" syntax menu.
Solution: Insert the backslash. (Zak Beck)
Files: runtime/makemenu.vim, runtime/synmenu.vim

Patch 6.1.315 (extra)

Problem: A very long hostname may lead to an unterminated string. Failing
to obtain a hostname may result in garbage. (Walter Briscoe)
Solution: Add a NUL at the end of the hostname buffer.
Files: src/os_mac.c, src/os_msdos.c, src/os_unix.c, src/os_win16.c,
src/os_win32.c

Patch 6.1.316

Problem: When exiting with "wq" and there is a hidden buffer, after the
"file changed" dialog there is a warning for a changed buffer.
(Ajit Thakkar)
Solution: Do update the buffer timestamps when exiting.
Files: src/fileio.c

Patch 6.1.317

Problem: Closing a window may cause some of the remaining windows to be
positioned wrong if there is a mix of horizontal and vertical
splits. (Stefan Ingi Valdimarsson)
Solution: Update the frame sizes before updating the window positions.
Files: src/window.c

Patch 6.1.318

Problem: auto/pathdef.c can include wrong quotes when a compiler flag
includes quotes.
Solution: Put a backslash before the quotes in compiler flags. (Shinra Aida)
Files: src/Makefile

Patch 6.1.319 (depends on 6.1.276)

Problem: Using "--remote +cmd file" does not execute "cmd".
Solution: Call inputrestore() in the same command line as inputsave(),
otherwise it will never get executed.

Files: src/main.c

Patch 6.1.320 (depends on 6.1.313)

Problem: When a ":drop one\ file" command is used the file "one\ file" is opened, the backslash is not removed. (Taro Muraoka)

Solution: Handle backslashes correctly. Always set the argument list to keep it simple.

Files: runtime/doc/windows.txt, src/ex_cmds.c

Patch 6.1.321

Problem: When '**mouse**' includes 'n' but not 'v', don't allow starting Visual mode with the mouse.

Solution: Don't use MOUSE_MAY_VIS when there is no 'v' in '**mouse**'. (Flemming Madsen)

Files: src/normal.c

Patch 6.1.322 (extra, depends on 6.1.315)

Problem: Win32: The host name is always "PC " plus the real host name.

Solution: Don't insert "PC " before the host name.

Files: src/os_win32.c

Patch 6.1.323

Problem: ":registers" doesn't stop listing for a "q" at the more prompt. (Hari Krishna Dara)

Solution: Check for interrupt and got_int.

Files: src/ops.c, src/proto/ops.pro

Patch 6.1.324

Problem: Crash when dragging a vertical separator when <LeftMouse> is remapped to jump to another window.

Solution: Pass the window pointer to the function doing the dragging instead of always using the current window. (Daniel Elstner)

Also fix that starting a drag changes window focus.

Files: src/normal.c, src/proto/window.pro, src/ui.c, src/vim.h, src/window.c

Patch 6.1.325

Problem: Shift-Tab is not automatically recognized in an xterm.

Solution: Add <Esc>[Z as the termcap code. (Andrew Pimlott)

Files: src/term.c

Patch 6.1.326

Problem: Using a search pattern may read from uninitialized data (Yasuhiro Matsumoto)

Solution: Initialize pointers to NULL.

Files: src/regexp.c

Patch 6.1.327

Problem: When opening the "mbyte.txt" help file the utf-8 characters are unreadable, because the fileencoding is forced to be latin1.

Solution: Check for utf-8 encoding first in help files. (Daniel Elstner)

Files: runtime/doc/mbyte.txt, src/fileio.c

Patch 6.1.328

Problem: Prototype for enc_canon_search() is missing.
Solution: Add the prototype. (Walter Briscoe)
Files: src/mbyte.c

Patch 6.1.329

Problem: When editing a file "a b c" replacing "%" in ":Cmd %" or ":next %" does not work properly. (Hari Krishna Dara)
Solution: Always escape spaces when expanding "%". Don't split argument for `<f-args>` in a user command when only one argument is used.
Files: src/ex_docmd.c

Patch 6.1.330

Problem: GTK, Motif and Athena: Keypad keys produce the same code as non-keypad keys, making it impossible to map them separately.
Solution: Use different termcap codes for the keypad keys. (Neil Bird)
Files: src/gui_gtk_x11.c, src/gui_x11.c

Patch 6.1.331

Problem: When translating the help files, "LOCAL ADDITIONS" no longer marks the spot where help files from plugins are to be listed.
Solution: Add a "local-additions" tag and use that to find the right spot.
Files: runtime/doc/help.txt, src/ex_cmds.c

Patch 6.1.332 (extra)

Problem: Win32: Loading Perl dynamically doesn't work with Perl 5.8. Perl 5.8 also does not work with Cygwin and Ming.
Solution: Adjust the function calls. (Taro Muraoka)
Adjust the cyg and ming makefiles. (Dan Sharp)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak, src/if_perl.xs

Patch 6.1.333 (extra)

Problem: Win32: Can't handle Unicode text on the clipboard. Can't pass NUL byte, it becomes a line break. (Bruce DeVisser)
Solution: Support Unicode for the clipboard (Ron Aaron and Glenn Maynard)
Also support copy/paste of NUL bytes.
Files: src/os_mswin.c, src/os_win16.c src/os_win32.c

Patch 6.1.334 (extra, depends on 6.1.303)

Problem: Problem with drawing Hebrew characters.
Solution: Only use ETO_PDY for Windows NT and the like. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 6.1.335 (extra)

Problem: Failure of obtaining the cursor position and window size is ignored.
Solution: Remove a semicolon after an "if". (Walter Briscoe)
Files: src/gui_w32.c

Patch 6.1.336 (extra)

Problem: Warning for use of function prototypes of msg().
Solution: Define HAVE_STDARG_H. (Walter Briscoe)
Files: src/os_win32.h

Patch 6.1.337

Problem: When using "finish" in debug mode in function B() for ":call A(B())" does not stop after B() is finished.
Solution: Increase debug_level while evaluating a function.
Files: src/ex_docmd.c

Patch 6.1.338

Problem: When using a menu that checks out the current file from Insert mode, there is no warning for the changed file until exiting Insert mode. (Srikanth Sankaran)
Solution: Add a check for need_check_timestamps in the Insert mode loop.
Files: src/edit.c

Patch 6.1.339

Problem: Completion doesn't allow "g:" in ":let g:did_<Tab>". (Benji Fisher)
Solution: Return "g:var" for global variables when that is what is being expanded. (Flemming Madsen)
Files: src/eval.c

Patch 6.1.340 (extra, depends on 6.1.332)

Problem: Win32: Can't compile the Perl interface with nmake.
Solution: Don't compare the version number as a string but as a number. (Juergen Kraemer)
Files: src/Make_mvc.mak

Patch 6.1.341

Problem: In Insert mode with '**rightleft**' set the cursor is drawn halfway a double-wide character. For **CTRL-R** and **CTRL-K** in Insert mode the " or ? is not displayed.
Solution: Draw the cursor in the next character cell. Display the " or ? over the right half of the double-wide character. (Yasuhiro Matsumoto) Also fix that cancelling a digraph doesn't redraw a double-byte character correctly.
Files: src/edit.c, src/gui.c, src/mbyte.c

Patch 6.1.342 (depends on 6.1.341)

Problem: With '**rightleft**' set typing "c" on a double-wide character causes the cursor to be displayed one cell to the left.
Solution: Draw the cursor in the next character cell. (Yasuhiro Matsumoto)
Files: src/gui.c

Patch 6.1.343 (depends on 6.1.342)

Problem: Cannot compile with the +multi_byte feature but without +rightleft. Cannot compile without the GUI.
Solution: Fix the #ifdefs. (partly by Nam SungHyun)
Files: src/gui.c, src/mbyte.c, src/ui.c

Patch 6.1.344

Problem: When using ":silent filetype" the output is still put in the message history. (Hari Krishna Dara)
Solution: Don't add messages in the history when ":silent" is used.
Files: src/message.c

Patch 6.1.345 (extra)

Problem: Win32: 'imdisable' doesn't work.
Solution: Make 'imdisable' work. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 6.1.346

Problem: The scroll wheel can only scroll the current window.
Solution: Make the scroll wheel scroll the window that the mouse points to.
(Daniel Elstner)
Files: src/edit.c, src/gui.c, src/normal.c, src/term.c

Patch 6.1.347

Problem: When using cscope to list matching tags, the listed number is sometimes not equal to what cscope uses. (Vihren Milev)
Solution: For cscope tags use only one table, don't give tags in the current file a higher priority.
Files: src/tag.c

Patch 6.1.348

Problem: Wildmode with wildmenu: ":set wildmode=list,full", ":colorscheme <tab>" results in "zellner" instead of the first entry. (Anand Hariharan)
Solution: Don't call ExpandOne() from globpath(). (Flemming Madsen)
Files: src/ex_getln.c

Patch 6.1.349

Problem: "vim --serverlist" when no server was ever started gives an error message without "\n".
"vim --serverlist" doesn't exit when the X server can't be contacted, it starts Vim unexpectedly. (Ricardo Signes)
Solution: Don't give an error when no Vim server was ever started.
Treat failing of opening the display equal to errors inside the remote*() functions. (Flemming Madsen)
Files: src/if_xcmdsrv.c, src/main.c

Patch 6.1.350

Problem: When entering a buffer with ":bnext" for the first time, using an autocommand to restore the last used cursor position doesn't work. (Paolo Giarusso)
Solution: Don't use the last known cursor position of the current Vim invocation if an autocommand changed the position.
Files: src/buffer.c

Patch 6.1.351 (depends on 6.1.349)

Problem: Crash when starting Vim the first time in an X server. (John McGowan)
Solution: Don't call xFree() with a fixed string.
Files: src/if_xcmdsrv.c

Patch 6.1.352 (extra, depends on 6.1.345)

Problem: Win32: Crash when setting "imdisable" in _vimrc.
Solution: Don't call IME functions when imm32.dll was not loaded (yet). Also add typecasts to avoid Compiler warnings for ImmAssociateContext() argument.

Files: src/gui_w32.c

Patch 6.1.353 (extra, depends on 6.1.334)

Problem: Problem with drawing Arabic characters.

Solution: Don't use ETO_PDY, do use padding.

Files: src/gui_w32.c

Patch 6.1.354 (extra, depends on 6.1.333)

Problem: MS-Windows 98: Notepad can't paste text copied from Vim when **'encoding'** is "utf-8".

Solution: Also make CF_TEXT available on the clipboard. (Ron Aaron)

Files: src/os_mswin.c

Patch 6.1.355

Problem: In a regexp '\n' will never match anything in a string.

Solution: Make '\n' match a newline character.

Files: src/buffer.c, src/edit.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/misc1.c, src/option.c, src/os_mac.c, src/os_unix.c, src/quickfix.c, src/regexp.c, src/search.c, src/syntax.c, src/tag.c, src/vim.h

Patch 6.1.356 (extra, depends on, well, eh, several others)

Problem: Compiler warnings for using convert_setup() and a few other things.

Solution: Add typecasts.

Files: src/mbyte.c, src/os_mswin.c, src/proto/os_win32.pro, src/os_win32.c

Patch 6.1.357

Problem: CR in the quickfix window jumps to the error under the cursor, but this doesn't work in Insert mode. (Srikanth Sankaran)

Solution: Handle CR in Insert mode in the quickfix window.

Files: src/edit.c

Patch 6.1.358

Problem: The tutor doesn't select another locale version properly.

Solution: Insert the "let" command. (Yasuhiro Matsumoto)

Files: runtime/tutor/tutor.vim

Patch 6.1.359 (extra)

Problem: Mac Carbon: Vim doesn't get focus when started from the command line. Crash when using horizontal scroll bar.

Solution: Set Vim as the frontprocess. Fix scrolling. (Peter Cucka)

Files: src/gui_mac.c

Patch 6.1.360 (depends on 6.1.341)

Problem: In Insert mode **CTRL-K ESC** messes up a multibyte character. (Anders Helmersson)

Solution: Save all bytes of a character when displaying a character temporarily.

Files: src/edit.c, src/proto/screen.pro, src/screen.c

Patch 6.1.361

Problem: Cannot jump to a file mark with ":'M".

Solution: Allow jumping to another file for a mark in an Ex address when it

is the only thing in the command line.
Files: src/ex_docmd.c

Patch 6.1.362

Problem: tgetent() may return zero for success. tgetflag() may return -1 for an error.

Solution: Check tgetflag() for returning a positive value. Add an autoconf check for the value that tgetent() returns.

Files: src/auto/configure, src/config.h.in, src/configure.in, src/term.c

Patch 6.1.363

Problem: byte2line() can return one more than the number of lines.

Solution: Return -1 if the offset is one byte past the end.

Files: src/memline.c

Patch 6.1.364

Problem: That the FileChangedShell autocommand event never nests makes it difficult to reload a file in a normal way.

Solution: Allow nesting for the FileChangedShell event but do not allow triggering itself again.

Also avoid autocommands for the cmdline window in rare cases.

Files: src/ex_getln.c, src/fileio.c, src/window.c

Patch 6.1.365 (depends on 6.1.217)

Problem: Setting a breakpoint in a sourced file with a relative path name doesn't work. (Servatius Brandt)

Solution: Expand the file name to a full path.

Files: src/ex_cmds2.c

Patch 6.1.366

Problem: Can't use Vim with Netbeans.

Solution: Add the Netbeans interface. Includes support for sign icons and "-fg" and "-bg" arguments for GTK. Add the **'autochdir'** option. (Gordon Prieur, George Hernandez, Dave Weatherford)
Make it possible to display both a sign with a text and one with line highlighting in the same line.
Add support for Agide, interface version 2.1.
Also fix that when **'iskeyword'** includes '?' the "*" command doesn't work properly on a word that includes "?" (Bill McCarthy):
Don't escape "?" to "\?" when searching forward.

Files: runtime/doc/Makefile, runtime/doc/netbeans.txt, runtime/doc/options.txt, runtime/doc/vars.txt, src/Makefile, src/auto/configure, src/buffer.c, src/config.h.in, src/config.mk.in, src/configure.in, src/edit.c, src/ex_cmds.c, src/ex_docmd.c, src/feature.h, src/fileio.c, src/globals.h, src/gui.c, src/gui_beval.c, src/gui_gtk_x11.c, src/gui_x11.c, src/main.c, src/memline.c, src/misc1.c, src/misc2.c, src/move.c, src/nbdebug.c, src/nbdebug.h, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/option.h, src/proto/buffer.pro, src/proto/gui_beval.pro, src/proto/gui_gtk_x11.pro, src/proto/gui_x11.pro, src/proto/misc2.pro, src/proto/netbeans.pro, src/proto/normal.pro, src/proto/ui.pro, src/proto.h, src/screen.c, src/structs.h, src/ui.c, src/undo.c, src/vim.h, src/window.c, src/workshop.c

Patch 6.1.367 (depends on 6.1.365)

Problem: Setting a breakpoint in a function doesn't work. For a sourced file it doesn't work when symbolic links are involved. (Servatius Brandt)

Solution: Expand the file name in the same way as do_source() does. Don't prepend the path to a function name.

Files: src/ex_cmds2.c

Patch 6.1.368

Problem: Completion for ":map" does not include <silent> and <script>. ":mkexrc" do not save the <silent> attribute of mappings.

Solution: Add "<silent>" to the generated map commands when appropriate. (David Elstner)

Add <silent> and <script> to command line completion.

Files: src/getchar.c

Patch 6.1.369 (extra)

Problem: VMS: Vim hangs when attempting to edit a read-only file in the terminal. Problem with VMS filenames for quickfix.

Solution: Rewrite low level input. Remove version number from file name in a couple more places. Fix crash after patch 6.1.362. Correct return code for system(). (Zoltan Arpadffy, Tomas Stehlik)

Files: src/misc1.c, src/os_unix.c, src/os_vms.c, src/proto/os_vms.pro, src/os_vms_conf.h, src/quickfix.c, src/ui.c

Patch 6.1.370

Problem: #ifdef nesting is unclear.

Solution: Insert spaces to indicate the nesting.

Files: src/os_unix.c

Patch 6.1.371

Problem: "%V" in 'statusline' doesn't show "0-1" in an empty line.

Solution: Add one to the column when comparing with virtual column (Andrew Pimlott)

Files: src/buffer.c

Patch 6.1.372

Problem: With 16 bit ints there are compiler warnings. (Walter Briscoe)

Solution: Change int into long.

Files: src/structs.h, src/syntax.c

Patch 6.1.373

Problem: The default page header for printing is not translated.

Solution: Add _() around the two places where "Page" is used. (Mike Williams) Translate the default value of the 'titleold' and 'printhead' options.

Files: src/ex_cmds2.c, src/option.c

Patch 6.1.374 (extra)

Problem: MS-Windows: Cannot build GvimExt with MingW or Cygwin.

Solution: Add makefile and modified resource files. (Rene de Zwart)

Also support Cygwin. (Alejandro Lopez_Valencia)

Files: GvimExt/Make_cyg.mak, GvimExt/Make_ming.mak, GvimExt/Makefile,

GvimExt/gvimext_ming.def, GvimExt/gvimext_ming.rc

Patch 6.1.375

Problem: MS-Windows: '!dir "%'" does not work for a file name with spaces.
(Xiangjiang Ma)

Solution: Don't insert backslashes for spaces in a shell command.

Files: src/ex_docmd.c

Patch 6.1.376

Problem: "vim --version" and "vim --help" have a non-zero exit code.
That is unusual. (Petesea)

Solution: Use a zero exit code.

Files: src/main.c

Patch 6.1.377

Problem: Can't add words to '**lispwords**' option.

Solution: Add P_COMMA and P_NODUP flags. (Haakon Riiser)

Files: src/option.c

Patch 6.1.378

Problem: When two buffer-local user commands are ambiguous, a full match
with a global user command isn't found. (Hari Krishna Dara)

Solution: Detect this situation and accept the global command.

Files: src/ex_docmd.c

Patch 6.1.379

Problem: Linux with kernel 2.2 can't use the alternate stack in combination
with threading, causes an infinite loop.

Solution: Don't use the alternate stack in this situation.

Files: src/os_unix.c

Patch 6.1.380

Problem: When '**winminheight**' is zero and the quickfix window is zero lines,
entering the window doesn't make it higher. (Christian J.
Robinson)

Solution: Make sure the current window is at least one line high.

Files: src/window.c

Patch 6.1.381

Problem: When a BufWriteCmd is used and it leaves the buffer modified, the
window may still be closed. (Hari Krishna Dara)

Solution: Return FAIL from buf_write() when the buffer is still modified
after a BufWriteCmd autocommand was used.

Files: src/fileio.c

Patch 6.1.382 (extra)

Problem: Win32 GUI: When using two monitors, the code that checks/fixes the
window size and position (e.g. when a font changes) doesn't work
properly. (George Reilly)

Solution: Handle a double monitor situation. (Helmut Stiegler)

Files: src/gui_w32.c

Patch 6.1.383

Problem: The filling of the status line doesn't work properly for

multibyte characters. (Nam SungHyun)

There is no check for going past the end of the buffer.

Solution: Properly distinguish characters and bytes. Properly check for running out of buffer space.

Files: src/buffer.c, src/ex_cmds2.c, src/proto/buffer.pro, src/screen.c

Patch 6.1.384

Problem: It is not possible to find if a certain patch has been included. (Lubomir Host)

Solution: Support using has() to check if a patch was included.

Files: runtime/doc/eval.txt, src/eval.c, src/proto/version.pro, src/version.c

Patch 6.1.385 (depends on 6.1.383)

Problem: Can't compile without the multibyte feature.

Solution: Move an #ifdef. (Christian J. Robinson)

Files: src/buffer.c

Patch 6.1.386

Problem: Get duplicate tags when running ":helptags".

Solution: Do the other half of moving a section to another help file.

Files: runtime/tagsrch.txt

Patch 6.1.387 (depends on 6.1.373)

Problem: Compiler warning for pointer cast.

Solution: Add (char_u *).

Files: src/option.c

Patch 6.1.388 (depends on 6.1.384)

Problem: Compiler warning for pointer cast.

Solution: Add (char *). Only include has_patch() when used.

Files: src/eval.c, src/version.c

Patch 6.1.389 (depends on 6.1.366)

Problem: Balloon evaluation doesn't work for GTK. has("balloon_eval") doesn't work.

Solution: Add balloon evaluation for GTK. Also improve displaying of signs. (Daniel Elstner)

Also make ":gui" start the netbeans connection and avoid using netbeans functions when the connection is not open.

Files: src/Makefile, src/feature.h, src/gui.c, src/gui.h, src/gui_beval.c, src/gui_beval.h, src/gui_gtk.c, src/gui_gtk_x11.c, src/eval.c, src/memline.c, src/menu.c, src/netbeans.c, src/proto/gui_beval.pro, src/proto/gui_gtk.pro, src/structs.h, src/syntax.c, src/ui.c, src/workshop.c

Patch 6.1.390 (depends on 6.1.389)

Problem: It's not possible to tell Vim to save and exit through the Netbeans interface. Would still try to send balloon eval text after the connection is closed.

Can't use Unicode characters for sign text.

Solution: Add functions "saveAndExit" and "getModified". Check for a working connection before sending a balloonText event. various other cleanups.

Support any character for sign text. (Daniel Elstner)
Files: runtime/doc/netbeans.txt, runtime/doc/sign.txt, src/ex_cmds.c,
src/netbeans.c, src/screen.c

Patch 6.1.391

Problem: ml_get() error when using virtualedit. (Charles Campbell)
Solution: Get a line from a specific window, not the current one.
Files: src/charset.c

Patch 6.1.392 (depends on 6.1.383)

Problem: Highlighting in the '**statusline**' is in the wrong position when an
item is truncated. (Zak Beck)
Solution: Correct the start of '**statusline**' items properly for a truncated
item.
Files: src/buffer.c

Patch 6.1.393

Problem: When compiled with Python and threads, detaching the terminal may
cause Vim to loop forever.
Solution: Add -pthread to \$CFLAGS when using Python and gcc. (Daniel
Elstner)
Files: src/auto/configure,, src/configure.in

Patch 6.1.394 (depends on 6.1.390)

Problem: The netbeans interface doesn't recognize multibyte glyph names.
Solution: Check the number of cells rather than bytes to decide
whether a glyph name is not a filename. (Daniel Elstner)
Files: src/netbeans.c

Patch 6.1.395 (extra, depends on 6.1.369)

Problem: VMS: OLD_VMS is never defined. Missing function prototype.
Solution: Define OLD_VMS in Make_vms.mms. Add vms_sys_status() to
os_vms.pro. (Zoltan Arpadffy)
Files: src/Make_vms.mms, src/proto/os_vms.pro

Patch 6.1.396 (depends on 6.1.330)

Problem: Compiler warnings for using enum.
Solution: Add typecast to char_u.
Files: src/gui_gtk_x11.c, src/gui_x11.c

Patch 6.1.397 (extra)

Problem: The install program may use a wrong path for the diff command if
there is a space in the install directory path.
Solution: Use double quotes around the path if necessary. (Alejandro
Lopez-Valencia) Also use double quotes around the file name
arguments.
Files: src/dosinst.c

Patch 6.1.398

Problem: Saving the typeahead for debug mode causes trouble for a test
script. (Servatius Brandt)
Solution: Add the ":debuggreedy" command to avoid saving the typeahead.
Files: runtime/doc/repeat.txt, src/ex_cmds.h, src/ex_cmds2.c,
src/ex_docmd.c, src/proto/ex_cmds2.pro

Patch 6.1.399

Problem: Warning for unused variable.
Solution: Remove the variable two_or_more.
Files: src/ex_cmds.c

Patch 6.1.400 (depends on 6.1.381)

Problem: When a BufWriteCmd wipes out the buffer it may still be accessed.
Solution: Don't try accessing a buffer that has been wiped out.
Files: src/fileio.c

Patch 6.1.401 (extra)

Problem: Building the Win16 version with Borland 5.01 doesn't work.
"make test" doesn't work with Make_dos.mak. (Walter Briscoe)
Solution: Various fixes to the w16 makefile. (Walter Briscoe)
Don't use deltree. Use "mkdir \tmp" instead of "mkdir /tmp".
Files: src/Make_w16.mak, src/testdir/Make_dos.mak

Patch 6.1.402

Problem: When evaluating a function name with curly braces, an error is not handled consistently.
Solution: Accept the result of a curly braces expression when an error was encountered. Skip evaluating an expression in curly braces when skipping. (Servatius Brandt)
Files: src/eval.c

Patch 6.1.403 (extra)

Problem: MS-Windows 16 bit: compiler warnings.
Solution: Add typecasts. (Walter Briscoe)
Files: src/ex_cmds2.c, src/gui_w48.c, src/os_mswin.c, src/os_win16.c, src/syntax.c

Patch 6.1.404 (extra)

Problem: Various small problems.
Solution: Fix comments. Various small additions, changes in indent, removal of unused items and fixes.
Files: Makefile, README.txt, runtime/menu.vim, runtime/vimrc_example.vim, src/INSTALL, src/INSTALLole.txt, src/Make_bc5.mak, src/Make_cyg.mak, src/Make_ming.mak, src/Makefile, src/config.h.in, src/edit.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/gui.c, src/gui_gtk.c, src/gui_photon.c, src/if_cscope.c, src/if_python.c, src/keymap.h, src/mark.c, src/mbyte.c, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/option.c, src/os_os2_cfg.h, src/os_win32.c, src/proto/getchar.pro, src/proto/message.pro, src/proto/regexp.pro, src/screen.c, src/structs.h, src/syntax.c, src/term.c, src/testdir/test15.in, src/testdir/test15.ok, src/vim.rc, src/xxd/Make_cyg.mak, src/xxd/Makefile

Patch 6.1.405

Problem: A few files are missing from the toplevel Makefile.
Solution: Add the missing files.
Files: Makefile

Patch 6.1.406 (depends on 6.1.392)

Problem: When a statusline item doesn't fit arbitrary text appears.
(Christian J. Robinson)

Solution: When there is just enough room but not for the "<" truncate the statusline item like there is no room.

Files: src/buffer.c

Patch 6.1.407

Problem: ":set scrollbind | help" scrollbinds the help window. (Andrew Pimlott)

Solution: Reset '**scrollbind**' when opening a help window.

Files: src/ex_cmds.c

Patch 6.1.408

Problem: When '**rightleft**' is set unprintable character 0x0c is displayed as ">c0<".

Solution: Reverse the text of the hex character.

Files: src/screen.c

Patch 6.1.409

Problem: Generating tags for the help doesn't work for some locales.

Solution: Set LANG=C LC_ALL=C in the environment for "sort". (Daniel Elstner)

Files: runtime/doc/Makefile

Patch 6.1.410 (depends on 6.1.390)

Problem: Linking error when compiling with Netbeans but without sign icons.
(Malte Neumann)

Solution: Don't define buf_signcount() when sign icons are unavailable.

Files: src/buffer.c

Patch 6.1.411

Problem: When '**virtualedit**' is set, highlighting a Visual block beyond the end of a line may be wrong.

Solution: Correct the virtual column when the end of the line is before the displayed part of the line. (Muraoka Taro)

Files: src/screen.c

Patch 6.1.412

Problem: When swapping terminal screens and using ":gui" to start the GUI, the shell prompt may be after a hit-enter prompt.

Solution: Output a newline in the terminal when starting the GUI and there was a hit-enter prompt..

Files: src/gui.c

Patch 6.1.413

Problem: When '**clipboard**' contains "unnamed", "p" in Visual mode doesn't work correctly.

Solution: Save the register before overwriting it and put the resulting text on the clipboard afterwards. (Muraoka Taro)

Files: src/normal.c, src/ops.c

Patch 6.1.414 (extra, depends on 6.1.369)

Problem: VMS: Vim busy waits when waiting for input.
Solution: Delay for a short while before getting another character. (Zoltan Arpadffy)
Files: src/os_vms.c

Patch 6.1.415

Problem: When there is a vertical split and a quickfix window, reducing the size of the Vim window may result in a wrong window layout and a crash.
Solution: When reducing the window size and there is not enough space for **'winfixheight'** set the frame height to the larger height, so that there is a retry while ignoring **'winfixheight'**. (Yasuhiro Matsumoto)
Files: src/window.c

Patch 6.1.416 (depends on 6.1.366)

Problem: When using the Netbeans interface, a line with a sign cannot be changed.
Solution: Respect the GUARDEDOFFSET for sign IDs when checking for a guarded area.
Files: src/netbeans.c

Patch 6.1.417

Problem: Unprintable multibyte characters are not handled correctly. Multi-byte characters above 0xffff are displayed as another character.
Solution: Handle unprintable multibyte characters. Display multibyte characters above 0xffff with a marker. Recognize UTF-16 words and BOM words as unprintable. (Daniel Elstner)
Files: src/charset.c, src/mbyte.c, src/screen.c

Patch 6.1.418

Problem: The result of strftime() is in the current locals. Need to convert it to **'encoding'**.
Solution: Obtain the current locale and convert the argument for strftime() to it and the result back to **'encoding'**. (Daniel Elstner)
Files: src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/mbyte.c, src/proto/mbyte.pro, src/option.c, src/os_mswin.c

Patch 6.1.419

Problem: Vim doesn't compile on AIX 5.1.
Solution: Don't define _NO_PROTO on this system. (Uribarri)
Files: src/auto/configure, src/configure.in

Patch 6.1.420 (extra)

Problem: convert_input() has an unnecessary STRLEN(). Conversion from UCS-2 to a codepage uses word count instead of byte count.
Solution: Remove the STRLEN() call. (Daniel Elstner)
Always use byte count for string_convert().
Files: src/gui_w32.c, src/mbyte.c

Patch 6.1.421 (extra, depends on 6.1.354)

Problem: MS-Windows 9x: When putting text on the clipboard it can be in

the wrong encoding.
Solution: Convert text to the active codepage for CF_TEXT. (Glenn Maynard)
Files: src/os_mswin.c

Patch 6.1.422

Problem: Error in .vimrc doesn't cause hit-enter prompt when swapping screens. (Neil Bird)
Solution: Set msg_didany also when sending a message to the terminal directly.
Files: src/message.c

Patch 6.1.423

Problem: Can't find arbitrary text in help files.
Solution: Added the ":helpgrep" command.
Files: runtime/doc/various.txt, src/ex_cmds.h, src/ex_docmd.c, src/proto/quickfix.pro, src/quickfix.c

Patch 6.1.424 (extra)

Problem: Win32: gvim compiled with VC++ 7.0 run on Windows 95 does not show menu items.
Solution: Define \$WINVER to avoid an extra item is added to MENUITEMINFO. (Muraoka Taro)
Files: src/Make_mvc.mak

Patch 6.1.425

Problem: ":helptags \$VIMRUNTIME/doc" does not add the "help-tags" tag.
Solution: Do add the "help-tags" tag for that specific directory.
Files: src/ex_cmds.c

Patch 6.1.426

Problem: "--remote-wait +cmd file" waits forever. (Valery Kondakoff)
Solution: Don't wait for the "+cmd" argument to have been edited.
Files: src/main.c

Patch 6.1.427

Problem: Several error messages for regexp patterns are not translated.
Solution: Use _() properly. (Muraoka Taro)
Files: src/regexp.c

Patch 6.1.428

Problem: FreeBSD: wait() may hang when compiled with Python support and doing a system() call in a startup script.
Solution: Use waitpid() instead of wait() and poll every 10 msec, just like what is done in the GUI.
Files: src/os_unix.c

Patch 6.1.429 (depends on 6.1.390)

Problem: Crash when using showmarks.vim plugin. (Charles Campbell)
Solution: Check for sign_get_text() returning a NULL pointer.
Files: src/screen.c

Patch 6.1.430

Problem: In Lisp code backslashed parens should be ignored for "%". (Dorai)
Solution: Skip over backslashed parens.

Files: src/search.c

Patch 6.1.431

Problem: Debug commands end up in redirected text.

Solution: Disable redirection while handling debug commands.

Files: src/ex_cmds2.c

Patch 6.1.432 (depends on 6.1.375)

Problem: MS-Windows: ":make %:p" inserts extra backslashes. (David Rennalls)

Solution: Don't add backslashes, handle it like "!:cmd".

Files: src/ex_docmd.c

Patch 6.1.433

Problem: ":popup" only works for Win32.

Solution: Add ":popup" support for GTK. (Daniel Elstner)

Files: runtime/doc/gui.txt, src/ex_docmd.c, src/gui_gtk.c, src/menu.c,
src/proto/gui_gtk.pro

Patch 6.1.434 (extra)

Problem: Win32: When there are more than 32767 lines, the scrollbar has a roundoff error.

Solution: Make a click on an arrow move one line. Also move the code to gui_w48.c, there is hardly any difference between the 16 bit and 32 bit versions. (Walter Briscoe)

Files: src/gui_w16.c, src/gui_w32.c, src/gui_w48.c

Patch 6.1.435

Problem: ":winsize x" resizes the Vim window to the minimal size. (Andrew Pimlott)

Solution: Give an error message for wrong arguments of ":winsize" and ":winpos".

Files: src/ex_docmd.c

Patch 6.1.436

Problem: When a long UTF-8 file contains an illegal byte it's hard to find out where it is. (Ron Aaron)

Solution: Add the line number to the error message.

Files: src/fileio.c

Patch 6.1.437 (extra, depends on 6.1.421)

Problem: Using multibyte functions when they are not available.

Solution: Put the clipboard conversion inside an #ifdef. (Vince Negri)
Also fix a pointer type mistake. (Walter Briscoe)

Files: src/os_mswin.c

Patch 6.1.438

Problem: When Perl has thread support Vim cannot use the Perl interface.

Solution: Add a configure check and disable Perl when it will not work. (Aron Griffis)

Files: src/auto/configure, src/configure.in

Patch 6.1.439

Problem: Netbeans: A "create" function doesn't actually create a buffer, following functions may fail.

Solution: Create a Vim buffer without a name when "create" is called.
(Gordon Prieur)
Files: runtime/doc/netbeans.txt, src/netbeans.c

Patch 6.1.440

Problem: The "@*" command doesn't obtain the actual contents of the clipboard. (Hari Krishna Dara)
Solution: Obtain the clipboard text before executing the command.
Files: src/ops.c

Patch 6.1.441

Problem: "zj" and "zk" cannot be used as a motion command after an operator. (Ralf Hetzel)
Solution: Accept these commands as motion commands.
Files: src/normal.c

Patch 6.1.442

Problem: Unicode 3.2 defines more space and punctuation characters.
Solution: Add the new characters to the Unicode tables. (Raphael Finkel)
Files: src/mbyte.c

Patch 6.1.443 (extra)

Problem: Win32: The gvimext.dll build with Borland 5.5 requires another DLL.
Solution: Build a statically linked version by default. (Dan Sharp)
Files: GvimExt/Make_bc5.mak

Patch 6.1.444 (extra)

Problem: Win32: Enabling a build with gettext support is not consistent.
Solution: Use "GETTEXT" for Borland and msvc makefiles. (Dan Sharp)
Files: src/Make_bc5.mak, src/Make_mvc.mak

Patch 6.1.445 (extra)

Problem: DJGPP: get warning for argument of putenv()
Solution: Define HAVE_PUTENV to use DJGPP's putenv(). (Walter Briscoe)
Files: src/os_msdos.h

Patch 6.1.446 (extra)

Problem: Win32: The MingW makefile uses a different style of arguments than other makefiles.
Dynamic IME is not supported for Cygwin.
Solution: Use "no" and "yes" style arguments. Remove the use of the dyn-ming.h include file. (Dan Sharp)
Do not include the ime.h file and adjust the makefile. (Alejandro Lopez-Valencia)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/gui_w32.c, src/if_perl.xs, src/if_python.c, src/if_ruby.c, src/os_win32.c

Patch 6.1.447

Problem: "make install" uses "make" directly for generating help tags.
Solution: Use \$(MAKE) instead of "make". (Tim Mooney)
Files: src/Makefile

Patch 6.1.448

Problem: **'titlestring'** has a default maximum width of 50 chars per item.
Solution: Remove the default maximum (also for **'statusline'**).
Files: src/buffer.c

Patch 6.1.449

Problem: When "1" and "a" are in **'formatoptions'**, auto-formatting always moves a newly added character to the next line. (Servatius Brandt)
Solution: Don't move a single character to the next line when it was just typed.
Files: src/edit.c

Patch 6.1.450

Problem: Termcap entry "kB" for back-tab is not recognized.
Solution: Use back-tab as the shift-tab code.
Files: src/keymap.h, src/misc2.c, src/term.c

Patch 6.1.451

Problem: GUI: When text in the find dialog contains a slash, a backslash is inserted the next time it is opened. (Mezz)
Solution: Remove escaped backslashes and question marks. (Daniel Elstner)
Files: src/gui.c

Patch 6.1.452 (extra, after 6.1.446)

Problem: Win32: IME support doesn't work for MSVC.
Solution: Use `_MSC_VER` instead of `__MSVC`. (Alejandro Lopez-Valencia)
Files: src/gui_w32.c

Patch 6.1.453 (after 6.1.429)

Problem: When compiled without sign icons but with sign support, adding a sign may cause a crash.
Solution: Check for the text sign to exist before using it. (Kamil Burzynski)
Files: src/screen.c

Patch 6.1.454 (extra)

Problem: Win32: pasting Russian text in Vim with **'enc'** set to cp1251 results in utf-8 bytes. (Perelyubskiy)
Conversion from DBCS to UCS2 does not work when **'encoding'** is not the active codepage.
Solution: Introduce `enc_codepage` and use it for conversion to **'encoding'** (Glenn Maynard)
Use `MultiByteToWideChar()` and `WideCharToMultiByte()` instead of `iconv()`. Should do most needed conversions without `iconv.dll`.
Files: src/globals.h, src/gui_w32.c, src/mbyte.c, src/os_mswin.c, src/proto/mbyte.pro, src/proto/os_mswin.pro, src/structs.h

Patch 6.1.455

Problem: Some Unicode characters can be one or two character cells wide.
Solution: Add the **'ambiwidth'** option to tell Vim how to display these characters. (Jungshik Shin)
Also reset the script ID when setting an option to its default value, so that `":verbose set"` won't give wrong info.
Files: runtime/doc/options.txt, src/mbyte.c, src/option.c, src/option.h

Patch 6.1.456 (extra, after 6.1.454)

Problem: Win32: IME doesn't work.

Solution: ImmGetCompositionStringW() returns the size in bytes, not words.
(Yasuhiro Matsumoto) Also fix typecast problem.

Files: src/gui_w32.c, src/os_mswin.c

Patch 6.1.457

Problem: An empty register in viminfo causes conversion to fail.

Solution: Don't convert an empty string. (Yasuhiro Matsumoto)

Files: src/ex_cmds.c, src/mbyte.c

Patch 6.1.458

Problem: Compiler warning for pointer.

Solution: Add a typecast.

Files: src/ex_cmds.c

Patch 6.1.459 (extra)

Problem: Win32: libcall() may return an invalid pointer and cause Vim to crash.

Solution: Add a strict check for the returned pointer. (Bruce Mellows)

Files: src/os_mswin.c

Patch 6.1.460

Problem: GTK: after scrolling the text one line with a key, clicking the arrow of the scrollbar does not always work. (Nam SungHyun)

Solution: Always update the scrollbar thumb when the value changed, even when it would not move, like for RISCOS. (Daniel Elstner)

Files: src/gui.c, src/gui.h

Patch 6.1.461

Problem: When a keymap is active, typing a character in Select mode does not use it. (Benji Fisher)

Solution: Apply Insert mode mapping to the character typed in Select mode.

Files: src/normal.c

Patch 6.1.462

Problem: When autocommands wipe out a buffer, a crash may happen. (Hari Krishna Dara)

Solution: Don't decrement the window count of a buffer before calling the autocommands for it. When re-using the current buffer, watch out for autocommands changing the current buffer.

Files: src/buffer.c, src/ex_cmds.c, src/proto/buffer.pro

Patch 6.1.463

Problem: When writing a compressed file, the file name that gzip stores in the file is the weird temporary file name. (David Rennalls)

Solution: Use the real file name when possible.

Files: runtime/plugin/gzip.vim

Patch 6.1.464

Problem: Crash when using C++ syntax highlighting. (Gerhard Hochholzer)

Solution: Check for a negative index.

Files: src/syntax.c

Patch 6.1.465 (after 6.1.454)

Problem: Compile error when using cygwin.

Solution: Change #ifdef WIN32 to #ifdef WIN3264. (Alejandro Lopez-Valencia)
Undefine WIN32 after including windows.h

Files: src/mbyte.c

Patch 6.1.466

Problem: The "-f" argument is a bit obscure.

Solution: Add the "--nofork" argument. Improve the help text a bit.

Files: runtime/doc/starting.txt, src/main.c

Patch 6.1.467

Problem: Setting the window title doesn't work for Chinese.

Solution: Use an X11 function to convert text to a text property. (Kentaro Nakazawa)

Files: src/os_unix.c

Patch 6.1.468

Problem: ":mksession" also stores folds for buffers which will not be restored.

Solution: Only store folds for a buffer with **'buftype'** empty and help files.

Files: src/ex_docmd.c

Patch 6.1.469

Problem: **'listchars'** cannot contain multibyte characters.

Solution: Handle multibyte UTF-8 list characters. (Matthew Samsonoff)

Files: src/message.c, src/option.c, src/screen.c

Patch 6.1.470 (lang)

Problem: Polish messages don't show up correctly on MS-Windows.

Solution: Convert messages to cp1250. (Mikolaj Machowski)
Also add English message translations, because it got in the way of the patch.

Files: Makefile, src/po/Makefile, src/po/en_gb.po, src/po/pl.po

Patch 6.1.471

Problem: ":jumps" output continues after pressing "q" at the more-prompt. (Hari Krishna Dara)

Solution: Check for "got_int" being set.

Files: src/mark.c

Patch 6.1.472

Problem: When there is an authentication error when connecting to the X server Vim exits.

Solution: Use XSetIOErrorHandler() to catch the error and longjmp() to avoid the exit. Also do this in the main loop, so that when the X server exits a Vim running in a console isn't killed.

Files: src/globals.h, src/main.c, src/os_unix.c

Patch 6.1.473

Problem: Referring to \$curwin or \$curbuf in Perl 5.6 causes a crash.

Solution: Add "pTHX_" to cur_val(). (Yasuhiro Matsumoto)

Files: src/if_perl.xs

Patch 6.1.474

Problem: When opening the command-line window in Ex mode it's impossible to go back. (Pavol Juhas)
Solution: Reset "exmode_active" and restore it when the command-line window is closed.
Files: src/ex_getln.c

Patch 6.2f.001

Problem: The configure check for Ruby didn't work properly for Ruby 1.8.0.
Solution: Change the way the Ruby check is done. (Aron Griffis)
Files: src/auto/configure, src/configure.in

Patch 6.2f.002

Problem: The output of ":ls" doesn't show whether a buffer had read errors.
Solution: Add the "x" flag in the ":ls" output.
Files: runtime/doc/windows.txt, src/buffer.c

Patch 6.2f.003

Problem: Test49 doesn't properly test the behavior of ":catch" without an argument.
Solution: Update test49. (Servatius Brandt)
Files: src/testdir/test49.ok, src/testdir/test49.vim

Patch 6.2f.004

Problem: "vim --version" always uses CR/LF in the output.
Solution: Omit the CR.
Files: src/message.c, src/os_unix.c

Patch 6.2f.005

Problem: Two error messages without a colon after the number.
Solution: Add the colon. (Taro Muraoka)
Files: src/if_cscope.c

Patch 6.2f.006

Problem: When saving a file takes a while and Vim regains focus this can result in a "file changed outside of Vim" warning and ml_get() errors. (Mike Williams)
Solution: Add the "b_saving" flag to avoid checking the timestamp while the buffer is being saved. (Michael Schaap)
Files: src/fileio.c, src/structs.h

Patch 6.2f.007

Problem: Irix compiler complains about multiple defined symbols. vsnprintf() is not available. (Charles Campbell)
Solution: Insert EXTERN for variables in globals.h. Change the configure check for vsnprintf() from compiling to linking.
Files: src/auto/configure, src/configure.in, src/globals.h

Patch 6.2f.008

Problem: The Aap recipe doesn't work with Aap 0.149.
Solution: Change targetarg to TARGETARG. Update the mysign file.
Files: src/main.aap, src/mysign

Patch 6.2f.009 (extra)

Problem: Small problem when building with Borland 5.01.
Solution: Use mkdir() instead of _mkdir(). (Walter Briscoe)
Files: src/dosinst.h

Patch 6.2f.010

Problem: Warning for missing prototypes.
Solution: Add missing prototypes. (Walter Briscoe)
Files: src/if_cscope.c

Patch 6.2f.011

Problem: The configure script doesn't work with autoconf 2.5x.
Solution: Add square brackets around a header check. (Aron Griffis)
Note: touch src/auto/configure after applying this patch.
Files: src/configure.in

Patch 6.2f.012

Problem: ":echoerr" doesn't work correctly inside try/endtry.
Solution: Don't reset did_emsg inside a try/endtry. (Servatius Brandt)
Files: src/eval.c

Patch 6.2f.013 (extra)

Problem: Macintosh: Compiler warning for a trigraph.
Solution: Insert a backslash before each question mark. (Peter Cucka)
Files: src/os_mac.h

Patch 6.2f.014 (extra)

Problem: Macintosh: ex_eval is not included in the project file.
Solution: Add ex_eval. (Dany St-Amant)
Files: src/os_mac.pbproj/project.pbxproj

Patch 6.2f.015 (extra)

Problem: Win32: When changing header files not all source files involved are recompiled.
Solution: Improve the dependency rules. (Dan Sharp)
Files: src/Make_cyg.mak, src/Make_ming.mak

Patch 6.2f.016

Problem: "vim --version > ff" on non-Unix systems results in a file with a missing line break at the end. (Bill McCarthy)
Solution: Add a line break.
Files: src/main.c

Patch 6.2f.017

Problem: Unix: starting Vim in the background and then bringing it to the foreground may cause the terminal settings to be wrong.
Solution: Check for tcsetattr() to return an error, retry when it does. (Paul Tapper)
Files: src/os_unix.c

Patch 6.2f.018

Problem: Mac OS X 10.2: OK is defined to zero in curses.h while Vim uses one. Redefining it causes a warning message.
Solution: Undefine OK before defining it to one. (Taro Muraoka)

Files: src/vim.h

Patch 6.2f.019

Problem: Mac OS X 10.2: COLOR_BLACK and COLOR_WHITE are defined in curses.h.

Solution: Rename them to PRCOLOR_BLACK and PRCOLOR_WHITE.

Files: src/ex_cmds2.c

Patch 6.2f.020

Problem: Win32: test50 produces beeps and fails with some versions of diff.

Solution: Remove empty lines and convert the output to dos fileformat.

Files: src/testdir/test50.in

Patch 6.2f.021

Problem: Running configure with "--enable-netbeans" disables Netbeans. (Gordon Prieur)

Solution: Fix the tests in configure.in where the default is to enable a feature. Fix that "--enable-acl" reported "yes" confusingly.

Files: src/auto/configure, src/configure.in, src/mysign

Patch 6.2f.022

Problem: A bogus value for 'foldmarker' is not rejected, possibly causing a hang. (Derek Wyatt)

Solution: Check for a non-empty string before and after the comma.

Files: src/option.c

Patch 6.2f.023

Problem: When the help files are not in \$VIMRUNTIME but 'helpfile' is correct Vim still can't find the help files.

Solution: Also look for a tags file in the directory of 'helpfile'.

Files: src/tag.c

Patch 6.2f.024

Problem: When 'delcombine' is set and a character has more than two composing characters "x" deletes them all.

Solution: Always delete only the last composing character.

Files: src/misc1.c

Patch 6.2f.025

Problem: When reading a file from stdin that has DOS line endings but a missing end-of-line for the last line 'fileformat' becomes "unix". (Bill McCarthy)

Solution: Don't add the missing line break when re-reading the text from the buffer.

Files: src/fileio.c

Patch 6.2f.026

Problem: When typing new text at the command line, old composing characters may be displayed.

Solution: Don't read composing characters from after the end of the text to be displayed.

Files: src/ex_getln.c, src/mbyte.c, src/message.c, src/proto/mbyte.pro, src/screen.c

Patch 6.2f.027

Problem: Compiler warnings for unsigned char pointers. (Tony Leneis)
Solution: Add typecasts to char pointer.
Files: src/quickfix.c

Patch 6.2f.028

Problem: GTK: When **'imactivatekey'** is empty and XIM is inactive it can't be made active again. Cursor isn't updated immediately when changing XIM activation. Japanese XIM may hang when using **'imactivatekey'**. Can't activate XIM after typing fFtT command or ":sh".
Solution: Properly set the flag that indicates the IM is active. Update the cursor right away. Do not send a key-release event. Handle Normal mode and running an external command differently. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 6.2f.029

Problem: Mixing use of int and enum.
Solution: Adjust argument type of cs_usage_msg(). Fix wrong typedef.
Files: src/if_cscope.c, src/if_cscope.h

Patch 6.2f.030 (after 6.2f.028)

Problem: Cursor moves up when using XIM.
Solution: Reset im_preedit_cursor. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 6.2f.031

Problem: Crash when listing a function argument in the debugger. (Ron Aaron)
Solution: Init the name field of an argument to NULL.
Files: src/eval.c

Patch 6.2f.032

Problem: When a write fails for a ":silent!" while inside try/entry the BufWritePost autocommands are not triggered.
Solution: Check the emsg_silent flag in should_abort(). (Servatius Brandt)
Files: src/ex_eval.c, src/testdir/test49.ok, src/testdir/test49.vim

Patch 6.2f.033

Problem: Cscope: re-entrance problem for ":cscope" command. Checking for duplicate database didn't work well for Win95. Didn't check for duplicate databases after an empty entry.
Solution: Don't set postponed_split too early. Remember first empty database entry. (Sergey Khorev)
Files: src/if_cscope.c

Patch 6.2f.034

Problem: The netbeans interface cannot be used on systems without vsnprintf(). (Tony Leneis)
Solution: Use EMSG(), EMSGN() and EMSG2() instead.
Files: src/auto/configure, src/configure.in, src/netbeans.c

Patch 6.2f.035

Problem: The configure check for the netbeans interface doesn't work if the socket and nsl libraries are required.

Solution: Check for the socket and nsl libraries before the netbeans check.
Files: src/auto/configure, src/configure.in

Patch 6.2f.036

Problem: Moving leftwards over text with an illegal UTF-8 byte moves one byte instead of one character.

Solution: Ignore an illegal byte after the cursor position.

Files: src/mbyte.c

Patch 6.2f.037

Problem: When receiving a Netbeans command at the hit-enter or more prompt the screen is redrawn but Vim is still waiting at the prompt.

Solution: Quit the prompt like a **CTRL-C** was typed.

Files: src/netbeans.c

Patch 6.2f.038

Problem: The dependency to run autoconf causes a patch for configure.in to run autoconf, even though the configure script was updated as well.

Solution: Only run autoconf with "make autoconf".

Files: src/Makefile

Patch 6.2f.039

Problem: **CTRL-W K** makes the new top window very high.

Solution: When '**equalalways**' is set equalize the window heights.

Files: src/window.c

VERSION 6.3

version-6.3

This section is about improvements made between version 6.2 and 6.3.

This is mainly a bug-fix release. There are also a few new features. The major number of new items is in the runtime files and translations.

Changed

changed-6.3

The intro message also displays a [note](#) about sponsoring Vim, mixed randomly with the message about helping children in Uganda.

Included the translated menus, keymaps and tutors with the normal runtime files. The separate "lang" archive now only contains translated messages.

Made the translated menu file names a bit more consistent. Use "latin1" for "iso_8859-1" and "iso_8859-15".

Removed the "file_select.vim" script from the distribution. It's not more useful than other scripts that can be downloaded from www.vim.org.

The "runtime/doc/tags" file is now always in unix fileformat. On MS-Windows it used to be dos fileformat, but ":helptags" generates a unix format file.

Added

added-6.3

New commands:

:cNfile	go to last error in previous file
:cpfile	idem
:changes	print the change list
:keepmarks	following command keeps marks where they are
:keepjumps	following command keeps jumplist and marks
:lockmarks	following command keeps marks where they are
:redrawstatus	force a redraw of the status line(s)

New options:

'antialias'	Mac OS X: use smooth, antialiased fonts
'helplang'	preferred help languages

Syntax files:

- Arch inventory (Nikolai Weibull)
- Calendar (Nikolai Weibull)
- Ch (Wayne Cheng)
- Controllable Regex Mutilator (Nikolai Weibull)
- D (Jason Mills)
- Desktop (Mikolaj Machowski)
- Dircolors (Nikolai Weibull)
- Elinks configuration (Nikolai Weibull)
- FASM (Ron Aaron)
- GrADS scripts (Stefan Fronzek)
- Icwm menu (James Mahler)
- LDIF (Zak Johnson)
- Locale input, fdcc. (Dwayne Bailey)
- Pinfo config (Nikolai Weibull)
- Pyrex (Marco Barisione)
- Relax NG Compact (Nikolai Weibull)
- Slice (Morel Bodin)
- VAX Macro Assembly (Tom Uijldert)
- grads (Stefan Fronzek)
- libao (Nikolai Weibull)
- mplayer (Nikolai Weibull)
- rst (Nikolai Weibull)
- tcsh (Gautam Iyer)
- yaml (Nikolai Weibull)

Compiler plugins:

- ATT dot (Marcos Macedo)
- Apple Project Builder (Alexander von Below)
- Intel (David Harrison)
- bdf (Nikolai Weibull)
- icc (Peter Puck)
- javac (Doug Kearns)
- neato (Marcos Macedo)
- onsgmls (Robert B. Rowsome)
- perl (Christian J. Robinson)

rst (Nikolai Weibull)
se (SmartEiffel) (Doug Kearns)
tcl (Doug Kearns)
xmlwf (Robert B. Rowsome)

Filetype plugins:

Aap (Bram Moolenaar)
Ch (Wayne Cheng)
Css (Nikolai Weibull)
Pyrex (Marco Barisione)
Rst (Nikolai Weibull)

Indent scripts:

Aap (Bram Moolenaar)
Ch (Wayne Cheng)
DocBook (Nikolai Weibull)
MetaPost (Eugene Minkovskii)
Objective-C (Kazunobu Kuriyama)
Pyrex (Marco Barisione)
Rst (Nikolai Weibull)
Tcsh (Gautam Iyer)
XFree86 configuration file (Nikolai Weibull)
Zsh (Nikolai Weibull)

Keymaps:

Greek for cp1253 (Panagiotis Louridas)
Hungarian (Magyar) (Laszlo Zavaleta)
Persian-Iranian (Behnam Esfahbod)

Message translations:

Catalan (Ernest Adroque)
Russian (Vassily Ragozin)
Swedish (Johan Svedberg)

Menu translations:

Catalan (Ernest Adroque)
Russian (Tim Alexeevsky)
Swedish (Johan Svedberg)

Tutor translations:

Catalan (Ernest Adroque)
Russian in cp1251 (Alexey Froloff)
Slovak in cp1250 and iso8859-2 (Lubos Celko)
Swedish (Johan Svedberg)
Korean (Kee-Won Seo)
UTF-8 version of the Japanese tutor (Yasuhiro Matsumoto) Use this as
the original, create the other Japanese tutor by conversion.

Included "russian.txt" help file. (Vassily Ragozin)

Include Encapsulated PostScript and PDF versions of the Vim logo in the extra archive.

The help highlighting finds the highlight groups and shows them in the color

that is actually being used. (idea from Yakov Lerner)

The big Win32 version is now compiled with Ruby interface, version 1.8. For Python version 2.3 is used. For Perl version 5.8 is used.

The "ftdetect" directory is mentioned in the documentation. The DOS install program creates it.

Fixed

fixed-6.3

Test 42 failed on MS-Windows. Set and reset **'fileformat'** and **'binary'** options here and there. (Walter Briscoe)

The explorer plugin didn't work for double-byte **'encoding'**s.

Use "copy /y" in Make_bc5.mak to avoid a prompt for overwriting.

Patch 6.2.001

Problem: The ":stopinsert" command doesn't have a help tag.

Solution: Add the tag. (Antoine J. Mechelynck)

Files: runtime/doc/insert.txt, runtime/doc/tags

Patch 6.2.002

Problem: When compiled with the +multi_byte feature but without +eval, displaying UTF-8 characters may cause a crash. (Karsten Hopp)

Solution: Also set the default for **'ambiwidth'** when compiled without the +eval feature.

Files: src/option.c

Patch 6.2.003

Problem: GTK 2: double-wide characters below 256 are not displayed correctly.

Solution: Check the cell width for characters above 127. (Yasuhiro Matsumoto)

Files: src/gui_gtk_x11.c

Patch 6.2.004

Problem: With a line-Visual selection at the end of the file a "p" command puts the text one line upwards.

Solution: Detect that the last line was deleted and put forward. (Taro Muraoka)

Files: src/normal.c

Patch 6.2.005

Problem: GTK: the "Find" and "Find and Replace" tools don't work. (Aschwin Marsman)

Solution: Show the dialog after creating it. (David Necas)

Files: src/gui_gtk.c

Patch 6.2.006

Problem: The Netbeans code contains an obsolete function that uses "vim61" and sets the fall-back value for \$VIMRUNTIME.

Solution: Delete the obsolete function.
Files: src/main.c, src/netbeans.c, src/proto/netbeans.pro

Patch 6.2.007

Problem: Listing tags for Cscope doesn't always work.
Solution: Avoid using smgs_attr(). (Sergey Khorev)
Files: src/if_cscope.c

Patch 6.2.008

Problem: XIM with GTK 2: After backspacing preedit characters are wrong.
Solution: Reset the cursor position. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 6.2.009

Problem: Win32: The self-installing executable "Full" selection only selects some of the items to install. (Salman Mohsin)
Solution: Change commas to spaces in between section numbers.
Files: nsis/gvim.nsi

Patch 6.2.010

Problem: When '**virtualedit**' is effective and a line starts with a multibyte character, moving the cursor right doesn't work.
Solution: Obtain the right character to compute the column offset. (Taro Muraoka)
Files: src/charset.c

Patch 6.2.011

Problem: Alpha OSF1: stat() is a macro and doesn't allow an #ifdef halfway. (Moshe Kaminsky)
Solution: Move the #ifdef outside of stat().
Files: src/os_unix.c

Patch 6.2.012

Problem: May hang when polling for a character.
Solution: Break the wait loop when not waiting for a character.
Files: src/os_unix.c

Patch 6.2.013 (extra)

Problem: Win32: The registry key for uninstalling GvimExt still uses "6.1".
Solution: Change the version number to "6.2". (Ajit Thakkar)
Files: src/GvimExt/GvimExt.reg

Patch 6.2.014 (after 6.2.012)

Problem: XSMP doesn't work when using poll().
Solution: Use xsmp_idx instead of gpm_idx. (Neil Bird)
Files: src/os_unix.c

Patch 6.2.015

Problem: The +xsmp feature is never enabled.
Solution: Move the #define for USE_XSMP to below where WANT_X11 is defined. (Alexey Froloff)
Files: src/feature.h

Patch 6.2.016

Problem: Using `":cscope find"` with `'cscopequickfix'` does not always split the window. (Gary Johnson)
Win32: `":cscope add"` could make the script that contains it read-only until the corresponding `":cscope kill"`.
Errors during `":cscope add"` may not be handled properly.

Solution: When using the quickfix window may need to split the window.
Avoid file handle inheritance for the script.
Check for a failed connection and/or process. (Sergey Khorev)

Files: `src/ex_cmds2.c`, `src/if_cscope.c`

Patch 6.2.017

Problem: `Test11` sometimes prompts the user, because a file would have been changed outside of Vim. (Antonio Colombo)

Solution: Add a `FileChangedShell` autocommand to avoid the prompt.

Files: `src/testdir/test11.in`

Patch 6.2.018

Problem: When using the XSMP protocol and reading from stdin Vim may wait for a key to be pressed.

Solution: Avoid that `RealWaitForChar()` is used recursively.

Files: `src/os_unix.c`

Patch 6.2.019 (lang)

Problem: Loading the Portuguese menu causes an error message.

Solution: Join two lines. (Jose Pedro Oliveira, José de Paula)

Files: `runtime/lang/menu_pt_br.vim`

Patch 6.2.020

Problem: The `"Syntax/Set syntax only"` menu item causes an error message. (Oyvind Holm)

Solution: Set the script-local variable in a function. (Benji Fisher)

Files: `runtime/synmenu.vim`

Patch 6.2.021

Problem: The user manual section on exceptions contains small mistakes.

Solution: Give a good example of an error that could be missed and other improvements. (Servatius Brandt)

Files: `runtime/doc/usr_41.txt`

Patch 6.2.022 (extra)

Problem: Win32: After deleting a menu item it still appears in a tear-off window.

Solution: Set the mode to zero for the deleted item. (Yasuhiro Matsumoto)

Files: `src/gui_w32.c`

Patch 6.2.023 (extra)

Problem: Win32: `Make_ivc.mak` does not clean everything.

Solution: Delete more files in the clean rule. (Walter Briscoe)

Files: `src/Make_ivc.mak`

Patch 6.2.024 (extra)

Problem: Win32: Compiler warnings for typecasts.

Solution: Use `DWORD` instead of `WORD`. (Walter Briscoe)

Files: `src/gui_w32.c`

Patch 6.2.025

Problem: Missing prototype for sigaltstack().
Solution: Add the prototype when it is not found in a header file.
Files: src/os_unix.c

Patch 6.2.026

Problem: Warning for utimes() argument.
Solution: Add a typecast.
Files: src/fileio.c

Patch 6.2.027

Problem: Warning for uninitialized variable.
Solution: Set mb_l to one when not using multibyte characters.
Files: src/message.c

Patch 6.2.028

Problem: Cscope connection may kill Vim process and others.
Solution: Check for pid being larger than one. (Khorev Sergey)
Files: src/if_cscope.c

Patch 6.2.029

Problem: When using the remote server functionality Vim may leak memory.
(Srikanth Sankaran)
Solution: Free the result of XListProperties().
Files: src/if_xcmdsrv.c

Patch 6.2.030

Problem: Mac: Warning for not being able to use precompiled header files.
Solution: Don't redefine select. Use -no-cpp-precomp for compiling, so that
function prototypes are still found.
Files: src/os_unix.c, src/osdef.sh

Patch 6.2.031

Problem: The langmenu entry in the options window doesn't work. (Rodolfo
Lima)
With GTK 1 the ":options" command causes an error message.
(Michael Naumann)
Solution: Change "lmenu" to "langmenu". Only display the 'tbis' option for
GTK 2.
Files: runtime/optwin.vim

Patch 6.2.032

Problem: The lpc filetype is never recognized. (Shizhu Pan)
Solution: Check for g:lpc_syntax_for_c instead of the local variable
lpc_syntax_for_c. (Benji Fisher)
Files: runtime/filetype.vim

Patch 6.2.033 (extra)

Problem: Mac: Various compiler warnings.
Solution: Don't include Classic-only headers in Unix version.
Remove references to several unused variables. (Ben Fowler)
Fix double definition of DEFAULT_TERM.
Use int instead of unsigned short for pixel values, so that the

negative error values are recognized.
Files: src/gui_mac.c, src/term.c

Patch 6.2.034

Problem: Mac: Compiler warning for redefining DEFAULT_TERM.
Solution: Fix double definition of DEFAULT_TERM.
Files: src/term.c

Patch 6.2.035

Problem: Mac: Compiler warnings in Python interface.
Solution: Make a difference between pure Mac and Unix-Mac. (Peter Cucka)
Files: src/if_python.c

Patch 6.2.036 (extra)

Problem: Mac Unix version: If foo is a directory, then ":e f<Tab>" should expand to ":e foo/" instead of ":e foo" . (Vadim Zeitlin)
Solution: Define DONT_ADD_PATHSEP_TO_DIR only for pure Mac. (Benji Fisher)
Files: src/os_mac.h

Patch 6.2.037

Problem: Win32: converting an encoding name to a codepage could result in an arbitrary number.
Solution: make encname2codepage() return zero if the encoding name doesn't contain a codepage number.
Files: src/mbyte.c

Patch 6.2.038 (extra)

Problem: Warning messages when using the MingW compiler. (Bill McCarthy)
Can't compile console version without +mouse feature.
Solution: Initialize variables, add parenthesis.
Add an #ifdef around g_nMouseClicked. (Ajit Thakkar)
Files: src/eval.c, src/os_win32.c, src/gui_w32.c, src/dosinst.c

Patch 6.2.039 (extra)

Problem: More warning messages when using the MingW compiler.
Solution: Initialize variables. (Bill McCarthy)
Files: src/os_mswin.c

Patch 6.2.040

Problem: FreeBSD: Crash while starting up when compiled with +xsmp feature.
Solution: Pass a non-NULL argument to IceAddConnectionWatch().
Files: src/os_unix.c

Patch 6.2.041 (extra, after 6.2.033)

Problem: Mac: Compiler warnings for conversion types, missing prototype, missing return type.
Solution: Change sscanf "%hd" to "%d", the argument is an int now. Add gui_mch_init_check() prototype. Add "int" to termlib functions.
Files: src/gui_mac.c, src/proto/gui_mac.pro, src/termlib.c.

Patch 6.2.042 (extra)

Problem: Cygwin: gcc 3.2 has an optimizer problem, sometimes causing a crash.
Solution: Add -fno-strength-reduce to the compiler arguments. (Dan Sharp)

Files: src/Make_cyg.mak

Patch 6.2.043

Problem: Compiling with both netbeans and workshop doesn't work.
Solution: Move the shellRectangle() function to gui_x11.c. (Gordon Prieur)
Files: src/gui_x11.c, src/integration.c, src/netbeans.c,
src/proto/netbeans.pro

Patch 6.2.044

Problem: ":au filetypepedetect" gives an error for a non-existing event name,
but it's actually a non-existing group name. (Antoine Mechelynck)
Solution: Make the error message clearer.
Files: src/fileio.c

Patch 6.2.045

Problem: Obtaining the '(mark changes the ' mark. (Gary Holloway)
Solution: Don't set the ' mark when searching for the start/end of the
current sentence/paragraph.
Files: src/mark.c

Patch 6.2.046

Problem: When evaluating an argument of a function throws an exception the
function is still called. (Hari Krishna Dara)
Solution: Don't call the function when an exception was thrown.
Files: src/eval.c

Patch 6.2.047 (extra)

Problem: Compiler warnings when using MingW. (Bill McCarthy)
Solution: Give the s_dwLastClickTime variable a type. Initialize dwEndTime.
Files: src/os_win32.c

Patch 6.2.048

Problem: The Python interface doesn't compile with Python 2.3 when
dynamically loaded.
Solution: Use dll_PyObject_Malloc and dll_PyObject_Free. (Paul Moore)
Files: src/if_python.c

Patch 6.2.049

Problem: Using a "--range=" argument with ":command" doesn't work and
doesn't generate an error message.
Solution: Generate an error message.
Files: src/ex_docmd.c

Patch 6.2.050

Problem: Test 32 didn't work on MS-Windows.
Solution: Write the temp file in Unix fileformat. (Walter Briscoe)
Files: src/testdir/test32.in

Patch 6.2.051

Problem: When using "\=submatch(0)" in a ":s" command, line breaks become
NUL characters.
Solution: Change NL to CR characters, so that they become line breaks.
Files: src/regexp.c

Patch 6.2.052

Problem: A few messages are not translated.
Solution: Add _() to the messages. (Muraoka Taro)
Files: src/ex_cmds.c

Patch 6.2.053

Problem: Prototype for bzero() doesn't match most systems.
Solution: Use "void *" instead of "char *" and "size_t" instead of "int".
Files: src/osdef1.h.in

Patch 6.2.054

Problem: A double-byte character with a second byte that is a backslash causes problems inside a string.
Solution: Skip over multibyte characters in a string properly. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 6.2.055

Problem: Using col('.') from **CTRL-O** in Insert mode does not return the correct value for multibyte characters.
Solution: Correct the cursor position when it is necessary, move to the first byte of a multibyte character. (Yasuhiro Matsumoto)
Files: src/edit.c

Patch 6.2.056 (extra)

Problem: Building with Sniff++ doesn't work.
Solution: Use the multi-threaded libc when needed. (Holger Ditting)
Files: src/Make_mvc.mak

Patch 6.2.057 (extra)

Problem: Mac: With -DMACOS_X putenv() is defined twice, it is in a system library. Get a warning for redefining OK. Unused variables in os_mac.c
Solution: Define HAVE_PUTENV. Undefine OK after including curses.h. Remove declarations for unused variables.
Files: src/os_mac.c, src/os_mac.h, src/vim.h

Patch 6.2.058

Problem: When **'autochdir'** is set ":bnext" to a buffer without a name causes a crash.
Solution: Don't call vim_chdirfile() when the file name is NULL. (Taro Muraoka)
Files: src/buffer.c

Patch 6.2.059

Problem: When **'scrolloff'** is a large number and listing completion results on the command line, then executing a command that jumps close to where the cursor was before, part of the screen is not updated. (Yakov Lerner)
Solution: Don't skip redrawing part of the window when it was scrolled.
Files: src/screen.c

Patch 6.2.060 (extra)

Problem: Win32: When **'encoding'** is set to "iso-8859-7" copy/paste to/from

the clipboard gives a lalloc(0) error. (Kriton Kyrimis)
Solution: When the string length is zero allocate one byte. Also fix that when the length of the Unicode text is zero (conversion from **'encoding'** to UCS-2 was not possible) the normal text is used.
Files: src/os_mswin.c

Patch 6.2.061

Problem: GUI: Using the left mouse button with the shift key should work like "*" but it scrolls instead. (Martin Beller)
Solution: Don't recognize an rxvt scroll wheel event when using the GUI.
Files: src/term.c

Patch 6.2.062

Problem: When one buffer uses a syntax with "containedin" and another buffer does not, redrawing depends on what the current buffer is. (Brett Pershing Stahlman)
Solution: Use "syn_buf" instead of "curbuf" to get the b_syn_containedin flag.
Files: src/syntax.c

Patch 6.2.063

Problem: When using custom completion end up with no matches.
Solution: Make cmd_numfiles and cmd_files local to completion to avoid that they are overwritten when ExpandOne() is called recursively by f_glob().
Files: src/eval.c, src/ex_docmd.c, src/ex_getln.c, src/proto/ex_getln.pro, src/misc1.c, src/structs.h, src/tag.c

Patch 6.2.064

Problem: resolve() only handles one symbolic link, need to repeat it to resolve all of them. Then need to simplify the file name.
Solution: Make resolve() resolve all symbolic links and simplify the result. Add simplify() to just simplify a file name. Fix that test49 doesn't work if /tmp is a symbolic link. (Servatius Brandt)
Files: runtime/doc/eval.txt, src/eval.c, src/tag.c, src/testdir/test49.vim

Patch 6.2.065

Problem: ":windo 123" only updates other windows when entering them. (Walter Briscoe)
Solution: Update the topline before going to the next window.
Files: src/ex_cmds2.c

Patch 6.2.066 (extra)

Problem: Ruby interface doesn't work with Ruby 1.8.0.
Solution: Change "defout" to "stdout". (Aron Griffis)
Change dynamic loading. (Taro Muraoka)
Files: src/if_ruby.c, src/Make_mvc.mak

Patch 6.2.067

Problem: When searching for a string that starts with a composing character the command line isn't drawn properly.
Solution: Don't count the space to draw the composing character on and adjust the cursor column after drawing the string.

Files: src/message.c

Patch 6.2.068

Problem: Events for the netbeans interface that include a file name with special characters don't work properly.

Solution: Use nb_quote() on the file name. (Sergey Khorev)

Files: src/netbeans.c

Patch 6.2.069 (after 6.2.064)

Problem: Unused variables "limit" and "new_st" and unused label "fail" in some situation. (Bill McCarthy)

Solution: Put the declarations inside an #ifdef. (Servatius Brandt)

Files: src/eval.c, src/tag.c

Patch 6.2.070 (after 6.2.069)

Problem: Still unused variable "new_st". (Bill McCarthy)

Solution: Move the declaration to the right block this time.

Files: src/tag.c

Patch 6.2.071

Problem: **'statusline'** can only contain 50 % items. (Antony Scriven)

Solution: Allow 80 items and mention it in the docs.

Files: runtime/doc/option.txt, src/vim.h

Patch 6.2.072

Problem: When using expression folding, foldexpr() mostly returns -1 for the previous line, which makes it difficult to write a fold expression.

Solution: Make the level of the previous line available while still looking for the end of a fold.

Files: src/fold.c

Patch 6.2.073

Problem: When adding detection of a specific filetype for a plugin you need to edit "filetype.vim".

Solution: Source files from the "ftdetect" directory, so that a filetype detection plugin only needs to be dropped in a directory.

Files: runtime/doc/filetype.txt, runtime/doc/usr_05.txt, runtime/doc/usr_41.txt, runtime/filetype.vim

Patch 6.2.074

Problem: Warnings when compiling the Python interface. (Ajit Thakkar)

Solution: Use ANSI function declarations.

Files: src/if_python.c

Patch 6.2.075

Problem: When the temp file for writing viminfo can't be used "NULL" appears in the error message. (Ben Lavender)

Solution: Print the original file name when there is no temp file name.

Files: src/ex_cmds.c

Patch 6.2.076

Problem: The tags listed for cscope are in the wrong order. (Johannes Stezenbach)

Solution: Remove the reordering of tags for the current file. (Sergey Khorev)
Files: src/if_cscope.c

Patch 6.2.077

Problem: When a user function specifies custom completion, the function gets a zero argument instead of an empty string when there is no word before the cursor. (Preben Guldberg)
Solution: Don't convert an empty string to a zero.
Files: src/eval.c

Patch 6.2.078

Problem: "make test" doesn't work if Vim wasn't compiled yet. (Ed Avis)
Solution: Build Vim before running the tests.
Files: src/Makefile

Patch 6.2.079

Problem: ":w ++enc=utf-8 !cmd" doesn't work.
Solution: Check for the "++" argument before the "!".
Files: src/ex_docmd.c

Patch 6.2.080

Problem: When '**t_ti**' is not empty but doesn't swap screens, using "ZZ" in an unmodified file doesn't clear the last line.
Solution: Call msg_clr_eos() when needed. (Michael Schroeder)
Files: src/os_unix.c

Patch 6.2.081

Problem: Problem when using a long multibyte string for the statusline.
Solution: Use the right pointer to get the cell size. (Taro Muraoka)
Files: src/buffer.c

Patch 6.2.082

Problem: Can't compile with Perl 5.8.1.
Solution: Rename "e_number" to "e_number_exp". (Sascha Blank)
Files: src/digraph.c, src/globals.h

Patch 6.2.083

Problem: When a compiler uses ^^^^ to mark a word the information is not visible in the quickfix window. (Srikanth Sankaran)
Solution: Don't remove the indent for a line that is not recognized as an error message.
Files: src/quickfix.c

Patch 6.2.084

Problem: "g_" in Visual mode always goes to the character after the line. (Jean-Rene David)
Solution: Ignore the NUL at the end of the line.
Files: src/normal.c

Patch 6.2.085

Problem: ":verbose set ts" doesn't say an option was set with a "-c" or "--cmd" argument.
Solution: Remember the option was set from a Vim argument.

Files: src/main.c, src/ex_cmds2.c, src/vim.h

Patch 6.2.086

Problem: "{" and }" stop inside a closed fold.

Solution: Only stop once inside a closed fold. (Stephen Riehm)

Files: src/search.c

Patch 6.2.087

Problem: CTRL-^ doesn't use the 'confirm' option. Same problem with ":bnext". (Yakov Lerner)

Solution: Put up a dialog for a changed file when 'confirm' is set in more situations.

Files: src/buffer.c, src/ex_cmds.c

Patch 6.2.088

Problem: When 'sidescrolloff' is set 'showmatch' doesn't work correctly if the match is less than 'sidescrolloff' off from the side of the window. (Roland Stahn)

Solution: Set 'sidescrolloff' to zero while displaying the match.

Files: src/search.c

Patch 6.2.089

Problem: ":set isk+=" adds a comma. (Mark Waggoner)

Solution: Don't add a comma when the added value is empty.

Files: src/option.c

Patch 6.2.090 (extra)

Problem: Win32: MingW compiler complains about #pragmas. (Bill McCarthy)

Solution: Put an #ifdef around the #pragmas.

Files: src/os_win32.c

Patch 6.2.091

Problem: When an autocommand is triggered when a file is dropped on Vim and it produces output, messages from a following command may be scrolled unexpectedly. (David Rennalls)

Solution: Save and restore msg_scroll in handle_drop().

Files: src/ex_docmd.c

Patch 6.2.092

Problem: Invalid items appear in the help file tags. (Antonio Colombo)

Solution: Only accept tags with white space before the first "*".

Files: runtime/doc/doctags.c, src/ex_cmds.c

Patch 6.2.093

Problem: ":nnoremenu" also defines menu for Visual mode. (Klaus Bosau)

Solution: Check the second command character for an "o", not the third.

Files: src/menu.c

Patch 6.2.094

Problem: Can't compile with GTK and tiny features.

Solution: Include handle_drop() and vim_chdirfile() when FEAT_DND is defined. Do not try to split the window.

Files: src/ex_docmd.c, src/misc2.c

Patch 6.2.095

Problem: The message "Cannot go to buffer x" is confusing for ":buf 6".
(Frans English)
Solution: Make it "Buffer x does not exist".
Files: src/buffer.c

Patch 6.2.096

Problem: Win32: ":let @* = '" put a newline on the clipboard. (Klaus Bosau)
Solution: Put zero bytes on the clipboard for an empty string.
Files: src/ops.c

Patch 6.2.097

Problem: Setting or resetting '**insertmode**' in a BufEnter autocommand doesn't always have immediate effect. (Nagger)
Solution: When '**insertmode**' is set, set need_start_insertmode, when it's reset set stop_insert_mode.
Files: src/option.c

Patch 6.2.098 (after 6.2.097)

Problem: Can't build Vim with tiny features. (Christian J. Robinson)
Solution: Declare stop_insert_mode always.
Files: src/edit.c, src/globals.h

Patch 6.2.099 (extra)

Problem: Test 49 fails. (Mikolaj Machowski)
Solution: The Polish translation must not change "E116" to "R116".
Files: src/po/pl.po

Patch 6.2.100

Problem: "make proto" fails when compiled with the Perl interface.
Solution: Remove "-fno.*" from PERL_CFLAGS, cproto sees it as its option.
Files: src/auto/configure, src/configure.in

Patch 6.2.101

Problem: When using syntax folding, opening a file slows down a lot when it's size increases by only 20%. (Gary Johnson)
Solution: The array with cached syntax states is leaking entries. After cleaning up the list obtain the current entry again.
Files: src/syntax.c

Patch 6.2.102

Problem: The macros equal() and CR conflict with a Carbon header file.
Solution: Rename equal() to equalpos(). Rename CR to CAR.
Do this in the non-extra files only.
Files: src/ascii.h, src/buffer.c, src/charset.c, src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/gui.c, src/gui_athena.c, src/gui_gtk_x11.c, src/gui_motif.c, src/macros.h, src/mark.c, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/ops.c, src/os_unix.c, src/regexp.c, src/search.c, src/ui.c, src/workshop.c

Patch 6.2.103 (extra)

Problem: The macros equal() and CR conflict with a Carbon header file.

Solution: Rename equal() to equalpos(). Rename CR to CAR.
Do this in the extra files only.
Files: src/gui_photon.c, src/gui_w48.c

Patch 6.2.104

Problem: Unmatched braces in the table with options.
Solution: Move the "}," outside of the #ifdef. (Yakov Lerner)
Files: src/option.c

Patch 6.2.105

Problem: When the cursor is past the end of the line when calling
get_c_indent() a crash might occur.
Solution: Don't look past the end of the line. (NJ Verenini)
Files: src/misc1.c

Patch 6.2.106

Problem: Tag searching gets stuck on a very long line in the tags file.
Solution: When skipping back to search the first matching tag remember the
offset where searching started looking for a line break.
Files: src/tag.c

Patch 6.2.107 (extra)

Problem: The NetBeans interface cannot be used on Win32.
Solution: Add support for the NetBeans for Win32. Add support for reading
XPM files on Win32. Also fixes that a sign icon with a space in
the file name did not work through the NetBeans interface.
(Sergey Khorev)
Also: avoid repeating error messages when the connection is lost.
Files: Makefile, runtime/doc/netbeans.txt, src/Make_bc5.mak,
src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak,
src/bigvim.bat, src/feature.h, src/gui_beval.c, src/gui_beval.h,
src/gui_w32.c, src/gui_w48.c, src/menu.c, src/nbdebug.c,
src/nbdebug.h, src/netbeans.c, src/os_mswin.c, src/os_win32.h,
src/proto/gui_beval.pro, src/proto/gui_w32.pro,
src/proto/netbeans.pro, src/proto.h, src/version.c, src/vim.h,
src/xpm_w32.c, src/xpm_w32.h

Patch 6.2.108

Problem: Crash when giving a message about ignoring case in a tag. (Manfred
Kuehn)
Solution: Use a longer buffer for the message.
Files: src/tag.c

Patch 6.2.109

Problem: Compiler warnings with various Amiga compilers.
Solution: Add typedef, prototypes, et al. that are also useful for other
systems. (Flavio Stanchina)
Files: src/eval.c, src/ops.c

Patch 6.2.110

Problem: When \$LANG includes the encoding, a menu without an encoding name
is not found.
Solution: Also look for a menu file without any encoding.
Files: runtime/menu.vim

Patch 6.2.111

Problem: Encoding "cp1251" is not recognized.
Solution: Add "cp1251" to the table of encodings. (Alexey Froloff)
Files: src/mbyte.c

Patch 6.2.112

Problem: After applying patches test32 fails. (Antonio Colombo)
Solution: Have "make clean" in the testdir delete *.rej and *.orig files.
Use this when doing "make clean" in the src directory.
Files: src/Makefile, src/testdir/Makefile

Patch 6.2.113

Problem: Using ":startinsert" after "\$" works like "a" instead of "i".
(Ajit Thakkar)
Solution: Reset "w_curswant" for ":startinsert" and reset o_eol in edit().
Files: src/edit.c, src/ex_docmd.c

Patch 6.2.114

Problem: When stdout is piped through "tee", the size of the screen may not
be correct.
Solution: Use stdin instead of stdout for ioctl() when stdin is a tty and
stdout isn't.
Files: src/os_unix.c

Patch 6.2.115 (extra)

Problem: Compiler warnings with various Amiga compilers.
Solution: Add typecast, prototypes, et al. Those changes that are
Amiga-specific. (Flavio Stanchina)
Files: src/fileio.c, src/memfile.c, src/os_amiga.c, src/os_amiga.h,
src/vim.h

Patch 6.2.116 (extra)

Problem: German keyboard with Numlock set different from system startup
causes problems.
Solution: Ignore keys with code 0xff. (Helmut Stiegler)
Files: src/gui_w48.c

Patch 6.2.117

Problem: Breakpoints in loops of sourced files and functions are not
detected. (Hari Krishna Dara)
Solution: Check for breakpoints when using lines that were previously read.
(Servatius Brandt)
Files: src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/proto/eval.pro,
src/proto/ex_cmds2.pro

Patch 6.2.118 (extra)

Problem: Mac: Compiling is done in a non-standard way.
Solution: Use the Unix method for Mac OS X, with autoconf. Add "CARBONGUI"
to Makefile and configure. (Eric Kow)
Move a few prototypes from os_mac.pro to gui_mac.pro.
Files: src/Makefile, src/auto/configure, src/configure.in,
src/config.mk.in, src/gui_mac.c, src/os_mac.h, src/os_macosx.c,
src/proto/gui_mac.pro, src/proto/os_mac.pro,

src/infplist.xml, src/vim.h

Patch 6.2.119 (after 6.2.107)

Problem: When packing the MS-Windows archives a few files are missing.
(Guopeng Wen)

Solution: Add gui_beval.* to the list of generic source files.

Files: Makefile

Patch 6.2.120

Problem: Win32 GUI: The console dialogs are not supported on MS-Windows,
disabling the 'c' flag of **'guioptions'**. (Servatius Brandt)

Solution: Define FEAT_CON_DIALOG also for GUI-only builds.

Files: src/feature.h

Patch 6.2.121 (after 6.2.118)

Problem: Not all make programs support "+=". (Charles Campbell)

Solution: Use a normal assignment.

Files: src/Makefile

Patch 6.2.122 (after 6.2.119)

Problem: Not all shells can expand [^~]. File missing. (Guopeng Wen)

Solution: Use a simpler pattern. Add the Aap recipe for the maze program
and a clean version of the source code.

Files: Makefile, runtime/macros/maze/Makefile,
runtime/macros/maze/README.txt, runtime/macros/maze/main.aap,
runtime/macros/maze/mazeclean.c

Patch 6.2.123 (after 6.2.118)

Problem: Running configure fails. (Tony Leneis)

Solution: Change "==" to "=" for a test.

Files: src/auto/configure, src/configure.in

Patch 6.2.124 (after 6.2.121)(extra)

Problem: Mac: Recursive use of M4FLAGS causes problems. When running Vim
directly it can't find the runtime files. (Emily Jackson)
Using GNU constructs causes warnings with other make programs.
(Ronald Schild)

Solution: Use another name for the M4FLAGS variable.

Don't remove "Vim.app" from the path.

Update the explanation for compiling on the Mac. (Eric Kow)

Don't use \$(shell) and \$(addprefix).

Files: src/INSTALLmac.txt, src/Makefile, src/misc1.c

Patch 6.2.125 (after 6.2.107)

Problem: The "winsock2.h" file isn't always available.

Solution: Don't include this header file.

Files: src/netbeans.c

Patch 6.2.126

Problem: Typing **CTRL-C** at a confirm() prompt doesn't throw an exception.

Solution: Reset "mapped_ctrl_c" in get_keystroke(), so that "got_int" is set
in _OnChar().

Files: src/misc1.c

Patch 6.2.127 (extra)

Problem: Win32 console: Typing **CTRL-C** doesn't throw an exception.
Solution: Set got_int immediately when **CTRL-C** is typed, don't wait for mch_breakcheck() being called.
Files: src/os_win32.c

Patch 6.2.128 (after 6.2.118)

Problem: src/auto/configure is not consistent with src/configure.in.
Solution: Use the newly generated configure script.
Files: src/auto/configure

Patch 6.2.129

Problem: When **'number'** is set **'wrapmargin'** does not work Vi-compatible. (Yasuhiro Matsumoto)
Solution: Reduce the textwidth when **'number'** is set. Also for **'foldcolumn'** and similar things.
Files: src/edit.c

Patch 6.2.130 (extra)

Problem: Win32 console: When **'restorescreen'** is not set exiting Vim causes the screen to be cleared. (Michael A. Mangino)
Solution: Don't clear the screen when exiting and **'restorescreen'** isn't set.
Files: src/os_win32.c

Patch 6.2.131 (extra)

Problem: Win32: Font handles are leaked.
Solution: Free italic, bold and bold-italic handles before overwriting them. (Michael Wookey)
Files: src/gui_w48.c

Patch 6.2.132 (extra)

Problem: Win32: console version doesn't work on latest Windows Server 2003.
Solution: Copy 12000 instead of 15000 cells at a time to avoid running out of memory.
Files: src/os_win32.c

Patch 6.2.133

Problem: When starting the GUI a bogus error message about **'imactivatekey'** may be given.
Solution: Only check the value of **'imactivatekey'** when the GUI is running.
Files: src/gui.c, src/option.c

Patch 6.2.134 (extra)

Problem: Win32: When scrolling parts of the window are redrawn when this isn't necessary.
Solution: Only invalidate parts of the window when they are obscured by other windows. (Michael Wookey)
Files: src/gui_w48.c

Patch 6.2.135

Problem: An item **<>** in the ":command" argument is interpreted as **<args>**.
Solution: Avoid that **<>** is recognized as **<args>**.
Files: src/ex_docmd.c

Patch 6.2.136

Problem: ":e ++enc=latin1 newfile" doesn't set **'fenc'** when the file doesn't exist. (Miroslaw Dobrzanski-Neumann)
Solution: Set **'fileencoding'** to the specified encoding when editing a file that does not exist.
Files: src/fileio.c

Patch 6.2.137

Problem: "d:cmd<CR>" cannot be repeated with ".". Breaks repeating "d%" when using the matchit plugin.
Solution: Store the command to be repeated. This is restricted to single-line commands.
Files: src/ex_docmd.c, src/globals.h, src/normal.c, src/vim.h

Patch 6.2.138 (extra)

Problem: Compilation problem on VMS with dynamic buffer on the stack.
Solution: Read one byte less than the size of the buffer, so that we can check for the string length without an extra buffer.
Files: src/os_vms.c

Patch 6.2.139

Problem: Code is repeated in the two Perl files.
Solution: Move common code from if_perl.xs and if_perlsfio.c to vim.h. Also fix a problem with generating prototypes.
Files: src/if_perl.xs, src/if_perlsfio.c, src/vim.h

Patch 6.2.140 (after 6.2.121)

Problem: Mac: Compiling with Python and Perl doesn't work.
Solution: Adjust the configure check for Python to use "--framework Python" for Python 2.3 on Mac OS/X.
Move "-ldl" after "DynaLoader.a" in the link command.
Change "perllibs" to "PERL_LIBS".
Files: src/auto/configure, src/configure.in, src/config.mk.in

Patch 6.2.141 (extra)

Problem: Mac: The b_FSSpec field is sometimes unused.
Solution: Change the #ifdef to FEAT_CW_EDITOR and defined it in feature.h
Files: src/fileio.c, src/gui_mac.c, src/structs.h, src/feature.h

Patch 6.2.142 (after 6.2.124)

Problem: Mac: building without GUI through configure doesn't work. When the system is slow, unpacking the resource file takes too long.
Solution: Don't always define FEAT_GUI_MAC when MACOS is defined, define it in the Makefile.
Add a configure option to skip Darwin detection.
Use a Python script to unpack the resources to avoid a race condition. (Taro Muraoka)
Files: Makefile, src/Makefile, src/auto/configure, src/configure.in, src/dehqx.py, src/vim.h

Patch 6.2.143

Problem: Using "K" on Visually selected text doesn't work if it ends in a multibyte character.

Solution: Include all the bytes of the last character. (Taro Muraoka)
Files: src/normal.c

Patch 6.2.144

Problem: When "g:html_use_css" is set the HTML header generated by the 2html script is wrong.

Solution: Add the header after adding HREF for links.
Also use ":normal!" instead of ":normal" to avoid mappings getting in the way.

Files: runtime/syntax/2html.vim

Patch 6.2.145 (after 6.2.139)

Problem: Undefined "bool" doesn't work for older systems. (Wojtek Pilorz)

Solution: Only undefine "bool" on Mac OS.

Files: src/vim.h

Patch 6.2.146

Problem: On some systems the prototype for iconv() is wrong, causing a warning message.

Solution: Use a cast (void *) to avoid the warning. (Charles Campbell)

Files: src/fileio.c, src/mbyte.c

Patch 6.2.147

Problem: ":s/pat/\=col('.')" always replaces with "1".

Solution: Set the cursor to the start of the match before substituting.
(Helmut Stiegler)

Files: src/ex_cmds.c

Patch 6.2.148

Problem: Can't break an Insert into several undoable parts.

Solution: Add the **CTRL-G** u command.

Files: runtime/doc/insert.txt, src/edit.c

Patch 6.2.149

Problem: When the cursor is on a line past 21,474,748 the indicated percentage of the position is invalid. With that many lines "100%" causes a negative cursor line number, resulting in a crash.
(Daniel Goujot)

Solution: Divide by 100 instead of multiplying. Avoid overflow when computing the line number for "100%".

Files: src/buffer.c, src/ex_cmds2.c, src/normal.c

Patch 6.2.150

Problem: When doing "vim - < file" lines are broken at NUL chars.
(Daniel Goujot)

Solution: Change NL characters back to NUL when reading from the temp buffer.

Files: src/fileio.c

Patch 6.2.151

Problem: When doing "vim --remote +startinsert file" some commands are inserted as text. (Klaus Bosau)

Solution: Put all the init commands in one Ex line, not using a <CR>, so that Insert mode isn't started too early.

Files: src/main.c

Patch 6.2.152

Problem: The cursor() function doesn't reset the column offset for **'virtualedit'**.

Solution: Reset the offset to zero. (Helmut Stiegler)

Files: src/eval.c

Patch 6.2.153

Problem: Win32: ":lang german" doesn't use German messages.

Solution: Add a table to translate the Win32 language names to two-letter language codes.

Files: src/ex_cmds2.c

Patch 6.2.154

Problem: Python bails out when giving a warning message. (Eugene Minkovskii)

Solution: Set sys.argv[] to an empty string.

Files: src/if_python.c

Patch 6.2.155

Problem: Win32: Using ":tjump www" in a help file gives two results. (Dave Roberts)

Solution: Ignore differences between slashes and backslashes when checking for identical tag matches.

Files: src/tag.c

Patch 6.2.156 (after 6.2.125)

Problem: Win32: Netbeans fails to build, EINTR is not defined.

Solution: Redefine EINTR to WSAEINTR. (Mike Williams)

Files: src/netbeans.c

Patch 6.2.157

Problem: Using "%p" in **'errorformat'** gives a column number that is too high.

Solution: Set the flag to use the number as a virtual column. (Lefteris Koutsoloukas)

Files: src/quickfix.c

Patch 6.2.158

Problem: The sed command on Solaris and HP-UX doesn't work for a line that doesn't end in a newline.

Solution: Add a newline when feeding text to sed. (Mark Waggoner)

Files: src/configure.in, src/auto/configure

Patch 6.2.159

Problem: When using expression folding and **'foldopen'** is "undo" an undo command doesn't always open the fold.

Solution: Save and restore the KeyTyped variable when evaluating **'foldexpr'**. (Taro Muraoka)

Files: src/fold.c

Patch 6.2.160

Problem: When **'virtualedit'** is "all" and **'selection'** is "exclusive",

selecting a double-width character below a single-width character may cause a crash.

Solution: Avoid overflow on unsigned integer decrement. (Taro Muraoka)
Files: src/normal.c

Patch 6.2.161 (extra)

Problem: VMS: Missing header file. Reading input busy loops.
Solution: Include termdef.h. Avoid the use of a wait function in vms_read(). (Frank Ries)
Files: src/os_unix.h, src/os_vms.c

Patch 6.2.162

Problem: ":redraw" doesn't always display the text that includes the cursor position, e.g. after ":call cursor(1, 0)". (Eugene Minkovskii)
Solution: Call update_topline() before redrawing.
Files: src/ex_docmd.c

Patch 6.2.163

Problem: "make install" may also copy AAPDIR directories.
Solution: Delete AAPDIR directories, just like CVS directories.
Files: src/Makefile

Patch 6.2.164 (after 6.2.144)

Problem: When "g:html_use_css" is set the HTML header generated by the 2html script is still wrong.
Solution: Search for a string instead of jumping to a fixed line number. Go to the start of the line before inserting the header. (Jess Thrysoee)
Files: runtime/syntax/2html.vim

Patch 6.2.165

Problem: The configure checks hang when using autoconf 2.57.
Solution: Invoke AC_PROGRAM_EGREP to set \$EGREP. (Aron Griffis)
Files: src/auto/configure, src/configure.in

Patch 6.2.166

Problem: When \$GZIP contains "-N" editing compressed files doesn't work properly.
Solution: Add "-n" to "gzip -d" to avoid restoring the file name. (Oyvind Holm)
Files: runtime/plugin/gzip.vim

Patch 6.2.167

Problem: The Python interface leaks memory when assigning lines to a buffer. (Sergey Khorev)
Solution: Do not copy the line when calling ml_replace().
Files: src/if_python.c

Patch 6.2.168

Problem: Python interface: There is no way to get the indices from a range object.
Solution: Add the "start" and "end" attributes. (Maurice S. Barnum)
Files: src/if_python.c, runtime/doc/if_pyth.txt

Patch 6.2.169

Problem: The prototype for `_Xmblen()` appears in a recent XFree86 header file, causing a warning for our prototype. (Hisashi T Fujinaka)
Solution: Move the prototype to an `osdef` file, so that it's filtered out.
Files: `src/mbyte.c`, `src/osdef2.h.in`

Patch 6.2.170

Problem: When using Sun WorkShop the current directory isn't changed to where the file is.
Solution: Set the `'autochdir'` option when using WorkShop. And avoid using the basename when `'autochdir'` is not set.
Files: `src/gui_x11.c`, `src/ex_cmds.c`

Patch 6.2.171 (after 6.2.163)

Problem: The `"-or"` argument of `"find"` doesn't work for SysV systems.
Solution: Use `"-o"` instead. (Gordon Prieur)
Files: `src/Makefile`

Patch 6.2.172 (after 6.2.169)

Problem: The prototype for `_Xmblen()` still causes trouble.
Solution: Include the X11 header file that defines the prototype.
Files: `src/osdef2.h.in`, `src/osdef.sh`

Patch 6.2.173 (extra)

Problem: Win32: Ruby interface doesn't work with Ruby 1.8.0 for other compilers than MSVC.
Solution: Fix the BC5, Cygwin and Mingw makefiles. (Dan Sharp)
Files: `src/Make_bc5.mak`, `src/Make_cyg.mak`, `src/Make_ming.mak`

Patch 6.2.174

Problem: After the `":intro"` message only a mouse click in the last line gets past the hit-return prompt.
Solution: Accept a click at or below the hit-return prompt.
Files: `src/gui.c`, `src/message.c`

Patch 6.2.175

Problem: Changing `'backupext'` in a `*WritePre` autocommand doesn't work. (William Natter)
Solution: Move the use of `p_bex` to after executing the `*WritePre` autocommands. Also avoids reading allocated memory after freeing.
Files: `src/fileio.c`

Patch 6.2.176

Problem: Accented characters in translated help files are not handled correctly. (Fabien Vayssiere)
Solution: Include `"192-255"` in `'iskeyword'` for the help window.
Files: `src/ex_cmds.c`

Patch 6.2.177 (extra)

Problem: VisVim: Opening a file with a space in the name doesn't work. (Rob Retter) Arbitrary commands are being executed. (Neil Bird)
Solution: Put a backslash in front of every space in the file name. (Gerard Blais) Terminate the `CTRL-\ CTRL-N` command with a NUL.
Files: `src/VisVim/Commands.cpp`, `src/VisVim/VisVim.rc`

Patch 6.2.178

Problem: People who don't know how to exit Vim try pressing **CTRL-C**.
Solution: Give a message how to exit Vim when **CTRL-C** is pressed and it doesn't cancel anything.
Files: src/normal.c

Patch 6.2.179 (extra)

Problem: The en_gb messages file isn't found on case sensitive systems.
Solution: Rename en_gb to en_GB. (Mike Williams)
Files: src/po/en_gb.po, src/po/en_GB.po, src/po/Make_ming.mak, src/po/Make_mvc.mak, src/po/Makefile, src/po/README_mvc.txt

Patch 6.2.180

Problem: Compiling with GTK2 on Win32 doesn't work.
Solution: Include gdkwin32.h instead of gdkx.h. (Srinath Avadhanula)
Files: src/gui_gtk.c, src/gui_gtk_f.c, src/gui_gtk_x11.c, src/mbyte.c

Patch 6.2.181 (after 6.2.171)

Problem: The "-o" argument of "find" has lower priority than the implied "and" with "-print".
Solution: Add parenthesis around the "-o" expression. (Gordon Prieur)
Files: src/Makefile

Patch 6.2.182 (after 6.2.094)

Problem: Compilation with tiny features fails because of missing get_past_head() function.
Solution: Adjust the #ifdef for get_past_head().
Files: src/misc1.c

Patch 6.2.183 (after 6.2.178)

Problem: Warning for char/unsigned char mixup.
Solution: Use MSG() instead of msg(). (Tony Leneis)
Files: src/normal.c

Patch 6.2.184

Problem: With '**formatoptions**' set to "l~~a~~w" inserting text may cause the paragraph to be ended. (Alan Schmitt)
Solution: Temporarily add an extra space to make the paragraph continue after moving the word after the cursor to the next line. Also format when pressing Esc.
Files: src/edit.c, src/normal.c, src/proto/edit.pro

Patch 6.2.185

Problem: Restoring a session with zero-height windows does not work properly. (Charles Campbell)
Solution: Accept a zero argument to ":resize" as intended. Add a window number argument to ":resize" to be able to set the size of other windows, because the current window cannot be zero-height. Fix the explorer plugin to avoid changing the window sizes. Add the winrestcmd() function for this.
Files: runtime/doc/eval.txt, runtime/plugin/explorer.vim, src/eval.c, src/ex_cmds.h, src/ex_docmd.c, src/proto/window.pro, src/window.c

Patch 6.2.186 (after 6.2.185)

Problem: Documentation file eval.txt contains examples without indent.
Solution: Insert the indent. Also fix other mistakes.
Files: runtime/doc/eval.txt

Patch 6.2.187

Problem: Using Insure++ reveals a number of bugs. (Dominique Pelle)
Solution: Initialize variables where needed. Free allocated memory to avoid leaks. Fix comparing tags to avoid reading past allocated memory.
Files: src/buffer.c, src/diff.c, src/fileio.c, src/mark.c, src/misc1.c, src/misc2.c, src/ops.c, src/option.c, src/tag.c, src/ui.c

Patch 6.2.188 (extra)

Problem: MS-Windows: Multi-byte characters in a filename cause trouble for the window title.
Solution: Return when the wide function for setting the title did its work.
Files: src/gui_w48.c

Patch 6.2.189

Problem: When setting '**viminfo**' after editing a new buffer its marks are not stored. (Keith Roberts)
Solution: Set the "b_marks_read" flag when skipping to read marks from the viminfo file.
Files: src/fileio.c

Patch 6.2.190

Problem: When editing a compressed files, marks are lost.
Solution: Add the ":lockmarks" modifier and use it in the gzip plugin. Make exists() also check for command modifiers, so that the existence of ":lockmarks" can be checked for. Also add ":keepmarks" to avoid that marks are deleted when filtering text. When deleting lines put marks 'A - 'Z and '0 - '9 at the first deleted line instead of clearing the mark. They were kept in the viminfo file anyway. Avoid that the gzip plugin puts deleted text in registers.
Files: runtime/doc/motion.txt, runtime/plugin/gzip.vim, src/ex_cmds.c, src/ex_docmd.c, src/mark.c, src/structs.h

Patch 6.2.191

Problem: The intro message is outdated. Information about sponsoring and registering is missing.
Solution: Show info about sponsoring and registering Vim in the intro message now and then. Add help file about sponsoring.
Files: runtime/doc/help.txt, runtime/doc/sponsor.txt, runtime/doc/tags, runtime/menu.vim, src/version.c

Patch 6.2.192

Problem: Using **CTRL-T** and **CTRL-D** with "gR" messes up the text. (Jonathan Hankins)
Solution: Avoid calling change_indent() recursively.
Files: src/edit.c

Patch 6.2.193

Problem: When recalling a search pattern from the history from a ":s,a/c," command the '/' ends the search string. (JC van Winkel)
Solution: Store the separator character with the history entries. Escape characters when needed, replace the old separator with the new one. Also fixes that recalling a "/" search for a "?" command messes up trailing flags.
Files: src/eval.c, src/ex_getln.c, src/normal.c, src/proto/ex_getln.pro, src/search.c, src/tag.c

Patch 6.2.194 (after 6.2.068)

Problem: For NetBeans, instead of writing the file and sending an event about it, tell NetBeans to write the file.
Solution: Add the "save" command, "netbeansBuffer" command and "buttonRelease" event to the netbeans protocol. Updated the interface to version 2.2. (Gordon Prieur)
Also: open a fold when the cursor has been positioned.
Also: fix memory leak, free result of nb_quote().
Files: runtime/doc/netbeans.txt, src/fileio.c, src/netbeans.c, src/normal.c, src/proto/netbeans.pro, src/structs.h

Patch 6.2.195 (after 6.2.190)

Problem: Compiling fails for missing CPO_REMMARK symbol.
Solution: Add the patch I forgot to include...
Files: src/option.h

Patch 6.2.196 (after 6.2.191)

Problem: Rebuilding the documentation doesn't use the sponsor.txt file.
Solution: Add sponsor.txt to the Makefile. (Christian J. Robinson)
Files: runtime/doc/Makefile

Patch 6.2.197

Problem: It is not possible to force a redraw of status lines. (Gary Johnson)
Solution: Add the ":redrawstatus" command.
Files: runtime/doc/various.txt, src/ex_cmds.h, src/ex_docmd.c, src/screen.c

Patch 6.2.198

Problem: A few messages are not translated. (Ernest Adrogué)
Solution: Mark the messages to be translated.
Files: src/ex_cmds.c

Patch 6.2.199 (after 6.2.194)

Problem: Vim doesn't work perfectly well with NetBeans.
Solution: When NetBeans saves the file, reset the timestamp to avoid "file changed" warnings. Close a buffer in a proper way. Don't try giving a debug message with an invalid pointer. Send a newDotAndMark message when needed. Report a change by the "r" command to NetBeans. (Gordon Prieur)
Files: src/netbeans.c, src/normal.c

Patch 6.2.200

Problem: When recovering a file, '**fileformat**' is always the default, thus writing the file may result in differences. (Penelope Fudd)

Solution: Before recovering the file try reading the original file to obtain the values of **'fileformat'**, **'fileencoding'**, etc.
Files: src/memline.c

Patch 6.2.201

Problem: When **'autowriteall'** is set `":qall"` still refuses to exit if there is a modified buffer. (Antoine Mechelynck)
Solution: Attempt writing modified buffers as intended.
Files: src/ex_cmds2.c

Patch 6.2.202

Problem: Filetype names of CHILL and ch script are confusing.
Solution: Rename "ch" to "chill" and "chscript" to "ch".
Files: runtime/filetype.vim, runtime/makemenu.vim, runtime/synmenu.vim
runtime/syntax/ch.vim, runtime/syntax/chill.vim

Patch 6.2.203

Problem: With characterwise text that has more than one line, "3P" works wrong. "3p" has the same problem. There also is a display problem. (Daniel Goujot)
Solution: Perform characterwise puts with a count in the right position.
Files: src/ops.c

Patch 6.2.204 (after 6.2.086)

Problem: `"]]"` in a file with closed folds moves to the end of the file. (Nam SungHyun)
Solution: Find one position in each closed fold, then move to after the fold.
Files: src/search.c

Patch 6.2.205 (extra)

Problem: MS-Windows: When the taskbar is at the left or top of the screen, the Vim window placement is wrong.
Solution: Compute the size and position of the window correctly. (Taro Muraoka)
Files: src/gui_w32.c, src/gui_w48.c

Patch 6.2.206

Problem: Multi-byte characters cannot be used as hotkeys in a console dialog. (Mattias Erkişson)
Solution: Handle multibyte characters properly. Also put `()` or `[]` around default hotkeys.
Files: src/message.c, src/macros.h

Patch 6.2.207

Problem: When **'encoding'** is a multibyte encoding, expanding an abbreviation that starts where insertion started results in characters before the insertion to be deleted. (Xiangjiang Ma)
Solution: Stop searching leftwards for the start of the word at the position where insertion started.
Files: src/getchar.c

Patch 6.2.208

Problem: When using fold markers, three lines in a row have the start marker and deleting the first one with `"dd"`, a nested fold is not

deleted. (Kamil Burzynski)
Using marker folding, a level 1 fold doesn't stop when it is followed by "{{{2", starting a level 2 fold.

Solution: Don't stop updating folds at the end of a change when the nesting level of folds is larger than the fold level.
Correctly compute the number of folds that start at "{{{2".
Also avoid a crash for a NULL pointer.

Files: src/fold.c

Patch 6.2.209

Problem: A bogus fold is created when using "P" while the cursor is in the middle of a closed fold. (Kamil Burzynski)

Solution: Correct the line number where marks are modified for closed folds.

Files: src/ops.c

Patch 6.2.210 (extra)

Problem: Mac OSX: antialiased fonts are not supported.

Solution: Add the '**antialias**' option to switch on antialiasing on Mac OSX 10.2 and later. (Peter Cucka)

Files: runtime/doc/options.txt, src/gui_mac.c, src/option.h, src/option.c

Patch 6.2.211 (extra)

Problem: Code for handling file dropped on Vim is duplicated.

Solution: Move the common code to gui_handle_drop().
Add code to drop the files in the window under the cursor.
Support drag&drop on the Macintosh. (Taro Muraoka)
When dropping a directory name edit that directory (using the explorer plugin)
Fix that changing directory with Shift pressed didn't work for relative path names.

Files: src/fileio.c, src/gui.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_w48.c, src/proto/fileio.pro, src/proto/gui.pro

Patch 6.2.212 (after 6.2.199)

Problem: NetBeans: Replacing with a count is not handled correctly.

Solution: Move reporting the change outside of the loop for the count.
(Gordon Prieur)

Files: src/normal.c

Patch 6.2.213 (after 6.2.208)

Problem: Using marker folding, "{{{1" doesn't start a new fold when already at fold level 1. (Servatius Brandt)

Solution: Correctly compute the number of folds that start at "{{{1".

Files: src/fold.c

Patch 6.2.214 (after 6.2.211) (extra)

Problem: Warning for an unused variable.

Solution: Delete the declaration. (Bill McCarthy)

Files: src/gui_w48.c

Patch 6.2.215

Problem: NetBeans: problems saving an unmodified file.

Solution: Add isNetbeansModified() function. Disable netbeans_unmodified().
(Gordon Prieur)

Files: src/fileio.c, src/netbeans.c, src/proto/netbeans.pro,
runtime/doc/netbeans.txt, runtime/doc/tags

Patch 6.2.216 (after 6.2.206)

Problem: Multi-byte characters still cannot be used as hotkeys in a console dialog. (Mattias Erkiison)

Solution: Make get_keystroke() handle multibyte characters.

Files: src/misc1.c

Patch 6.2.217

Problem: GTK: setting the title doesn't always work correctly.

Solution: Invoke gui_mch_settitle(). (Tomas Stehlik)

Files: src/os_unix.c

Patch 6.2.218

Problem: Warning for function without prototype.

Solution: Add argument types to the msgCB field of the BalloonEval struct.

Files: src/gui_beval.h

Patch 6.2.219

Problem: Syntax highlighting hangs on an empty match of an item with a nextgroup. (Charles Campbell)

Solution: Remember that the item has already matched and don't match it again at the same position.

Files: src/syntax.c

Patch 6.2.220

Problem: When a Vim server runs in a console a remote command isn't handled before a key is typed. (Joshua Neuheisel)

Solution: Don't try reading more input when a client-server command has been received.

Files: src/os_unix.c

Patch 6.2.221

Problem: No file name completion for ":cscope add".

Solution: Add the XFILE flag to ":cscope". (Gary Johnson)

Files: src/ex_cmds.h

Patch 6.2.222

Problem: Using "--remote" several times on a row only opens some of the files. (Dany St-Amant)

Solution: Don't delete all typeahead when the server receives a command from a client, only delete typed characters.

Files: src/main.c

Patch 6.2.223

Problem: Cscope: Avoid a hang when cscope waits for a response while Vim waits for a prompt.

Error messages from Cscope mess up the display.

Solution: Detect the hit-enter message and respond by sending a return character to cscope. (Gary Johnson)

Use EMSG() and strerror() when possible. Replace perror() with PERROR() everywhere, add emsg3().

Files: src/diff.c, src/if_cscope.c, src/integration.c, src/message.c,

src/proto/message.pro, src/misc2.c, src/netbeans.c, src/vim.h

Patch 6.2.224

Problem: Mac: Can't compile with small features. (Axel Kielhorn)
Solution: Also include vim_chdirfile() when compiling for the Mac.
Files: src/misc2.c

Patch 6.2.225

Problem: NetBeans: Reported modified state isn't exactly right.
Solution: Report a file being modified in the NetBeans way.
Files: src/netbeans.c

Patch 6.2.226 (after 6.2.107) (extra)

Problem: The "ws2-32.lib" file isn't always available.
Solution: Use "WSock32.lib" instead. (Taro Muraoka, Dan Sharp)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak

Patch 6.2.227 (extra)

Problem: The "PC" symbol is defined but not used anywhere.
Solution: Remove "-DPC" from the makefiles.
Files: src/Make_bc3.mak, src/Make_bc5.mak, src/Make_cyg.mak,
src/Make_ming.mak

Patch 6.2.228

Problem: Receiving CTRL-\ CTRL-N after typing "f" or "m" doesn't switch Vim back to Normal mode. Same for CTRL-\ CTRL-G.
Solution: Check if the character typed after a command is CTRL-\ and obtain another character to check for CTRL-N or CTRL-G, waiting up to **'ttimeoutlen'** msec.
Files: src/normal.c

Patch 6.2.229

Problem: ":function" with a name that uses magic curlies does not work inside a function. (Servatius Brandt)
Solution: Skip over the function name properly.
Files: src/eval.c

Patch 6.2.230 (extra)

Problem: Win32: a complex pattern may cause a crash.
Solution: Use __try and __except to catch the exception and handle it gracefully, when possible. Add myresetstkoflw() to reset the stack overflow. (Benjamin Peterson)
Files: src/Make_bc5.mak, src/os_mswin.c src/os_win32.c, src/os_win32.h,
src/proto/os_win32.pro, src/regexp.c

Patch 6.2.231 (after 6.2.046)

Problem: Various problems when an error exception is raised from within a builtin function. When it is invoked while evaluating arguments to a function following arguments are still evaluated. When invoked with a line range it will be called for remaining lines.
Solution: Update "force_abort" also after calling a builtin function, so that aborting() always returns the correct value. (Servatius Brandt)
Files: src/eval.c, src/ex_eval.c, src/proto/ex_eval.pro,

src/testdir/test49.ok, src/testdir/test49.vim

Patch 6.2.232

Problem: ":python vim.command('python print 2*2')" crashes Vim. (Eugene Minkovskii)
Solution: Disallow executing a Python command recursively and give an error message.
Files: src/if_python.c

Patch 6.2.233

Problem: On Mac OSX adding -pthread for Python only generates a warning. The test for Perl threads rejects Perl while it's OK. Tcl doesn't work at all. The test for Ruby fails if ruby exists but there are no header files. The Ruby library isn't detected properly
Solution: Avoid adding -pthread on Mac OSX. Accept Perl threads when it's not the 5.5 threads. Use the Tcl framework for header files. For Ruby rename cWindow to cVimWindow to avoid a name clash. (Ken Scott) Only enable Ruby when the header files can be found. Use "-lruby" instead of "libruby.a" when it can't be found.
Files: src/auto/configure, src/configure.in, src/if_ruby.c

Patch 6.2.234

Problem: GTK 2 GUI: ":sp" and the ":q" leaves the cursor on the command line.
Solution: Flush output before removing scrollbars. Also do this in other places where gui_mch_*() functions are invoked.
Files: src/ex_cmds.c, src/option.c, src/window.c

Patch 6.2.235 (extra)

Problem: Win32: Cursor isn't removed with a 25x80 window and doing: "1830ia<Esc>400a-<Esc>0w0". (Yasuhiro Matsumoto)
Solution: Remove the call to gui_undraw_cursor() from gui_mch_insert_lines().
Files: src/gui_w48.c

Patch 6.2.236

Problem: Using gvim with Agide gives "connection lost" error messages.
Solution: Only give the "connection lost" message when the buffer was once owned by NetBeans.
Files: src/netbeans.c, src/structs.h

Patch 6.2.237

Problem: GTK 2: Thai text is drawn wrong. It changes when moving the cursor over it.
Solution: Disable the shaping engine, it moves combining characters to a wrong position and combines characters, while drawing the cursor doesn't combine characters.
Files: src/gui_gtk_x11.c

Patch 6.2.238 (after 6.2.231)

Problem: ":function" does not work inside a while loop. (Servatius Brandt)
Solution: Add get_while_line() and pass it to do_one_cmd() when in a while loop, so that all lines are stored and can be used again when

repeating the loop.
Adjust test 49 so that it checks for the fixed problems.
(Servatius Brandt)

Files: src/digraph.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/proto/ex_cmds2.pro, src/proto/ex_docmd.pro,
src/testdir/test49.in, src/testdir/test49.ok,
src/testdir/test49.vim

Patch 6.2.239

Problem: GTK 2: With closed folds the arrow buttons of a vertical scrollbar often doesn't scroll. (Moshe Kaminsky)

Solution: Hackish solution: Detect that the button was pressed from the mouse pointer position.

Files: src/gui_gtk.c, src/gui.c

Patch 6.2.240

Problem: GTK 2: Searching for bitmaps for the toolbar doesn't work as with other systems. Need to explicitly use "icon=name". (Ned Konz, Christian J. Robinson)

Solution: Search for icons like done for Motif.

Files: src/gui_gtk.c

Patch 6.2.241

Problem: GTK 2: Search and Search/Replace dialogs are synced, that makes no sense. Buttons are sometimes greyed-out. (Jeremy Messenger)

Solution: Remove the code to sync the two dialogs. Adjust the code to react to an empty search string to also work for GTK2. (David Necas)

Files: src/gui_gtk.c

Patch 6.2.242

Problem: Gnome: "vim --help" only shows the Gnome arguments, not the Vim arguments.

Solution: Don't let the Gnome code remove the "--help" argument and don't exit at the end of usage().

Files: src/gui_gtk_x11.c, src/main.c

Patch 6.2.243 (extra)

Problem: Mac: Dropping a file on a Vim icon causes a hit-enter prompt.

Solution: Move the dropped files to the global argument list, instead of the usual drop handling. (Eckehard Berns)

Files: src/main.c, src/gui_mac.c

Patch 6.2.244

Problem: ':echo "\xf7"' displays the illegal byte as if it was a character and leaves "cho" after it.

Solution: When checking the length of a UTF-8 byte sequence and it's shorter than the number of bytes available, assume it's an illegal byte.

Files: src/mbyte.c

Patch 6.2.245

Problem: Completion doesn't work for ":keepmarks" and ":lockmarks".

Solution: Add the command modifiers to the table of commands. (Madoka Machitani)

Files: src/ex_cmds.h, src/ex_docmd.c

Patch 6.2.246

Problem: Mac: Starting Vim from Finder doesn't show error messages.
Solution: Recognize that output is being displayed by stderr being
"/dev/console". (Eckehard Berns)
Files: src/main.c, src/message.c

Patch 6.2.247 (after 6.2.193)

Problem: When using a search pattern from the viminfo file the last
character is replaced with a '/'.
Solution: Store the separator character in the right place. (Kelvin Lee)
Files: src/ex_getln.c

Patch 6.2.248

Problem: GTK: When XIM is enabled normal "2" and keypad "2" cannot be
distinguished.
Solution: Detect that XIM changes the keypad key to the expected ASCII
character and fall back to the non-XIM code. (Neil Bird)
Files: src/gui_gtk_x11.c, src/mbyte.c, src/proto/mbyte.pro

Patch 6.2.249

Problem: ":cnext" moves to the error in the next file, but there is no
method to go back.
Solution: Add ":cpfile" and ":cNfile".
Files: src/ex_cmds.h, src/quickfix.c, src/vim.h, runtime/doc/quickfix.txt

Patch 6.2.250

Problem: Memory leaks when using signs. (Xavier de Gaye)
Solution: Delete the list of signs when unloading a buffer.
Files: src/buffer.c

Patch 6.2.251

Problem: GTK: The 'v' flag in '**guioptions**' doesn't work. (Steve Hall)
Order of buttons is reversed for GTK 2.2.4. Don't always get
focus back after handling a dialog.
Solution: Make buttons appear vertically when desired. Reverse the order in
which buttons are added to a dialog. Move mouse pointer around
when the dialog is done and we don't have focus.
Files: src/gui_gtk.c

Patch 6.2.252 (extra, after 6.2.243)

Problem: Mac: Dropping a file on a Vim icon causes a hit-enter prompt for
Mac OS classic.
Solution: Remove the #ifdef from the code that fixes it for Mac OSX.
Files: src/gui_mac.c

Patch 6.2.253

Problem: When '**tagstack**' is not set a ":tag id" command does not work after
a ":tjump" command.
Solution: Set "new_tag" when '**tagstack**' isn't set. (G. Narendran)
Files: src/tag.c

Patch 6.2.254

Problem: May run out of space for error messages.

Solution: Keep room for two more bytes.
Files: src/quickfix.c

Patch 6.2.255

Problem: GTK: A new item in the popup menu is put just after instead of just before the right item. (Gabriel Zachmann)
Solution: Don't increment the menu item index.
Files: src/gui_gtk.c

Patch 6.2.256

Problem: Mac: "macroman" encoding isn't recognized, need to use "8bit-macroman".
Solution: Recognize "macroman" with an alias "mac". (Eckehard Berns)
Files: src/mbyte.c

Patch 6.2.257 (after 6.2.250)

Problem: Signs are deleted for ":bdel", but they could still be useful.
Solution: Delete signs only for ":bwipe".
Files: src/buffer.c

Patch 6.2.258

Problem: GUI: can't disable (grey-out) a popup menu item. (Ajit Thakkar)
Solution: Loop over the popup menus for all modes.
Files: src/menu.c

Patch 6.2.259

Problem: If there are messages when exiting, on the console there is a hit-enter prompt while the message can be read; in the GUI the message may not be visible.
Solution: Use the hit-enter prompt when there is an error message from writing the viminfo file or autocommands, or when there is any output in the GUI and **'verbose'** is set. Don't use a hit-enter prompt for the non-GUI version unless there is an error message.
Files: src/main.c

Patch 6.2.260

Problem: GTK 2: Can't quit a dialog with <Esc>.
GTK 1 and 2: <Enter> always gives a result, even when the default button has been disabled.
Solution: Handle these keys explicitly. When no default button is specified use the first one (works mostly like it was before).
Files: src/gui_gtk.c

Patch 6.2.261

Problem: When **'autoindent'** and **'cindent'** are set and a line is recognized as a comment, starting a new line won't do **'cindent'** formatting.
Solution: Also use **'cindent'** formatting for lines that are used as a comment. (Servatius Brandt)
Files: src/misc1.c

Patch 6.2.262

Problem: 1 **CTRL-W** w beeps, even though going to the first window is possible. (Charles Campbell)
Solution: Don't beep.

Files: src/window.c

Patch 6.2.263

Problem: Lint warnings: Duplicate function prototypes, duplicate macros, use of a zero character instead of a zero pointer, unused variable. Clearing allocated memory in a complicated way.

Solution: Remove the function prototypes from farsi.h. Remove the duplicated lines in keymap.h. Change getvcol() argument from NUL to NULL. Remove the "col" variable in regmatch(). Use lalloc_clear() instead of lalloc(). (Walter Briscoe)

Files: src/farsi.h, src/keymap.h, src/ops.c, src/regexp.c, src/search.c

Patch 6.2.264 (after 6.2.247)

Problem: Writing past allocated memory when using a command line from the viminfo file.

Solution: Store the NUL in the right place.

Files: src/ex_getln.c

Patch 6.2.265

Problem: Although ":set" is not allowed in the sandbox, ":let &opt = val" works.

Solution: Do allow changing options in the sandbox, but not the ones that can't be changed from a modeline.

Files: src/ex_cmds.h, src/options.c

Patch 6.2.266

Problem: When redirecting output and using ":silent", line breaks are missing from output of ":map" and ":tselect". Alignment of columns is wrong.

Solution: Insert a line break where "msg_didout" was tested. Update msg_col when redirecting and using ":silent".

Files: src/getchar.c, src/message.c

Patch 6.2.267 (extra)

Problem: Win32: "&&" in a tearoff menu is not shown. (Luc Hermitte)

Solution: Use the "name" item from the menu instead of the "dname" item.

Files: src/gui_w32.c, src/menu.c

Patch 6.2.268

Problem: GUI: When changing **'guioptions'** part of the window may be off screen. (Randall Morris)

Solution: Adjust the size of the window when changing **'guioptions'**, but only when adding something.

Files: src/gui.c

Patch 6.2.269

Problem: Diff mode does not highlight a change in a combining character. (Raphael Finkel)

Solution: Make diff_find_change() multibyte aware: find the start byte of a character that contains a change.

Files: src/diff.c

Patch 6.2.270

Problem: Completion in Insert mode, then repeating with ".", doesn't handle

Solution: composing characters in the completed text. (Raphael Finkel)
Don't skip over composing chars when adding completed text to the redo buffer.
Files: src/getchar.c

Patch 6.2.271

Problem: NetBeans: Can't do "tail -f" on the log. Passing socket info with an argument or environment variable is not secure.
Solution: Wait after initializing the log. Allow passing the socket info through a file. (Gordon Prieur)
Files: runtime/doc/netbeans.txt, src/main.c, src/netbeans.c

Patch 6.2.272

Problem: When the "po" directory exists, but "po/Makefile" doesn't, building fails. Make loops when the "po" directory has been deleted after running configure.
Solution: Check for the "po/Makefile" instead of just the "po" directory. Check this again before trying to run make with that Makefile.
Files: src/auto/configure, src/configure.in, src/Makefile

Patch 6.2.273

Problem: Changing the sort order in an explorer window for an empty directory produces error messages. (Doug Kearns)
Solution: When an invalid range is used for a function that is not going to be executed, skip over the arguments anyway.
Files: src/eval.c

Patch 6.2.274

Problem: ":print" skips empty lines when 'list' is set and there is no "eol" in 'listchars'. (Yakov Lerner)
Solution: Skip outputting a space for an empty line only when 'list' is set and the end-of-line character is not empty.
Files: src/message.c

Patch 6.2.275 (extra, after 6.2.267)

Problem: Warning for uninitialized variable when using gcc.
Solution: Initialize "acLen" to zero. (Bill McCarthy)
Files: src/gui_w32.c

Patch 6.2.276

Problem: ":echo X()" does not put a line break between the message that X() displays and the text that X() returns. (Yakov Lerner)
Solution: Invoke msg_start() after evaluating the argument.
Files: src/eval.c

Patch 6.2.277

Problem: Vim crashes when a ":runtime ftplugin/ada.vim" causes a recursive loop. (Robert Nowotniak)
Solution: Restore "msg_list" before returning from do_cmdline().
Files: src/ex_docmd.c

Patch 6.2.278

Problem: Using "much" instead of "many".
Solution: Correct the error message.

Files: src/eval.c

Patch 6.2.279

Problem: There is no default choice for a confirm() dialog, now that it is possible not to have a default choice.

Solution: Make the first choice the default choice.

Files: runtime/doc/eval.txt, src/eval.c

Patch 6.2.280

Problem: "do" and ":diffget" don't work in the first line and the last line of a buffer. (Aron Griffis)

Solution: Find a difference above the first line and below the last line. Also fix a few display updating bugs.

Files: src/diff.c, src/fold.c, src/move.c

Patch 6.2.281

Problem: PostScript printing doesn't work on Mac OS X 10.3.2.

Solution: Adjust the header file. (Mike Williams)

Files: runtime/print/prolog.ps

Patch 6.2.282

Problem: When using **CTRL-O** to go back to a help file, it becomes listed. (Andrew Nesbit)

Using ":tag" or ":tjump" in a help file doesn't keep the help file settings (e.g. for '**iskeyword**').

Solution: Don't mark a buffer as listed when its help flag is set. Put all the option settings for a help buffer together in do_ecmd().

Files: src/ex_cmds.c

Patch 6.2.283

Problem: The "local additions" in help.txt are used without conversion, causing latin1 characters showing up wrong when '**enc**' is utf-8. (Antoine J. Mechelynck)

Solution: Convert the text to '**encoding**'.

Files: src/ex_cmds.c

Patch 6.2.284

Problem: Listing a function puts "endfunction" in the message history. Typing "q" at the more prompt isn't handled correctly when listing variables and functions. (Hara Krishna Dara)

Solution: Don't use msg() for "endfunction". Check "got_int" regularly.

Files: src/eval.c

Patch 6.2.285

Problem: GUI: In a single wrapped line that fills the window, "gj" in the last screen line leaves the cursor behind. (Ivan Tarasov)

Solution: Undraw the cursor before scrolling the text up.

Files: src/gui.c

Patch 6.2.286

Problem: When trying to rename a file and it doesn't exist, the destination file is deleted anyway. (Luc Deux)

Solution: Don't delete the destination when the source doesn't exist. (Taro Muraoka)

Files: src/fileio.c

Patch 6.2.287 (after 6.2.264)

Problem: Duplicate lines are added to the viminfo file.

Solution: Compare with existing entries without an offset. Also fixes reading very long history lines from viminfo.

Files: src/ex_getln.c

Patch 6.2.288 (extra)

Problem: Mac: An external program can't be interrupted.

Solution: Don't use the 'c' key for backspace. (Eckehard Berns)

Files: src/gui_mac.c

Patch 6.2.289

Problem: Compiling the Tcl interface with thread support causes ":make" to fail. (Juergen Salk)

Solution: Use \$TCL_DEFS from the Tcl config script to obtain the required compile flags for using the thread library.

Files: src/auto/configure, src/configure.in

Patch 6.2.290 (extra)

Problem: Mac: The mousewheel doesn't work.

Solution: Add mousewheel support. Also fix updating the thumb after a drag and then using another way to scroll. (Eckehard Berns)

Files: src/gui_mac.c

Patch 6.2.291 (extra)

Problem: Mac: the plus button and close button don't do anything.

Solution: Make the plus button maximize the window and the close button close Vim. (Eckehard Berns)

Files: src/gui.c, src/gui_mac.c

Patch 6.2.292

Problem: Motif: When removing GUI arguments from argv[] a "ps -ef" shows the last argument repeated.

Solution: Set argv[argc] to NULL. (Michael Jarvis)

Files: src/gui_x11.c

Patch 6.2.293 (after 6.2.255)

Problem: GTK: A new item in a menu is put before the tearoff item.

Solution: Do increment the menu item index for non-popup menu items.

Files: src/gui_gtk.c

Patch 6.2.294 (extra)

Problem: Mac: Cannot use modifiers with Space, Tab, Enter and Escape.

Solution: Handle all modifiers for these keys. (Eckehard Berns)

Files: src/gui_mac.c

Patch 6.2.295

Problem: When in debug mode, receiving a message from a remote client causes a crash. Evaluating an expression causes Vim to wait for "cont" to be typed, without a prompt. (Hari Krishna Dara)

Solution: Disable debugging when evaluating an expression for a client. (Michael Geddes) Don't try reading into the typeahead buffer when

it may have been filled in another way.
Files: src/ex_getln.c, src/getchar.c, src/if_xcmds.c, src/main.c,
src/misc1.c, src/proto/getchar.pro, src/proto/main.pro,
src/proto/os_unix.pro, src/proto/ui.pro, src/structs.h,
src/os_unix.c, src/ui.c

Patch 6.2.296 (extra)

Problem: Same as 6.2.295.

Solution: Extra files for patch 6.2.295.

Files: src/os_amiga.c, src/os_msdos.c, src/os_riscos.c, src/os_win32.c,
src/proto/os_amiga.pro, src/proto/os_msdos.pro,
src/proto/os_riscos.pro, src/proto/os_win32.pro

Patch 6.2.297 (after 6.2.232)

Problem: Cannot invoke Python commands recursively.

Solution: With Python 2.3 and later use the available mechanisms to invoke
Python recursively. (Matthew Mueller)

Files: src/if_python.c

Patch 6.2.298

Problem: A change always sets the '.' mark and an insert always sets the '^'
mark, even when this is not wanted.
Cannot go back to the position of older changes without undoing
those changes.

Solution: Add the ":keepjumps" command modifier.
Add the "g," and "g;" commands.

Files: runtime/doc/motion.txt, src/ex_cmds.h, src/ex_doccmd.c, src/edit.c,
src/mark.c, src/misc1.c, src/normal.c, src/proto/mark.pro,
src/structs.h, src/undo.c

Patch 6.2.299

Problem: Can only use one language for help files.

Solution: Add the '**helplang**' option to select the preferred language(s).
Make ":helptags" generate tags files for all languages.

Files: runtime/doc/options.txt, runtime/doc/various.txt, src/Makefile,
src/ex_cmds.c, src/ex_cmds2.c, src/ex_cmds.h, src/ex_getln.c,
src/normal.c, src/option.c, src/option.h, src/proto/ex_cmds.pro,
src/proto/ex_cmds2.pro, src/proto/option.pro, src/structs.h,
src/tag.c, src/vim.h

Patch 6.2.300 (after 6.2.297)

Problem: Cannot build Python interface with Python 2.2 or earlier.

Solution: Add a semicolon.

Files: src/if_python.c

Patch 6.2.301

Problem: The "select all" item from the popup menu doesn't work for Select
mode.

Solution: Use the same commands as for the "Edit.select all" menu.
(Benji Fisher)

Files: runtime/menu.vim

Patch 6.2.302

Problem: Using "**CTRL-O** ." in Insert mode doesn't work properly. (Benji

Fisher)
Solution: Restore "restart_edit" after an insert command that was not typed. Avoid waiting with displaying the mode when there is no text to be overwritten.
Fix that "CTRL-O ." sometimes doesn't put the cursor back after the end-of-line. Only reset the flag that CTRL-O was used past the end of the line when restarting editing. Update "o_lnum" number when inserting text and "o_eol" is set.
Files: src/edit.c, src/normal.c

Patch 6.2.303

Problem: Cannot use Unicode digraphs while 'encoding' is not Unicode.
Solution: Convert the character from Unicode to 'encoding' when needed. Use the Unicode digraphs for the Macintosh. (Eckehard Berns)
Files: src/digraph.c

Patch 6.2.304 (extra, after 6.2.256)

Problem: Mac: No proper support for 'encoding'. Conversion without iconv() is not possible.
Solution: Convert input from 'termencoding' to 'encoding'. Add mac_string_convert(). Convert text for the clipboard when needed. (Eckehard Berns)
Files: src/gui_mac.c, src/mbyte.c, src/structs.h, src/vim.h

Patch 6.2.305 (after 6.2.300)

Problem: Win32: Cannot build Python interface with Python 2.3. (Ajit Thakkar)
Solution: Add two functions to the dynamic loading feature.
Files: src/if_python.c

Patch 6.2.306 (extra)

Problem: Win32: Building console version with BCC 5.5 gives a warning for get_cmd_args() prototype missing. (Ajit Thakkar)
Solution: Don't build os_w32exe.c for the console version.
Files: src/Make_bc5.mak

Patch 6.2.307 (after 6.2.299)

Problem: Installing help files fails.
Solution: Expand wildcards for translated help files separately.
Files: src/Makefile

Patch 6.2.308

Problem: Not all systems have "whoami", resulting in an empty user name.
Solution: Use "logname" when possible, "whoami" otherwise. (David Boyce)
Files: src/Makefile

Patch 6.2.309

Problem: "3grx" waits for two ESC to be typed. (Jens Paulus)
Solution: Append the ESC to the stuff buffer when redoing the "gr" insert.
Files: src/edit.c

Patch 6.2.310

Problem: When setting 'undolevels' to -1, making a change and setting 'undolevels' to a positive value an "undo list corrupt" error

occurs. (Madoka Machitani)
Solution: Sync undo before changing **'undolevels'**.
Files: src/option.c

Patch 6.2.311 (after 6.2.298)

Problem: When making several changes in one line the changelist grows quickly. There is no error message for reaching the end of the changelist. Reading changelist marks from viminfo doesn't work properly.

Solution: Only make a new entry in the changelist when making a change in another line or **'textwidth'** columns away. Add E662, E663 and E664 error messages. Put a changelist mark from viminfo one position before the end.

Files: runtime/doc/motion.txt, src/mark.c, src/misc1.c, src/normal.c

Patch 6.2.312 (after 6.2.299)

Problem: "make install" clears the screen when installing the docs.

Solution: Execute ":helptags" in silent mode.

Files: runtime/doc/Makefile

Patch 6.2.313

Problem: When opening folds in a diff window, other diff windows no longer show the same text.

Solution: Sync the folds in diff windows.

Files: src/diff.c, src/fold.c, src/move.c, src/proto/diff.pro, src/proto/move.pro

Patch 6.2.314

Problem: When **'virtualedit'** is set "rx" may cause a crash with a blockwise selection and using "\$". (Moritz Orbach)

Solution: Don't try replacing chars in a line that has no characters in the block.

Files: src/ops.c

Patch 6.2.315

Problem: Using **CTRL-C** in a Visual mode mapping while **'insertmode'** is set stops Vim from returning to Insert mode.

Solution: Don't reset "restart_edit" when a **CTRL-C** is found and **'insertmode'** is set.

Files: src/normal.c

Patch 6.2.316 (after 6.2.312)

Problem: "make install" tries connecting to the X server when installing the docs. (Stephen Thomas)

Solution: Add the "-X" argument.

Files: runtime/doc/Makefile

Patch 6.2.317 (after 6.2.313)

Problem: When using "zi" in a diff window, other diff windows are not adjusted. (Richard Curnow)

Solution: Distribute a change in **'foldenable'** to other diff windows.

Files: src/normal.c

Patch 6.2.318

Problem: When compiling with `_THREAD_SAFE` external commands don't echo typed characters.
Solution: Don't set the terminal mode to `TMODE_SLEEP` when it's already at `TMODE_COOK`.
Files: `src/os_unix.c`

Patch 6.2.319 (extra)

Problem: Building `gvimext.dll` with Mingw doesn't work properly.
Solution: Use `gcc` instead of `dllwrap`. Use long option names. (Alejandro Lopez-Valencia)
Files: `src/GvimExt/Make_ming.mak`

Patch 6.2.320

Problem: Win32: Adding and removing the menubar resizes the Vim window. (Jonathon Merz)
Solution: Don't let a resize event change `'lines'` unexpectedly.
Files: `src/gui.c`

Patch 6.2.321

Problem: When using modeless selection, wrapping lines are not recognized, a line break is always inserted.
Solution: Add `LineWraps[]` to remember whether a line wrapped or not.
Files: `src/globals.h`, `src/screen.c`, `src/ui.c`

Patch 6.2.322

Problem: With `'showcmd'` set, after typing `"dd"` the next `"d"` may not be displayed. (Jens Paulus)
Solution: Redraw the command line after updating the screen, scrolling may have set `"clear_cmdline"`.
Files: `src/screen.c`

Patch 6.2.323

Problem: Win32: expanding `"~/file"` in an autocommand pattern results in backslashes, while this pattern should only have forward slashes.
Solution: Make expanding environment variables respect `'shellslash'` and set `p_ssl` when expanding the autocommand pattern.
Files: `src/fileio.c`, `src/misc1.c`, `src/proto/fileio.pro`

Patch 6.2.324 (extra)

Problem: Win32: when `"vimrun.exe"` has a path with white space, such as `"Program Files"`, executing external commands may fail.
Solution: Put double quotes around the path to `"vimrun"`.
Files: `src/os_win32.c`

Patch 6.2.325

Problem: When `$HOME` includes a space, doing `":set tags=~/tags"` doesn't work, the space is used to separate file names. (Brett Stahlman)
Solution: Escape the space with a backslash.
Files: `src/option.c`

Patch 6.2.326

Problem: `":windo set syntax=foo"` doesn't work. (Tim Chase)
Solution: Don't change `'eventignore'` for `":windo"`.
Files: `src/ex_cmds2.c`

Patch 6.2.327

Problem: When formatting text all marks in the formatted lines are lost. A word is not joined to a previous line when this would be possible. (Mikolaj Machowski)

Solution: Try to keep marks in the same position as much as possible. Also keep mark positions when joining lines. Start auto-formatting in the previous line when appropriate. Add the "gw" operator: Like "gq" but keep the cursor where it is.

Files: runtime/doc/change.txt, src/edit.c, src/globals.h, src/mark.c, src/misc1.c, src/normal.c, src/ops.c, src/proto/edit.pro, src/proto/mark.pro, src/proto/ops.pro, src/structs.h, src/vim.h

Patch 6.2.328

Problem: XIM with GTK: It is hard to understand what XIM is doing.

Solution: Add xim_log() to log XIM events and help with debugging.

Files: src/mbyte.c

Patch 6.2.329

Problem: ":@" does not work Vi compatible. (Antony Scriven)

Solution: Print the last line number instead of the current line. Don't print "line".

Files: src/ex_cmds.h, src/ex_docmd.c

Patch 6.2.330 (extra, after 6.2.267)

Problem: Win32: Crash when tearing off a menu.

Solution: Terminate a string with a NUL. (Yasuhiro Matsumoto)

Files: src/gui_w32.c

Patch 6.2.331 (after 6.2.327)

Problem: "gwap" leaves cursor in the wrong line.

Solution: Remember the cursor position before finding the ends of the paragraph.

Files: src/normal.c, src/ops.c, src/structs.h

Patch 6.2.332 (extra)

Problem: Amiga: Compile error for string array. Compiling the Amiga GUI doesn't work.

Solution: Use a char pointer instead. Move including "gui_amiga.h" to after including "vim.h". Add a semicolon. (Ali Akcaagac)

Files: src/gui_amiga.c, src/os_amiga.c

Patch 6.2.333 (extra)

Problem: Win32: printing doesn't work with specified font charset.

Solution: Use the specified font charset. (Mike Williams)

Files: src/os_mswin.c

Patch 6.2.334 (extra, after 6.2.296)

Problem: Win32: evaluating client expression in debug mode requires typing "cont".

Solution: Use eval_client_expr_to_string().

Files: src/os_mswin.c

Patch 6.2.335

Problem: The ":sign" command cannot be followed by another command.
Solution: Add TRLBAR to the command flags.
Files: src/ex_cmds.h

Patch 6.2.336 (after 6.2.327)
Problem: Mixup of items in an expression.
Solution: Move "== NUL" to the right spot.
Files: src/edit.c

Patch 6.2.337 (extra, after 6.2.319)
Problem: Building gvimext.dll with Mingw doesn't work properly.
Solution: Fix white space and other details. (Alejandro Lopez-Valencia)
Files: src/GvimExt/Make_ming.mak

Patch 6.2.338 (after 6.2.331)
Problem: When undoing "gwap" the cursor is always put at the start of the paragraph. When undoing auto-formatting the cursor may be above the change.
Solution: Try to move the cursor back to where it was or to the first line that actually changed.
Files: src/normal.c, src/ops.c, src/undo.c

Patch 6.2.339
Problem: Crash when using many different highlight groups and a User highlight group. (Juergen Kraemer)
Solution: Do not use the sg_name_u pointer when it is NULL. Also simplify use of the highlight group table.
Files: src/syntax.c

Patch 6.2.340
Problem: ":reg" doesn't show the actual contents of the clipboard if it was filled outside of Vim. (Stuart MacDonald)
Solution: Obtain the clipboard contents before displaying it.
Files: src/ops.c

Patch 6.2.341 (extra)
Problem: Win32: When the path to diff.exe contains a space and using the vimrc generated by the install program, diff mode does not work.
Solution: Put the first double quote just before the space instead of before the path.
Files: src/dosinst.c

Patch 6.2.342 (extra)
Problem: Win32: macros are not always used as expected.
Solution: Define WINVER to 0x0400 instead of 0x400. (Alejandro Lopez-Valencia)
Files: src/Make_bc5.mak, src/Make_cyg.mak, src/Make_mvc.mak

Patch 6.2.343
Problem: Title doesn't work with some window managers. X11: Setting the text property for the window title is hard coded.
Solution: Use STRING format when possible. Use the UTF-8 function when it's available and 'encoding' is utf-8. Use XStringListToTextProperty(). Do the same for the icon name.

(David Harrison)
Files: src/os_unix.c

Patch 6.2.344 (extra, after 6.2.337)

Problem: Cannot build gvimext.dll with MingW on Linux.
Solution: Add support for cross compiling. (Ronald Hoellwarth)
Files: src/GvimExt/Make_ming.mak

Patch 6.2.345 (extra)

Problem: Win32: Copy/paste between two Vims fails if **'encoding'** is not set properly or there are illegal bytes.
Solution: Use a raw byte format. Always set it when copying. When pasting use the raw format if **'encoding'** is the same.
Files: src/os_mswin.c, src/os_win16.c, src/os_win32.c, src/vim.h

Patch 6.2.346

Problem: Win32 console: After using "chcp" Vim does not detect the different codepage.
Solution: Use GetConsoleCP() and when it is different from GetACP() set **'termencoding'**.
Files: src/option.c

Patch 6.2.347 (extra)

Problem: Win32: XP theme support is missing.
Solution: Add a manifest and refer to it from the resource file. (Michael Wookey)
Files: Makefile, src/gvim.exe.mnf, src/vim.rc

Patch 6.2.348

Problem: Win32: "vim c:\dir\(\test)" doesn't work, because the **'isfname'** default value doesn't contain parentheses.
Solution: Temporarily add '(' and ')' to **'isfname'** when expanding file name arguments.
Files: src/main.c

Patch 6.2.349

Problem: Finding a match using **'matchpairs'** may cause a crash. **'matchpairs'** is not used for **'showmatch'**.
Solution: Don't look past the NUL in **'matchpairs'**. Use **'matchpairs'** for **'showmatch'**. (Dave Olszewski)
Files: src/misc1.c, src/normal.c, src/proto/search.pro, src/search.c

Patch 6.2.350

Problem: Not enough info about startup timing.
Solution: Add a few more TIME_MSG() calls.
Files: src/main.c

Patch 6.2.351

Problem: Win32: \$HOME may be set to %USERPROFILE%.
Solution: Expand %VAR% at the start of \$HOME.
Files: src/misc1.c

Patch 6.2.352 (after 6.2.335)

Problem: ":sign texthl=||" does not work.

Solution: Remove the check for a following command. Give an error for extra arguments after "buff=1".
Files: src/ex_cmds.c, src/ex_cmds.h

Patch 6.2.353 (extra)

Problem: Win32: Supported server name length is limited. (Paul Bossi)
Solution: Use MAX_PATH instead of 25.
Files: src/os_mswin.c

Patch 6.2.354 (extra)

Problem: Win32: When the mouse pointer is on a tear-off menu it is hidden when typing but is not redisplayed when moved. (Markx Hackmann)
Solution: Handle the pointer move event for the tear-off menu window.
Files: src/gui_w32.c

Patch 6.2.355 (after 6.2.303)

Problem: When '**encoding**' is a double-byte encoding different from the current locale, the width of characters is not correct. Possible failure and memory leak when using iconv, Unicode digraphs and '**encoding**' is not "utf-8".
Solution: Use iconv() to discover the actual width of characters. Add the "vc_fail" field to vimconv_T. When converting a digraph, init the conversion type to NONE and cleanup afterwards.
Files: src/digraph.c, src/mbyte.c, src/structs.h

Patch 6.2.356

Problem: When using a double-byte '**encoding**' and '**selection**' is "exclusive", "vy" only yanks the first byte of a double-byte character. (Xiangjiang Ma)
Solution: Correct the column in unadjust_for_sel() to position on the first byte, always include the trailing byte of the selected text.
Files: src/normal.c

Patch 6.2.357 (after 6.2.321)

Problem: Memory leak when resizing the Vim window.
Solution: Free the LineWraps array.
Files: src/screen.c

Patch 6.2.358 (after 6.2.299)

Problem: Memory leak when using ":help" and the language doesn't match.
Solution: Free the array with matching tags.
Files: src/ex_cmds.c

Patch 6.2.359 (after 6.2.352)

Problem: Compiler warning for long to int type cast.
Solution: Add explicit type cast.
Files: src/ex_cmds.c

Patch 6.2.360

Problem: "100|" in an empty line results in a ruler "1,0-100". (Pavol Juhas)
Solution: Recompute w_virtcol if the target column was not reached.
Files: src/misc2.c

Patch 6.2.361 (extra)

Problem: Win32: Run gvim, " :set go-=m", use Alt-Tab, keep Alt pressed while pressing Esc, then release Alt: Cursor disappears and typing a key causes a beep. (Hari Krishna Dara)

Solution: Don't ignore the WM_SYSKEYUP event when the menu is disabled.

Files: src/gui_w32.c

Patch 6.2.362 (extra, after 6.2.347)

Problem: Win32: The manifest causes gvim not to work. (Dave Roberts)

Solution: Change "x86" to "X86". (Serge Pirotte)

Files: src/gvim.exe.mnf

Patch 6.2.363

Problem: In an empty file with **'showmode'** off, "i" doesn't change the ruler from "0-1" to "1". Typing "x<BS>" does show "1", but then <Esc> doesn't make it "0-1" again. Same problem for ruler in statusline. (Andrew Pimlott)

Solution: Remember the "empty line" flag with Insert mode and'ed to it.

Files: src/screen.c

Patch 6.2.364

Problem: HTML version of the documentation doesn't mention the encoding, which is a problem for mbyte.txt.

Solution: Adjust the awk script. (Ilya Sher)

Files: runtime/doc/makehtml.awk

Patch 6.2.365

Problem: The configure checks for Perl and Python may add compile and link arguments that break building Vim.

Solution: Do a sanity check: try building with the arguments.

Files: src/auto/configure, src/configure.in

Patch 6.2.366

Problem: When the GUI can't start because no valid font is found, there is no error message. (Ugen)

Solution: Add an error message.

Files: src/gui.c

Patch 6.2.367

Problem: Building the help tags file while installing may fail if there is another Vim in \$PATH.

Solution: Specify the just installed Vim executable. (Gordon Prieur)

Files: src/Makefile

Patch 6.2.368

Problem: When **'autochdir'** is set, closing a window doesn't change to the directory of the new current window. (Salman Halim)

Solution: Handle **'autochdir'** always when a window becomes the current one.

Files: src/window.c

Patch 6.2.369

Problem: Various memory leaks: when using globpath(), when searching for help tags files, when defining a function inside a function, when

giving an error message through an exception, for the final "." line in ":append", in expression "cond ? a : b" that fails and for missing ")" in an expression. Using NULL pointer when adding first user command and for pointer computations with regexp. (tests by Dominique Pelle)

Solution: Fix the leaks by freeing the allocated memory. Don't use the array of user commands when there are no entries. Use a macro instead of a function call for saving and restoring regexp states.
Files: src/eval.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/misc2.c, src/regexp.c, src/screen.c, src/tag.c

Patch 6.2.370 (extra, after 6.2.341)

Problem: Win32: When the path to diff.exe contains a space and using the vimrc generated by the install program, diff mode may not work. (Alejandro Lopez-Valencia)

Solution: Do not use double quotes for arguments that do not have a space.
Files: src/dosinst.c

Patch 6.2.371

Problem: When '**virtualedit**' is set and there is a Tab before the next "x", "dtx" does not delete the whole Tab. (Ken Hashishi)

Solution: Move the cursor to the last position of the Tab. Also for "df<Tab>".

Files: src/normal.c

Patch 6.2.372

Problem: When using balloon evaluation, no value is displayed for members of structures and items of an array.

Solution: Include "->", "." and "[*]" in the expression.

Files: src/gui_beval.c, src/normal.c, src/vim.h

Patch 6.2.373

Problem: When '**winminheight**' is zero and a window is reduced to zero height, the ruler always says "Top" instead of the cursor position. (Antoine J. Mechelynck)

Solution: Don't recompute w_toplevel for a zero-height window.

Files: src/window.c

Patch 6.2.374

Problem: ":echo "hello" | silent normal n" removes the "hello" message. (Servatius Brandt)

Solution: Don't echo the search string when ":silent" was used. Also don't show the mode. In general: don't clear to the end of the screen.

Files: src/gui.c, src/message.c, src/os_unix.c, src/proto/message.pro, src/screen.c, src/search.c, src/window.c

Patch 6.2.375

Problem: When changing '**guioptions**' the hit-enter prompt may be below the end of the Vim window.

Solution: Call screen_alloc() before showing the prompt.

Files: src/message.c

Patch 6.2.376

Problem: Win32: Ruby interface cannot be dynamically linked with Ruby 1.6.

Solution: Add #ifdefs around use of rb_w32_snprintf(). (Benoît Cerrina)
Files: src/if_ruby.c

Patch 6.2.377 (after 6.2.372)

Problem: Compiler warnings for signed/unsigned compare. (Michael Wookey)
Solution: Add type cast.
Files: src/normal.c

Patch 6.2.378 (extra, after 6.2.118)

Problem: Mac: cannot build with Project Builder.
Solution: Add remove_tail_with_ext() to locate and remove the "build" directory from the runtime path. Include os_unix.c when needed. (Dany St Amant)
Files: src/misc1.c, src/os_macosx.c, src/vim.h

Patch 6.2.379

Problem: Using ":mkvimrc" in the ":options" window sets 'bufhidden' to "delete". (Michael Naumann)
Solution: Do not add buffer-specific option values to a global vimrc file.
Files: src/option.c

Patch 6.2.380 (extra)

Problem: DOS: "make test" fails when running it again. Can't "make test" with Borland C.
Solution: Make sure ".out" files are deleted when they get in the way. Add a "test" target to the Borland C Makefile.
Files: src/Make_bc5.mak, src/testdir/Make_dos.mak

Patch 6.2.381

Problem: Setting 'fileencoding' to a comma-separated list (confusing it with 'fileencodings') does not result in an error message. Setting 'fileencoding' in an empty file marks it as modified. There is no "+" in the title after setting 'fileencoding'.
Solution: Check for a comma in 'fileencoding'. Only consider a non-empty file modified by changing 'fileencoding'. Update the title after changing 'fileencoding'.
Files: src/option.c

Patch 6.2.382

Problem: Running "make test" puts marks from test files in viminfo.
Solution: Specify a different viminfo file to use.
Files: src/testdir/test15.in, src/testdir/test49.in

Patch 6.2.383

Problem: ":hi foo term='bla" crashes Vim. (Antony Scriven)
Solution: Check that the closing ' is there.
Files: src/syntax.c

Patch 6.2.384

Problem: ":menu a.&b" ":unmenu a.b" only works if "&b" isn't translated.
Solution: Also compare the names without '&' characters.
Files: src/menu.c

Patch 6.2.385 (extra)

Problem: Win32: forward_slash() and trash_input_buf() are undefined when compiling with small features. (Ajit Thakkar)
Solution: Change the #ifdefs for forward_slash(). Don't call trash_input_buf() if the input buffer isn't used.
Files: src/fileio.c, src/os_win32.c

Patch 6.2.386

Problem: Wasting time trying to read marks from the viminfo file for a buffer without a name.
Solution: Skip reading marks when the buffer has no name.
Files: src/fileio.c

Patch 6.2.387

Problem: There is no highlighting of translated items in help files.
Solution: Search for a "help_ab.vim" syntax file when the help file is called "*.abx". Also improve the help highlighting a bit.
Files: runtime/syntax/help.vim

Patch 6.2.388

Problem: GTK: When displaying some double-width characters they are drawn as single-width, because of conversion to UTF-8.
Solution: Check the width that GTK uses and add a space if it's one instead of two.
Files: src/gui_gtk_x11.c

Patch 6.2.389

Problem: When working over a slow connection, it's very annoying that the last line is partly drawn and then cleared for every change.
Solution: Don't redraw the bottom line if no rows were inserted or deleted. Don't draw the line if we know "@" lines will be used.
Files: src/screen.c

Patch 6.2.390

Problem: Using "r*" in Visual mode on multibyte characters only replaces every other character. (Tyson Roberts)
Solution: Correct the cursor position after replacing each character.
Files: src/ops.c

Patch 6.2.391 (extra)

Problem: The ":highlight" command is not tested.
Solution: Add a test script for ":highlight".
Files: src/testdir/Makefile, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/test51.in, src/testdir/test51.ok

Patch 6.2.392 (after 6.2.384)

Problem: Unused variable.
Solution: Remove "dlen".
Files: src/menu.c

Patch 6.2.393

Problem: When using very long lines the viminfo file can become very big.
Solution: Add the "s" flag to 'viminfo': skip registers with more than the

specified Kbyte of text.
Files: runtime/doc/options.txt, src/ops.c, src/option.c

Patch 6.2.394 (after 6.2.391)

Problem: Test 51 fails on a terminal with 8 colors. (Tony Leneis)
Solution: Use "DarkBlue" instead of "Blue" to avoid the "bold" attribute.
Files: src/testdir/test51.in

Patch 6.2.395

Problem: When using ":tag" or ":pop" the previous matching tag is used.
But since the current file is different, the ordering of the tags
may change.
Solution: Remember what the current buffer was for when re-using cur_match.
Files: src/edit.c, src/ex_cmds.c, src/proto/tag.pro, src/structs.h,
src/tag.c

Patch 6.2.396

Problem: When **CTRL-T** jumps to another file and an autocommand moves the
cursor to the '"' mark, don't end up on the right line. (Michal
Malecki)
Solution: Set the line number after loading the file.
Files: src/tag.c

Patch 6.2.397

Problem: When using a double-byte '**encoding**' mapping <M-x> doesn't work.
(Yasuhiro Matsumoto)
Solution: Do not set the 8th bit of the character but use a modifier.
Files: src/gui_gtk_x11.c, src/gui_x11.c, src/misc2.c

Patch 6.2.398 (extra)

Problem: Win32 console: no extra key modifiers are supported.
Solution: Encode the modifiers into the input stream. Also fix that special
keys are converted and stop working when '**tenc**' is set. Also fix
that when '**tenc**' is initialized the input and output conversion is
not setup properly until '**enc**' or '**tenc**' is set.
Files: src/getchar.c, src/option.c, src/os_win32.c

Patch 6.2.399

Problem: A ":set" command that fails still writes a message when it is
inside a try/catch block.
Solution: Include all the text of the message in the error message.
Files: src/charset.c, src/option.c

Patch 6.2.400

Problem: Can't compile if_xcmdsrv.c on HP-UX 11.0.
Solution: Include header file poll.h. (Malte Neumann)
Files: src/if_xcmdsrv.c

Patch 6.2.401

Problem: When opening a buffer that was previously opened, Vim does not
restore the cursor position if the first line starts with white
space. (Gregory Margo)
Solution: Don't skip restoring the cursor position if it is past the blanks
in the first line.

Files: src/buffer.c

Patch 6.2.402

Problem: Mac: "make install" doesn't generate help tags. (Benji Fisher)

Solution: Generate help tags before copying the runtime files.

Files: src/Makefile

Patch 6.2.403

Problem: ":@y" checks stdin if there are more commands to execute. This fails if stdin is not connected, e.g., when starting the GUI from KDE. (Ned Konz)

Solution: Only check for a next command if there still is typeahead.

Files: src/ex_docmd.c

Patch 6.2.404

Problem: Our own function to determine width of Unicode characters may get outdated. (Markus Kuhn)

Solution: Use wwidth() when it is available. Also use iswprint().

Files: src/auto/configure, src/configure.in, src/config.h.in, src/mbyte.c

Patch 6.2.405

Problem: Cannot map zero without breaking the count before a command. (Benji Fisher)

Solution: Disable mapping zero when entering a count.

Files: src/getchar.c, src/globals.h, src/normal.c

Patch 6.2.406

Problem: ":help \zs", ":help \@=" and similar don't find useful help.

Solution: Prepend "/" to the arguments to find the desired help tag.

Files: src/ex_cmds.c

Patch 6.2.407 (after 6.2.299)

Problem: ":help \@<=" doesn't find help.

Solution: Avoid that ":help \@<=" searches for the "<=" language.

Files: src/tag.c

Patch 6.2.408

Problem: ":compiler" is not consistent: Sets local options and a global variable. (Douglas Potts) There is no error message when a compiler is not supported.

Solution: Use ":compiler!" to set a compiler globally, otherwise it's local to the buffer and "b:current_compiler" is used. Give an error when no compiler script could be found.

Note: updated compiler plugins can be found at

<ftp://ftp.vim.org/pub/vim/runtime/compiler/>

Files: runtime/compiler/msvc.vim, runtime/doc/quickfix.txt, src/eval.c, src/ex_cmds2.c

Patch 6.2.409

Problem: The cursor ends up in the last column instead of after the line when doing "i//<Esc>o" with '**indentexpr**' set to "cindent(v:lnum)". (Toby Allsopp)

Solution: Adjust the cursor as if in Insert mode.

Files: src/misc1.c

Patch 6.2.410 (after 6.2.389)

Problem: In diff mode, when there are more filler lines than fit in the window, they are not drawn.

Solution: Check for filler lines when skipping to draw a line that doesn't fit.

Files: src/screen.c

Patch 6.2.411

Problem: A "\n" inside a string is not seen as a line break by the regular expression matching. (Hari Krishna Dara)

Solution: Add the vim_regexec_nl() function for strings where "\n" is to be matched with a line break.

Files: src/eval.c, src/ex_eval.c, src/proto/regexp.c, src/regexp.c

Patch 6.2.412

Problem: Ruby: "ruby << EOF" inside a function doesn't always work. Also for ":python", ":tcl" and ":perl".

Solution: Check for "<< marker" and skip until "marker" before checking for "endfunction".

Files: src/eval.c

Patch 6.2.413 (after 6.2.411)

Problem: Missing prototype for vim_regexec_nl(). (Marcel Svitalsky)

Solution: Now really include the prototype.

Files: src/proto/regexp.pro

Patch 6.2.414

Problem: The function used for custom completion of user commands cannot have <SID> to make it local. (Hari Krishna Dara)

Solution: Pass the SID of the script where the user command was defined on to the completion. Also clean up #ifdefs.

Files: src/ex_docmd.c, src/eval.c, src/ex_getln.c, src/structs.h

Patch 6.2.415

Problem: Vim may crash after a sequence of events that change the window size. The window layout assumes a larger window than is actually available. (Servatius Brandt)

Solution: Invoke win_new_shellsize() from screenalloc() instead of from set_shellsize().

Files: src/screen.c, src/term.c

Patch 6.2.416

Problem: Compiler warning for incompatible pointer.

Solution: Remove the "&" in the call to poll(). (Xavier de Gaye)

Files: src/os_unix.c

Patch 6.2.417 (after 6.2.393)

Problem: Many people forget that the ''' item in 'viminfo' needs to be preceded with a backslash,

Solution: Add '<' as an alias for the ''' item.

Files: runtime/doc/options.txt, src/ops.c, src/option.c

Patch 6.2.418

Problem: Using ":noremap <F12> :echo "cheese" and ":cabbr cheese xxx": when pressing <F12> still uses the abbreviation. (Hari Krishna)
Solution: Also apply "noremap" to abbreviations.
Files: src/getchar.c

Patch 6.2.419 (extra)

Problem: Win32: Cannot open the Vim window inside another application.
Solution: Add the "-P" argument to specify the window title of the application to run inside. (Zibo Zhao)
Files: runtime/doc/starting.txt, src/main.c, src/gui_w32.c, src/gui_w48.c, src/if_ole.cpp, src/os_mswin.c, src/proto/gui_w32.pro

Patch 6.2.420

Problem: Cannot specify a file to be edited in binary mode without setting the global value of the '**binary**' option.
Solution: Support ":edit ++bin file".
Files: runtime/doc/editing.txt, src/buffer.c, src/eval.c, src/ex_cmds.h, src/ex_docmd.c, src/fileio.c, src/misc2.c

Patch 6.2.421

Problem: Cannot set the '[' and ']' mark, which may be necessary when an autocommand simulates reading a file.
Solution: Allow using "m[" and "m]".
Files: runtime/doc/motion.txt, src/mark.c

Patch 6.2.422

Problem: In **CTRL-X** completion messages the "/" makes them less readable.
Solution: Remove the slashes. (Antony Scriven)
Files: src/edit.c

Patch 6.2.423

Problem: ":vertical wincmd]" does not split vertically.
Solution: Add "postponed_split_flags".
Files: src/ex_docmd.c, src/globals.h, src/if_cscope.c, src/tag.c

Patch 6.2.424

Problem: A BufEnter autocommand that sets an option stops '**mousefocus**' from working in Insert mode (Normal mode is OK). (Gregory Seidman)
Solution: In the Insert mode loop invoke gui_mouse_correct() when needed.
Files: src/edit.c

Patch 6.2.425

Problem: Vertical split and command line window: can only drag status line above the cmdline window on the righthand side, not lefthand side.
Solution: Check the status line row instead of the window pointer.
Files: src/ui.c

Patch 6.2.426

Problem: A syntax region end match with a matchgroup that includes a line break only highlights the last line with matchgroup. (Gary Holloway)
Solution: Also use the line number of the position where the region highlighting ends.

Files: src/syntax.c

Patch 6.2.427 (extra)

Problem: When pasting a lot of text in a multibyte encoding, conversion from **'termencoding'** to **'encoding'** may fail for some characters. (Kuang-che Wu)

Solution: When there is an incomplete byte sequence at the end of the read text keep it for the next time.

Files: src/mbyte.c, src/os_amiga.c, src/os_mswin.c, src/proto/mbyte.pro, src/proto/os_mswin.pro, src/ui.c

Patch 6.2.428

Problem: The X11 clipboard supports the Vim selection for char/line/block mode, but since the encoding is not included can't copy/paste between two Vims with a different **'encoding'**.

Solution: Add a new selection format that includes the **'encoding'**. Perform conversion when necessary.

Files: src/gui_gtk_x11.c, src/ui.c, src/vim.h

Patch 6.2.429

Problem: Unix: glob() doesn't work for a directory with a single quote in the name. (Nazri Ramliy)

Solution: When using the shell to expand, only put double quotes around spaces and single quotes, not the whole thing.

Files: src/os_unix.c

Patch 6.2.430

Problem: BOM at start of a vim script file is not recognized and causes an error message.

Solution: Detect the BOM and skip over it. Also fix that after using `":scriptencoding"` the `iconv()` file descriptor was not closed (memory leak).

Files: src/ex_cmds2.c

Patch 6.2.431

Problem: When using the horizontal scrollbar, the scrolling is limited to the length of the cursor line.

Solution: Make the scroll limit depend on the longest visible line. The cursor is moved when necessary. Including the 'h' flag in **'guioptions'** disables this.

Files: runtime/doc/gui.txt, runtime/doc/options.txt, src/gui.c, src/misc2.c, src/option.h

Patch 6.2.432 (after 6.2.430 and 6.2.431)

Problem: Lint warnings.

Solution: Add type casts.

Files: src/ex_cmds2.c, src/gui.c

Patch 6.2.433

Problem: Translating "VISUAL" and "BLOCK" separately doesn't give a good result. (Alejandro Lopez Valencia)

Solution: Use a string for each combination.

Files: src/screen.c

Patch 6.2.434 (after 6.2.431)

Problem: Compiler warning. (Salman Halim)

Solution: Add type casts.

Files: src/gui.c

Patch 6.2.435

Problem: When there are vertically split windows the minimal Vim window height is computed wrong.

Solution: Use frame_minheight() to correctly compute the minimal height.

Files: src/window.c

Patch 6.2.436

Problem: Running the tests changes the user's viminfo file.

Solution: In test 49 tell the extra Vim to use the test viminfo file.

Files: src/testdir/test49.vim

Patch 6.2.437

Problem: ":mksession" always puts "set nocompatible" in the session file. This changes option settings. (Ron Aaron)

Solution: Add an "if" to only change '**compatible**' when needed.

Files: src/ex_docmd.c

Patch 6.2.438

Problem: When the 'v' flag is present in '**coptions**', backspacing and then typing text again: one character too much is overtyped before inserting is done again.

Solution: Set "dollar_vcol" to the right column.

Files: src/edit.c

Patch 6.2.439

Problem: GTK 2: Changing '**lines**' may cause a mismatch between the window layout and the size of the window.

Solution: Disable the hack with force_shell_resize_idle().

Files: src/gui_gtk_x11.c

Patch 6.2.440

Problem: When '**lazyredraw**' is set the window title is still updated. The size of the Visual area and the ruler are displayed too often.

Solution: Postpone redrawing the window title. Only show the Visual area size when waiting for a character. Don't draw the ruler unnecessary.

Files: src/buffer.c, src/normal.c, src/screen.c

Patch 6.2.441

Problem: ":unabbreviate foo " doesn't work, because of the trailing space, while an abbreviation with a trailing space is not possible. (Paul Jolly)

Solution: Accept a match with the lhs of an abbreviation without the trailing space.

Files: src/getchar.c

Patch 6.2.442

Problem: Cannot manipulate the command line from a function.

Solution: Add getcmdline(), getcmdpos() and setcmdpos() functions and the

Files: `CTRL-\ e` command.
runtime/doc/cmdline.txt, runtime/doc/eval.txt, src/eval.c
src/ex_getln.c, src/ops.c, src/proto/ex_getln.pro,
src/proto/ops.pro

Patch 6.2.443

Problem: With `":silent! echoerr something"` you don't get the position of the error. `emsg()` only writes the message itself and returns.
Solution: Also redirect the position of the error.
Files: `src/message.c`

Patch 6.2.444

Problem: When adding the 'c' flag to a `":substitute"` command it may replace more times than without the 'c' flag. Happens for a match that starts with `"\ze"` (Marcel Svitalsky) and when using `"\@<="` (Klaus Bosau).
Solution: Correct `"prev_matchcol"` when replacing the line. Don't replace the line when the pattern uses look-behind matching.
Files: `src/ex_cmds.c, src/proto/regexp.pro, src/regexp.c`

Patch 6.2.445

Problem: Copying `vimtutor` to `/tmp/something` is not secure, a symlink may cause trouble.
Solution: Create a directory and create the file in it. Use `"umask"` to create the directory with mode 700. (Stefan Nordhausen)
Files: `src/vimtutor`

Patch 6.2.446 (after 6.2.404)

Problem: Using library functions `wcwidth()` and `iswprint()` results in display problems for Hebrew characters. (Ron Aaron)
Solution: Disable the code to use the library functions, use our own.
Files: `src/mbyte.c`

Patch 6.2.447 (after 6.2.440)

Problem: Now that the title is only updated when redrawing, it is no longer possible to show it while executing a function. (Madoka Machitani)
Solution: Make `":redraw"` also update the title.
Files: `src/ex_docmd.c`

Patch 6.2.448 (after 6.2.427)

Problem: Mac: conversion done when `'termencoding'` differs from `'encoding'` fails when pasting a longer text.
Solution: Check for an incomplete sequence at the end of the chunk to be converted. (Eckehard Berns)
Files: `src/mbyte.c`

Patch 6.2.449 (after 6.2.431)

Problem: Get error messages when switching files.
Solution: Check for a valid line number when calculating the width of the horizontal scrollbar. (Helmut Stiegler)
Files: `src/gui.c`

Patch 6.2.450

Problem: `" #include"` and `" #define"` are not recognized with the default

option values for **'include'** and **'defined'**. (RG Kiran)
Solution: Adjust the default values to allow white space before the #.
Files: runtime/doc/options.txt, src/option.c

Patch 6.2.451

Problem: GTK: when using XIM there are various problems, including setting **'modified'** and breaking undo at the wrong moment.
Solution: Add "xim_changed_while_preediting", "preedit_end_col" and im_is_preediting(). (Yasuhiro Matsumoto)
Files: src/ex_getln.c, src/globals.h, src/gui_gtk.c, src/gui_gtk_x11.c, src/mbyte.c, src/misc1.c, src/proto/mbyte.pro, src/screen.c, src/undo.c

Patch 6.2.452

Problem: In diff mode, when DiffAdd and DiffText highlight settings are equal, an added line is highlighted with DiffChange. (Tom Schumm)
Solution: Remember the diff highlight type instead of the attributes.
Files: src/screen.c

Patch 6.2.453

Problem: ":s/foo\|\nbar/x/g" does not replace two times in "foo\nbar". (Pavel Papishev)
Solution: When the pattern can match a line break also try matching at the NUL at the end of a line.
Files: src/ex_cmds.c, src/regexp.c

Patch 6.2.454

Problem: ":let b:changedtick" doesn't work. (Alan Schmitt) ":let b:changedtick = 99" does not give an error message.
Solution: Add code to recognize ":let b:changedtick".
Files: src/eval.c

Patch 6.2.455 (after 6.2.297)

Problem: In Python commands the current locale changes how certain Python functions work. (Eugene M. Minkovskii)
Solution: Set the LC_NUMERIC locale to "C" while executing a Python command.
Files: src/if_python.c

Patch 6.2.456 (extra)

Problem: Win32: Editing a file by its Unicode name (dropping it on Vim or using the file selection dialog) doesn't work. (Yakov Lerner, Alex Jakushev)
Solution: Use wide character functions when file names are involved and convert from/to **'encoding'** where needed.
Files: src/gui_w48.c, src/macros.h, src/memfile.c, src/memline.c, src/os_mswin.c, src/os_win32.c

Patch 6.2.457 (after 6.2.244)

Problem: When **'encoding'** is "utf-8" and writing text with chars above 0x80 in latin1, conversion is wrong every 8200 bytes. (Oyvind Holm)
Solution: Correct the utf_ptr2len_check_len() function and fix the problem of displaying 0xf7 in utfc_ptr2len_check_len().
Files: src/mbyte.c

Patch 6.2.458

Problem: When **'virtualedit'** is set "\$" doesn't move to the end of an unprintable character, causing "y\$" not to include that character. (Fred Ma)

Solution: Set "coladd" to move the cursor to the end of the character.

Files: src/misc2.c

Patch 6.2.459 (after 6.2.454)

Problem: Variable "b" cannot be written. (Salman Halim)

Solution: Compare strings properly.

Files: src/eval.c

Patch 6.2.460 (extra, after 6.2.456)

Problem: Compiler warnings for missing prototypes.

Solution: Include the missing prototypes.

Files: src/proto/os_win32.pro

Patch 6.2.461

Problem: After using a search command "x" starts putting single characters in the numbered registers.

Solution: Reset "use_reg_one" at the right moment.

Files: src/normal.c

Patch 6.2.462

Problem: Finding a matching parenthesis does not correctly handle a backslash in a trailing byte.

Solution: Handle multibyte characters correctly. (Taro Muraoka)

Files: src/search.c

Patch 6.2.463 (extra)

Problem: Win32: An NTFS file system may contain files with extra info streams. The current method to copy them creates one and then deletes it again. (Peter Toennies) Also, only three streams with hard coded names are copied.

Solution: Use BackupRead() to check which info streams the original file contains and only copy these streams.

Files: src/os_win32.c

Patch 6.2.464 (extra, after 6.2.427)

Problem: Amiga: Compilation error with gcc. (Ali Akcaagac)

Solution: Move the #ifdef outside of Read().

Files: src/os_amiga.c

Patch 6.2.465

Problem: When resizing the GUI window the window manager sometimes moves it left of or above the screen. (Michael McCarty)

Solution: Check the window position after resizing it and move it onto the screen when it isn't.

Files: src/gui.c

Patch 6.2.466 (extra, after 6.2.456)

Problem: Win32: Compiling with Borland C fails, and an un/signed warning.

Solution: Redefine wcsicmp() to wcscmpi() and add type casts. (Yasuhiro Matsumoto)

Files: src/os_win32.c

Patch 6.2.467 (extra, after 6.2.463)

Problem: Win32: can't compile without multibyte feature. (Ajit Thakkar)

Solution: Add #ifdefs around the info stream code.

Files: src/os_win32.c

Patch 6.2.468

Problem: Compiler warnings for shadowed variables. (Matthias Mohr)

Solution: Delete superfluous variables and rename others.

Files: src/eval.c, src/ex_docmd.c, src/ex_eval.c, src/if_cscope.c,
src/fold.c, src/option.c, src/os_unix.c, src/quickfix.c,
src/regexp.c

Patch 6.2.469 (extra, after 6.2.456)

Problem: Win32: Can't create swap file when **'encoding'** differs from the active code page. (Kriton Kyrimis)

Solution: In enc_to_ucs2() terminate the converted string with a NUL

Files: src/os_mswin.c

Patch 6.2.470

Problem: The name returned by tempname() may be equal to the file used for shell output when ignoring case.

Solution: Skip 'O' and 'I' in tempname().

Files: src/eval.c

Patch 6.2.471

Problem: "-L/usr/lib" is used in the link command, even though it's supposed to be filtered out. "-lw" and "-ldl" are not automatically added when needed for "-lXmu". (Antonio Colombo)

Solution: Check for a space after the argument instead of before. Also remove "-R/usr/lib" if it's there. Check for "-lw" and "-ldl" before trying "-lXmu".

Files: src/auto/configure, src/configure.in, src/link.sh

Patch 6.2.472

Problem: When using a FileChangedShell autocommand that changes the current buffer, a buffer exists that can't be wiped out.

Also, Vim sometimes crashes when executing an external command that changes the buffer and a FileChangedShell autocommand is used. (Hari Krishna Dara)

Users are confused by the warning for a file being changed outside of Vim.

Solution: Avoid that the window counter for a buffer is incremented twice.

Avoid that buf_check_timestamp() is used recursively.

Add a hint to look in the help for more info.

Files: src/ex_cmds.c, src/fileio.c

Patch 6.2.473

Problem: Using **CTRL-]** in a help buffer without a name causes a crash.

Solution: Check for name to be present before using it. (Taro Muraoka)

Files: src/tag.c

Patch 6.2.474 (extra, after 6.2.456)

Problem: When Vim is starting up conversion is done unnecessarily. Failure to find the runtime files on Windows 98. (Randall W. Morris)
Solution: Init enc_codepage negative, only use it when not negative. Don't use GetFileAttributesW() on Windows 98 or earlier.
Files: src/globals.h, src/gui_w32.c, src/gui_w48.c, src/os_mswin.c, src/os_win32.c

Patch 6.2.475

Problem: Commands after "perl <<EOF" are parsed as Vim commands when they are not executed.
Solution: Properly skip over the perl commands.
Files: src/ex_docmd.c, src/ex_getln.c, src/if_perl.xs, src/if_python.c, src/if_ruby.c, src/if_tcl.c, src/misc2.c

Patch 6.2.476

Problem: When reloading a hidden buffer changed outside of Vim and the current buffer is read-only, the reloaded buffer becomes read-only. (Hari Krishna Dara)
Solution: Save the '**readonly**' flag of the reloaded buffer instead of the current buffer.
Files: src/fileio.c

Patch 6.2.477

Problem: Using remote_send(v:servername, "\<C-V>") causes Vim to hang. (Yakov Lerner)
Solution: When the resulting string is empty don't set received_from_client.
Files: src/main.c

Patch 6.2.478

Problem: Win32: "--remote file" fails changing directory if the current directory name starts with a single quote. (Iestyn Walters)
Solution: Add a backslash where it will be removed later.
Files: src/main.c, src/misc2.c, src/proto/misc2.pro

Patch 6.2.479

Problem: The error message for errors during recovery goes unnoticed.
Solution: Avoid that the hit-enter prompt overwrites the message. Add a few lines to make the error stand out.
Files: src/main.c, src/message.c, src/memline.c

Patch 6.2.480

Problem: NetBeans: Using negative index in array. backslash at end of message may cause Vim to crash. (Xavier de Gaye)
Solution: Initialize buf_list_used to zero. Check for trailing backslash.
Files: src/netbeans.c

Patch 6.2.481

Problem: When writing a file it is not possible to specify that hard and/or symlinks are to be broken instead of preserved.
Solution: Add the "breaksymlink" and "breakhardlink" values to '**backupcopy**'. (Simon Ekstrand)
Files: runtime/doc/options.txt, src/fileio.c, src/option.c, src/option.h

Patch 6.2.482

Problem: Repeating insert of CTRL-K 1 S doesn't work. The superscript 1 is considered to be a digit. (Juergen Kraemer)

Solution: In vim_isdigit() only accept '0' to '9'. Use VIM_ISDIGIT() for speed where possible. Also add vim_isxdigit().

Files: src/buffer.c, src/charset.c, src/diff.c, src/digraph.c, src/edit.c, src/eval.c,, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/if_xcmdsrv.c, src/farsi.c, src/fileio.c, src/fold.c, src/getchar.c, src/gui.c, src/if_cscope.c, src/macros.h, src/main.c, src/mark.c, src/mbyte.c, src/menu.c, src/misc1.c, src/misc2.c, src/normal.c, src/ops.c, src/option.c, src/proto/charset.pro, src/regexp.c, src/screen.c, src/search.c, src/syntax.c, src/tag.c, src/term.c, src/termlib.c

Patch 6.2.483 (extra, after 6.2.482)

Problem: See 6.2.482.

Solution: Extra part of patch 6.2.482.

Files: src/gui_photon.c, src/gui_w48.c, src/os_msdos.c, src/os_mswin.c

Patch 6.2.484

Problem: MS-Windows: With the included diff.exe, differences after a CTRL-Z are not recognized. (Peter Keresztes)

Solution: Write the files with unix fileformat and invoke diff with --binary if possible.

Files: src/diff.c

Patch 6.2.485

Problem: A BufWriteCmd autocommand cannot know if "!" was used or not. (Hari Krishna Dara)

Solution: Add the v:cmdbang variable.

Files: runtime/doc/eval.txt, src/eval.c, src/proto/eval.pro, src/fileio.c, src/vim.h

Patch 6.2.486 (6.2.482)

Problem: Diff for eval.c is missing.

Solution: Addition to patch 6.2.482.

Files: src/eval.c

Patch 6.2.487 (extra, after 6.2.456)

Problem: Compiler warnings for wrong prototype. (Alejandro Lopez Valencia)

Solution: Delete the prototype for Handle_WM_Notify().

Files: src/proto/gui_w32.pro

Patch 6.2.488

Problem: Missing ")" in *.ch filetype detection.

Solution: Add the ")". (Ciaran McCreesh)

Files: runtime/filetype.vim

Patch 6.2.489

Problem: When accidentally opening a session in Vim which has already been opened in another Vim there is a long row of ATTENTION prompts. Need to quit each of them to get out. (Robert Webb)

Solution: Add the "Abort" alternative to the dialog.

Files: src/memline.c

Patch 6.2.490

Problem: With **'paragraph'** it is not possible to use a single dot as a paragraph boundary. (Dorai Sitaram)
Solution: Allow using " " (two spaces) in **'paragraph'** to match ".\$" or ". \$"
Files: src/search.c

Patch 6.2.491

Problem: Decrementing a position doesn't take care of multibyte chars.
Solution: Adjust the column for multibyte characters. Remove mb_dec(). (Yasuhiro Matsumoto)
Files: src/mbyte.c, src/misc2.c, src/proto/mbyte.pro

Patch 6.2.492

Problem: When using ":redraw" while there is a message, the next ":echo" still causes text to scroll. (Yasuhiro Matsumoto)
Solution: Reset msg_didout and msg_col, so that after ":redraw" the next message overwrites an existing one.
Files: src/ex_docmd.c

Patch 6.2.493

Problem: "@x" doesn't work when **'insertmode'** is set. (Benji Fisher)
Solution: Put "restart_edit" in the typeahead buffer, so that it's used after executing the register contents.
Files: src/ops.c

Patch 6.2.494

Problem: Using diff mode with two windows, when moving horizontally in inserted lines, a fold in the other window may open.
Solution: Compute the line number in the other window correctly.
Files: src/diff.c

Patch 6.2.495 (extra, after 6.2.456)

Problem: Win32: The file dialog doesn't work on Windows 95.
Solution: Put the wide code of gui_mch_browse() in gui_mch_browseW() and use it only on Windows NT/2000/XP.
Files: src/gui_w32.c, src/gui_w48.c

Patch 6.2.496

Problem: FreeBSD 4.x: When compiled with the pthread library (Python) a complicated pattern may cause Vim to crash. Catching the signal doesn't work.
Solution: When compiled with threads, instead of using the normal stacksize limit, use the size of the initial stack.
Files: src/auto/configure, src/config.h.in, src/configure.in, src/os_unix.c

Patch 6.2.497 (extra)

Problem: Russian messages are only available in one encoding.
Solution: Convert the messages to MS-Windows codepages. (Vassily Ragosin)
Files: src/po/Makefile

Patch 6.2.498

Problem: Non-latin1 help files are not properly supported.
Solution: Support utf-8 help files and convert them to **'encoding'** when needed.
Files: src/fileio.c

Patch 6.2.499

Problem: When writing a file and halting the system, the file might be lost when using a journaling file system.
Solution: Use fsync() to flush the file data to disk after writing a file. (Radim Kolar)
Files: src/fileio.c

Patch 6.2.500 (extra)

Problem: The DOS/MS-Windows the installer doesn't use the --binary flag for diff.
Solution: Add --binary to the diff argument in MyDiff(). (Alejandro Lopez-Valencia)
Files: src/dosinst.c

Patch 6.2.501

Problem: Vim does not compile with MorphOS.
Solution: Add a Makefile and a few changes to make Vim work with MorphOS. (Ali Akcaagac)
Files: runtime/doc/os_amiga.txt, src/INSTALLami.txt, src/Make_morphos.mak, src/memfile.c, src/term.c

Patch 6.2.502

Problem: Building fails for generating message files.
Solution: Add dummy message files.
Files: src/po/ca.po, src/po/ru.po, src/po/sv.po

Patch 6.2.503

Problem: Mac: Can't compile MacRoman conversions without the GUI.
Solution: Also link with the Carbon framework for the terminal version, for the MacRoman conversion functions. (Eckehard Berns)
Remove -ltermcap from the GUI link command, it is not needed.
Files: src/auto/configure, src/Makefile, src/configure.in

Patch 6.2.504

Problem: Various problems with **'cindent'**, among which that a list of variable declarations is not indented properly.
Solution: Fix the wrong indenting. Improve indenting of C++ methods. Add the 'i', 'b' and 'W' options to **'cinoptions'**. (mostly by Helmut Stiegler)
Improve indenting of preprocessor-continuation lines.
Files: runtime/doc/indent.txt, src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 6.2.505

Problem: Help for -P argument is missing. (Ronald Hoellwarth)
Solution: Add the patch that was missing in 6.2.419.
Files: runtime/doc/starting.txt

Patch 6.2.506 (extra)

Problem: Win32: When **'encoding'** is a codepage then reading a utf-8 file only works when iconv is available. Writing a file in another codepage uses the wrong kind of conversion.

Solution: Use internal conversion functions. Enable reading and writing files with **'fileencoding'** different from **'encoding'** for all valid codepages and utf-8 without the need for iconv.

Files: src/fileio.c, src/testdir/Make_dos.mak, src/testdir/test52.in, src/testdir/test52.ok

Patch 6.2.507

Problem: The ownership of the file with the password for the NetBeans connection is not checked. "-nb={file}" doesn't work for GTK.

Solution: Only accept the file when owned by the user and not accessible by others. Detect "-nb=" for GTK.

Files: src/netbeans.c, src/gui_gtk_x11.c

Patch 6.2.508

Problem: Win32: "v:lang" does not show the current language for messages if it differs from the other locale settings.

Solution: Use the value of the \$LC_MESSAGES environment variable.

Files: src/ex_cmds2.c

Patch 6.2.509 (after 6.2.508)

Problem: Crash when \$LANG is not set.

Solution: Add check for NULL pointer. (Ron Aaron)

Files: src/ex_cmds2.c

Patch 6.2.510 (after 6.2.507)

Problem: Warning for pointer conversion.

Solution: Add a type cast.

Files: src/gui_gtk_x11.c

Patch 6.2.511

Problem: Tags in Russian help files are in utf-8 encoding, which may be different from **'encoding'**.

Solution: Use the "TAG_FILE_ENCODING" field in the tags file to specify the encoding of the tags. Convert help tags from **'encoding'** to the tag file encoding when searching for matches, do the reverse when listing help tags.

Files: runtime/doc/tagsrch.txt, src/ex_cmds.c, src/tag.c

Patch 6.2.512

Problem: Translating "\"\n" is useless. (Gerfried Fuchs)

Solution: Remove the _() around it.

Files: src/main.c, src/memline.c

Patch 6.2.513 (after 6.2.507)

Problem: NetBeans: the check for owning the connection info file can be simplified. (Nikolay Molchanov)

Solution: Only check if the access mode is right.

Files: src/netbeans.c

Patch 6.2.514

Problem: When a highlight/syntax group name contains invalid characters

there is no warning.
Solution: Add an error for unprintable characters and a warning for other invalid characters.
Files: src/syntax.c

Patch 6.2.515

Problem: When using the options window **'swapfile'** is reset.
Solution: Use `":setlocal"` instead of `":set"`.
Files: runtime/optwin.vim

Patch 6.2.516

Problem: The sign column cannot be seen, looks like there are two spaces before the text. (Rob Retter)
Solution: Add the SignColumn highlight group.
Files: runtime/doc/options.txt, runtime/doc/sign.txt, src/option.c, src/screen.c, src/syntax.c, src/vim.h

Patch 6.2.517

Problem: Using `"r*` in Visual mode on multibyte characters replaces too many characters. In Visual Block mode replacing with a multibyte character doesn't work.
Solution: Adjust the operator end for the difference in byte length of the original and the replaced character. Insert all bytes of a multibyte character, take care of double-wide characters.
Files: src/ops.c

Patch 6.2.518

Problem: Last line of a window is not updated after using `"J"` and then `"D"`. (Adri Verhoef)
Solution: When no line is found below a change that doesn't need updating, update all lines below the change.
Files: src/screen.c

Patch 6.2.519

Problem: Mac: cannot read/write files in MacRoman format.
Solution: Do internal conversion from/to MacRoman to/from utf-8 and latin1. (Eckehard Berns)
Files: src/fileio.c

Patch 6.2.520 (extra)

Problem: The NSIS installer is outdated.
Solution: Make it work with NSIS 2.0. Also include console executables for Win 95/98/ME and Win NT/2000/XP. Use LZWA compression. Use `"/oname"` to avoid having to rename files before running NSIS.
Files: Makefile, nsis/gvim.nsi

Patch 6.2.521

Problem: When using silent Ex mode the "changing a readonly file" warning is omitted but the one second wait isn't. (Yakov Lerner)
Solution: Skip the delay when `"silent_mode"` is set.
Files: src/misc1.c

Patch 6.2.522

Problem: GUI: when changing **'cmdheight'** in the gvimrc file the window

layout is messed up. (Keith Dart)
Solution: Skip updating the window layout when changing **'cmdheight'** while still starting up.
Files: src/option.c

Patch 6.2.523

Problem: When loading a session and aborting when a swap file already exists, the user is left with useless windows. (Robert Webb)
Solution: Load one file before creating the windows.
Files: src/ex_docmd.c

Patch 6.2.524 (extra, after 6.2.520)

Problem: Win32: (un)installing gvimext.dll may fail if it was used. The desktop and start menu links are created for the current user instead of all users. Using the home directory as working directory for the links is a bad idea for multi-user systems. Cannot use Vim from the "Open With..." menu.
Solution: Force a reboot if necessary. (Alejandro Lopez-Valencia) Also use macros for the directory of the source and runtime files. Use "CSIDL_COMMON_*" instead of "CSIDL_*" when possible. Do not specify a working directory in the links. Add Vim to the "Open With..." menu. (Giuseppe Bilotta)
Files: nsis/gvim.nsi, src/dosinst.c, src/dosinst.h, src/uninstal.c

Patch 6.2.525

Problem: When the history contains a very long line ":history" causes a crash. (Volker Kiefel)
Solution: Shorten the history entry to fit it in one line.
Files: src/ex_getln.c

Patch 6.2.526

Problem: When s:lang is "ja" the Japanese menus are not used.
Solution: Add **'encoding'** to the language when there is no charset.
Files: runtime/menu.vim

Patch 6.2.527

Problem: The 2html script uses ":wincmd p", which breaks when using some autocommands.
Solution: Remember the window numbers and jump to them with ":wincmd w". Also add XHTML support. (Panagiotis Issaris)
Files: runtime/syntax/2html.vim

Patch 6.2.528

Problem: NetBeans: Changes of the "~" command are not reported.
Solution: Call netbeans_inserted() after performing "~". (Gordon Prieur) Also change NetBeans debugging to append to the log file. Also fix that "~" in Visual block mode changes too much if there are multibyte characters.
Files: src/nbdebug.c, src/normal.c, src/ops.c

Patch 6.2.529 (extra)

Problem: VisVim only works for Admin. Doing it for one user doesn't work. (Alexandre Gouraud)

Solution: When registering the module fails, simply continue.
Files: src/VisVim/VisVim.cpp

Patch 6.2.530

Problem: Warning for missing prototype on the Amiga.
Solution: Include time.h
Files: src/version.c

Patch 6.2.531

Problem: In silent ex mode no messages are given, which makes debugging very difficult.
Solution: Do output messages when **'verbose'** is set.
Files: src/message.c, src/ui.c

Patch 6.2.532 (extra)

Problem: Compiling for Win32s with VC 4.1 doesn't work.
Solution: Don't use CP_UTF8 if it's not defined. Don't use CSIDL_COMMON* when not defined.
Files: src/dosinst.h, src/fileio.c

Win32 console: After patch 6.2.398 Ex mode did not work. (Yasuhiro Matsumoto)

Patch 6.3a.001

Problem: Win32: if testing for the "--binary" option fails, diff isn't used at all.
Solution: Handle the "ok" flag properly. (Yasuhiro Matsumoto)
Files: src/diff.c

Patch 6.3a.002

Problem: NetBeans: An insert command from NetBeans beyond the end of a buffer crashes Vim. (Xavier de Gaye)
Solution: Use a local pos_T structure for the position.
Files: src/netbeans.c

Patch 6.3a.003

Problem: E315 error with auto-formatting comments. (Henry Van Roessel)
Solution: Pass the line number to same_leader().
Files: src/ops.c

Patch 6.3a.004

Problem: Test32 fails on Windows XP for the DJGPP version. Renaming test11.out fails.
Solution: Don't try renaming, create new files to use for the test.
Files: src/testdir/test32.in, src/testdir/test32.ok

Patch 6.3a.005

Problem: ":checkpath!" does not use **'includeexpr'**.
Solution: Use a file name that was found directly. When a file was not found and the located name is empty, use the rest of the line.
Files: src/search.c

Patch 6.3a.006

Problem: "yip" moves the cursor to the first yanked line, but not to the first column. Looks like not all text was yanked. (Jens Paulus)

Solution: Move the cursor to the first column.
Files: src/search.c

Patch 6.3a.007

Problem: **'cindent'** recognizes "enum" but not "typedef enum".
Solution: Skip over "typedef" before checking for "enum". (Helmut Stiegler)
Also avoid that searching for this item goes too far back.
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 6.3a.008 (extra)

Problem: Windows 98: Some of the wide functions are not implemented, resulting in file I/O to fail. This depends on what Unicode support is installed.
Solution: Handle the failure and fall back to non-wide functions.
Files: src/os_win32.c

Patch 6.3a.009

Problem: Win32: Completion of filenames does not work properly when **'encoding'** differs from the active code page.
Solution: Use wide functions for expanding wildcards when appropriate.
Files: src/misc1.c

Patch 6.3a.010 (extra)

Problem: Win32: Characters in the window title that do not appear in the active codepage are replaced by a question mark.
Solution: Use DefWindowProcW() instead of DefWindowProc() when possible.
Files: src/glbl_ime.cpp, src/globals.h, src/proto/gui_w16.pro, src/proto/gui_w32.pro, src/gui_w16.c, src/gui_w32.c, src/gui_w48.c

Patch 6.3a.011

Problem: Using the explorer plugin changes a local directory to the global directory.
Solution: Don't use ":chdir" to restore the current directory. Make "expand('%:~p~') remove "/.~/ and "/.~/ items from the path.
Files: runtime/plugin/explorer.vim, src/eval.c, src/os_unix.c

Patch 6.3a.012 (extra)

Problem: On Windows 98 the installer doesn't work, don't even get the "I agree" button. The check for the path ending in "vim" makes the browse dialog hard to use. The default path when no previous Vim is installed is "c:\vim" instead of "c:\Program Files\Vim".
Solution: Remove the background gradient command. Change the .onVerifyInstDir function to a leave function for the directory page. Don't let the install program default to c:\vim when no path could be found.
Files: nsis/gvim.nsi, src/dosinst.c

Patch 6.3a.013 (extra)

Problem: Win32: Characters in the menu that are not in the active codepage are garbled.
Solution: Convert menu strings from **'encoding'** to the active codepage.
Files: src/gui_w32.c, src/gui_w48.c

Patch 6.3a.014

Problem: Using multibyte text and highlighting in a statusline causes gaps to appear. (Helmut Stiegler)
Solution: Advance the column by text width instead of number of bytes. Add the vim_strnsize() function.
Files: src/charset.c, src/proto/charset.pro, src/screen.c

Patch 6.3a.015

Problem: Using the "select all" menu item when **'insertmode'** is set and clicking the mouse button doesn't return to Insert mode. The Buffers/Delete menu doesn't offer a choice to abandon a changed buffer. (Jens Paulus)
Solution: Don't use CTRL-\ **CTRL-N**. Add ":confirm" for the Buffers menu items.
Files: runtime/menu.vim

Patch 6.3a.016

Problem: After cancelling the ":confirm" dialog the error message and hit-enter prompt may not be displayed properly.
Solution: Flush output after showing the dialog.
Files: src/message.c

Patch 6.3a.017

Problem: servername() doesn't work when Vim was started with the "-X" argument or when the "exclude" in **'clipboard'** matches the terminal name. (Robert Nowotniak)
Solution: Force connecting to the X server when using client-server commands.
Files: src/eval.c, src/globals.h, src/os_unix.c

Patch 6.3a.018 (after 6.3a.017)

Problem: Compiler warning for return value of make_connection().
Solution: Use void return type.
Files: src/eval.c

Patch 6.3a.019 (extra)

Problem: Win32: typing non-latin1 characters doesn't work.
Solution: Invoke _OnChar() directly to avoid that the argument is truncated to a byte. Convert the UTF-16 character to bytes according to **'encoding'** and ignore **'termencoding'**. Same for _OnSysChar().
Files: src/gui_w32.c, src/gui_w48.c

Patch 6.3a.020 (extra)

Problem: Missing support for AROS (AmigaOS reimplementation). Amiga GUI doesn't work.
Solution: Add AROS support. (Adam Chodorowski)
Fix Amiga GUI problems. (Georg Steger, Ali Akcaagac)
Files: Makefile, src/Make_aros.mak, src/gui_amiga.c, src/gui_amiga.h, src/memfile.c, src/os_amiga.c, src/term.c

Patch 6.3a.021 (after 6.3a.017)

Problem: Can't compile with X11 but without GUI.
Solution: Put use of "gui.in_use" inside an #ifdef.
Files: src/eval.c

Patch 6.3a.022

Problem: When typing Tabs when **'softtabstop'** is used and **'list'** is set a tab is counted for two spaces.
Solution: Use the "L" flag in **'coptions'** to tell whether a tab is counted as two spaces or as **'tabstop'**. (Antony Scriven)
Files: runtime/doc/options.txt, src/edit.c

Patch 6.3a.023

Problem: Completion on the command line doesn't handle backslashes properly. Only the tail of matches is shown, even when not completing filenames.
Solution: When turning the string into a pattern double backslashes. Don't omit the path when not expanding files or directories.
Files: src/ex_getln.c

Patch 6.3a.024

Problem: The "save all" toolbar item fails for buffers that don't have a name. When using ":wa" or closing the Vim window and there are nameless buffers, browsing for a name may cause the name being given to the wrong buffer or not stored properly. ":browse" only worked for one file.
Solution: Use ":confirm browse" for "save all".
Pass buffer argument to setfname(). Restore "browse" flag and "forceit" after doing the work for one file.
Files: runtime/menu.vim, src/buffer.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/memline.c, src/message.c, src/window.c, src/proto/buffer.pro, src/proto/ex_cmds2.pro, src/proto/memline.pro

Patch 6.3a.025

Problem: Setting **'virtualedit'** moves the cursor. (Benji Fisher)
Solution: Update the virtual column before using it.
Files: src/option.c

Patch 6.3a.026 (extra, after 6.3a.008)

Problem: Editing files on Windows 98 doesn't work when **'encoding'** is "utf-8" (Antoine Mechelynck)
Warning for missing function prototype.
Solution: For all wide functions check if it failed because it is not implemented. Use ANSI function declaration for char_to_string().
Files: src/gui_w48.c, src/os_mswin.c, src/os_win32.c

Patch 6.3a.027 (extra, after 6.3a.026)

Problem: Compiler warning for function argument.
Solution: Declare both char and WCHAR arrays.
Files: src/gui_w48.c

Patch 6.3a.028

Problem: ":normal ." doesn't work inside a function, because redo is saved and restored. (Benji Fisher)
Solution: Make a copy of the redo buffer when executing a function.
Files: src/getchar.c

Patch 6.3b.001 (extra)

Problem: Bcc 5: The generated auto/pathdef can't be compiled.
Solution: Fix the way quotes and backslashes are escaped.
Files: src/Make_bc5.mak

Patch 6.3b.002

Problem: Win32: conversion during file write fails when a double-byte character is split over two writes.
Solution: Fix the conversion retry without a trailing byte. (Taro Muraoka)
Files: src/fileio.c

Patch 6.3b.003 (extra)

Problem: Win32: When compiling with Borland C 5.5 and **'encoding'** is "utf-8" then Vim can't open files under MS-Windows 98. (Antoine J. Mechelynck)
Solution: Don't use _wstat(), _wopen() and _wfopen() in this situation.
Files: src/os_mswin.c, src/os_win32.c

Patch 6.3b.004

Problem: ":helpgrep" includes a trailing CR in the text line.
Solution: Remove the CR.
Files: src/quickfix.c

Patch 6.3b.005

Problem: ":echo &g:ai" results in the local option value. (Salman Halim)
Solution: Pass the flags from find_option_end() to get_option_value().
Files: src/eval.c

Patch 6.3b.006

Problem: When using "mswin.vim", **CTRL-V** in Insert mode leaves cursor before last pasted character. (Mathew Davis)
Solution: Use the same Paste() function as in menu.vim.
Files: runtime/mswin.vim

Patch 6.3b.007

Problem: Session file doesn't restore view on windows properly. (Robert Webb)
Solution: Restore window sizes both before and after restoring the view, so that the view, cursor position and size are restored properly.
Files: src/ex_docmd.c

Patch 6.3b.008

Problem: Using ":finally" in a user command doesn't always work. (Hari Krishna Dara)
Solution: Don't assume that using getexline() means the command was typed.
Files: src/ex_docmd.c

Patch 6.3b.009 (extra)

Problem: Win32: When the -P argument is not found in a window title, there is no error message.
Solution: When the window can't be found give an error message and exit. Also use try/except to catch failing to open the MDI window. (Michael Wookey)
Files: src/gui_w32.c

Patch 6.3b.010

Problem: Win32: Using the "-D" argument and expanding arguments may cause a hang, because the terminal isn't initialized yet. (Vince Negri)
Solution: Don't go into debug mode before the terminal is initialized.
Files: src/main.c

Patch 6.3b.011

Problem: Using CTRL-\ e while obtaining an expression aborts the command line. (Hari Krishna Dara)
Solution: Insert the CTRL-\ e as typed.
Files: src/ex_getln.c

Patch 6.3b.012 (after 6.3b.010)

Problem: Can't compile with tiny features. (Norbert Tretkowski)
Solution: Add #ifdefs.
Files: src/main.c

Patch 6.3b.013

Problem: Loading a session file results in editing the wrong file in the first window when this is not the file at the current position in the argument list. (Robert Webb)
Solution: Check w_arg_idx_invalid to decide whether to edit a file.
Files: src/ex_docmd.c

Patch 6.3b.014

Problem: ":runtime! foo*.vim" may using freed memory when a sourced script changes the value of 'runtimepath'.
Solution: Make a copy of 'runtimepath' when looping over the matches.
Files: src/ex_cmds2.c

Patch 6.3b.015

Problem: Get lalloc(0) error when using "p" in Visual mode while 'clipboard' contains "autoselect,unnamed". (Mark Wagonner)
Solution: Avoid allocating zero bytes. Obtain the clipboard when necessary.
Files: src/ops.c

Patch 6.3b.016

Problem: When 'virtualedit' is used "x" doesn't delete the last character of a line that has as many characters as 'columns'. (Yakov Lerner)
Solution: When the cursor isn't moved let oneright() return FAIL.
Files: src/edit.c

Patch 6.3b.017

Problem: Win32: "vim --remote-wait" doesn't exit when the server finished editing the file. (David Fishburn)
Solution: In the rrhelper plugin change backslashes to forward slashes and escape special characters.
Files: runtime/plugin/rrhelper.vim

Patch 6.3b.018

Problem: The list of help files in the "local additions" table doesn't recognize utf-8 encoding. (Yasuhiro Matsumoto)
Solution: Recognize utf-8 characters.
Files: src/ex_cmds.c

Patch 6.3b.019

Problem: When \$VIMRUNTIME is not a full path name the "local additions" table lists all the help files.

Solution: Use fullpathcmp() instead of fnamecmp() to compare the directory names.

Files: src/ex_cmds.c

Patch 6.3b.020

Problem: When using **CTRL-^** when entering a search string, the item in the statusline that indicates the keymap is not updated. (Ilya Dogolazky)

Solution: Mark the statuslines for updating.

Files: src/ex_getln.c

Patch 6.3b.021

Problem: The swapfile is not readable for others, the ATTENTION prompt does not show all info when someone else is editing the same file. (Marcel Svitalsky)

Solution: Use the protection of original file for the swapfile and set it after creating the swapfile.

Files: src/fileio.c

Patch 6.3b.022

Problem: Using "4v" to select four times the old Visual area may put the cursor beyond the end of the line. (Jens Paulus)

Solution: Correct the cursor column.

Files: src/normal.c

Patch 6.3b.023

Problem: When "3dip" starts in an empty line, white lines after the non-white lines are not deleted. (Jens Paulus)

Solution: Include the white lines.

Files: src/search.c

Patch 6.3b.024

Problem: "2daw" does not delete leading white space like "daw" does. (Jens Paulus)

Solution: Include the white space when a count is used.

Files: src/search.c

Patch 6.3b.025

Problem: Percentage in ruler isn't updated when a line is deleted. (Jens Paulus)

Solution: Check for a change in line count when deciding to update the ruler.

Files: src/screen.c, src/structs.h

Patch 6.3b.026

Problem: When selecting "abort" at the ATTENTION prompt for a file that is already being edited Vim crashes.

Solution: Don't abort creating a new buffer when we really need it.

Files: src/buffer.c, src/vim.h

Patch 6.3b.027

Problem: Win32: When enabling the menu in a maximized window, Vim uses more lines than what is room for. (Shizhu Pan)
Solution: When deciding to call shell_resized(), also compare the text area size with Rows and Columns, not just with screen_Rows and screen_Columns.
Files: src/gui.c

Patch 6.3b.028

Problem: When in diff mode, setting '**rightleft**' causes a crash. (Eddine)
Solution: Check for last column differently when '**rightleft**' is set.
Files: src/screen.c

Patch 6.3b.029

Problem: Win32: warning for uninitialized variable.
Solution: Initialize to zero.
Files: src/misc1.c

Patch 6.3b.030

Problem: After Visually selecting four characters, changing it to other text, Visually selecting and yanking two characters: "." changes four characters, another "." changes two characters. (Robert Webb)
Solution: Don't store the size of the Visual area when redo is active.
Files: src/normal.c

VERSION 6.4

version-6.4

This section is about improvements made between version 6.3 and 6.4.

This is a bug-fix release. There are also a few new features. The major number of new items is in the runtime files and translations.

The big MS-Windows version now uses:

Ruby version 1.8.3
Perl version 5.8.7
Python version 2.4.2

Changed

changed-6.4

Removed runtime/tools/tcltags, Exuberant ctags does it better.

Added

added-6.4

Alsaconf syntax file (Nikolai Weibull)
Eruby syntax, indent, compiler and ftplugin file (Doug Kearns)
Esterel syntax file (Maurizio Tranchero)
Mathematica indent file (Steve Layland)
Netrc syntax file (Nikolai Weibull)
PHP compiler file (Doug Kearns)
Pascal indent file (Neil Carter)

Prescribe syntax file (Klaus Muth)
Rubyunit compiler file (Doug Kearns)
SMTPrc syntax file (Kornel Kielczewski)
Sudoers syntax file (Nikolai Weibull)
TPP syntax file (Gerfried Fuchs)
VHDL ftplugin file (R. Shankar)
Verilog-AMS syntax file (S. Myles Prather)

Bulgarian keymap (Alberto Mardegan)
Canadian keymap (Eric Joanis)

Hungarian menu translations in UTF-8 (Kantra Gergely)
Ukrainian menu translations (Bohdan Vlasyuk)

Irish message translations (Kevin Patrick Scannell)

Configure also checks for tclsh8.4.

Fixed

fixed-6.4

"dFxd;" deleted the character under the cursor, "d;" didn't remember the exclusiveness of the motion.

When using "set laststatus=2 cmdheight=2" in the .gvimrc you may only get one line for the cmdline. (Christian Robinson) Invoke command_height() after the GUI has started up.

Gcc would warn "dereferencing type-punned pointer will break strict -aliasing rules". Avoid using typecasts for variable pointers.

Gcc 3.x interprets the -MM argument differently. Change "-I /path" to "-isystem /path" for "make depend".

Patch 6.3.001

Problem: ":browse split" gives the file selection dialog twice. (Gordon Bazeley) Same problem for ":browse diffpatch".

Solution: Reset cmdmod.browse before calling do_ecmd().

Files: src/diff.c, src/ex_docmd.c

Patch 6.3.002

Problem: When using translated help files with non-ASCII latin1 characters in the first line the utf-8 detection is wrong.

Solution: Properly detect utf-8 characters. When a mix of encodings is detected continue with the next language and avoid a "no matches" error because of "got_int" being set. Add the directory name to the error message for a duplicate tag.

Files: src/ex_cmds.c

Patch 6.3.003

Problem: Crash when using a console dialog and the first choice does not have a default button. (Darin Ohashi)

Solution: Allocate two more characters for the [] around the character for the default choice.
Files: src/message.c

Patch 6.3.004

Problem: When searching for a long string (140 chars in a 80 column terminal) get three hit-enter prompts. (Robert Webb)
Solution: Avoid the hit-enter prompt when giving the message for wrapping around the end of the buffer. Don't give that message again when the string was not found.
Files: src/message.c, src/search.c

Patch 6.3.005

Problem: Crash when searching for a pattern with a character offset and starting in a closed fold. (Frank Butler)
Solution: Check for the column to be past the end of the line. Also fix that a pattern with a character offset relative to the end isn't read back from the viminfo properly.
Files: src/search.c

Patch 6.3.006

Problem: ":breakadd file *foo" prepends the current directory to the file pattern. (Hari Krishna Dara)
Solution: Keep the pattern as-is.
Files: src/ex_cmds2.c

Patch 6.3.007

Problem: When there is a buffer with 'buftype' set to "nofile" and using a ":cd" command, the swap file is not deleted when exiting.
Solution: Use the full path of the swap file also for "nofile" buffers.
Files: src/fileio.c

Patch 6.3.008

Problem: Compiling fails under OS/2.
Solution: Include "e_screenmode" also for OS/2. (David Sanders)
Files: src/globals.h

Patch 6.3.009 (after 6.3.006)

Problem: ":breakadd file /path/foo.vim" does not match when a symbolic link is involved. (Servatius Brandt)
Solution: Do expand the pattern when it does not start with "*".
Files: runtime/doc/repeat.txt, src/ex_cmds2.c

Patch 6.3.010

Problem: When writing to a named pipe there is an error for fsync() failing.
Solution: Ignore the fsync() error for devices.
Files: src/fileio.c

Patch 6.3.011

Problem: Crash when the completion function of a user-command uses a "normal :cmd" command. (Hari Krishna Dara)
Solution: Save the command line when invoking the completion function.
Files: src/ex_getln.c

Patch 6.3.012

Problem: Internal lalloc(0) error when using a complicated multi-line pattern in a substitute command. (Luc Hermitte)
Solution: Avoid going past the end of a line.
Files: src/ex_cmds.c

Patch 6.3.013

Problem: Crash when editing a command line and typing **CTRL-R** = to evaluate a function that uses "normal :cmd". (Hari Krishna Dara)
Solution: Save and restore the command line when evaluating an expression for **CTRL-R** =.
Files: src/ex_getln.c, src/ops.c, src/proto/ex_getln.pro, src/proto/ops.pro

Patch 6.3.014

Problem: When using Chinese or Taiwanese the default for **'helplang'** is wrong. (Simon Liang)
Solution: Use the part of the locale name after "zh_".
Files: src/option.c

Patch 6.3.015

Problem: The string that winrestcmd() returns may end in garbage.
Solution: NUL-terminate the string. (Walter Briscoe)
Files: src/eval.c

Patch 6.3.016

Problem: The default value for **'define'** has "\s" before '#'.
Solution: Add a star after "\s". (Herculano de Lima Einloft Neto)
Files: src/option.c

Patch 6.3.017

Problem: "8zz" may leave the cursor beyond the end of the line. (Niko Maatjes)
Solution: Correct the cursor column after moving to another line.
Files: src/normal.c

Patch 6.3.018

Problem: ":0argadd zero" added the argument after the first one, instead of before it. (Adri Verhoef)
Solution: Accept a zero range for ":argadd".
Files: src/ex_cmds.h

Patch 6.3.019

Problem: Crash in startup for debug version. (David Rennals)
Solution: Move the call to nbdebug_wait() to after allocating NameBuff.
Files: src/main.c

Patch 6.3.020

Problem: When **'encoding'** is "utf-8" and **'delcombine'** is set, "dw" does not delete a word but only a combining character of the first character, if there is one. (Raphael Finkel)
Solution: Correctly check that one character is being deleted.
Files: src/misc1.c

Patch 6.3.021

Problem: When the last character of a file name is a multibyte character and the last byte is a path separator, the file cannot be edited.

Solution: Check for the last byte to be part of a multibyte character.
(Taro Muraoka)

Files: src/fileio.c

Patch 6.3.022 (extra)

Problem: Win32: When the last character of a file name is a multibyte character and the last byte is a path separator, the file cannot be written. A trail byte that is a space makes that a file cannot be opened from the command line.

Solution: Recognize double-byte characters when parsing the command line. In mch_stat() check for the last byte to be part of a multibyte character. (Taro Muraoka)

Files: src/gui_w48.c, src/os_mswin.c

Patch 6.3.023

Problem: When the "to" part of a mapping starts with its "from" part, abbreviations for the same characters is not possible. For example, when <Space> is mapped to something that starts with a space, typing <Space> does not expand abbreviations.

Solution: Only disable expanding abbreviations when a mapping is not remapped, don't disable it when the RHS of a mapping starts with the LHS.

Files: src/getchar.c, src/vim.h

Patch 6.3.024

Problem: In a few places a string in allocated memory is not terminated with a NUL.

Solution: Add ga_append(NUL) in script_get(), gui_do_findrepl() and serverGetVimNames().

Files: src/ex_getln.c, src/gui.c, src/if_xcmdsrv.c, src/os_mswin.c

Patch 6.3.025 (extra)

Problem: Missing NUL for list of server names.

Solution: Add ga_append(NUL) in serverGetVimNames().

Files: src/os_mswin.c

Patch 6.3.026

Problem: When ~/.vim/after/syntax/syncolor.vim contains a command that reloads the colors an endless loop and/or a crash may occur.

Solution: Only free the old value of an option when it was originally allocated. Limit recursiveness of init_highlight() to 5 levels.

Files: src/option.c, src/syntax.c

Patch 6.3.027

Problem: VMS: Writing a file may insert extra CR characters. Not all terminals are recognized correctly. Vt320 doesn't support colors. Environment variables are not expanded correctly.

Solution: Use another method to write files. Add vt320 termcap codes for colors. (Zoltan Arpadffy)

Files: src/fileio.c, src/misc1.c, src/os_unix.c, src/structs.h,

src/term.c

Patch 6.3.028

Problem: When appending to a file the BOM marker may be written. (Alex Jakushev)

Solution: Do not write the BOM marker when appending.

Files: src/fileio.c

Patch 6.3.029

Problem: Crash when inserting a line break. (Walter Briscoe)

Solution: In the syntax highlighting code, don't use an old state after a change was made, current_col may be past the end of the line.

Files: src/syntax.c

Patch 6.3.030

Problem: GTK 2: Crash when sourcing a script that deletes the menus, sets **'encoding'** to "utf-8" and loads the menus again. GTK error message when tooltip text is in a wrong encoding.

Solution: Don't copy characters from the old screen to the new screen when switching **'encoding'** to utf-8, they may be invalid. Only set the tooltip when it is valid utf-8.

Files: src/gui_gtk.c, src/mbyte.c, src/proto/mbyte.pro, src/screen.c

Patch 6.3.031

Problem: When entering a mapping and pressing Tab halfway the command line isn't redrawn properly. (Adri Verhoef)

Solution: Reposition the cursor after drawing over the "... " of the completion attempt.

Files: src/ex_getln.c

Patch 6.3.032

Problem: Using Python 2.3 with threads doesn't work properly.

Solution: Release the lock after initialization.

Files: src/if_python.c

Patch 6.3.033

Problem: When a mapping ends in a Normal mode command of more than one character Vim doesn't return to Insert mode.

Solution: Check that the mapping has ended after obtaining all characters of the Normal mode command.

Files: src/normal.c

Patch 6.3.034

Problem: VMS: crash when using ":help".

Solution: Avoid using "tags-??", some Open VMS systems can't handle the "?" wildcard. (Zoltan Arpadffy)

Files: src/tag.c

Patch 6.3.035 (extra)

Problem: RISC OS: Compile errors.

Solution: Change e_screnmode to e_screenmode. Change the way __riscosify_control is set. Improve the makefile. (Andy Wingate)

Files: src/os_riscos.c, src/search.c, src/Make_ro.mak

Patch 6.3.036

Problem: ml_get errors when the whole file is a fold, switching 'foldmethod' and doing "zj". (Christian J. Robinson) Was not deleting the fold but creating a fold with zero lines.

Solution: Delete the fold properly.

Files: src/fold.c

Patch 6.3.037 (after 6.3.032)

Problem: Warning for unused variable.

Solution: Change the #ifdefs for the saved thread stuff.

Files: src/if_python.c

Patch 6.3.038 (extra)

Problem: Win32: When the "file changed" dialog pops up after a click that gives gvim focus and not moving the mouse after that, the effect of the click may occur when moving the mouse later. (Ken Clark) Happened because the release event was missed.

Solution: Clear the s_button_pending variable when any input is received.

Files: src/gui_w48.c

Patch 6.3.039

Problem: When 'number' is set and inserting lines just above the first displayed line (in another window on the same buffer), the line numbers are not updated. (Hitier Sylvain)

Solution: When 'number' is set and lines are inserted/deleted redraw all lines below the change.

Files: src/screen.c

Patch 6.3.040

Problem: Error handling does not always work properly and may cause a buffer to be marked as if it's viewed in a window while it isn't. Also when selecting "Abort" at the attention prompt.

Solution: Add enter_cleanup() and leave_cleanup() functions to move saving/restoring things for error handling to one place. Clear a buffer read error when it's unloaded.

Files: src/buffer.c, src/ex_docmd.c, src/ex_eval.c, src/proto/ex_eval.pro, src/structs.h, src/vim.h

Patch 6.3.041 (extra)

Problem: Win32: When the path to a file has Russian characters, ":cd %:p:h" doesn't work. (Valery Kondakoff)

Solution: Use a wide function to change directory.

Files: src/os_mswin.c

Patch 6.3.042

Problem: When there is a closed fold at the top of the window, **CTRL-X** **CTRL-E** in Insert mode reduces the size of the fold instead of scrolling the text up. (Gautam)

Solution: Scroll over the closed fold.

Files: src/move.c

Patch 6.3.043

Problem: 'hlsearch' highlighting sometimes disappears when inserting text in PHP code with syntax highlighting. (Marcel Svitalsky)

Solution: Don't use pointers to remember where a match was found, use an index. The pointers may become invalid when searching in other lines.

Files: src/screen.c

Patch 6.3.044 (extra)

Problem: Mac: When '**linespace**' is non-zero the Insert mode cursor leaves pixels behind. (Richard Sandilands)

Solution: Erase the character cell before drawing the text when needed.

Files: src/gui_mac.c

Patch 6.3.045

Problem: Unusual characters in an option value may cause unexpected behavior, especially for a modeline. (Ciaran McCreesh)

Solution: Don't allow setting termcap options or '**printdevice**' in a modeline. Don't list options for "termcap" and "all" in a modeline. Don't allow unusual characters in '**filetype**', '**syntax**', '**backupext**', '**keymap**', '**patchmode**' and '**langmenu**'.

Files: src/option.c, runtime/doc/options.txt

Patch 6.3.046

Problem: ":registers" doesn't show multibyte characters properly. (Valery Kondakoff)

Solution: Get the length of each character before displaying it.

Files: src/ops.c

Patch 6.3.047 (extra)

Problem: Win32 with Borland C 5.5 on Windows XP: A new file is created with read-only attributes. (Tony Mechelynck)

Solution: Don't use the _wopen() function for Borland.

Files: src/os_win32.c

Patch 6.3.048 (extra)

Problem: Build problems with VMS on IA64.

Solution: Add dependencies to the build file. (Zoltan Arpadffy)

Files: src/Make_vms.mms

Patch 6.3.049 (after 6.3.045)

Problem: Compiler warning for "char" vs "char_u" mixup. (Zoltan Arpadffy)

Solution: Add a typecast.

Files: src/option.c

Patch 6.3.050

Problem: When SIGHUP is received while busy exiting, non-reentrant functions such as free() may cause a crash.

Solution: Ignore SIGHUP when exiting because of an error. (Scott Anderson)

Files: src/misc1.c, src/main.c

Patch 6.3.051

Problem: When '**wildmenu**' is set and completed file names contain multibyte characters Vim may crash.

Solution: Reserve room for multibyte characters. (Yasuhiro Matsumoto)

Files: src/screen.c

Patch 6.3.052 (extra)

Problem: Windows 98: typed keys that are not ASCII may not work properly. For example with a Russian input method. (Jiri Jezdinsky)

Solution: Assume that the characters arrive in the current codepage instead of UCS-2. Perform conversion based on that.

Files: src/gui_w48.c

Patch 6.3.053

Problem: Win32: ":loadview" cannot find a file with non-ASCII characters. (Valerie Kondakoff)

Solution: Use mch_open() instead of open() to open the file.

Files: src/ex_cmds2.c

Patch 6.3.054

Problem: When '**insertmode**' is set <C-L>4ixxx<C-L> hangs Vim. (Jens Paulus) Vim is actually still working but redraw is disabled.

Solution: When stopping Insert mode with **CTRL-L** don't put an Esc in the redo buffer but a **CTRL-L**.

Files: src/edit.c

Patch 6.3.055 (after 6.3.013)

Problem: Can't use getcmdline(), getcmdpos() or setcmdpos() with <C-R>= when editing a command line. Using <C-\>e may crash Vim. (Peter Winters)

Solution: When moving ccline out of the way for recursive use, make it available to the functions that need it. Also save and restore ccline when calling get_expr_line(). Make ccline.cmdbuf NULL at the end of getcmdline().

Files: src/ex_getln.c

Patch 6.3.056

Problem: The last characters of a multibyte file name may not be displayed in the window title.

Solution: Avoid to remove a multibyte character where the last byte looks like a path separator character. (Yasuhiro Matsumoto)

Files: src/buffer.c, src/ex_getln.c

Patch 6.3.057

Problem: When filtering lines folds are not updated. (Carl Osterwisch)

Solution: Update folds for filtered lines.

Files: src/ex_cmds.c

Patch 6.3.058

Problem: When '**foldcolumn**' is equal to the window width and '**wrap**' is on Vim may crash. Disabling the vertical split feature breaks compiling. (Peter Winters)

Solution: Check for zero room for wrapped text. Make compiling without vertical splits possible.

Files: src/move.c, src/quickfix.c, src/screen.c, src/netbeans.c

Patch 6.3.059

Problem: Crash when expanding an ":edit" command containing several spaces with the shell. (Brian Hirt)

Solution: Allocate enough space for the quotes.
Files: src/os_unix.c

Patch 6.3.060

Problem: Using **CTRL-R CTRL-O** in Insert mode with an invalid register name still causes something to be inserted.

Solution: Check the register name for being valid.
Files: src/edit.c

Patch 6.3.061

Problem: When editing a utf-8 file in an utf-8 xterm and there is a multibyte character in the last column, displaying is messed up. (Joël Rio)

Solution: Check for a multibyte character, not a multi-column character.
Files: src/screen.c

Patch 6.3.062

Problem: ":normal! gQ" hangs.

Solution: Quit getcmdline() and do_exmode() when out of typeahead.
Files: src/ex_getln.c, src/ex_docmd.c

Patch 6.3.063

Problem: When a CursorHold autocommand changes to another window (temporarily) **'mousefocus'** stops working.

Solution: Call gui_mouse_correct() after triggering CursorHold.
Files: src/gui.c

Patch 6.3.064

Problem: line2byte(line("\$") + 1) sometimes returns the wrong number. (Charles Campbell)

Solution: Flush the cached line before counting the bytes.
Files: src/memline.c

Patch 6.3.065

Problem: The euro digraph doesn't always work.

Solution: Add an "e=" digraph for Unicode euro character and adjust the help files.

Files: src/digraph.c, runtime/doc/digraph.txt

Patch 6.3.066

Problem: Backup file may get wrong permissions.

Solution: Use permissions of original file for backup file in more places.
Files: src/fileio.c

Patch 6.3.067 (after 6.3.066)

Problem: Newly created file gets execute permission.

Solution: Check for "perm" to be negative before using it.
Files: src/fileio.c

Patch 6.3.068

Problem: When editing a compressed file xxx.gz which is a symbolic link to the actual file a ":write" renames the link.

Solution: Resolve the link, so that the actual file is renamed and compressed.

Files: runtime/plugin/gzip.vim

Patch 6.3.069

Problem: When converting text with illegal characters Vim may crash.

Solution: Avoid that too much is subtracted from the length. (Da Woon Jung)

Files: src/mbyte.c

Patch 6.3.070

Problem: After ":set number linebreak wrap" and a vertical split, moving the vertical separator far left will crash Vim. (Georg Dahn)

Solution: Avoid dividing by zero.

Files: src/charset.c

Patch 6.3.071

Problem: The message for **CTRL-X** mode is still displayed after an error for **'thesaurus'** or **'dictionary'** being empty.

Solution: Clear "edit_submode".

Files: src/edit.c

Patch 6.3.072

Problem: Crash in giving substitute message when language is Chinese and encoding is utf-8. (Yongwei)

Solution: Make the msg_buf size larger when using multibyte.

Files: src/vim.h

Patch 6.3.073

Problem: Win32 GUI: When the Vim window is partly above or below the screen, scrolling causes display errors when the taskbar is not on that side.

Solution: Use the SW_INVALIDATE flag when the Vim window is partly below or above the screen.

Files: src/gui_w48.c

Patch 6.3.074

Problem: When mswin.vim is used and **'insertmode'** is set, typing text in Select mode and then using **CTRL-V** results in **<SNR>99_Pastegi**. (Georg Dahn)

Solution: When restart_edit is set use "d" instead of "c" to remove the selected text to avoid calling edit() twice.

Files: src/normal.c

Patch 6.3.075

Problem: After unloading another buffer, syntax highlighting in the current buffer may be wrong when it uses "containedin". (Eric Arnold)

Solution: Use "buf" instead of "curbuf" in syntax_clear().

Files: src/syntax.c

Patch 6.3.076

Problem: Crash when using cscope and there is a parse error (e.g., line too long). (Alexey I. Froloff)

Solution: Pass the actual number of matches to cs_manage_matches() and correctly handle the error situation.

Files: src/if_cscope.c

Patch 6.3.077 (extra)

Problem: VMS: First character input after ESC was not recognized.
Solution: Added TRM\$M_TM_TIMED in vms_read(). (Zoltan Arpadffy)
Files: src/os_vms.c

Patch 6.3.078 (extra, after 6.3.077)

Problem: VMS: Performance issue after patch 6.3.077
Solution: Add a timeout in the itemlist. (Zoltan Arpadffy)
Files: src/os_vms.c

Patch 6.3.079

Problem: Crash when executing a command in the command line window while syntax highlighting is enabled. (Pero Brbora)
Solution: Don't use a pointer to a buffer that has been deleted.
Files: src/syntax.c

Patch 6.3.080 (extra)

Problem: Win32: With '**encoding**' set to utf-8 while the current codepage is Chinese editing a file with some specific characters in the name fails.
Solution: Use _wfullpath() instead of _fullpath() when necessary.
Files: src/os_mswin.c

Patch 6.3.081

Problem: Unix: glob() may execute a shell command when it's not wanted. (Georgi Guninski)
Solution: Verify the sandbox flag is not set.
Files: src/os_unix.c

Patch 6.3.082 (after 6.3.081)

Problem: Unix: expand() may execute a shell command when it's not wanted. (Georgi Guninski)
Solution: A more generic solution than 6.3.081.
Files: src/os_unix.c

Patch 6.3.083

Problem: VMS: The vt320 termcap entry is incomplete.
Solution: Add missing function keys. (Zoltan Arpadffy)
Files: src/term.c

Patch 6.3.084 (extra)

Problem: Cygwin: compiling with DEBUG doesn't work. Perl path was ignored. Failure when \$(OUTDIR) already exists. "po" makefile is missing.
Solution: Use changes tested in Vim 7. (Tony Mechelynck)
Files: src/Make_cyg.mak, src/po/Make_cyg.mak

Patch 6.3.085

Problem: Crash in syntax highlighting code. (Marc Espie)
Solution: Prevent current_col going past the end of the line.
Files: src/syntax.c

Patch 6.3.086 (extra)

Problem: Can't produce message translation file with msgfmt that checks printf strings.

Solution: Fix the Russian translation.
Files: src/po/ru.po, src/po/ru.cp1251.po

Patch 6.3.087

Problem: MS-DOS: Crash. (Jason Hood)
Solution: Don't call fname_case() with a NULL pointer.
Files: src/ex_cmds.c

Patch 6.3.088

Problem: Editing ".in" causes error E218. (Stefan Karlsson)
Solution: Require some characters before ".in". Same for ".orig" and others.
Files: runtime/filetype.vim

Patch 6.3.089

Problem: A session file doesn't work when created while the current directory contains a space or the directory of the session files contains a space. (Paolo Giarrusso)
Solution: Escape spaces with a backslash.
Files: src/ex_docmd.c

Patch 6.3.090

Problem: A very big value for '**columns**' or '**lines**' may cause a crash.
Solution: Limit the values to 10000 and 1000.
Files: src/option.c

Patch 6.4a.001

Problem: The Unix Makefile contained too many dependencies and a few uncommented lines.
Solution: Run "make depend" with manual changes to avoid a gcc incompatibility. Comment a few lines.
Files: src/Makefile

Patch 6.4b.001

Problem: Vim reports "Vim 6.4a" in the ":version" output.
Solution: Change "a" to "b". (Tony Mechelynck)
Files: src/version.h

Patch 6.4b.002

Problem: In Insert mode, pasting a multibyte character after the end of the line leaves the cursor just before that character.
Solution: Make sure "gP" leaves the cursor in the right place when '**virtualedit**' is set.
Files: src/ops.c

Patch 6.4b.003 (after 6.4b.002)

Problem: The problem still exists when '**encoding**' is set to "cp936".
Solution: Fix the problem in getvcol(), compute the coladd field correctly.
Files: src/charset.c, src/ops.c

Patch 6.4b.004

Problem: Selecting a {} block with "viB" includes the '}' when there is an empty line before it.
Solution: Don't advance the cursor to include a line break when it's already at the line break.

Files: src/search.c

[version7.txt](#) For Vim version 9.1. Last change: 2021 May 17

VIM REFERENCE MANUAL by Bram Moolenaar

[vim7](#) [version-7.0](#) [version7.0](#)

Welcome to Vim 7! A large number of features has been added. This file mentions all the new items, changes to existing features and bug fixes since Vim 6.x. Use this command to see the version you are using:

`:version`

See [vi_diff.txt](#) for an overview of differences between Vi and Vim 7.0.
See [version4.txt](#) for differences between Vim 3.x and Vim 4.x.
See [version5.txt](#) for differences between Vim 4.x and Vim 5.x.
See [version6.txt](#) for differences between Vim 5.x and Vim 6.x.

INCOMPATIBLE CHANGES

[incompatible-7](#)

NEW FEATURES

[new-7](#)

Vim script enhancements
Spell checking
Omni completion
MzScheme interface
Printing multibyte text
Tab pages
Undo branches
Extended Unicode support
More highlighting
Translated manual pages
Internal grep
Scroll back in messages
Cursor past end of the line
POSIX compatibility
Debugger support
Remote file explorer
Define an operator
Mapping to an expression
Visual and Select mode mappings
Location list
Various new items

[new-vim-script](#)
[new-spell](#)
[new-omni-completion](#)
[new-MzScheme](#)
[new-print-multibyte](#)
[new-tab-pages](#)
[new-undo-branches](#)
[new-more-unicode](#)
[new-more-highlighting](#)
[new-manpage-trans](#)
[new-vimgrep](#)
[new-scroll-back](#)
[new-onemore](#)
[new-posix](#)
[new-debug-support](#)
[new-netrw-explore](#)
[new-define-operator](#)
[new-map-expression](#)
[new-map-select](#)
[new-location-list](#)
[new-items-7](#)

IMPROVEMENTS

[improvements-7](#)

COMPILE TIME CHANGES

[compile-changes-7](#)

BUG FIXES

[bug-fixes-7](#)

VERSION 7.1

[version-7.1](#)

Changed

[changed-7.1](#)

Added

[added-7.1](#)

Fixed

[fixed-7.1](#)

VERSION 7.2

[version-7.2](#)

Changed [changed-7.2](#)
Added [added-7.2](#)
Fixed [fixed-7.2](#)

VERSION 7.3 [version-7.3](#)

Persistent undo [new-persistent-undo](#)
More encryption [new-more-encryption](#)
Conceal text [new-conceal](#)
Lua interface [new-lua](#)
Python3 interface [new-python3](#)

Changed [changed-7.3](#)
Added [added-7.3](#)
Fixed [fixed-7.3](#)

VERSION 7.4 [version-7.4](#)

New regexp engine [new-regexp-engine](#)
Better Python interface [better-python-interface](#)
Changed [changed-7.4](#)
Added [added-7.4](#)
Fixed [fixed-7.4](#)

INCOMPATIBLE CHANGES

[incompatible-7](#)

These changes are incompatible with previous releases. Check this list if you run into a problem when upgrading from Vim 6.x to 7.0.

A `":write file"` command no longer resets the `'modified'` flag of the buffer, unless the `+` flag is in `'coptions'` `cpo+`. This was illogical, since the buffer is still modified compared to the original file. And when undoing all changes the file would actually be marked modified. It does mean that `":quit"` fails now.

`":helpgrep"` now uses a help window to display a match.

In an argument list double quotes could be used to include spaces in a file name. This caused a difference between `":edit"` and `":next"` for escaping double quotes and it is incompatible with some versions of Vi.

Command	Vim 6.x file name	Vim 7.x file name
<code>:edit foo\"888</code>	<code>foo"888</code>	<code>foo"888</code>
<code>:next foo\"888</code>	<code>foo888</code>	<code>foo"888</code>
<code>:next a\"b c\"d</code>	<code>ab cd</code>	<code>a"b and c"d</code>

In a `literal-string` a single quote can be doubled to get one.

`":echo 'a''b'"` would result in `"a b"`, but now that two quotes stand for one it results in `"a'b"`.

When overwriting a file with `":w! fname"` there was no warning for when `"fname"` was being edited by another Vim. Vim now gives an error message [E768](#).

The support for Mac OS 9 has been removed.

Files ending in .tex now have **'filetype'** set to "context", "plaintex", or "tex". `ft-tex-plugin`

Minor incompatibilities:

For filetype detection: For many types, use `*./dir/filename` instead of `~/dir/filename`, so that it also works for other user's files.

For quite a few filetypes the indent settings have been moved from the filetype plugin to the indent plugin. If you used:

`:filetype plugin on`

Then some indent settings may be missing. You need to use:

`:filetype plugin indent on`

`":@verbose"` now sets **'verbose'** to zero instead of one.

Removed the old and incomplete "VimBuddy" code.

Buffers without a name report "No Name" instead of "No File". It was confusing for buffers with a name and **'buftype'** set to "nofile".

When `":file xxx"` is used in a buffer without a name, the alternate file name isn't set. This avoids creating buffers without a name, they are not useful.

The "2html.vim" script now converts closed folds to HTML. This means the HTML looks like it's displayed, with the same folds open and closed. Use `"zR"`, or `"let html_ignore_folding=1"`, if no folds should appear in the HTML. (partly by Carl Osterwisch)

Diff mode is now also converted to HTML as it is displayed.

Win32: The effect of the `<F10>` key depended on **'winaltkeys'**. Now it depends on whether `<F10>` has been mapped or not. This allows mapping `<F10>` without changing **'winaltkeys'**.

When **'octal'** is in **'nrformats'** and using **CTRL-A** on `"08"` it became `"018"`, which is illogical. Now it becomes `"9"`. The leading zero(s) is(are) removed to avoid the number becoming octal after incrementing `"009"` to `"010"`.

When **'encoding'** is set to a Unicode encoding, the value for **'fileencodings'** now includes "default" before "latin1". This means that for files with 8-bit encodings the default is to use the encoding specified by the environment, if possible. Previously latin1 would always be used, which is wrong in a non-latin1 environment, such as Russian.

Previously Vim would exit when there are two windows, both of them displaying a help file, and using `":quit"`. Now only the window is closed.

`"-w {scriptout}"` only works when `{scriptout}` doesn't start with a digit. Otherwise it's used to set the **'window'** option.

Previously `<Home>` and `<xHome>` could be mapped separately. This had the disadvantage that all mappings (with modifiers) had to be duplicated, since

you can't be sure what the keyboard generates. Now all `<xHome>` are internally translated to `<Home>`, both for the keys and for mappings. Also for `<xEnd>`, `<xF1>`, etc.

`":put"` now leaves the cursor on the last inserted line.

When a `.gvimrc` file exists then `'compatible'` is off, just like when a `".vimrc"` file exists.

When making a string upper-case with `"vlllU"` or similar then the German sharp `s` is replaced with `"SS"`. This does not happen with `"~"` to avoid backwards compatibility problems and because `"SS"` can't be changed back to a sharp `s`.

`"gd"` previously found the very first occurrence of a variable in a function, that could be the function argument without type. Now it finds the position where the type is given.

The line continuation in functions was not taken into account, line numbers in errors were logical lines, not lines in the sourced file. That made it difficult to locate errors. Now the line number in the sourced file is reported, relative to the function start. This also means that line numbers for `":breakadd func"` are different.

When defining a user command with `:command` the special items could be abbreviated. This caused unexpected behavior, such as `` being recognized as `<line1>`. The items can no longer be abbreviated.

When executing a `FileChangedRO` autocommand it is no longer allowed to switch to another buffer or edit another file. This is to prevent crashes (the event is triggered deep down in the code where changing buffers is not anticipated). It is still possible to reload the buffer.

At the `more-prompt` and the `hit-enter-prompt`, when the `'more'` option is set, the `'k'`, `'u'`, `'g'` and `'b'` keys are now used to scroll back to previous messages. Thus they are no longer used as typeahead.

NEW FEATURES new-7

Vim script enhancements new-vim-script

In Vim scripts the following types have been added:

- | | |
|-------------------------|----------------------------|
| <code>List</code> | ordered list of items |
| <code>Dictionary</code> | associative array of items |
| <code>Funcref</code> | reference to a function |

Many functions and commands have been added to support the new types.

The `string()` function can be used to get a string representation of a variable. Works for Numbers, Strings and composites of them. Then `eval()` can be used to turn the string back into the variable value.

The `:let` command can now use "+=", "-=" and " .=":

```
:let var += expr      " works like :let var = var + expr
:let var -= expr      " works like :let var = var - expr
:let var .= string    " works like :let var = var . string
```

With the `:profile` command you can find out where your function or script is wasting time.

In the Python interface `vim.eval()` also handles Dictionaries and Lists.
`python-eval` (G. Sumner Hayes)

The `getscript` plugin was added as a convenient way to update scripts from www.vim.org automatically. (Charles Campbell)

The `vimball` plugin was added as a convenient way to distribute a set of files for a plugin (plugin file, autoload script, documentation). (Charles Campbell)

Spell checking

[new-spell](#)

Spell checking has been integrated in Vim. There were a few implementations with scripts, but they were slow and/or required an external program.

The `'spell'` option is used to switch spell checking on or off
The `'spelllang'` option is used to specify the accepted language(s)
The `'spellfile'` option specifies where new words are added
The `'spellsuggest'` option specifies the methods used for making suggestions

The `]s` and `[s` commands can be used to move to the next or previous error
The `zg` and `zw` commands can be used to add good and wrong words
The `z=` command can be used to list suggestions and correct the word
The `:mkspell` command is used to generate a Vim spell file from word lists

The "undercurl" highlighting attribute was added to nicely point out spelling mistakes in the GUI (based on patch from Marcin Dalecki).

The "guisp" color can be used to give it a color different from foreground and background.

The number of possible different highlight attributes was raised from about 220 to over 30000. This allows for the attributes of spelling to be combined with syntax highlighting attributes. This is also used for syntax highlighting and marking the Visual area.

Much more info here: [spell](#) .

Omni completion

[new-omni-completion](#)

This could also be called "intellisense", but that is a trademark. It is a smart kind of completion. The text in front of the cursor is inspected to figure out what could be following. This may suggest struct and class members, system functions, etc.

Use **CTRL-X CTRL-O** in Insert mode to start the completion. [i_CTRL-X_CTRL-O](#)

The **'omnifunc'** option is set by filetype plugins to define the function that figures out the completion.

Currently supported languages:

C	ft-c-omni
(X)HTML with CSS	ft-html-omni
JavaScript	ft-javascript-omni
PHP	ft-php-omni
Python	
Ruby	ft-ruby-omni
SQL	ft-sql-omni
XML	ft-xml-omni
any language with syntax highlighting	ft-syntax-omni

You can add your own omni completion scripts.

When the **'completeopt'** option contains "menu" then matches for Insert mode completion are displayed in a (rather primitive) popup menu.

MzScheme interface

[new-MzScheme](#)

The MzScheme interpreter is supported. [MzScheme](#)

The [:mzscheme](#) command can be used to execute MzScheme commands

The [:mzfile](#) command can be used to execute an MzScheme script file

This depends on Vim being compiled with the [+mzscheme](#) feature.

Printing multibyte text

[new-print-multibyte](#)

The [:hardcopy](#) command now supports printing multibyte characters when using PostScript.

The **'printmbcharset'** and **'printmbfont'** options are used for this.

Also see [postscript-cjk-printing](#) . (Mike Williams)

Tab pages

[new-tab-pages](#)

A tab page is a page with one or more windows with a label (aka tab) at the top. By clicking on the label you can quickly switch between the tab pages. And with the keyboard, using the [gt](#) (Goto Tab) command. This is a convenient way to work with many windows.

To start Vim with each file argument in a separate tab page use the [-p](#) argument. The maximum number of pages can be set with **'tabpagemax'**.

The line with tab labels is either made with plain text and highlighting or with a GUI mechanism. The GUI labels look better but are only available on a few systems. The line can be customized with `'tabline'`, `'guitablabel'` and `'guitabtooltip'`. Whether it is displayed is set with `'showtabline'`. Whether to use the GUI labels is set with the "e" flag in `'guioptions'`.

The `:tab` command modifier can be used to have most commands that open a new window open a new tab page instead.

The `--remote-tab` argument can be used to edit a file in a new tab page in an already running Vim server.

Variables starting with "t:" are local to a tab page.

More info here: [tabpage](#)

Most of the GUI stuff was implemented by Yegappan Lakshmanan.

Undo branches

[new-undo-branches](#)

Previously there was only one line of undo-redo. If, after undoing a number of changes, a new change was made all the undone changes were lost. This could lead to accidentally losing work.

Vim now makes an undo branch in this situation. Thus you can go back to the text after any change, even if they were undone. So long as you do not run into `'undolevels'`, when undo information is freed up to limit the memory used.

To be able to navigate the undo branches each change is numbered sequentially. The commands `g-` and `:earlier` go back in time, to older changes. The commands `g+` and `:later` go forward in time, to newer changes.

The changes are also timestamped. Use `":earlier 10m"` to go to the text as it was about ten minutes earlier.

The `:undolist` command can be used to get an idea of which undo branches exist. The `:undo` command now takes an argument to directly jump to a specific position in this list. The `changenr()` function can be used to obtain the change number.

There is no graphical display of the tree with changes, navigation can be quite confusing.

Extended Unicode support

[new-more-unicode](#)

Previously only two combining characters were displayed. The limit is now raised to 6. This can be set with the `'maxcombine'` option. The default is still 2.

`ga` now shows all combining characters, not just the first two.

Previously only 16 bit Unicode characters were supported for displaying. Now the full 32 bit character set can be used. Unless manually disabled at compile time to save a bit of memory.

For pattern matching it is now possible to search for individual composing characters. [patterns-composing](#)

The `8g8` command searches for an illegal UTF-8 byte sequence.

More highlighting

[new-more-highlighting](#)

Highlighting matching parens:

When moving the cursor through the text and it is on a paren, then the matching paren can be highlighted. This uses the new `CursorMoved` autocommand event.

This means some commands are executed every time you move the cursor. If this slows you down too much switch it off with:

`:NoMatchParen`

See [matchparen](#) for more information.

The plugin uses the `:match` command. It now supports three match patterns. The plugin uses the third one. The first one is for the user and the second one can be used by another plugin.

Highlighting the cursor line and column:

The `'cursorline'` and `'cursorcolumn'` options have been added. These highlight the screen line and screen column of the cursor. This makes the cursor position easier to spot. `'cursorcolumn'` is also useful to align text. This may make screen updating quite slow. The `CursorColumn` and `CursorLine` highlight groups allow changing the colors used. [hl-CursorColumn](#)
[hl-CursorLine](#)

The number of possible different highlight attributes was raised from about 220 to over 30000. This allows for the attributes of spelling to be combined with syntax highlighting attributes. This is also used for syntax highlighting, marking the Visual area, `CursorColumn`, etc.

Translated manual pages

[new-manpage-trans](#)

The manual page of Vim and associated programs is now also available in several other languages.

French - translated by David Blanchet
Italian - translated by Antonio Colombo
Russian - translated by Vassily Ragosin

Polish - translated by Mikolaj Machowski

The Unix Makefile installs the Italian manual pages in `.../man/it/man1/`, `.../man/it.ISO8859-1/man1/` and `.../man/it.UTF-8/man1/`. There appears to be no standard for what encoding goes in the "it" directory, the 8-bit encoded file is used there as a best guess.

Other languages are installed in similar places.

The translated pages are not automatically installed when Vim was configured with `--disable-nls`, but `"make install-languages install-tool-languages"` will do it anyway.

Internal grep

new-vimgrep

The `":vimgrep"` command can be used to search for a pattern in a list of files. This is like the `":grep"` command, but no external program is used. Besides better portability, handling of different file encodings and using multi-line patterns, this also allows grepping in compressed and remote files.

`:vimgrep .`

If you want to use the search results in a script you can use the `getqflist()` function.

To grep files in various directories the `"**"` pattern can be used. It expands into an arbitrary depth of directories. `"**"` can be used in all places where file names are expanded, thus also with `:next` and `:args`.

Scroll back in messages

new-scroll-back

When displaying messages, at the `more-prompt` and the `hit-enter-prompt`, The `'k'`, `'u'`, `'g'` and `'b'` keys can be used to scroll back to previous messages. This is especially useful for commands such as `":syntax"`, `":autocommand"` and `":highlight"`. This is implemented in a generic way thus it works for all commands and highlighting is kept. Only works when the `'more'` option is set. Previously it only partly worked for `":clist"`.

The `g<` command can be used to see the last page of messages after you have hit `<Enter>` at the `hit-enter-prompt`. Then you can scroll further back.

Cursor past end of the line

new-onemore

When the `'virtualedit'` option contains `"onemore"` the cursor can move just past the end of the line. As if it's on top of the line break.

This makes some commands more consistent. Previously the cursor was always past the end of the line if the line was empty. But it is far from Vi compatible. It may also break some plugins or Vim scripts. Use with care!

The patch was provided by Mattias Flodin.

POSIX compatibility

new-posix

The POSIX test suite was used to verify POSIX compatibility. A number of problems have been fixed to make Vim more POSIX compatible. Some of them conflict with traditional Vi or expected behavior. The `$VIM_POSIX` environment variable can be set to get POSIX compatibility. See [posix](#).

Items that were fixed for both Vi and POSIX compatibility:

- repeating "R" with a count only overwrites text once; added the 'X' flag to **'coptions'** `cpo-X`
- a vertical movement command that moves to a non-existing line fails; added the '-' flag to **'coptions'** `cpo--`
- when preserving a file and doing ":q!" the file can be recovered; added the '&' flag to **'coptions'** `cpo-&`
- The **'window'** option is partly implemented. It specifies how much **CTRL-F** and **CTRL-B** scroll when there is one window. The `"-w {number}"` argument is now accepted. `"-w {scriptout}"` only works when `{scriptout}` doesn't start with a digit.
- Allow `"-c{command}"` argument, no space between `"-c"` and `{command}`.
- When writing a file with ":w!" don't reset **'readonly'** when 'Z' is present in **'coptions'**.
- Allow 'l' and '#' flags for ":list", ":print" and ":number".
- Added the '.' flag to **'coptions'**: ":cd" fails when the buffer is modified.
- In Ex mode with an empty buffer ":read file" doesn't keep an empty line above or below the new lines.
- Remove a backslash before a NL for the ":global" command.
- When ":append", ":insert" or ":change" is used with ":global", get the inserted lines from the command. Can use backslash-NL to separate lines.
- Can use `":global /pat/ visual"` to execute Normal mode commands at each matched line. Use "Q" to continue and go to the next line.
- The **:open** command has been partially implemented. It stops Ex mode, but redraws the whole screen, not just one line as open mode is supposed to do.
- Support using a pipe to read the output from and write input to an external command. Added the **'shelltemp'** option and `has("filterpipe")`.
- In ex silent mode the ":set" command output is displayed.
- The ":@" and ":*" give an error message when no register was used before.
- The search pattern "[]-`]" matches ']', '^', '_' and ``.
- Autoindent for ":insert" is using the line below the insert.
- Autoindent for ":change" is using the first changed line.
- Editing Ex command lines is not done in cooked mode, because **CTRL-D** and **CTRL-T** cannot be handled then.
- In Ex mode, "1,3" prints three lines. "%" prints all lines.
- In Ex mode "undo" would undo all changes since Ex mode was started.
- Implemented the **'prompt'** option.

Debugger support

new-debug-support

The **'balloonexpr'** option has been added. This is a generic way to implement balloon functionality. You can use it to show info for the word under the

mouse pointer.

Remote file explorer

[new-netrw-explore](#)

The netrw plugin now also supports viewing a directory, when "scp://" is used. Deleting and renaming files is possible.

To avoid duplicating a lot of code, the previous file explorer plugin has been integrated in the netrw plugin. This means browsing local and remote files works the same way.

":browse edit" and ":browse split" use the netrw plugin when it's available and a GUI dialog is not possible.

The netrw plugin is maintained by Charles Campbell.

Define an operator

[new-define-operator](#)

Previously it was not possible to define your own operator; a command that is followed by a {motion}. Vim 7 introduces the '**operatorfunc**' option and the **g@** operator. This makes it possible to define a mapping that works like an operator. The actual work is then done by a function, which is invoked through the **g@** operator.

See [:map-operator](#) for the explanation and an example.

Mapping to an expression

[new-map-expression](#)

The {rhs} argument of a mapping can be an expression. That means the resulting characters can depend on the context. Example:

```
:inoremap <expr> . InsertDot()
```

Here the dot will be mapped to whatever InsertDot() returns.

This also works for abbreviations. See [:map-<expr>](#) for the details.

Visual and Select mode mappings

[new-map-select](#)

Previously Visual mode mappings applied both to Visual and Select mode. With a trick to have the mappings work in Select mode like they would in Visual mode.

Commands have been added to define mappings for Visual and Select mode separately: **:xmap** and **:smap**. With the associated "noremap" and "unmap" commands.

The same is done for menus: **:xmenu**, **:smenu**, etc.

Location list

new-location-list

The support for a per-window quickfix list (location list) is added. The location list can be displayed in a location window (similar to the quickfix window). You can open more than one location list window. A set of commands similar to the quickfix commands are added to browse the location list. (Yegappan Lakshmanan)

Various new items

new-items-7

Normal mode commands:

a", a' and a` New text objects to select quoted strings. a'
i", i' and i` (Taro Muraoka)

CTRL-W <Enter> In the quickfix window: opens a new window to show the location of the error under the cursor.

at and it text objects select a block of text between HTML or XML tags.

<A-LeftMouse> ('mousemodel' "popup" or "popup-setpos")

<A-RightMouse> ('mousemodel' "extend")

Make a blockwise selection. <A-LeftMouse>

gF Start editing the filename under the cursor and jump to the line number following the file name. (Yegappan Lakshmanan)

CTRL-W F Start editing the filename under the cursor in a new window and jump to the line number following the file name. (Yegappan Lakshmanan)

Insert mode commands:

CTRL-\ **CTRL-O** Execute a Normal mode command. Like **CTRL-O** but without moving the cursor. i_CTRL-_CTRL-O

Options:

'balloonexpr' expression for text to show in evaluation balloon
'completefunc' The name of the function used for user-specified Insert mode completion. **CTRL-X CTRL-U** can be used in Insert mode to do any kind of completion. (Taro Muraoka)

'completeopt' Enable popup menu and other settings for Insert mode completion.

'cursorcolumn' highlight column of the cursor

'cursorline' highlight line of the cursor

'formatexpr' expression for formatting text with gq and when text

<code>'formatlistpat'</code>	goes over <code>'textwidth'</code> in Insert mode. pattern to recognize a numbered list for formatting. (idea by Hugo Haas)
<code>'fsync'</code>	Whether <code>fsync()</code> is called after writing a file. (Ciaran McCreesh)
<code>'guitablabel'</code>	expression for text to display in GUI tab page label
<code>'guitabtooltip'</code>	expression for text to display in GUI tab page tooltip
<code>'macatsui'</code>	Mac: use ATSUI text display functions
<code>'maxcombine'</code>	maximum number of combining characters displayed
<code>'maxmempattern'</code>	maximum amount of memory to use for pattern matching
<code>'mkspellmem'</code>	parameters for <code>:mkspell</code> memory use
<code>'mzquantum'</code>	Time in msec to schedule MzScheme threads.
<code>'numberwidth'</code>	Minimal width of the space used for the <code>'number'</code> and <code>'relativenumber'</code> option. (Emmanuel Renieris)
<code>'omnifunc'</code>	The name of the function used for omni completion.
<code>'operatorfunc'</code>	function to be called for <code>g@</code> operator
<code>'printmbcharset'</code>	CJK character set to be used for <code>:hardcopy</code>
<code>'printmbfont'</code>	font names to be used for CJK output of <code>:hardcopy</code>
<code>'pumheight'</code>	maximum number of items to show in the popup menu
<code>'quoteescape'</code>	Characters used to escape quotes inside a string. Used for the <code>a"</code> , <code>a'</code> and <code>a`</code> text objects. <code>a'</code>
<code>'shelltemp'</code>	whether to use a temp file or pipes for shell commands
<code>'showtabline'</code>	whether to show the tab pages line
<code>'spell'</code>	switch spell checking on/off
<code>'spellcapcheck'</code>	pattern to locate the end of a sentence
<code>'spellfile'</code>	file where good and wrong words are added
<code>'spelllang'</code>	languages to check spelling for
<code>'spellsuggest'</code>	methods for spell suggestions
<code>'synmaxcol'</code>	maximum column to look for syntax items; avoids very slow redrawing when there are very long lines
<code>'tabline'</code>	expression for text to display in the tab pages line
<code>'tabpagemax'</code>	maximum number of tab pages to open for <code>-p</code>
<code>'verbosefile'</code>	Log messages in a file.
<code>'wildoptions'</code>	"tagfile" value enables listing the file name of matching tags for CTRL-D command line completion. (based on an idea from Yegappan Lakshmanan)
<code>'winfixwidth'</code>	window with fixed width, similar to <code>'winfixheight'</code>

Ex commands:

Win32: The `":winpos"` command now also works in the console. (Vipin Aravind)

<code>:startreplace</code>	Start Replace mode. (Charles Campbell)
<code>:startgreplace</code>	Start Virtual Replace mode.
<code>:@file</code>	Removes the name of the buffer. (Charles Campbell)
<code>:diffoff</code>	Switch off diff mode in the current window or in all windows.
<code>:delmarks</code>	Delete marks.
<code>:exusage</code>	Help for Ex commands (Nvi command).

`:viusage` Help for Vi commands (Nvi command).

`:sort` Sort lines in the buffer without depending on an external command. (partly by Bryce Wagner)

`:vimgrep` Internal grep command, search for a pattern in files.
`:vimgrepadd` Like `:vimgrep` but don't make a new list.

`:caddfile` Add error messages to an existing quickfix list (Yegappan Lakshmanan).
`:cbuffer` Read error lines from a buffer. (partly by Yegappan Lakshmanan)
`:cgetbuffer` Create a quickfix list from a buffer but don't jump to the first error.
`:caddbuffer` Add errors from the current buffer to the quickfix list.
`:cexpr` Read error messages from a Vim expression (Yegappan Lakshmanan).
`:caddexpr` Add error messages from a Vim expression to an existing quickfix list. (Yegappan Lakshmanan).
`:cgetexpr` Create a quickfix list from a Vim expression, but don't jump to the first error. (Yegappan Lakshmanan).

`:lfile` Like `:cfile` but use the location list.
`:lgetfile` Like `:cgetfile` but use the location list.
`:laddfile` Like `:caddfile` but use the location list.
`:lbuffer` Like `:cbuffer` but use the location list.
`:lgetbuffer` Like `:cgetbuffer` but use the location list.
`:laddbuffer` Like `:caddbuffer` but use the location list.
`:lexpr` Like `:cexpr` but use the location list.
`:lgetexpr` Like `:cgetexpr` but use the location list.
`:laddexpr` Like `:caddexpr` but use the location list.
`:ll` Like `:cc` but use the location list.
`:l|list` Like `:c|list` but use the location list.
`:lnext` Like `:cnext` but use the location list.
`:lprevious` Like `:cprevious` but use the location list.
`:lNext` Like `:cNext` but use the location list.
`:lfirst` Like `:cfirst` but use the location list.
`:lrewind` Like `:crewind` but use the location list.
`:llast` Like `:clast` but use the location list.
`:lnfile` Like `:cnfile` but use the location list.
`:lpfile` Like `:cpfile` but use the location list.
`:lNfile` Like `:cNfile` but use the location list.
`:lolder` Like `:colder` but use the location list.
`:lnewer` Like `:cnewer` but use the location list.
`:lwindow` Like `:cwindow` but use the location list.
`:lopen` Like `:copen` but use the location list.
`:lclose` Like `:cclose` but use the location list.
`:lmake` Like `:make` but use the location list.
`:lgrep` Like `:grep` but use the location list.
`:lgrepadd` Like `:grepadd` but use the location list.
`:lvimgrep` Like `:vimgrep` but use the location list.
`:lvimgrepadd` Like `:vimgrepadd` but use the location list.
`:lhelpgrep` Like `:helpgrep` but use the location list.

<code>:lscope</code>	Like <code>:cscope</code> but use the location list.
<code>:ltag</code>	Jump to a tag and add matching tags to a location list.
<code>:undojoin</code>	Join a change with the previous undo block.
<code>:undolist</code>	List the leaves of the undo tree.
<code>:earlier</code>	Go back in time for changes in the text.
<code>:later</code>	Go forward in time for changes in the text.
<code>:for</code>	Loop over a <code>List</code> .
<code>:endfor</code>	
<code>:lockvar</code>	Lock a variable, prevents it from being changed.
<code>:unlockvar</code>	Unlock a locked variable.
<code>:mkspell</code>	Create a Vim spell file.
<code>:spellgood</code>	Add a word to the list of good words.
<code>:spellwrong</code>	Add a word to the list of bad words
<code>:speldump</code>	Dump list of good words.
<code>:spellinfo</code>	Show information about the spell files used.
<code>:spellrepall</code>	Repeat a spelling correction for the whole buffer.
<code>:spellundo</code>	Remove a word from list of good and bad words.
<code>:mzscheme</code>	Execute MzScheme commands.
<code>:mzfile</code>	Execute an MzScheme script file.
<code>:nbkey</code>	Pass a key to NetBeans for processing.
<code>:profile</code>	Commands for Vim script profiling.
<code>:profdel</code>	Stop profiling for specified items.
<code>:smap</code>	Select mode mapping.
<code>:smapclear</code>	
<code>:snoremap</code>	
<code>:sunmap</code>	
<code>:xmap</code>	Visual mode mapping, not used for Select mode.
<code>:xmapclear</code>	
<code>:xnoremap</code>	
<code>:xunmap</code>	
<code>:smenu</code>	Select mode menu.
<code>:snoremenu</code>	
<code>:sunmenu</code>	
<code>:xmenu</code>	Visual mode menu, not used for Select mode.
<code>:xnoremenu</code>	
<code>:xunmenu</code>	
<code>:tabclose</code>	Close the current tab page.
<code>:tabdo</code>	Perform a command in every tab page.
<code>:tabedit</code>	Edit a file in a new tab page.
<code>:tabnew</code>	Open a new tab page.
<code>:tabfind</code>	Search for a file and open it in a new tab page.

<code>:tabnext</code>	Go to the next tab page.
<code>:tabprevious</code>	Go to the previous tab page.
<code>:tabNext</code>	Go to the previous tab page.
<code>:tabfirst</code>	Go to the first tab page.
<code>:tabrewind</code>	Go to the first tab page.
<code>:tablast</code>	Go to the last tab page.
<code>:tabmove</code>	Move the current tab page elsewhere.
<code>:tabonly</code>	Close all other tab pages.
<code>:tabs</code>	List the tab pages and the windows they contain.

Ex command modifiers:

<code>:keepalt</code>	Do not change the alternate file.
<code>:noautocmd</code>	Do not trigger autocommand events.
<code>:sandbox</code>	Execute a command in the sandbox.
<code>:tab</code>	When opening a new window create a new tab page.

Ex command arguments:

<code>++bad</code>	Specify what happens with characters that can't be converted and illegal bytes. (code example by Yasuhiro Matsumoto) Also, when a conversion error occurs or illegal bytes are found include the line number in the error message.
--------------------	---

New and extended functions:

<code>add()</code>	append an item to a List
<code>append()</code>	append List of lines to the buffer
<code>argv()</code>	without an argument return the whole argument list
<code>browsedir()</code>	dialog to select a directory
<code>bufnr()</code>	takes an extra argument: create buffer
<code>byteidx()</code>	index of a character (Ilya Sher)
<code>call()</code>	call a function with List as arguments
<code>changenr()</code>	number of current change
<code>complete()</code>	set matches for Insert mode completion
<code>complete_add()</code>	add match for 'completefunc'
<code>complete_check()</code>	check for key pressed, for 'completefunc'
<code>copy()</code>	make a shallow copy of a List or Dictionary
<code>count()</code>	count nr of times a value is in a List or Dictionary
<code>cursor()</code>	also accepts an offset for 'virtualedit' , and the first argument can be a list: [lnum, col, off]
<code>deepcopy()</code>	make a full copy of a List or Dictionary
<code>diff_filler()</code>	returns number of filler lines above line {lnum}.
<code>diff_hlID()</code>	returns the highlight ID for diff mode
<code>empty()</code>	check if List or Dictionary is empty
<code>eval()</code>	evaluate {string} and return the result
<code>extend()</code>	append one List to another or add items from one

feedkeys()	Dictionary to another
filter()	put characters in the typeahead buffer
finddir()	remove selected items from a List or Dictionary
findfile()	find a directory in 'path'
foldtextresult()	find a file in 'path' (Johannes Zellner)
function()	the text displayed for a closed fold at line "lnum"
garbagecollect()	make a Funcref out of a function name
get()	cleanup unused Lists and Dictionaries with circular references
getbufline()	get an item from a List or Dictionary
getcldtype()	get a list of lines from a specified buffer (Yegappan Lakshmanan)
getfontname()	return the current command-line type (Yegappan Lakshmanan)
getfperm()	get actual font name being used
getftype()	get file permission string (Nikolai Weibull)
getline()	get type of file (Nikolai Weibull)
getloclist()	with second argument: get List with buffer lines
getpos()	list of location list items (Yegappan Lakshmanan)
getqflist()	return a list with the position of cursor, mark, etc.
getreg()	list of quickfix errors (Yegappan Lakshmanan)
gettabwinvar()	get contents of a register
has_key()	get variable from window in specified tab page.
haslocaldir()	check whether a key appears in a Dictionary
hasmapto()	check if current window used :lcd
index()	check for a mapping to a string
inputlist()	index of item in List
insert()	prompt the user to make a selection from a list
islocked()	insert an item somewhere in a List
items()	check if a variable is locked
join()	get List of Dictionary key-value pairs
keys()	join List items into a String
len()	get List of Dictionary keys
map()	number of items in a List or Dictionary
maparg()	change each List or Dictionary item
mapcheck()	extra argument: use abbreviation
match()	extra argument: use abbreviation
matcharg()	extra argument: count
matchend()	return arguments of :match command
matchlist()	extra argument: count
matchstr()	list with match and submatches of a pattern in a string
max()	extra argument: count
min()	maximum value in a List or Dictionary
mkdir()	minimum value in a List or Dictionary
pathshorten()	create a directory
printf()	reduce directory names to a single character
pumvisible()	format text
range()	check whether the popup menu is displayed
readfile()	generate a List with numbers
reltime()	read a file into a list of lines
reltimestr()	get time value, possibly relative
remove()	turn a time value into a string
repeat()	remove one or more items from a List or Dictionary
	repeat "expr" "count" times (Christophe Poucet)

<code>reverse()</code>	reverse the order of a List
<code>search()</code>	extra argument:
<code>searchdecl()</code>	search for declaration of variable
<code>searchpair()</code>	extra argument: line to stop searching
<code>searchpairpos()</code>	return a List with the position of the match
<code>searchpos()</code>	return a List with the position of the match
<code>setloclist()</code>	modify a location list (Yegappan Lakshmanan)
<code>setpos()</code>	set cursor or mark to a position
<code>setqflist()</code>	modify a quickfix list (Yegappan Lakshmanan)
<code>settabwinvar()</code>	set variable in window of specified tab page
<code>sort()</code>	sort a List
<code>soundfold()</code>	get the sound-a-like equivalent of a word
<code>spellbadword()</code>	get a badly spelled word
<code>spellsuggest()</code>	get suggestions for correct spelling
<code>split()</code>	split a String into a List
<code>str2nr()</code>	convert a string to a number, base 2, 8, 10 or 16
<code>stridx()</code>	extra argument: start position
<code>strridx()</code>	extra argument: start position
<code>string()</code>	string representation of a List or Dictionary
<code>system()</code>	extra argument: filters <code>{input}</code> through a shell command
<code>tabpagebuflist()</code>	List of buffers in a tab page
<code>tabpagenr()</code>	number of current or last tab page
<code>tabpagewinnr()</code>	window number in a tab page
<code>tagfiles()</code>	List with tags file names
<code>taglist()</code>	get list of matching tags (Yegappan Lakshmanan)
<code>tr()</code>	translate characters (Ron Aaron)
<code>uniq()</code>	remove copies of repeated adjacent list items
<code>values()</code>	get List of Dictionary values
<code>winnr()</code>	takes an argument: what window to use
<code>winrestview()</code>	restore the view of the current window
<code>winsaveview()</code>	save the view of the current window
<code>writefile()</code>	write a list of lines into a file

User defined functions can now be loaded automatically from the "autoload" directory in `'runtimepath'`. See [autoload-functions](#) .

New Vim variables:

<code>v:insertmode</code>	used for <code>InsertEnter</code> and <code>InsertChange</code> autocommands
<code>v:val</code>	item value in a <code>map()</code> or <code>filter()</code> function
<code>v:key</code>	item key in a <code>map()</code> or <code>filter()</code> function
<code>v:profiling</code>	non-zero after a <code>":profile start"</code> command
<code>v:fcs_reason</code>	the reason why <code>FileChangedShell</code> was triggered
<code>v:fcs_choice</code>	what should happen after <code>FileChangedShell</code>
<code>v:beval_bufnr</code>	buffer number for <code>'balloonexpr'</code>
<code>v:beval_winnr</code>	window number for <code>'balloonexpr'</code>
<code>v:beval_lnum</code>	line number for <code>'balloonexpr'</code>
<code>v:beval_col</code>	column number for <code>'balloonexpr'</code>
<code>v:beval_text</code>	text under the mouse pointer for <code>'balloonexpr'</code>
<code>v:scrollstart</code>	what caused the screen to be scrolled up
<code>v:swapname</code>	name of the swap file for the <code>SwapExists</code> event
<code>v:swapchoice</code>	what to do for an existing swap file
<code>v:swapcommand</code>	command to be executed after handling <code>SwapExists</code>

`v:char` argument for evaluating `'formatexpr'`

New autocommand events:

<code>ColorScheme</code>	after loading a color scheme
<code>CursorHoldI</code>	the user doesn't press a key for a while in Insert mode
<code>CursorMoved</code>	the cursor was moved in Normal mode
<code>CursorMovedI</code>	the cursor was moved in Insert mode
<code>FileChangedShellPost</code>	after handling a file changed outside of Vim
<code>InsertEnter</code>	starting Insert or Replace mode
<code>InsertChange</code>	going from Insert to Replace mode or back
<code>InsertLeave</code>	leaving Insert or Replace mode
<code>MenuPopup</code>	just before showing popup menu
<code>QuickFixCmdPre</code>	before <code>:make</code> , <code>:grep</code> et al. (Ciaran McCreesh)
<code>QuickFixCmdPost</code>	after <code>:make</code> , <code>:grep</code> et al. (Ciaran McCreesh)
<code>SessionLoadPost</code>	after loading a session file. (Yegappan Lakshmanan)
<code>ShellCmdPost</code>	after executing a shell command
<code>ShellFilterPost</code>	after filtering with a shell command
<code>SourcePre</code>	before sourcing a Vim script
<code>SpellFileMissing</code>	when a spell file can't be found
<code>SwapExists</code>	found existing swap file when editing a file
<code>TabEnter</code>	just after entering a tab page
<code>TabLeave</code>	just before leaving a tab page
<code>VimResized</code>	after the Vim window size changed (Yakov Lerner)

New highlight groups:

<code>Pmenu</code>	Popup menu: normal item <code>hl-Pmenu</code>
<code>PmenuSel</code>	Popup menu: selected item <code>hl-PmenuSel</code>
<code>PmenuThumb</code>	Popup menu: scrollbar <code>hl-PmenuThumb</code>
<code>PmenuSbar</code>	Popup menu: Thumb of the scrollbar <code>hl-PmenuSbar</code>
<code>TabLine</code>	tab pages line, inactive label <code>hl-TabLine</code>
<code>TabLineSel</code>	tab pages line, selected label <code>hl-TabLineSel</code>
<code>TabLineFill</code>	tab pages line, filler <code>hl-TabLineFill</code>
<code>SpellBad</code>	badly spelled word <code>hl-SpellBad</code>
<code>SpellCap</code>	word with wrong caps <code>hl-SpellCap</code>
<code>SpellRare</code>	rare word <code>hl-SpellRare</code>
<code>SpellLocal</code>	word only exists in other region <code>hl-SpellLocal</code>

CursorColumn	' cursorcolumn '	hl-CursorColumn
CursorLine	' cursorline '	hl-CursorLine
MatchParen	matching parens	pi_paren.txt hl-MatchParen

New items in search patterns:

<code>/\%d</code>	<code>\%d123</code>	search for character with decimal number
<code>/\]</code>	<code>[\d123]</code>	idem, in a collection
<code>/\%o</code>	<code>\%o103</code>	search for character with octal number
<code>/\]</code>	<code>[\o103]</code>	idem, in a collection
<code>/\%x</code>	<code>\%x1a</code>	search for character with 2 pos. hex number
<code>/\]</code>	<code>[\x1a]</code>	idem, in a collection
<code>/\%u</code>	<code>\%u12ab</code>	search for character with 4 pos. hex number
<code>/\]</code>	<code>[\u12ab]</code>	idem, in a collection
<code>/\%U</code>	<code>\%U1234abcd</code>	search for character with 8 pos. hex number
<code>/\]</code>	<code>[\U1234abcd]</code>	idem, in a collection

(The above partly by Ciaran McCreesh)

<code>/[[=</code>	<code>[[=a=]</code>	an equivalence class (only for latin1 characters)
<code>/[[.</code>	<code>[[.a.]</code>	a collation element (only works with single char)

<code>/\%'m</code>	<code>\%'m</code>	match at mark m
<code>/\%<'m</code>	<code>\%<'m</code>	match before mark m
<code>/\%>'m</code>	<code>\%>'m</code>	match after mark m
<code>/\%V</code>	<code>\%V</code>	match in Visual area

Nesting `/multi` items no longer is an error when an empty match is possible.

It is now possible to use `\{0}`, it matches the preceding atom zero times. Not useful, just for compatibility.

New Syntax/Indent/FTplugin files:

Moved all the indent settings from the filetype plugin to the indent file. Implemented `b:undo_indent` to undo indent settings when setting '**filetype**' to a different value.

- a2ps syntax and ftplugin file. (Nikolai Weibull)
- ABAB/4 syntax file. (Marius van Wyk)
- alsaconf ftplugin file. (Nikolai Weibull)
- AppendMatchGroup ftplugin file. (Dave Silvia)
- arch ftplugin file. (Nikolai Weibull)
- asterisk and asteriskvm syntax file. (Tilghman Leshner)
- BDF ftplugin file. (Nikolai Weibull)
- BibTeX indent file. (Dorai Sitaram)
- BibTeX Bibliography Style syntax file. (Tim Pope)
- BTM ftplugin file. (Bram Moolenaar)
- calendar ftplugin file. (Nikolai Weibull)
- Changelog indent file. (Nikolai Weibull)
- ChordPro syntax file. (Niels Bo Andersen)
- Cmake indent and syntax file. (Andy Cedilnik)

conf ftplugin file. (Nikolai Weibull)
context syntax and ftplugin file. (Nikolai Weibull)
CRM114 ftplugin file. (Nikolai Weibull)
cvs RC ftplugin file. (Nikolai Weibull)
D indent file. (Jason Mills)
Debian Sources.list syntax file. (Matthijs Mohlmann)
dictconf and dictdconf syntax, indent and ftplugin files. (Nikolai Weibull)
diff ftplugin file. (Bram Moolenaar)
dircolors ftplugin file. (Nikolai Weibull)
django and htmdjango syntax file. (Dave Hodder)
doxygen syntax file. (Michael Geddes)
elinks ftplugin file. (Nikolai Weibull)
eterm ftplugin file. (Nikolai Weibull)
evIEWS syntax file. (Vaidotas Zemlys)
fetchmail RC ftplugin file. (Nikolai Weibull)
FlexWiki syntax and ftplugin file. (George Reilly)
Generic indent file. (Dave Silvia)
gpg ftplugin file. (Nikolai Weibull)
gretl syntax file. (Vaidotas Zemlys)
groovy syntax file. (Alessio Pace)
group syntax and ftplugin file. (Nikolai Weibull)
grub ftplugin file. (Nikolai Weibull)
Haskell ftplugin file. (Nikolai Weibull)
help ftplugin file. (Nikolai Weibull)
indent ftplugin file. (Nikolai Weibull)
Javascript ftplugin file. (Bram Moolenaar)
Kconfig ftplugin and syntax file. (Nikolai Weibull)
ld syntax, indent and ftplugin file. (Nikolai Weibull)
lftp ftplugin file. (Nikolai Weibull)
libao config ftplugin file. (Nikolai Weibull)
limits syntax and ftplugin file. (Nikolai Weibull)
Lisp indent file. (Sergey Khorev)
loginaccess and logindefs syntax and ftplugin file. (Nikolai Weibull)
m4 ftplugin file. (Nikolai Weibull)
mailaliases syntax file. (Nikolai Weibull)
mailcap ftplugin file. (Nikolai Weibull)
manconf syntax and ftplugin file. (Nikolai Weibull)
matlab ftplugin file. (Jake Wasserman)
Maxima syntax file. (Robert Dodier)
MGL syntax file. (Gero Kuhlmann)
modconf ftplugin file. (Nikolai Weibull)
mplayer config ftplugin file. (Nikolai Weibull)
Mrxvtrc syntax and ftplugin file. (Gautam Iyer)
MuPAD source syntax, indent and ftplugin. (Dave Silvia)
mutt RC ftplugin file. (Nikolai Weibull)
nanorc syntax and ftplugin file. (Nikolai Weibull)
netrc ftplugin file. (Nikolai Weibull)
pamconf syntax and ftplugin file. (Nikolai Weibull)
Pascal indent file. (Neil Carter)
passwd syntax and ftplugin file. (Nikolai Weibull)
PHP compiler plugin. (Doug Kearns)
pinfo ftplugin file. (Nikolai Weibull)
plaintex syntax and ftplugin files. (Nikolai Weibull, Benji Fisher)
procmail ftplugin file. (Nikolai Weibull)

prolog ftplugin file. (Nikolai Weibull)
protocols syntax and ftplugin file. (Nikolai Weibull)
quake ftplugin file. (Nikolai Weibull)
racc syntax and ftplugin file. (Nikolai Weibull)
readline ftplugin file. (Nikolai Weibull)
rhelp syntax file. (Johannes Ranke)
rnoweb syntax file. (Johannes Ranke)
Relax NG compact ftplugin file. (Nikolai Weibull)
Scheme indent file. (Sergey Khorev)
screen ftplugin file. (Nikolai Weibull)
sensors syntax and ftplugin file. (Nikolai Weibull)
services syntax and ftplugin file. (Nikolai Weibull)
setserial syntax and ftplugin file. (Nikolai Weibull)
sieve syntax and ftplugin file. (Nikolai Weibull)
SiSU syntax file (Ralph Amissah)
Sive syntax file. (Nikolai Weibull)
slp config, reg and spi syntax and ftplugin files. (Nikolai Weibull)
SML indent file. (Saikat Guha)
SQL anywhere syntax and indent file. (David Fishburn)
SQL indent file.
SQL-Informix syntax file. (Dean L Hill)
SQL: Handling of various variants. (David Fishburn)
sshconfig ftplugin file. (Nikolai Weibull)
Stata and SMCL syntax files. (Jeff Pitblado)
sudoers ftplugin file. (Nikolai Weibull)
sysctl syntax and ftplugin file. (Nikolai Weibull)
terminfo ftplugin file. (Nikolai Weibull)
trustees syntax file. (Nima Talebi)
Vera syntax file. (David Eggum)
udev config, permissions and rules syntax and ftplugin files. (Nikolai Weibull)
updatedb syntax and ftplugin file. (Nikolai Weibull)
VHDL indent file (Gerald Lai)
WSML syntax file. (Thomas Haselwanter)
Xdefaults ftplugin file. (Nikolai Weibull)
XFree86 config ftplugin file. (Nikolai Weibull)
xinetd syntax, indent and ftplugin file. (Nikolai Weibull)
xmodmap ftplugin file. (Nikolai Weibull)
Xquery syntax file. (Jean-Marc Vanel)
xsd (XML schema) indent file.
YAML ftplugin file. (Nikolai Weibull)
Zsh ftplugin file. (Nikolai Weibull)

New Keymaps:

Sinhala (Sri Lanka) (Harshula Jayasuriya)
Tamil in TSCII encoding (Yegappan Lakshmanan)
Greek in cp737 (Panagiotis Louridas)
Polish-slash (HS6_06)
Ukrainian-jcuken (Anatoli Sakhnik)
Kana (Edward L. Fox)

New message translations:

The Ukrainian messages are now also available in cp1251.
Vietnamese message translations and menu. (Phan Vinh Thinh)

Others:

The `:read` command has the `++edit` argument. This means it will use the detected `'fileformat'`, `'fileencoding'` and other options for the buffer. This also fixes the problem that editing a compressed file didn't set these options.

The Netbeans interface was updated for Sun Studio 10. The protocol number goes from 2.2 to 2.3. (Gordon Prieur)

Mac: When starting up Vim will load the `$VIMRUNTIME/macmap.vim` script to define default command-key mappings. (mostly by Benji Fisher)

Mac: Add the selection type to the clipboard, so that Block, line and character selections can be used between two Vims. (Eckehard Berns)
Also fixes the problem that setting `'clipboard'` to "unnamed" breaks using "yyp".

Mac: GUI font selector. (Peter Cucka)

Mac: support for multibyte characters. (Da Woon Jung)
This doesn't always work properly. If you see text drawing problems try switching the `'macatsui'` option off.

Mac: Support the xterm mouse in the non-GUI version.

Mac: better integration with Xcode. Post a fake mouse-up event after the `odoc` event and the drag receive handler to work around a stall after Vim loads a file. Fixed an off-by-one line number error. (Da Woon Jung)

Mac: When started from Finder change directory to the file being edited or the user home directory.

Added the `t_SI` and `t_EI` escape sequences for starting and ending Insert mode. To be used to set the cursor shape to a bar or a block. No default values, they are not supported by `termcap/terminfo`.

GUI font selector for Motif. (Marcin Dalecki)

Nicer toolbar buttons for Motif. (Marcin Dalecki)

Mnemonics for the Motif find/replace dialog. (Marcin Dalecki)

Included a few improvements for Motif from Marcin Dalecki. Draw label contents ourselves to make them handle fonts in a way configurable by Vim and a bit less dependent on the X11 font management.

Autocommands can be defined local to a buffer. This means they will also work when the buffer does not have a name or no specific name. See

`autocmd-buflocal` . (Yakov Lerner)

For xterm most combinations of modifiers with function keys are recognized.
`xterm-modifier-keys`

When **'verbose'** is set the output of `":highlight"` will show where a highlight item was last set.

When **'verbose'** is set the output of the `":map"`, `":abbreviate"`, `":command"`, `":function"` and `":autocmd"` commands will show where it was last defined.
(Yegappan Lakshmanan)

`":function /pattern"` lists functions matching the pattern.

`"lgd"` can be used like `"gd"` but ignores matches in a `{}` block that ends before the cursor position. Likewise for `"lgD"` and `"gD"`.

'scrolljump' can be set to a negative number to scroll a percentage of the window height.

The `v:scrollstart` variable has been added to help find the location in your script that causes the hit-enter prompt.

To make it possible to handle the situation that a file is being edited that is already being edited by another Vim instance, the `SwapExists` event has been added. The `v:swapname`, `v:swapchoice` and `v:swapcommand` variables can be used, for example to use the `client-server` functionality to bring the other Vim to the foreground.

When starting Vim with a `"-t tag"` argument, there is an existing swapfile and the user selects `"quit"` or `"abort"` then exit Vim.

Undo now also restores the `'<` and `'>` marks. `"gv"` selects the same area as before the change and undo.

When editing a search pattern for a `"/` or `"?"` command and **'incsearch'** is set **CTRL-L** can be used to add a character from the current match. **CTRL-R CTRL-W** will add a word, but exclude the part of the word that was already typed.

Ruby interface: add line number methods. (Ryan Paul)

The `$MYVIMRC` environment variable is set to the first found vimrc file.
The `$MYGVIMRC` environment variable is set to the first found gvimrc file.

IMPROVEMENTS

[improvements-7](#)

`":helpgrep"` accepts a language specifier after the pattern: `"pat@it"`.

Moved the help for printing to a separate help file. It's quite a lot now.

When doing completion for `":!cmd"`, `":r !cmd"` or `":w !cmd"` executable files are found in `$PATH` instead of looking for ordinary files in the current directory.

When `":silent"` is used and a backwards range is given for an Ex command the range is swapped automatically instead of asking if that is OK.

The pattern matching code was changed from a recursive function to an iterative mechanism. This avoids out-of-stack errors. State is stored in allocated memory, running out of memory can always be detected. Allows matching more complex things, but Vim may seem to hang while doing that.

Previously some options were always evaluated in the `sandbox`. Now that only happens when the option was set from a modeline or in secure mode. Applies to `'balloonexpr'`, `'foldexpr'`, `'foldtext'` and `'includeexpr'`. (Sumner Hayes)

Some commands and expressions could have nasty side effects, such as using **CTRL-R** = while editing a search pattern and the expression invokes a function that jumps to another window. The `textlock` has been added to prevent this from happening.

`":breakadd here"` and `":breakdel here"` can be used to set or delete a breakpoint at the cursor.

It is now possible to define a function with:
`:exe "func Test()\n ... \n endfunc"`

The tutor was updated to make it simpler to use and text was added to explain a few more important commands. Used ideas from Gabriel Zachmann.

Unix: When `libcall()` fails obtain an error message with `dlerror()` and display it. (Johannes Zellner)

Mac and Cygwin: When editing an existing file make the file name the same case of the edited file. Thus when typing `":e os_UNIX.c"` the file name becomes `"os_unix.c"`.

Added `"nbsp"` in `'listchars'`. (David Blanchet)

Added the `"acwrite"` value for the `'buftype'` option. This is for a buffer that does not have a name that refers to a file and is written with `BufWriteCmd` autocommands.

For lisp indenting and matching parenthesis: (Sergey Khorev)
- square brackets are recognized properly
- `#\()`, `#\)`, `#\[` and `#\]` are recognized as character literals
- Lisp line comments (delimited by semicolon) are recognized

Added the `"count"` argument to `match()`, `matchend()` and `matchstr()`. (Ilya Sher)

`winnr()` takes an optional `"$"` or `"#"` argument. (Nikolai Weibull, Yegappan Lakshmanan)

Added `'s'` flag to `search()`: set `'` mark if cursor moved. (Yegappan Lakshmanan)

Added `'n'` flag to `search()`: don't move the cursor. (Nikolai Weibull)

Added `'c'` flag to `search()`: accept match at the cursor.

Added `'e'` flag to `search()`: move to end of the match. (Benji Fisher)

Added `'p'` flag to `search()`: return number of sub-pattern. (Benji Fisher)

These also apply to `searchpos()`, `searchpair()` and `searchpairpos()`.

The search() and searchpair() functions have an extra argument to specify where to stop searching. Speeds up searches that should not continue too far.

When uncompressing fails in the gzip plugin, give an error message but don't delete the raw text. Helps if the file has a .gz extension but is not actually compressed. (Andrew Pimlott)

When C, C++ or IDL syntax is used, may additionally load doxygen syntax. (Michael Geddes)

Support setting **'filetype'** and **'syntax'** to "aaa.bbb" for "aaa" plus "bbb" filetype or syntax.

The ":registers" command now displays multibyte characters properly.

VMS: In the usage message mention that a slash can be used to make a flag upper case. Add color support to the builtin vt320 terminal codes. (Zoltan Arpadffy)

For the '%' item in **'viminfo'**, allow a number to set a maximum for the number of buffers.

For recognizing the file type: When a file looks like a shell script, check for an "exec" command that starts the tcl interpreter. (suggested by Alexios Zavras)

Support conversion between utf-8 and latin9 (iso-8859-15) internally, so that digraphs still work when iconv is not available.

When a session file is loaded while editing an unnamed, empty buffer that buffer is wiped out. Avoids that there is an unused buffer in the buffer list.

Win32: When libintl.dll supports bind_textdomain_codeset(), use it. (NAKADAIRA Yukihiro)

Win32: Vim was not aware of hard links on NTFS file systems. These are detected now for when **'backupcopy'** is "auto". Also fixed a bogus "file has been changed since reading it" error for links.

When foldtext() finds no text after removing the comment leader, use the second line of the fold. Helps for C-style /* */ comments where the first line is just "/*".

When editing the same file from two systems (e.g., Unix and MS-Windows) there mostly was no warning for an existing swap file, because the name of the edited file differs (e.g., y:\dir\file vs /home/me/dir/file). Added a flag to the swap file to indicate it is in the same directory as the edited file. The used path then doesn't matter and the check for editing the same file is much more reliable.

Unix: When editing a file through a symlink the swap file would use the name of the symlink. Now use the name of the actual file, so that editing the same file twice is detected. (suggestions by Stefano Zacchiroli and James Vega)

Client-server communication now supports **'encoding'**. When setting **'encoding'** in a Vim server to "utf-8", and using "vim --remote fname" in a console, "fname" is converted from the console encoding to utf-8. Also allows Vims with different **'encoding'** settings to exchange messages.

Internal: Changed ga_room into ga_maxlen, so that it doesn't need to be incremented/decremented each time.

When a register is empty it is not stored in the viminfo file.

Removed the tcltags script, it's obsolete.

":redir @*>>" and ":redir @+>>" append to the clipboard. Better check for invalid characters after the register name. **:redir**

":redir => variable" and ":redir =>> variable" write or append to a variable. (Yegappan Lakshmanan) **:redir**

":redir @[a-z]>>" appends to register a to z. (Yegappan Lakshmanan)

The **'verbosefile'** option can be used to log messages in a file. Verbose messages are not displayed then. The "-V{filename}" argument can be used to log startup messages.

":let g:" lists global variables.
":let b:" lists buffer-local variables.
":let w:" lists window-local variables.
":let v:" lists Vim variables.

The stridx() and stridx() functions take a third argument, where to start searching. (Yegappan Lakshmanan)

The getreg() function takes an extra argument to be able to get the expression for the '=' register instead of the result of evaluating it.

The setline() function can take a List argument to set multiple lines. When the line number is just below the last line the line is appended.

g **CTRL-G** also shows the number of characters if it differs from the number of bytes.

Completion for ":debug" and entering an expression for the '=' register. Skip ":" between range and command name. (Peter Winters)

CTRL-Q in Insert mode now works like **CTRL-V** by default. Previously it was ignored.

When "beep" is included in **'debug'** a function or script that causes a beep will result in a message with the source of the error.

When completing buffer names, match with "\(^|[\\/]\\)" instead of "^", so that ":buf stor<Tab>" finds both "include/storage.h" and "storage/main.c".

To count items (pattern matches) without changing the buffer the 'n' flag has been added to `:substitute`. See `count-items`.

In a `:substitute` command the `\u`, `\U`, `\l` and `\L` items now also work for multibyte characters.

The "screen.linux" \$TERM name is recognized to set the default for `'background'` to "dark". (Ciaran McCreesh) Also for "cygwin" and "putty".

The `FileChangedShell` autocommand event can now use the `v:fcs_reason` variable that specifies what triggered the event. `v:fcs_choice` can be used to reload the buffer or ask the user what to do.

Not all modifiers were recognized for xterm function keys. Added the possibility in term codes to end in `;*X` or `0*X`, where X is any character and the * stands for the modifier code. Added the `<xUp>`, `<xDown>`, `<xLeft>` and `<xRight>` keys, to be able to recognize the two forms that xterm can send their codes in and still handle all possible modifiers.

`getwinvar()` now also works to obtain a buffer-local option from the specified window.

Added the `"%s"` item to `'errorformat'`. (Yegappan Lakshmanan)
Added the `"%>"` item to `'errorformat'`.

For `'errorformat'` it was not possible to have a file name that contains the character that follows after `"%f"`. For example, in `"%f:%l:%m"` the file name could not contain `":"`. Now include the first `":"` where the rest of the pattern matches. In the example a `":"` not followed by a line number is included in the file name. (suggested by Emanuele Giaquinta)

GTK GUI: use the GTK file dialog when it's available. Mix from patches by Grahame Bowland and Evan Webb.

Added `":scriptnames"` to `bugreport.vim`, so that we can see what plugins were used.

Win32: If the user changes the setting for the number of lines a scroll wheel click scrolls it is now used immediately. Previously Vim would need to be restarted.

When using `@=` in an expression the value is expression `@=` contains. `":let @= value"` can be used to set the register contents.

A `!` can be added to `":popup"` to have the popup menu appear at the mouse pointer position instead of the text cursor.

The table with encodings has been expanded with many MS-Windows codepages, such as `cp1250` and `cp737`, so that these can also be used on Unix without prepending `"8bit-"`.
When an encoding name starts with `"microsoft-cp"` ignore the `"microsoft-"` part.

Added the `"customlist"` completion argument to a user-defined command. The

user-defined completion function should return the completion candidates as a Vim List and the returned results are not filtered by Vim. (Yegappan Lakshmanan)

Win32: Balloons can have multiple lines if common controls supports it. (Sergey Khorev)

For command-line completion the matches for various types of arguments are now sorted: user commands, variables, syntax names, etc.

When no locale is set, thus using the "C" locale, Vim will work with latin1 characters, using its own isupper()/toupper()/etc. functions.

When using an rxvt terminal emulator guess the value of **'background'** using the COLORFGBG environment variable. (Ciaran McCreesh)

Also support t_SI and t_EI on Unix with normal features. (Ciaran McCreesh)

When **'foldcolumn'** is one then put as much info in it as possible. This allows closing a fold with the mouse by clicking on the '-'.

input() takes an optional completion argument to specify the type of completion supported for the input. (Yegappan Lakshmanan)

"dp" works with more than two buffers in diff mode if there is only one where **'modifiable'** is set.

The **'diffopt'** option has three new values: "horizontal", "vertical" and "foldcolumn".

When the **'include'** option contains \zs the file name found is what is being matched from \zs to the end or \ze. Useful to pass more to **'includeexpr'**.

Loading plugins on startup now supports subdirectories in the plugin directory. [load-plugins](#)

In the foldcolumn always show the '+' for a closed fold, so that it can be opened easily. It may overwrite another character, esp. if **'foldcolumn'** is 1.

It is now possible to get the W10 message again by setting **'readonly'**. Useful in the FileChangedRO autocommand when checking out the file fails.

Unix: When open() returns EFBIG give an appropriate message.

":mksession" sets the SessionLoad variable to notify plugins. A modeline is added to the session file to set **'filetype'** to "vim".

In the ATTENTION prompt put the "Delete it" choice before "Quit" to make it more logical. (Robert Webb)

When appending to a file while the buffer has no name the name of the appended file would be used for the current buffer. But the buffer contents is actually different from the file content. Don't set the file name, unless the 'P' flag is present in **'coptions'**.

When starting to edit a new file and the directory for the file doesn't exist then Vim will report "[New DIRECTORY]" instead of "[New File]" to give the user a hint that something might be wrong.

Win32: Preserve the hidden attribute of the viminfo file.

In Insert mode **CTRL-A** didn't keep the last inserted text when using **CTRL-O** and then a cursor key. Now keep the previously inserted text if nothing is inserted after the **CTRL-O**. Allows using **CTRL-O** commands to move the cursor without losing the last inserted text.

The exists() function now supports checking for autocmd group definition and for supported autocommand events. (Yegappan Lakshmanan)

Allow using ":global" in the sandbox, it doesn't do anything harmful by itself.

":saveas asdf.c" will set 'filetype' to c when it's empty. Also for ":w asdf.c" when it sets the filename for the buffer.

Insert mode completion for whole lines now also searches unloaded buffers.

The colortest.vim script can now be invoked directly with ":source" or ":runtime syntax/colortest.vim".

The 'statusline' option can be local to the window, so that each window can have a different value. (partly by Yegappan Lakshmanan)

The 'statusline' option and other options that support the same format can now use these new features:

- When it starts with "%!" the value is first evaluated as an expression before parsing the value.
- "%#HLname#" can be used to start highlighting with HLname.

When 'statusline' is set to something that causes an error message then it is made empty to avoid an endless redraw loop. Also for other options, such as 'tabline' and 'titlestring'. ":verbose set statusline" will mention that it was set in an error handler.

When there are several matching tags, the ":tag <name>" and **CTRL-]** commands jump to the [count] matching tag. (Yegappan Lakshmanan)

Win32: In the batch files generated by the install program, use \$VIMRUNTIME or \$VIM if it's set. Example provided by Mathias Michaelis. Also create a vimtutor.bat batch file.

The 'balloonexpr' option is now **global-local**.

The system() function now runs in cooked mode, thus can be interrupted by **CTRL-C**.

=====

COMPILE TIME CHANGES compile-changes-7

Dropped the support for the BeOS and Amiga GUI. They were not maintained and probably didn't work. If you want to work on this: get the Vim 6.x version and merge it back in.

When running the tests and one of them fails to produce "test.out" the following tests are still executed. This helps when running out of memory.

When compiling with EXITFREE defined and the ccmalloc library, it is possible to detect memory leaks. Some memory will always be reported as leaked, such as allocated by X11 library functions and the memory allocated in alloc_cmdbuff() to store the ":quit" command.

Moved the code for printing to src/hardcopy.c.

Moved some code from main() to separate functions to make it easier to see what is being done. Using a structure to avoid a lot of arguments to the functions.

Moved unix_expandpath() to misc1.c, so that it can also be used by os_mac.c without copying the code.

--- Mac ---

"make" now creates the Vim.app directory and "make install" copies it to its final destination. (Raf)

Put the runtime directory not directly in Vim.app but in Vim.app/Contents/Resources/vim, so that it's according to Mac specs.

Made it possible to compile with Motif, Athena or GTK without tricks and still being able to use the MacRoman conversion. Added the os_mac_conv.c file.

When running "make install" the runtime files are installed as for Unix. Avoids that too many files are copied. When running "make" a link to the runtime files is created to avoid a recursive copy that takes much time.

Configure will attempt to build Vim for both Intel and PowerPC. The --with-mac-arch configure argument can change it.

--- Win32 ---

The Make_mvc.mak file was adjusted to work with the latest MS compilers, including the free version of Visual Studio 2005. (George Reilly)

INSTALLpc.txt was updated for the recent changes. (George Reilly)

The distributed executable is now produced with the free Visual C++ Toolkit 2003 and other free SDK chunks. msvcsetup.bat was added to support this.

Also generate the .pdb file that can be used to generate a useful crash report on MS-Windows. (George Reilly)

=====

BUG FIXES

bug-fixes-7

When using PostScript printing on MS-DOS the default **'printexpr'** used "lpr" instead of "copy". When **'printdevice'** was empty the copy command did not work. Use "LPT1" then.

The GTK font dialog uses a font size zero when the font name doesn't include a size. Use a default size of 10.

This example in the documentation didn't work:

```
:e `=foo . ".c"``
```

Skip over the expression in ``=expr`` when looking for comments, |, % and #.

When `":helpgrep"` doesn't find anything there is no error message.

"L" and "H" did not take closed folds into account.

Win32: The `"-P title"` argument stopped at the first title that matched, even when it doesn't support MDI.

Mac GUI: **CTRL-^** and **CTRL-@** did not work.

"2daw" on "word." at the end of a line didn't include the preceding white space.

Win32: Using `FindExecutable()` doesn't work to find a program. Use `SearchPath()` instead. For `executable()` use `$PATHEXT` when the program searched for doesn't have an extension.

When **'virtualedit'** is set, moving the cursor up after appending a character may move it to a different column. Was caused by auto-formatting moving the cursor and not putting it back where it was.

When indent was added automatically and then moving the cursor, the indent was not deleted (like when pressing ESC). The "I" flag in **'cptions'** can be used to make it work the old way.

When opening a command-line window, **'textwidth'** gets set to 78 by the Vim filetype plugin. Reset **'textwidth'** to 0 to avoid lines are broken.

After using `cursor(line, col)` moving up/down doesn't keep the same column.

Win32: Borland C before 5.5 requires using ".u." for LowPart and HighPart fields. (Walter Briscoe)

On Sinix `SYS_NMLN` isn't always defined. Define it ourselves. (Cristiano De Michele)

Printing with PostScript may keep the printer waiting for more. Append a **CTRL-D** to the printer output. (Mike Williams)

When converting a string with a hex or octal number the leading '-' was ignored. `":echo '-05' + 0"` resulted in 5 instead of -5.

Using "@" to repeat a command line didn't work when it contains control characters. Also remove "<,>" when in Visual mode to avoid that it appears twice.

When using file completion for a user command, it would not expand environment variables like for a regular command with a file argument.

'cindent': When the argument of a #define looks like a C++ class the next line is indented too much.

When **'comments'** includes multibyte characters inserting the middle part and alignment may go wrong. **'cindent'** also suffers from this for right-aligned items.

Win32: when **'encoding'** is set to "utf-8" getenv() still returns strings in the active codepage. Convert to utf-8. Also for \$HOME.

The default for **'helplang'** was "zh" for both "zh_cn" and "zh_tw". Now use "cn" or "tw" as intended.

When **'bin'** is set and **'eol'** is not set then line2byte() added the line break after the last line while it's not there.

Using foldlevel() in a WinEnter autocommand may not work. Noticed when resizing the GUI shell upon startup.

Python: Using buffer.append(f.readlines()) didn't work. Allow appending a string with a trailing newline. The newline is ignored.

When using the ":saveas f2" command for buffer "f1", the Buffers menu would contain "f2" twice, one of them leading to "f1". Also trigger the BuffFilePre and BuffFilePost events for the alternate buffer that gets the old name.

stridx() did not work well when the needle is empty. (Ciaran McCreesh)

GTK: Avoid a potential hang in gui_mch_wait_for_chars() when input arrives just before it is invoked

VMS: Occasionally CR characters were inserted in the file. Expansion of environment variables was not correct. (Zoltan Arpadffy)

UTF-8: When **'delcombine'** is set "dw" only deleted the last combining character from the first character of the word.

When using ":sball" in an autocommand only the filetype in one buffer was detected. Reset did_filetype in enter_buffer().

When using ":argdo" and the window already was at the first argument index, but not actually editing it, the current buffer would be used instead.

When ":next dir/*" includes many matches, adding the names to the argument list may take an awful lot of time and can't be interrupted. Allow interrupting this.

When editing a file that was already loaded in a buffer, modelines were not used. Now window-local options in the modeline are set. Buffer-local options and global options remain unmodified.

Win32: When **'encoding'** is set to "utf-8" in the vimrc file, files from the command line with non-ASCII characters are not used correctly. Recode the file names when **'encoding'** is set, using the Unicode command line.

Win32 console: When the default for **'encoding'** ends up to be "latin1", the default value of **'isprint'** was wrong.

When an error message is given while waiting for a character (e.g., when an xterm reports the number of colors), the hit-enter prompt overwrote the last line. Don't reset msg_didout in normal_cmd() for K_IGNORE.

Mac GUI: Shift-Tab didn't work.

When defining tooltip text, don't translate terminal codes, since it's not going to be used like a command.

GTK 2: Check the tooltip text for valid utf-8 characters to avoid getting a GTK error. Invalid characters may appear when **'encoding'** is changed.

GTK 2: Add a safety check for invalid utf-8 sequences, they can crash pango.

Win32: When **'encoding'** is changed while starting up, use the Unicode command line to convert the file arguments to **'encoding'**. Both for the GUI and the console version.

Win32 GUI: latin9 text (iso-8859-15) was not displayed correctly, because there is no codepage for latin9. Do our own conversion from latin9 to UCS2.

When two versions of GTK+ 2 are installed it was possible to use the header files from one and the library from the other. Use GTK_LIBDIR to put the directory for the library early in the link flags.

With the GUI find/replace dialog a replace only worked if the pattern was literal text. Now it works for any pattern.

When **'equalalways'** is set and **'eadirection'** is "hor", ":quit" would still cause equalizing window heights in the vertical direction.

When ":emenu" is used in a startup script the command was put in the typeahead buffer, causing a prompt for the crypt key to be messed up.

Mac OS/X: The default for **'isprint'** included characters 128-160, causes problems for Terminal.app.

When a syntax item with "containedin" is used, it may match in the start or end of a region with a matchgroup, while this doesn't happen for a "contains" argument.

When a transparent syntax items matches in another item where the highlighting has already stopped (because of a he= argument), the highlighting would come

back.

When cscope is used to set the quickfix error list, it didn't get set if there was only one match. (Sergey Khorev)

When **'confirm'** is set and using `":bdel"` in a modified buffer, then selecting "cancel", would still give an error message.

The PopUp menu items that started Visual mode didn't work when not in Normal mode. Switching between selecting a word and a line was not possible.

Win32: The keypad decimal point always resulted in a '.', while on some keyboards it's a ','. Use `MapVirtualKey(VK_DECIMAL, 2)`.

Removed unused function `DisplayCompStringOpaque()` from `gui_w32.c`

In Visual mode there is not always an indication whether the line break is selected or not. Highlight the character after the line when the line break is included, e.g., after `"\v$o"`.

GTK: The `<F10>` key can't be mapped, it selects the menu. Disable that with a GTK setting and do select the menu when `<F10>` isn't mapped. (David Necas)

After `"Y" '['` and `']` were not at start/end of the yanked text.

When a telnet connection is dropped Vim preserves files and exits. While doing that a SIGHUP may arrive and disturb us, thus ignore it. (Scott Anderson) Also postpone SIGHUP, SIGQUIT and SIGTERM until it's safe to handle. Added `handle_signal()`.

When completing a file name on the command line backslashes are required for white space. Was only done for a space, not for a Tab.

When configure could not find a terminal library, compiling continued for a long time before reporting the problem. Added a configure check for `tgetent()` being found in a library.

When the cursor is on the first char of the last line a `":g/pat/s///"` command may cause the cursor to be displayed below the text.

Win32: Editing a file with non-ASCII characters doesn't work when **'encoding'** is "utf-8". use `_wfullpath()` instead of `_fullpath()`. (Yu-sung Moon)

When recovering the **'fileformat'** and **'fileencoding'** were taken from the original file instead of from the swapfile. When the file didn't exist, was empty or the option was changed (e.g., with `":e ++fenc=cp123 file"`) it could be wrong. Now store **'fileformat'** and **'fileencoding'** in the swapfile and use the values when recovering.

`":bufdo g/something/p"` overwrites each last printed text line with the file message for the next buffer. Temporarily clear **'shortmess'** to avoid that.

Win32: Cannot edit a file starting with # with `--remote`. Do escape % and # when building the `":drop"` command.

A comment or | just after an expression-backtick argument was not recognized. E.g. in :e `="foo"`comment.

"(" does not stop at an empty sentence (single dot and white space) while ")" does. Also breaks "das" on that dot.

When doing "yy" with the cursor on a TAB the ruler could be wrong and "k" moved the cursor to another column.

When '**commentstring**' is '%" and there is a double quote in the line a double quote before the fold marker isn't removed in the text displayed for a closed fold.

In Visual mode, when '**bin**' and '**eol**' set, g **CTRL-G** counted the last line break, resulting in "selected 202 of 201 bytes".

Motif: fonts were not used for dialog components. (Marcin Dalecki)

Motif: After using a toolbar button the keyboard focus would be on the toolbar (Lesstif problem). (Marcin Dalecki)

When using "y<C-V>`x" where mark x is in the first column, the last line was not included.

Not all test scripts work properly on MS-Windows when checked out from CVS. Use a Vim command to fix all fileformats to dos before executing the tests.

When using ":new" and the file fits in the window, lines could still be above the window. Now remove empty lines instead of keeping the relative position.

Cmdline completion didn't work after ":let var1 var<Tab>".

When using ":startinsert" or ":startreplace" when already in Insert mode (possible when using **CTRL-R** =), pressing Esc would directly restart Insert mode. (Peter Winters)

"2daw" didn't work at end of file if the last word is a single character.

Completion for ":next a'<Tab>" put a backslash before single quote, but it was not removed when editing a file. Now halve backslashes in save_patterns(). Also fix expanding a file name with the shell that contains "\".

When doing "1,6d|put" only "fewer lines" was reported. Now a following "more lines" overwrites the message.

Configure could not handle "-Dfoo=long\ long" in the TCL config output.

When searching backwards, using a pattern that matches a newline and uses \zs after that, didn't find a match. Could also get a hang or end up in the right column in the wrong line.

When \$LANG is "sl" for slovenian, the slovak menu was used, since "slovak" starts with "sl".

When **'paste'** is set in the GUI the Paste toolbar button doesn't work. Clear **'paste'** when starting the GUI.

A message about a wrong viminfo line included the trailing NL.

When **'paste'** is set in the GUI the toolbar button doesn't work in Insert mode. Use `":exe"` in `menu.vim` to avoid duplicating the commands, instead of using a mapping.

Treat `"mlterm"` as an xterm-like terminal. (Seiichi Sato)

`":z.4"` and `":z=4"` didn't work Vi compatible.

When sourcing a file, editing it and sourcing it again, it could appear twice in `":scriptnames"` and get a new `<SID>`, because the inode has changed.

When `$SHELL` is set but empty the **'shell'** option would be empty. Don't use an empty `$SHELL` value.

A command `"w! file"` in `.vimrc` or `$EXINIT` didn't work. Now it writes an empty file.

When a **CTRL-F** command at the end of the file failed, the cursor was still moved to the start of the line. Now it remains where it is.

When using `":s"` or `"&"` to repeat the last substitute and `"$"` was used to put the cursor in the last column, put the cursor in the last column again. This is Vi compatible.

Vim is not fully POSIX compliant but sticks with traditional Vi behavior. Added a few flags in **'coptions'** to behave the POSIX way when wanted. The `$VIM_POSIX` environment variable is checked to set the default.

Appending to a register didn't insert a line break like Vi. Added the `'>'` flag to **'coptions'** for this.

Using `"I"` in a line with only blanks appended to the line. This is not Vi compatible. Added the `'H'` flag in **'coptions'** for this.

When joining multiple lines the cursor would be at the last joint, but Vi leaves it at the position where `"J"` would put it. Added the `'q'` flag in **'coptions'** for this.

Autoindent didn't work for `":insert"` and `":append"`.

Using `":append"` in an empty buffer kept the dummy line. Now it's deleted to be Vi compatible.

When reading commands from a file and `stdout` goes to a terminal, would still request the xterm version. Vim can't read it, thus the output went to the shell and caused trouble there.

When redirecting to a register with an invalid name the redirection would

still be done (after an error message). Now reset "redir_reg". (Yegappan Lakshmanan)

It was not possible to use a NL after a backslash in Ex mode. This is sometimes used to feed multiple lines to a shell command.

When '**cmdheight**' is set to 2 in .vimrc and the GUI uses the number of lines from the terminal we actually get 3 lines for the cmdline in gvim.

When setting \$HOME allocated memory would leak.

Win32: bold characters may sometimes write in another character cell. Use unicodepdy[] as for UTF-8. (Taro Muraoka)

":w fname" didn't work for files with '**buftype**' set to "nofile".

The method used to locate user commands for completion differed from when they are executed. Ambiguous command names were not completed properly.

Incremental search may cause a crash when there is a custom statusline that indirectly invokes ":normal".

Diff mode failed when \$DIFF_OPTIONS was set in the environment. Unset it before invoking "diff".

Completion didn't work after ":argdo", ":windo" and ":bufdo". Also for ":set &l:opt" and ":set &g:opt". (Peter Winters)

When setting '**ttymouse**' to "dec" in an xterm that supports the DEC mouse locator it doesn't work. Now switch off the mouse before selecting another mouse model.

When the CursorHold event is triggered and the commands peek for typed characters the typeahead buffer may be messed up, e.g., when a mouse-up event is received. Avoid invoking the autocommands from the function waiting for a character, let it put K_CURSORHOLD in the input buffer.

Removed the "COUNT" flag from ":argadd", to avoid ":argadd 1*" to be used like ":l:argadd *". Same for ":argdelete" and ":argedit".

Avoid that \$LANG is used for the menus when LC_MESSAGES is "en_US".

Added backslashes before dashes in the vim.1 manual page to make them appear as real dashes. (Pierre Habouzit)

Where "gq" left the cursor depended on the value of '**formatprg**'. Now "gq" always leaves the cursor at the last line of the formatted text.

When editing a compressed file, such as "changelog.Debian.gz" file, filetype detection may try to check the contents of the file while it's still compressed. Skip setting '**filetype**' for compressed files until they have been decompressed. Required for patterns that end in a "*".

Starting with an argument "+cmd" or "-S script" causes the cursor to be moved

to the first line. That breaks a BufReadPost autocommand that uses g`.
Don't move the cursor if it's somewhere past the first line.

"gg=G" while **'modifiable'** is off was uninterruptible.

When **'encoding'** is "sjis" inserting **CTRL-V** u d800 a few times causes a crash.
Don't insert a DBCS character with a NUL second byte.

In Insert mode **CTRL-O** <Home> didn't move the cursor. Made "ins_at_eol" global
and reset it in nv_home().

Wildcard expansion failed: ":w /tmp/\$\$.`echo test`". Don't put quotes around
spaces inside backticks.

After this sequence of commands: Y V p gv: the wrong line is selected. Now
let "gv" select the text that was put, since the original text is deleted.
This should be the most useful thing to do.

":sleep 100u" sleeps for 100 seconds, not 100 usec as one might expect. Give
an error message when the argument isn't recognized.

In gui_mch_draw_string() in gui_w32.c "unibuflen" wasn't static, resulting in
reallocating the buffer every time. (Alexei Alexandrov)

When using a Python "atexit" function it was not invoked when Vim exits. Now
call Py_Finalize() for that. (Ugo Di Girolamo)
This breaks the thread stuff though, fixed by Ugo.

GTK GUI: using a .vimrc with "set cmdheight=2 lines=43" and ":split" right
after startup, the window layout is messed up. (Michael Schaap) Added
win_new_shellsize() call in gui_init() to fix the topframe size.

Trick to get ...MOUSE_NM not used when there are vertical splits. Now pass
column -1 for the left most window and add MOUSE_COLOFF for others. Limits
mouse column to 10000.

searchpair() may hang when the end pattern has "\zs" at the end. Check that
we find the same position again and advance one character.

When in diff mode and making a change that causes the "changed" highlighting
to disappear or reappear, it was still highlighted in another window.

When a ":next" command fails because the user selects "Abort" at the ATTENTION
prompt the argument index was advanced anyway.

When "~" is in **'iskeyword'** the "gd" doesn't work, it's used for the previous
substitute pattern. Put "\V" in the pattern to avoid that.

Use of sprintf() sometimes didn't check properly for buffer overflow. Also
when using msg(). Included code for snprintf() to avoid having to do size
checks where invoking them

":help \=<Tab>" didn't find "sub-replace-\=". Wild menu for help tags didn't
show backslashes. ":he :s\" didn't work.

When reading an errorfile "~/\" in a file name was not expanded.

GTK GUI: When adding a scrollbar (e.g. when using ":vsplit") in a script or removing it the window size may change. GTK sends us resize events when we change the window size ourselves, but they may come at an unexpected moment. Peek for a character to get any window resize events and fix **'columns'** and **'lines'** to undo this.

When using the GTK plug mechanism, resizing and focus was not working properly. (Neil Bird)

After deleting files from the argument list a session file generated with ":mksession" may contain invalid ":next" commands.

When **'shortmess'** is empty and **'keymap'** set to accents, in Insert mode **CTRL-N** may cause the hit-enter prompt. Typing 'a then didn't result in the accented character. Put the character typed at the prompt back in the typeahead buffer so that mapping is done in the right mode.

setbufvar() and setwinvar() did not give error messages.

It was possible to set a variable with an illegal name, e.g. with setbufvar(). It was possible to define a function with illegal name, e.t. ":func F{-1}()"

CTRL-W F and "gf" didn't use the same method to get the file name.

When reporting a conversion error the line number of the last error could be given. Now report the first encountered error.

When using ":e ++enc=name file" and iconv() was used for conversion an error caused a fall-back to no conversion. Now replace a character with '?' and continue.

When opening a new buffer the local value of **'bomb'** was not initialized from the global value.

Win32: When using the "Edit with Vim" entry the file name was limited to about 200 characters.

When using command line completion for ":e *foo" and the file "+foo" exists the resulting command ":e +foo" doesn't work. Now insert a backslash: ":e \+foo".

When the translation of "-- More --" was not 10 characters long the following message would be in the wrong position.

At the more-prompt the last character in the last line wasn't drawn.

When deleting non-existing text while **'virtualedit'** is set the '[' and ']' marks were not set.

Win32: Could not use "**/" in **'path'**, it had to be "**\".

The search pattern "\n" did not match at the end of the last line.

Searching for a pattern backwards, starting on the NUL at the end of the line and **'encoding'** is "utf-8" would match the pattern just before it incorrectly. Affected searchpair('/*', '', '*/').

For the Find/Replace dialog it was possible that not finding the text resulted in an error message while redrawing, which cleared the syntax highlighting while it was being used, resulting in a crash. Now don't clear syntax highlighting, disable it with b_syn_error.

Win32: Combining UTF-8 characters were drawn on the previous character. Could be noticed with a Thai font.

Output of ":function" could leave some of the typed text behind. (Yegappan Lakshmanan)

When the command line history has only a few lines the command line window would be opened with these lines above the first window line.

When using a command line window for search strings ":qa" would result in searching for "qa" instead of quitting all windows.

GUI: When scrolling with the scrollbar and there is a line that doesn't fit redrawing may fail. Make sure w_skipcol is valid before redrawing.

Limit the values of **'columns'** and **'lines'** to avoid an overflow in Rows * Columns. Fixed bad effects when running out of memory (command line would be reversed, ":qa!" resulted in "!:aq").

Motif: "gvim -iconic" opened the window anyway. (David Harrison)

There is a tiny chance that a symlink gets created between checking for an existing file and creating a file. Use the O_NOFOLLOW for open() if it's available.

In an empty line "ix<CTRL-O>0" moved the cursor to after the line instead of sticking to the first column.

When using ":wq" and a BufWriteCmd autocmd uses inputsecret() the text was echoed anyway. Set terminal to raw mode in getcmdline().

Unix: ":w a;b~c" caused an error in expanding wildcards.

When appending to a file with ":w >>fname" in a buffer without a name, causing the buffer to use "fname", the modified flag was reset.

When appending to the current file the "not edited" flag would be reset. ":w" would overwrite the file accidentally.

Unix: When filtering text with an external command Vim would still read input, causing text typed for the command (e.g., a password) to be eaten and echoed. Don't read input when the terminal is in cooked mode.

The Cygwin version of xxd used CR/LF line separators. (Corinna Vinschen)

Unix: When filtering text through a shell command some resulting text may be dropped. Now after detecting that the child has exited try reading some more of its output.

When inside input(), using **CTRL-R** = and the expression throws an exception the command line was not abandoned but it wasn't used either. Now abandon typing the command line.

'delcombine' was also used in Visual and Select mode and for commands like "cl". That was illogical and has been disabled.

When recording while a CursorHold autocommand was defined special keys would appear in the register. Now the CursorHold event is not triggered while recording.

Unix: the src/configure script used \${srcdir-}, not all shells understand that. Use \${srcdir:-.} instead.

When editing file "a" which is a symlink to file "b" that doesn't exist, writing file "a" to create "b" and then ":split b" resulted in two buffers on the same file with two different swapfile names. Now set the inode in the buffer when creating a new file.

When 'esckeys' is not set don't send the xterm code to request the version string, because it may cause trouble in Insert mode.

When evaluating an expression for **CTRL-R** = on the command line it was possible to call a function that opens a new window, resulting in errors for incremental search, and many other nasty things were possible. Now use the `textlock` to disallow changing the buffer or jumping to another window to protect from unexpected behavior. Same for **CTRL-** e.

"d(" deleted the character under the cursor, while the documentation specified an exclusive motion. Vi also doesn't delete the character under the cursor.

Shift-Insert in Insert mode could put the cursor before the last character when it just fits in the window. In coladvance() don't stop at the window edge when filling with spaces and when in Insert mode. In mswin.vim avoid getting a beep from the "l" command.

Win32 GUI: When Alt-F4 is used to close the window and Cancel is selected in the dialog then Vim would insert <M-F4> in the text. Now it's ignored.

When ":silent! {cmd}" caused the swap file dialog, which isn't displayed, there would still be a hit-enter prompt.

Requesting the termresponse (`t_RV`) early may cause problems with "-c" arguments that invoke an external command or even "-c quit". Postpone it until after executing "-c" arguments.

When typing in Insert mode so that a new line is started, using **CTRL-G** u to break undo and start a new change, then joining the lines with <BS> caused

undo info to be missing. Now reset the insertion start point.

Syntax HL: When a region start match has a matchgroup and an offset that happens to be after the end of the line then it continued in the next line and stopped at the region end match, making the region continue after that. Now check for the column being past the end of the line in `syn_add_end_off()`.

When changing a file, setting `'swapfile'` off and then on again, making another change and killing Vim, then some blocks may be missing from the swapfile. When `'swapfile'` is switched back on mark all blocks in the swapfile as dirty. Added `mf_set_dirty()`.

Expanding wildcards in a command like `":e aap;<>!"` didn't work. Put backslashes before characters that are special to the shell. (Adri Verhoef)

A `CursorHold` autocommand would cause a message to be cleared. Don't show the special key for the event for `'showcmd'`.

When expanding a file name for a shell command, as in `"!cmd foo<Tab>"` or `":r !cmd foo<Tab>"` also escape characters that are special for the shell: `"!;&()<>"`.

When the name of the buffer was set by a `":r fname"` command `cpo-f` no autocommands were triggered to notify about the change in the buffer list.

In the quickfix buffer `'bufhidden'` was set to "delete", which caused closing the quickfix window to leave an unlisted "No Name" buffer behind every time.

Win32: when using two screens of different size, setting `'lines'` to a large value didn't fill the whole screen. (SungHyun Nam)

Win32 installer: The generated `_vimrc` contained an absolute path to `diff.exe`. After upgrading it becomes invalid. Now use `$VIMRUNTIME` instead.

The command line was cleared too often when `'showmode'` was set and `":silent normal vy"` was used. Don't clear the command line unless the mode was actually displayed. Added the `"mode_displayed"` variable.

The "load session" toolbar item could not handle a space or other special characters in `v:this_session`.

`":set sta ts=8 sw=4 sts=2"` deleted 4 spaces halfway a line instead of 2.

In a multibyte file the foldmarker could be recognized in the trail byte. (Taro Muraoka)

Pasting with **CTRL-V** and menu didn't work properly when some commands are mapped. Use `":normal!"` instead of `":normal"`. (Tony Apuzzo)

Crashed when expanding a file name argument in backticks.

In some situations the menu and scrollbar didn't work, when the value contains a CSI byte. (Yukihiro Nakadaira)

GTK GUI: When drawing the balloon focus changes and we might get a key release event that removed the balloon again. Ignore the key release event.

'**titleold**' was included in ":mkexrc" and ":mksession" files.

":set background&" didn't use the same logic as was used when starting up.

When "umask" is set such that nothing is writable then the viminfo file would be written without write permission. (Julian Bridle)

Motif: In diff mode dragging one scrollbar didn't update the scrollbar of the other diff'ed window.

When editing in an xterm with a different number of colors than expected the screen would be cleared and redrawn, causing the message about the edited file to be cleared. Now set "keep_msg" to redraw the last message.

For a color terminal: When the Normal HL uses bold, possibly to make the color lighter, and another HL group specifies a color it might become light as well. Now reset bold if a HL group doesn't specify bold itself.

When using 256 color xterm the color 255 would show up as color 0. Use a short instead of a char to store the color number.

ml_get errors when searching for "\n\zs" in an empty file.

When selecting a block and using "\$" to select until the end of every line and not highlighting the character under the cursor the first character of the block could be unhighlighted.

When counting words for the Visual block area and using "\$" to select until the end of every line only up to the length of the last line was counted.

"dip" in trailing empty lines left one empty line behind.

The script ID was only remembered globally for each option. When a buffer- or window-local option was set the same "last set" location was changed for all buffers and windows. Now remember the script ID for each local option separately.

GUI: The "Replace All" button didn't handle backslashes in the replacement in the same way as "Replace". Escape backslashes so that they are taken literally.

When using Select mode from Insert mode and typing a key, causing lines to be deleted and a message displayed, delayed the effect of inserting the key. Now overwrite the message without delay.

When '**whichwrap**' includes "l" then "dl" and "yl" on a single letter line worked differently. Now recognize all operators when using "l" at the end of a line.

GTK GUI: when the font selector returned a font name with a comma in it then it would be handled like two font names. Now put a backslash before the

comma.

MS-DOS, Win32: When **'encoding'** defaults to "latin1" then the value for **'iskeyword'** was still for CPxxx. And when **'nocompatible'** was set **'isprint'** would also be the wrong value.

When a command was defined not to take arguments and no '|' no warning message would be given for using a '|'. Also with ":loadkeymap".

Motif: When using a fontset and **'encoding'** is "utf-8" and sizeof(wchar_t) != sizeof(XChar2b) then display was wrong. (Yukihiro Nakadaira)

":all" always set the current window to the first window, even when it contains a buffer that is not in the argument list (can't be closed because it is modified). Now go to the window that has the first item of the argument list.

GUI: To avoid left-over pixels from bold text all characters after a character with special attributes were redrawn. Now only do this for characters that actually are bold. Speeds up displaying considerably.

When only highlighting changes and the text is scrolled at the same time everything is redrawn instead of using a scroll and updating the changed text. E.g., when using ":match" to highlight a paren that the cursor landed on. Added SOME_VALID: Redraw the whole window but also try to scroll to minimize redrawing.

Win32: When using Korean IME making it active didn't work properly. (Moon, Yu-sung, 2005 March 21)

Ruby interface: when inserting/deleting lines display wasn't updated. (Ryan Paul)

--- fixes since Vim 7.0b ---

Getting the GCC version in configure didn't work with Solaris sed. First strip any "darwin." and then get the version number.

The "autoload" directory was missing from the self-installing executable for MS-Windows.

The MS-Windows install program would find "vimtutor.bat" in the install directory. After changing to "c:" also change to "\" to avoid looking in the install directory.

To make the 16 bit DOS version compile exclude not used highlight initializations and build a tiny instead of small version.

finddir() and findfile() accept a negative count and return a List then.

The Python indent file contained a few debugging statements, removed.

Expanding {} for a function name, resulting in a name starting with "s:" was not handled correctly.

Spelling: renamed COMPOUNDMAX to COMPOUNDWORDMAX. Added several items to be able to handle the new Hungarian dictionary.

Mac: Default to building for the current platform only, that is much faster than building a universal binary. Also, using Perl/Python/etc. only works for the current platform.

The time on undo messages disappeared for someone. Using %T for strftime() apparently doesn't work everywhere. Use %H:%M:%S instead.

Typing BS at the "z=" prompt removed the prompt.

--- fixes and changes since Vim 7.0c ---

When jumping to another tab page the Vim window size was always set, even when nothing in the layout changed.

Win32 GUI tab pages line wasn't always enabled. Do a proper check for the compiler version.

Win32: When switching between tab pages the Vim window was moved when part of it was outside of the screen. Now only do that in the direction of a size change.

Win32: added menu to GUI tab pages line. (Yegappan Lakshmanan)

Mac: Added document icons. (Benji Fisher)

Insert mode completion: Using Enter to accept the current match causes confusion. Use **CTRL-Y** instead. Also, use **CTRL-E** to go back to the typed text.

GUI: When there are left and right scrollbars, ":tabedit" kept them instead of using the one that isn't needed.

Using "gP" to replace all the text could leave the cursor below the last line, causing ml_get errors.

When '**cursorline**' is set don't use the highlighting when Visual mode is active, otherwise it's difficult to see the selected area.

The matchparen plugin restricts the search to 100 lines, to avoid a long delay when there are closed folds.

Sometimes using **CTRL-X s** to list spelling suggestions used text from another line.

Win32: Set the default for '**isprint**' back to the wrong default "@,~-255", because many people use Windows-1252 while '**encoding**' is "latin1".

GTK: Added a workaround for gvim crashing when used over an untrusted ssh link, caused by GTK doing something nasty. (Ed Catmur)

Win32: The font used for the tab page labels is too big. Use the system menu font. (George Reilly)

Win32: Adjusting the window position and size to keep it on the screen didn't work properly when the taskbar is on the left or top of the screen.

The installman.sh and installml.sh scripts use `${10}`, that didn't work with old shells. And use `"test -f"` instead of `"test -e"`.

Win32: When **'encoding'** was set in the vimrc then a directory argument for diff mode didn't work.

GUI: at the inputlist() prompt the cursorshape was adjusted as if the windows were still at their old position.

The parenmatch plugin didn't remember the highlighting per window.

Using `":bd"` for a buffer that's the current window in another tab page caused a crash.

For a new tab page the **'scroll'** option wasn't set to a good default.

Using an end offset for a search `"/pat/e"` didn't work properly for multibyte text. (Yukihiro Nakadaira)

`":s/\n/,/"` doubled the text when used on the last line.

When `"search"` is in **'foldopen'** `"[s"` and `"]s"` now open folds.

When using a numbered function `"dict"` can be omitted, but `"self"` didn't work then. Always add `FC_DICT` to the function flags when it's part of a dictionary.

When `"--remote-tab"` executes locally it left an empty tab page.

`"gvim -u NONE"`, `":set cursorcolumn"`, `"C"` in the second line didn't update text. Do update further lines even though the `"$"` is displayed.

VMS: Support GTK better, also enable `+clientserver`. (Zoltan Arpadffy)

When highlighting of statusline or tabline is changed there was no redraw to show the effect.

Mac: Added `"CFBundleIdentifier"` to `infplist.xml`.

Added `tabpage-local` variables `t:var`.

Win32: Added double-click in tab pages line creates new tab. (Yegappan Lakshmanan)

Motif: Added GUI tab pages line. (Yegappan Lakshmanan)

Fixed crash when **'lines'** was set to 1000 in a modeline.

When `init_spellfile()` finds a writable directory in `'runtimepath'` but it doesn't contain a "spell" directory, create one.

Win32: `executable()` also finds "xxd" in the directory where Vim was started, but "!xxd" doesn't work. Append the Vim starting directory to \$PATH.

The tab page labels are shortened, directory names are reduced to a single letter by default. Added the `pathshorten()` function to allow a user to do the same.

`":saveas"` now resets `'readonly'` if the file was successfully written.

Set \$MYVIMRC file to the first found .vimrc file.
Set \$MYGVIMRC file to the first found .gvimrc file.
Added menu item "Startup Settings" that edits the \$MYVIMRC file

Added `matcharg()`.

Error message E745 appeared twice. Renamed one to E786.

Fixed crash when using "au BufRead * Sexplore" and doing `":help"`. Was wiping out a buffer that's still in a window.

`":hardcopy"` resulted in an error message when `'encoding'` is "utf-8" and `'printencoding'` is empty. Now it assumes latin1. (Mike Williams)

The check for the toolbar feature for Motif, depending on certain included files, wasn't detailed enough, causing building to fail in `gui_xmew.c`.

Using **CTRL-E** in Insert mode completion after **CTRL-P** inserted the first match instead of the original text.

When displaying a UTF-8 character with a zero lower byte Vim might think the previous character is double-wide.

The "nbsp" item of `'listchars'` didn't work when `'encoding'` was utf-8.

Motif: when `Xm/xpm.h` is missing `gui_xmew.c` would not compile.
`HAVE_XM_UNHIGHLIGHTT_H` was missing a T.

Mac: Moved the .icns files into `src/os_mac_rsrc`, so that they can all be copied at once. Adjusted the `Info.plist` file for three icons.

When Visual mode is active while switching to another tabpage could get `ml_get` errors.

When `'list'` is set, `'nowrap'` the \$ in the first column caused `'cursorcolumn'` to move to the right.

When a line wraps, `'cursorcolumn'` was never displayed past the end of the line.

`'autochdir'` was only available when compiled with NetBeans and GUI. Now it's a separate feature, also available in the "big" version.

Added **CTRL-W** gf: open file under cursor in new tab page.

When using the menu in the tab pages line, "New Tab" opens the new tab before where the click was. Beyond the labels the new tab appears at the end instead of after the current tab page.

Inside a mapping with an expression getchar() could not be used.

When vgetc is used recursively vgetc_busy protects it from being used recursively. But after a ":normal" command the protection was reset.

":s/a/b/n" didn't work when '**modifiable**' was off.

When \$VIMRUNTIME includes a multibyte character then rgb.txt could not be found. (Yukihiro Nakadaira)

":mkspell" didn't work correctly for non-ASCII affix flags when conversion is needed on the spell file.

glob('/dir/\\$ABC/*') didn't work.

When using several tab pages and changing '**cmdheight**' the display could become messed up. Now store the value of '**cmdheight**' separately for each tab page.

The user of the Enter key while the popup menu is visible was still confusing. Now use Enter to select the match after using a cursor key.

Added "usetab" to '**switchbuf**'.

--- fixes and changes since Vim 7.0d ---

Added **CTRL-W** T: move a window to a new tab page.

Using **CTRL-X** s in Insert mode to complete spelling suggestions and using BS deleted characters before the bad word.

A few small fixes for the VMS makefile. (Zoltan Arpadffy)

With a window of 91 lines 45 cols, ":vsp" scrolled the window. Copy w_wrow when splitting a window and skip setting the height when it's already at the right value.

Using <silent> in a mapping with a shell command and the GUI caused redraw to use wrong attributes.

Win32: Using MSVC 4.1 for install.exe resulted in the start menu items to be created in the administrator directory instead of "All Users". Define the CSIDL_ items if they are missing.

Motif: The GUI tabline did not use the space above the right scrollbar. Work around a bug in the Motif library. (Yegappan Lakshmanan)

The extra files for XML Omni completion are now also installed.

`xml-omni-datafile`

GTK GUI: when 'm' is missing from **'guioptions'** during startup and pressing <F10> GTK produced error messages. Now do create the menu but disable it just after the first `gui_mch_update()`.

":mkspell" doesn't work well with the Hungarian dictionary from the Hunspell project. Back to the Myspell dictionary.

In help files hide the | used around tags.

Renamed `pycomplete` to `pythoncomplete`.

Added "tabpages" to **'sessionoptions'**.

When **'guitablabel'** is set the effect wasn't visible right away.

Fixed a few **'cindent'** errors.

When completing menu names, e.g., after ":", don't sort the entries but keep them in the original order.

Fixed a crash when editing a directory in diff mode. Don't trigger autocommands when executing the diff command.

Getting a keystroke could get stuck if **'encoding'** is a multibyte encoding and typing a special key.

When **'foldignore'** is set the folds were not updated right away.

When a list is indexed with [a : b] and b was greater than the length an error message was given. Now silently truncate the result.

When using BS during Insert mode completion go back to the original text, so that **CTRL-N** selects the first matching entry.

Added the 'M' flag to **'cinoptions'**.

Win32: Make the "gvim --help" window appear in the middle of the screen instead of at an arbitrary position. (Randall W. Morris)

Added `gettabwinvar()` and `settabwinvar()`.

Command line completion: pressing <Tab> after ":", expands the whole tree, because it becomes ":". Don't add a star if there already is one.

Added `grey10` to `grey90` to all GUIs, so that they can all be used for initializing highlighting. Use `grey40` for `CursorColumn` and `CursorLine` when **'background'** is "dark".

When reading a file and using `iconv` for conversion, an incomplete byte sequence at the end caused problems. (Yukihiro Nakadaira)

--- fixes and changes since Vim 7.0e ---

Default color for MatchParen when **'background'** is "dark" is now DarkCyan.

":syn off" had to be used twice in a file that sets **'syntax'** in a modeline.
(Michael Geddes)

When using ":vsp" or ":sp" the available space wasn't used equally between windows. (Servatius Brandt)

Expanding <cWORD> on a trailing blank resulted in the first word in the line if **'encoding'** is a multibyte encoding.

Spell checking: spellbadword() didn't see a missing capital in the first word of a line. Popup menu now only suggest the capitalized word when appropriate.

When using whole line completion **CTRL-L** moves through the matches but it didn't work when at the original text.

When completion finds the longest match, don't go to the first match but stick at the original text, so that **CTRL-N** selects the first one.

Recognize "zsh-beta" like "zsh" for setting the **'shellpipe'** default. (James Vega)

When using ":map <expr>" and the expression results in something with a special byte (NUL or CSI) then it didn't work properly. Now escape special bytes.

The default Visual highlighting for a color xterm with 8 colors was a magenta background, which made magenta text disappear. Now use reverse in this specific situation.

After completing the longest match "." didn't insert the same text. Repeating also didn't work correctly for multibyte text.

When using Insert mode completion and BS the whole word that was completed would result in all possible matches. Now stop completion. Also fixes that for spell completion the previous word was deleted.

GTK: When **'encoding'** is "latin1" and using non-ASCII characters in a file name the tab page label was wrong and an error message would be given.

The taglist() function could hang on a tags line with a non-ASCII character.

Win32: When **'encoding'** differs from the system encoding tab page labels with non-ASCII characters looked wrong. (Yegappan Lakshmanan)

Motif: building failed when Xm/Notebook.h doesn't exist. Added a configure check, disable GUI tabline when it's missing.

Mac: When compiled without multibyte feature the clipboard didn't work.

It was possible to switch to another tab page when the cmdline window is open.

Completion could hang when **'lines'** is 6 and a preview window was opened.

Added **CTRL-W gF**: open file under cursor in new tab page and jump to the line number following the file name.

Added **'guitabtooltip'**. Implemented for Win32 (Yegappan Lakshmanan).

Added "throw" to **'debug'** option: throw an exception for error messages even when they would otherwise be ignored.

When **'keymap'** is set and a line contains an invalid entry could get a "No mapping found" warning instead of a proper error message.

Motif: default to using XpmAttributes instead of XpmAttributes_21.

A few more changes for 64 bit MS-Windows. (George Reilly)

Got ml_get errors when doing "o" and selecting in other window where there are less lines shorter than the cursor position in the other window. ins_mouse() was using position in wrong window.

Win32 GUI: Crash when giving a lot of messages during startup. Allocate twice as much memory for the dialog template.

Fixed a few leaks and wrong pointer use reported by coverity.

When showing menus the mode character was sometimes wrong.

Added feedkeys(). (Yakov Lerner)

Made matchlist() always return all submatches.

Moved triggering QuickFixCmdPost to before jumping to the first location.

Mac: Added the **'macatsui'** option as a temporary work around for text drawing problems.

Line completion on **"/**"** gave error messages when scanning an unloaded buffer.

--- fixes and changes since Vim 7.0f ---

Win32: The height of the tab page labels is now adjusted to the font height. (Yegappan Lakshmanan)

Win32: selecting the tab label was off by one. (Yegappan Lakshmanan)

Added tooltips for Motif and GTK tab page labels. (Yegappan Lakshmanan)

When **'encoding'** is "utf-8" then **":help spell"** would report an illegal byte and the file was not converted from latin1 to utf-8. Now retry with latin1 if reading the file as utf-8 results in illegal bytes.

Escape the argument of `feedkeys()` before putting it in the typeahead buffer. (Yukihiro Nakadaira)

Added the `v:char` variable for evaluating `'formatexpr'`. (Yukihiro Nakadaira)

With 8 colors Search highlighting combined with Statement highlighted text made the text disappear.

VMS: avoid warnings for redefining MAX and MIN. (Zoltan Arpadffy)

When `'virtualedit'` includes "onemore", stopping Visual selection would still move the cursor left.

Prevent that using `CTRL-R` = in Insert mode can start Visual mode.

Fixed a crash that occurred when in Insert mode with completion active and a mapping caused `edit()` to be called recursively.

When using `CTRL-O` in Insert mode just after the last character while `'virtualedit'` is "all", then typing CR moved the last character to the next line. Call `coladvance()` before starting the new line.

When using `:shell` ignore clicks on the tab page labels. Also when using the command line window.

When `'eventignore'` is "all" then adding more to ignoring some events, e.g., for `":vimgrep"`, would actually trigger more events.

Win32: When a running Vim uses server name GVIM1 then `"gvim --remote fname"` didn't find it. When looking for a server name that doesn't end in a digit and it is not found then use another server with that name and a number (just like on Unix).

When using "double" in `'spellsuggest'` when the language doesn't support sound folding resulted in too many suggestions.

Win32: Dropping a shortcut on the Vim icon didn't edit the referred file like editing it in another way would. Use `fname_expand()` in `buf_set_name()` instead of simply make the file name a full path.

Using `feedkeys()` could cause Vim to hang.

When closing another tab page from the tabline menu in Insert mode the tabline was not updated right away.

The syntax menu didn't work in compatible mode.

After using `":tag id"` twice with the same "id", `":ts"` and then `":pop"` a `":ts"` reported no matching tag. Clear the cached tag name.

In Insert mode the matchparen plugin highlighted the wrong paren when there is a string just next to a paren.

GTK: After opening a new tab page the text was sometimes not drawn correctly.

Flush output and catch up with events when updating the tab page labels.

In the GUI, using **CTRL-W** q to close the last window of a tab page could cause a crash.

GTK: The tab pages line menu was not converted from **'encoding'** to utf-8.

Typing a multibyte character or a special key at the hit-enter prompt did not work.

When **'virtualedit'** contains "onemore" **CTRL-O** in Insert mode still moved the cursor left when it was after the end of the line, even though it's allowed to be there.

Added test for using tab pages.

toupper() and tolower() were not used, because of checking for __STDC__ISO_10646__ instead of __STDC_ISO_10646__. (sertacyildiz)

For `":map <expr>"` forbid changing the text, jumping to another buffer and using `":normal"` to avoid nasty side effects.

--- fixes and changes since Vim 7.0g ---

Compilation error on HP-UX, use of "dlerr" must be inside a #ifdef.
(Gary Johnson)

Report +reltime feature in `":version"` output.

The tar and zip plugins detect failure to get the contents of the archive and edit the file as-is.

When the result of **'guitablabel'** is empty fall back to the default label.

Fixed crash when using `":insert"` in a while loop and missing "endwhile".

`"gt"` and other commands could move to another window when `textlock` active and when the command line window was open.

Spell checking a file with syntax highlighting and a bad word at the end of the line is ignored could make `"]s"` hang.

Mac: `inputdialog()` didn't work when compiled with big features.

Interrupting `":vimgrep"` while it is busy loading a file left a modified and hidden buffer behind. Use `enter_cleanup()` and `leave_cleanup()` around `wipe_buffer()`.

When making **'keymap'** empty the `b:keymap_name` variable wasn't deleted.

Using **CTRL-N** that searches a long time, pressing space to interrupt the searching and accept the first match, the popup menu was still displayed briefly.

When setting the Vim window height with `-geometry` the `'window'` option could be at a value that makes `CTRL-F` behave differently.

When opening a quickfix window in two tabs they used different buffers, causing redrawing problems later. Now use the same buffer for all quickfix windows. (Yegappan Lakshmanan)

When `'mousefocus'` is set moving the mouse to the text tab pages line would move focus to the first window. Also, the mouse pointer would jump to the active window.

In a session file, when an empty buffer is wiped out, do this silently.

When one window has the cursor on the last line and another window is resized to make that window smaller, the cursor line could go below the displayed lines. In `win_new_height()` subtract one from the available space. Also avoid that using `"~"` lines makes the window scroll down.

Mac: When sourcing the `"macmap.vim"` script and then finding a `.vimrc` file the `'cpo'` option isn't set properly, because it was already set and restored. Added the `<special>` argument to `":map"`, so that `'cpo'` doesn't need to be changed to be able to use `<>` notation. Also do this for `":menu"` for consistency.

When using `"/encoding=abc"` in a spell word list, only `"bc"` was used.

When `'encoding'` and `'printencoding'` were both `"utf-8"` then `":hardcopy"` didn't work. (Mike Williams)

Mac: When building with `"--disable-gui"` the install directory would still be `"/Applications"` and `Vim.app` would be installed. Now install in `/usr/local` as usual for a console application.

GUI: when doing completion and there is one match and still searching for another, the cursor was displayed at the end of the line instead of after the match. Now show the cursor after the match while still searching for matches.

GUI: The mouse shape changed on the statusline even when `'mouse'` was empty and they can't be dragged.

GTK2: Selecting a button in the `confirm()` dialog with Tab or cursor keys and hitting Enter didn't select that button. Removed GTK 1 specific code. (Neil Bird)

When evaluating `'balloonexpr'` takes a long time it could be called recursively, which could cause a crash.

`exists()` could not be used to detect whether `":2match"` is supported. Added a check for it specifically.

GTK1: Tab page labels didn't work. (Yegappan Lakshmanan)

Insert mode completion: When finding matches use `'ignorecase'`, but when adding matches to the list don't use it, so that all words with different case are

added, "word", "Word" and "WORD".

When **'cursorline'** and **'hlsearch'** are set and the search pattern is "x\n" the rest of the line was highlighted as a match.

Cursor moved while evaluating **'balloonexpr'** that invokes ":isearch" and redirects the output. Don't move the cursor to the command line if msg_silent is set.

exists() ignored text after a function name and option name, which could result in false positives.

exists() ignored characters after the recognized word, which can be wrong when using a name with non-keyword characters. Specifically, these calls no longer allow characters after the name: exists('*funcname') exists('*funcname(...)') exists('&option') exists(':cmd') exists('g:name') exists('g:name[n]') exists('g:name.n')

Trigger the TabEnter autocommand only after entering the current window of the tab page, otherwise the commands are executed with an invalid current window.

Win32: When using two monitors and Vim is on the second monitor, changing the width of the Vim window could make it jump to the first monitor.

When scrolling back at the more prompt and the quitting a line of text would be left behind when **'cmdheight'** is 2 or more.

Fixed a few things for Insert mode completion, especially when typing BS, **CTRL-N** or a printable character while still searching for matches.

=====

VERSION 7.1 version-7.1 version7.1

This section is about improvements made between version 7.0 and 7.1.

This is a bug-fix release, there are no fancy new features.

Changed changed-7.1

Added setting **'mouse'** in vimrc_example.vim.

When building with MZscheme also look for include files in the "plt" subdirectory. That's where they are for FreeBSD.

The Ruby interface module is now called "Vim" instead of "VIM". But "VIM" is an alias, so it's backwards compatible. (Tim Pope)

Added added-7.1

New syntax files:

- /var/log/messages (Yakov Lerner)
- Autohotkey (Nikolai Weibull)
- AutoIt v3 (Jared Breland)
- Bazaar commit file "bzc". (Dmitry Vasiliev)
- Cdrdao TOC (Nikolai Weibull)
- Cmusrc (Nikolai Weibull)
- Conary recipe (rPath Inc)
- Framescript (Nikolai Weibull)
- FreeBasic (Mark Manning)
- Hamster (David Fishburn)
- IBasic (Mark Manning)
- Initng (Elan Ruusamae)
- Ldapconf (Nikolai Weibull)
- Litestep (Nikolai Weibull)
- Privoxy actions file (Doug Kearns)
- Streaming Descriptors "sd" (Puria Nafisi Azizi)

New tutor files:

- Czech (Lubos Turek)
- Hungarian (Arpad Horvath)
- Turkish (Serkan kkk)
- utf-8 version of Greek tutor.
- utf-8 version of Russian tutor.
- utf-8 version of Slovak tutor.

New filetype plugins:

- Bst (Tim Pope)
- Cobol (Tim Pope)
- Fvwm (Gautam Iyer)
- Hamster (David Fishburn)
- Django HTML template (Dave Hodder)

New indent files:

- Bst (Tim Pope)
- Cobol (Tim Pope)
- Hamster (David Fishburn)
- Django HTML template (Dave Hodder)
- Javascript
- JSP (David Fishburn)

New keymap files:

- Bulgarian (Boyko Bantchev)
- Mongolian (Natsagdorj Shagdar)
- Thaana (Ibrahim Fayaz)
- Vietnamese (Samuel Thibault)

Other new runtime files:

- Ada support files. (Neil Bird, Martin Krischik)
- Slovenian menu translations (Mojca Miklavc)
- Mono C# compiler plugin (Jarek Sobiecki)

Fixed

fixed-7.1

Could not build the Win32s version. Added a few structure definitions in src/gui_w32.c

Patch 7.0.001

Problem: ":set spellsuggest+=10" does not work. (Suresh Govindachar)
Solution: Add P_COMMA to the '**spellsuggest**' flags.
Files: src/option.c

Patch 7.0.002

Problem: C omni completion has a problem with tags files with a path containing "#" or "%".
Solution: Escape these characters. (Sebastian Baberowski)
Files: runtime/autoload/ccomplete.vim

Patch 7.0.003

Problem: GUI: clicking in the lower part of a label in the tab pages line while '**mousefocus**' is set may warp the mouse pointer. (Robert Webb)
Solution: Check for a negative mouse position.
Files: src/gui.c

Patch 7.0.004

Problem: Compiler warning for debug_saved used before set. (Todd Blumer)
Solution: Remove the "else" for calling save_dbg_stuff().
Files: src/ex_docmd.c

Patch 7.0.005 (extra)

Problem: Win32: The installer doesn't remove the "autoload" and "spell" directories. (David Fishburn)
Solution: Add the directories to the list to be removed.
Files: nsis/gvim.nsi

Patch 7.0.006

Problem: Mac: "make shadow" doesn't make a link for infplist.xml. (Axel Kielhorn)
Solution: Make the link.
Files: src/Makefile

Patch 7.0.007

Problem: AIX: compiling fails for message.c. (Ruediger Hornig)
Solution: Move the #if outside of memchr().
Files: src/message.c

Patch 7.0.008

Problem: Can't call a function that uses both <SID> and {expr}. (Thomas)
Solution: Check both the expanded and unexpanded name for <SID>.
Files: src/eval.c

Patch 7.0.009

Problem: ml_get errors with both '**sidescroll**' and '**spell**' set.
Solution: Use ml_get_buf() instead of ml_get(), get the line from the right

buffer, not the current one.
Files: src/spell.c

Patch 7.0.010

Problem: The spellfile plugin required typing login name and password.
Solution: Use "anonymous" and "vim7user" by default. No need to setup a .netrc file.
Files: runtime/autoload/spellfile.vim

Patch 7.0.011

Problem: Can't compile without the folding and with the eval feature.
Solution: Add an #ifdef. (Vallimar)
Files: src/option.c

Patch 7.0.012

Problem: Using the matchparen plugin, moving the cursor in Insert mode to a shorter line that ends in a brace, changes the preferred column
Solution: Use winsaveview()/winrestview() instead of getpos()/setpos().
Files: runtime/plugin/matchparen.vim

Patch 7.0.013

Problem: Insert mode completion: using **CTRL-L** to add an extra character also deselected the current match, making it impossible to use **CTRL-L** a second time.
Solution: Keep the current match. Also make **CTRL-L** work at the original text, using the first displayed match.
Files: src/edit.c

Patch 7.0.014

Problem: Compiling gui_xmew.c fails on Dec Alpha Tru64. (Rolfe)
Solution: Disable some code for Motif 1.2 and older.
Files: src/gui_xmew.c

Patch 7.0.015

Problem: Athena: compilation problems with modern compiler.
Solution: Avoid type casts for lvalue. (Alexey Froloff)
Files: src/gui_at_fs.c

Patch 7.0.016

Problem: Printing doesn't work for "dec-mcs" encoding.
Solution: Add "dec-mcs", "mac-roman" and "hp-roman8" to the list of recognized 8-bit encodings. (Mike Williams)
Files: src/mbyte.c

Patch 7.0.017 (after 7.0.014)

Problem: Linking gui_xmew.c fails on Dec Alpha Tru64. (Rolfe)
Solution: Adjust defines for Motif 1.2 and older.
Files: src/gui_xmew.c

Patch 7.0.018

Problem: VMS: plugins are not loaded on startup.
Solution: Remove "*" from the path. (Zoltan Arpadffy)
Files: src/main.c

Patch 7.0.019

Problem: Repeating "VjA789" may cause a crash. (James Vega)
Solution: Check the cursor column after moving it to another line.
Files: src/ops.c

Patch 7.0.020

Problem: Crash when using **'mousefocus'**. (William Fulton)
Solution: Make buffer for mouse coordinates 2 bytes longer. (Juergen Weigert)
Files: src/gui.c

Patch 7.0.021

Problem: Crash when using "\\[" and "\\]" in **'errorformat'**. (Marc Weber)
Solution: Check for valid submatches after matching the pattern.
Files: src/quickfix.c

Patch 7.0.022

Problem: Using buffer.append() in Ruby may append the line to the wrong buffer. (Alex Norman)
Solution: Properly switch to the buffer to do the appending. Also for buffer.delete() and setting a buffer line.
Files: src/if_ruby.c

Patch 7.0.023

Problem: Crash when doing spell completion in an empty line and pressing **CTRL-E**.
Solution: Check for a zero pointer. (James Vega)
Also handle a situation without a matching pattern better, report "No matches" instead of remaining in undefined **CTRL-X** mode. And get out of **CTRL-X** mode when typing a letter.
Files: src/edit.c

Patch 7.0.024

Problem: It is possible to set arbitrary "v:" variables.
Solution: Disallow setting "v:" variables that are not predefined.
Files: src/eval.c

Patch 7.0.025

Problem: Crash when removing an element of a:000. (Nikolai Weibull)
Solution: Mark the a:000 list with VAR_FIXED.
Files: src/eval.c

Patch 7.0.026

Problem: Using libcall() may show an old error.
Solution: Invoke dlerror() to clear a previous error. (Yukihiro Nakadaira)
Files: src/os_unix.c

Patch 7.0.027 (extra)

Problem: Win32: When compiled with SNIFF gvim may hang on exit.
Solution: Translate and dispatch the WM_USER message. (Mathias Michaelis)
Files: src/gui_w48.c

Patch 7.0.028 (extra)

Problem: OS/2: Vim doesn't compile with gcc 3.2.1.
Solution: Add argument to after_pathsep(), don't define vim_handle_signal(),

define HAVE_STDARG_H. (David Sanders)
Files: src/os_unix.c, src/vim.h, src/os_os2_cfg.h

Patch 7.0.029

Problem: getchar() may not position the cursor after a space.
Solution: Position the cursor explicitly.
Files: src/eval.c

Patch 7.0.030

Problem: The ":compiler" command can't be used in a FileChangedRO event.
(Hari Krishna Dara)
Solution: Add the CMDWIN flag to the ":compiler" command.
Files: src/ex_cmds.h

Patch 7.0.031

Problem: When deleting a buffer the buffer-local mappings for Select mode remain.
Solution: Add the Select mode bit to MAP_ALL_MODES. (Edwin Steiner)
Files: src/vim.h

Patch 7.0.032 (extra, after 7.0.027)

Problem: Missing semicolon.
Solution: Add the semicolon.
Files: src/gui_w48.c

Patch 7.0.033

Problem: When pasting text, with the menu or **CTRL-V**, autoindent is removed.
Solution: Use "x<BS>" to avoid indent to be removed. (Benji Fisher)
Files: runtime/autoload/paste.vim

Patch 7.0.034

Problem: After doing completion and typing more characters or using BS repeating with "." didn't work properly. (Martin Stubenschrott)
Solution: Don't put BS and other characters in the redo buffer right away, do this when finishing completion.
Files: src/edit.c

Patch 7.0.035

Problem: Insert mode completion works when typed but not when replayed from a register. (Hari Krishna Dara)
Also: Mappings for Insert mode completion don't always work.
Solution: When finding a non-completion key in the input don't interrupt completion when it wasn't typed.
Do use mappings when checking for typeahead while still finding completions. Avoids that completion is interrupted too soon.
Use "compl_pending" in a different way.
Files: src/edit.c

Patch 7.0.036

Problem: Can't compile with small features and syntax highlighting or the diff feature.
Solution: Define LINE_ATTR whenever syntax highlighting or the diff feature is enabled.
Files: src/screen.c

Patch 7.0.037

Problem: Crash when resizing the GUI window vertically when there is a line that doesn't fit.
Solution: Don't redraw while the screen data is invalid.
Files: src/screen.c

Patch 7.0.038

Problem: When calling complete() from an Insert mode expression mapping text could be inserted in an improper way.
Solution: Make undo_allowed() global and use it in complete().
Files: src/undo.c, src/proto/undo.pro, src/eval.c

Patch 7.0.039

Problem: Calling inputdialog() with a third argument in the console doesn't work.
Solution: Make a separate function for input() and inputdialog(). (Yegappan Lakshmanan)
Files: src/eval.c

Patch 7.0.040

Problem: When '**cmdheight**' is larger than 1 using inputlist() or selecting a spell suggestion with the mouse gets the wrong entry.
Solution: Start listing the first alternative on the last line of the screen.
Files: src/eval.c, src/spell.c

Patch 7.0.041

Problem: cursor([1, 1]) doesn't work. (Peter Hodge)
Solution: Allow leaving out the third item of the list and use zero for the virtual column offset.
Files: src/eval.c

Patch 7.0.042

Problem: When pasting a block of text in Insert mode Vim hangs or crashes. (Noam Halevy)
Solution: Avoid that the cursor is positioned past the NUL of a line.
Files: src/ops.c

Patch 7.0.043

Problem: Using "%!" at the start of '**statusline**' doesn't work.
Solution: Recognize the special item when the option is being set.
Files: src/option.c

Patch 7.0.044

Problem: Perl: setting a buffer line in another buffer may result in changing the current buffer.
Solution: Properly change to the buffer to be changed.
Files: src/if_perl.xs

Patch 7.0.045 (extra)

Problem: Win32: Warnings when compiling OLE version with MSVC 2005.
Solution: Move including vim.h to before windows.h. (Ilya Bobir)
Files: src/if_ole.cpp

Patch 7.0.046

Problem: The matchparen plugin ignores parens in strings, but not in single quotes, often marked with "character".
Solution: Also ignore parens in syntax items matching "character".
Files: runtime/plugin/matchparen.vim

Patch 7.0.047

Problem: When running configure the exit status is wrong.
Solution: Handle the exit status properly. (Matthew Woehlke)
Files: configure, src/configure

Patch 7.0.048

Problem: Writing a compressed file fails when there are parens in the name. (Wang Jian)
Solution: Put quotes around the temp file name.
Files: runtime/autoload/gzip.vim

Patch 7.0.049

Problem: Some TCL scripts are not recognized. (Steven Atkinson)
Solution: Check for "exec wish" in the file.
Files: runtime/scripts.vim

Patch 7.0.050

Problem: After using the netbeans interface close command a stale pointer may be used.
Solution: Clear the pointer to the closed buffer. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.0.051 (after 7.0.44)

Problem: The Perl interface doesn't compile or doesn't work properly.
Solution: Remove the spaces before #ifdef and avoid an empty line above it.
Files: src/if_perl.xs

Patch 7.0.052

Problem: The user may not be aware that the Vim server allows others more functionality than desired.
Solution: When running Vim as root don't become a Vim server without an explicit --servername argument.
Files: src/main.c

Patch 7.0.053

Problem: Shortening a directory name may fail when there are multibyte characters.
Solution: Copy the correct bytes. (Titov Anatoly)
Files: src/misc1.c

Patch 7.0.054

Problem: Mac: Using a menu name that only has a mnemonic or accelerator causes a crash. (Elliot Shank)
Solution: Check for an empty menu name. Also delete empty submenus that were created before detecting the error.
Files: src/menu.c

Patch 7.0.055

Problem: ":startinsert" in a CmdwinEnter autocommand doesn't take immediate effect. (Bradley White)
Solution: Put a NOP key in the typeahead buffer. Also avoid that using **CTRL-C** to go back to the command line moves the cursor left.
Files: src/edit.c, src/ex_getln.c

Patch 7.0.056

Problem: "#!something" gives an error message.
Solution: Ignore this line, so that it can be used in an executable Vim script.
Files: src/ex_docmd.c

Patch 7.0.057 (extra, after 7.0.45)

Problem: Win32: Compilation problem with Borland C 5.5.
Solution: Include vim.h as before. (Mark S. Williams)
Files: src/if_ole.cpp

Patch 7.0.058

Problem: The gbk and gb18030 encodings are not recognized.
Solution: Add aliases to cp936. (Edward L. Fox)
Files: src/mbyte.c

Patch 7.0.059

Problem: The Perl interface doesn't compile with ActiveState Perl 5.8.8.
Solution: Remove the __attribute__() items. (Liu Yubao)
Files: src/if_perl.xs

Patch 7.0.060 (after 7.0.51)

Problem: Code for temporarily switching to another buffer is duplicated in quite a few places.
Solution: Use aucmd_prepbuf() and aucmd_restbuf() also when FEAT_AUTOCMD is not defined.
Files: src/buffer.c, src/eval.c, src/fileio.c, src/if_ruby.c, src/if_perl.xs, src/quickfix.c, src/structs.h

Patch 7.0.061

Problem: Insert mode completion for Vim commands may crash if there is nothing to complete.
Solution: Instead of freeing the pattern make it empty, so that a "not found" error is given. (Yukihiro Nakadaira)
Files: src/edit.c

Patch 7.0.062

Problem: Mac: Crash when using the popup menu for spell correction. The popup menu appears twice when letting go of the right mouse button early.
Solution: Don't show the popup menu on the release of the right mouse button. Also check that a menu pointer is actually valid.
Files: src/proto/menu.pro, src/menu.c, src/normal.c, src/term.c

Patch 7.0.063

Problem: Tiny chance for a memory leak. (coverity)
Solution: Free pointer when next memory allocation fails.
Files: src/eval.c

Patch 7.0.064

Problem: Using uninitialized variable. (Tony Mechelynck)
Solution: When not used set "temp" to zero. Also avoid a warning for "files" in ins_compl_dictionaries().
Files: src/edit.c

Patch 7.0.065 (extra)

Problem: Mac: left-right movement of the scrollwheel causes up-down scrolling.
Solution: Ignore mouse wheel events that are not up-down. (Nicolas Weber)
Files: src/gui_mac.c

Patch 7.0.066

Problem: After the popup menu for Insert mode completion overlaps the tab pages line it is not completely removed.
Solution: Redraw the tab pages line after removing the popup menu. (Ori Avtalion)
Files: src/popupmnu.c

Patch 7.0.067

Problem: Undo doesn't always work properly when using "scim" input method. Undo is split up when using preediting.
Solution: Reset xim_has_preediting also when preedit_start_col is not MAXCOL. Don't split undo when <Left> is used while preediting. (Yukihiro Nakadaira)
Files: src/edit.c, src/mbyte.c

Patch 7.0.068

Problem: When '**ignorecase**' is set and using Insert mode completion, typing characters to change the list of matches, case is not ignored. (Hugo Ahlenius)
Solution: Store the '**ignorecase**' flag with the matches where needed.
Files: src/edit.c, src/search.c, src/spell.c

Patch 7.0.069

Problem: Setting '**guitablabel**' to %!expand(\%) causes Vim to free an invalid pointer. (Kim Schulz)
Solution: Don't try freeing a constant string pointer.
Files: src/buffer.c

Patch 7.0.070

Problem: Compiler warnings for shadowed variables and uninitialized variables.
Solution: Rename variables such as "index", "msg" and "dup". Initialize variables.
Files: src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_x11.c, src/hardcopy.c, src/if_cscope.c, src/main.c, src/mbyte.c, src/memline.c, src/netbeans.c, src/normal.c, src/option.c, src/os_unix.c, src/quickfix.c, src/regexp.c, src/screen.c, src/search.c, src/spell.c, src/ui.c, src/undo.c, src/window.c, src/version.c

Patch 7.0.071

Problem: Using an empty search pattern may cause a crash.
Solution: Avoid using a NULL pointer.
Files: src/search.c

Patch 7.0.072

Problem: When starting the GUI fails there is no way to adjust settings or do something else.
Solution: Add the GUIFailed autocommand event.
Files: src/fileio.c, src/gui.c, src/vim.h

Patch 7.0.073

Problem: Insert mode completion: Typing <CR> sometimes selects the original text instead of keeping what was typed. (Justin Constantino)
Solution: Don't let <CR> select the original text if there is no popup menu.
Files: src/edit.c

Patch 7.0.074 (extra)

Problem: Win32: tooltips were not converted from 'encoding' to Unicode.
Solution: Set the tooltip to use Unicode and do the conversion. Also cleanup the code for the tab pages tooltips. (Yukihiro Nakadaira)
Files: src/gui_w32.c, src/gui_w48.c

Patch 7.0.075

Problem: winsaveview() did not store the actual value of the desired cursor column. This could move the cursor in the matchparen plugin.
Solution: Call update_curswant() before using the value w_curswant.
Files: src/eval.c

Patch 7.0.076 (after 7.0.010)

Problem: Automatic downloading of spell files only works for ftp.
Solution: Don't add login and password for non-ftp URLs. (Alexander Patrakov)
Files: runtime/autoload/spellfile.vim

Patch 7.0.077

Problem: ":unlet v:this_session" causes a crash. (Marius Roets)
Solution: When trying to unlet a fixed variable give an error message.
Files: src/eval.c

Patch 7.0.078

Problem: There are two error messages E46.
Solution: Change the number for the sandbox message to E794.
Files: src/globals.h

Patch 7.0.079

Problem: Russian tutor doesn't work when 'encoding' is "utf-8".
Solution: Use tutor.ru.utf-8 as the master, and generate the other encodings from it. Select the right tutor depending on 'encoding'. (Alexey Froloff)
Files: runtime/tutor/Makefile, runtime/tutor/tutor.vim, runtime/tutor/tutor.ru.utf-8

Patch 7.0.080

Problem: Generating auto/pathdef.c fails for CFLAGS with a backslash.

Solution: Double backslashes in the string. (Alexey Froloff)
Files: src/Makefile

Patch 7.0.081

Problem: Command line completion doesn't work for a shell command with an absolute path.

Solution: Don't use \$PATH when there is an absolute path.

Files: src/ex_getln.c

Patch 7.0.082

Problem: Calling a function that waits for input may cause List and Dictionary arguments to be freed by the garbage collector.

Solution: Keep a list of all arguments to internal functions.

Files: src/eval.c

Patch 7.0.083

Problem: Clicking with the mouse on an item for inputlist() doesn't work when '**compatible**' is set and/or when '**cmdheight**' is more than one. (Christian J. Robinson)

Solution: Also decrement "lines_left" when '**more**' isn't set. Set "cmdline_row" to zero to get all mouse events.

Files: src/message.c, src/misc1.c

Patch 7.0.084

Problem: The garbage collector may do its work while some Lists or Dictionaries are used internally, e.g., by ":echo" that runs into the more-prompt or ":echo [garbagecollect()]".

Solution: Only do garbage collection when waiting for a character at the toplevel. Let garbagecollect() set a flag that is handled at the toplevel before waiting for a character.

Files: src/eval.c, src/getchar.c, src/globals.h, src/main.c

Patch 7.0.085

Problem: When doing "make test" the viminfo file is modified.

Solution: Use another viminfo file after setting '**compatible**'.

Files: src/testdir/test56.in

Patch 7.0.086

Problem: getqflist() returns entries for pattern and text with the number zero. Passing these to setqflist() results in the string "0".

Solution: Use an empty string instead of the number zero.

Files: src/quickfix.c

Patch 7.0.087

Problem: After ":file fname" and ":saveas fname" the '**autochdir**' option does not take effect. (Yakov Lerner)

Commands for handling '**autochdir**' are repeated many times.

Solution: Add the DO_AUTOCHDIR macro and do_autochdir(). Use it for ":file fname" and ":saveas fname".

Files: src/proto/buffer.pro, src/buffer.c, src/ex_cmds.c, src/macros.h, src/netbeans.c, src/option.c, src/window.c

Patch 7.0.088

Problem: When compiled with Perl the generated prototypes have "extern"

unnecessarily added.
Solution: Remove the "-pipe" argument from PERL_CFLAGS.
Files: src/auto/configure, src/configure.in

Patch 7.0.089

Problem: "ga" does not work properly for a non-Unicode multibyte encoding.
Solution: Only check for composing chars for utf-8. (Taro Muraoka)
Files: src/ex_cmds.c

Patch 7.0.090

Problem: Cancelling the conform() dialog on the console with Esc requires typing it twice. (Benji Fisher)
Solution: When the start of an escape sequence is found use 'timeoutlen' or 'ttimeoutlen'.
Files: src/misc1.c

Patch 7.0.091

Problem: Using winrestview() while 'showcmd' is set causes the cursor to be displayed in the wrong position. (Yakov Lerner)
Solution: Set the window topline properly.
Files: src/eval.c

Patch 7.0.092 (after 7.0.082 and 7.0.084)

Problem: The list of internal function arguments is obsolete now that garbage collection is only done at the toplevel.
Solution: Remove the list of all arguments to internal functions.
Files: src/eval.c

Patch 7.0.093

Problem: The matchparen plugin can't handle a 'matchpairs' value where a colon is matched.
Solution: Change the split() that is used to change 'matchpairs' into a List.
Files: runtime/plugin/matchparen.vim

Patch 7.0.094

Problem: When a hidden buffer is made the current buffer and another file edited later, the file message will still be given. Using ":silent" also doesn't prevent the file message. (Marvin Renich)
Solution: Reset the need_fileinfo flag when reading a file. Don't set need_fileinfo when msg_silent is set.
Files: src/buffer.c, src/fileio.c

Patch 7.0.095

Problem: The Greek tutor is not available in utf-8. "el" is used for the language, only "gr" for the country is recognized.
Solution: Add the utf-8 Greek tutor. Use it for conversion to iso-8859-7 and cp737. (Lefteris Dimitroulakis)
Files: runtime/tutor/Makefile, runtime/tutor/tutor.gr.utf-8, runtime/tutor/tutor.vim

Patch 7.0.096

Problem: taglist() returns the filename relative to the tags file, while the directory of the tags file is unknown. (Hari Krishna Dara)

Solution: Expand the file name. (Yegappan Lakshmanan)
Files: src/tag.c

Patch 7.0.097

Problem: ":tabclose N" that closes another tab page does not remove the tab pages line. Same problem when using the mouse.

Solution: Adjust the tab pages line when needed in tabpage_close_other().
Files: src/ex_docmd.c

Patch 7.0.098

Problem: Redirecting command output in a cmdline completion function doesn't work. (Hari Krishna Dara)

Solution: Enable redirection when redirection is started.
Files: src/ex_docmd.c, src/ex_getln.c

Patch 7.0.099

Problem: GUI: When the popup menu is visible using the scrollbar messes up the display.

Solution: Disallow scrolling the current window. Redraw the popup menu after scrolling another window.

Files: src/gui.c

Patch 7.0.100

Problem: "zug" may report the wrong filename. (Lawrence Kesteloot)

Solution: Call home_replace() to fill NameBuff[].
Files: src/spell.c

Patch 7.0.101

Problem: When the "~/vim/spell" directory does not exist "zg" may create a wrong directory. "zw" doesn't work.

Solution: Use the directory of the file name instead of NameBuff. For "zw" not only remove a good word but also add the word with "!".

Files: src/spell.c

Patch 7.0.102

Problem: Redrawing cmdline is not correct when using SCIM.

Solution: Don't call im_get_status(). (Yukihiro Nakadaira)
Files: src/ex_getln.c

Patch 7.0.103 (after 7.0.101)

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)

Solution: Init variable.

Files: src/spell.c

Patch 7.0.104

Problem: The CursorHoldI event only triggers once in Insert mode. It also triggers after **CTRL-V** and other two-key commands.

Solution: Set "did_cursorhold" before getting a second key. Reset "did_cursorhold" after handling a command.

Files: src/edit.c, src/fileio.c

Patch 7.0.105

Problem: When using incremental search the statusline ruler isn't updated. (Christoph Koegl)

Solution: Update the statusline when it contains the ruler.
Files: src/ex_getln.c

Patch 7.0.106

Problem: The spell popup menu uses ":amenu", triggering mappings. Other PopupMenu autocommands are removed. (John Little)

Solution: Use ":anoremenu" and use an autocmd group.
Files: runtime/menu.vim

Patch 7.0.107

Problem: Incremental search doesn't redraw the text tabline. (Ilya Bobir)
Also happens in other situations with one window in a tab page.

Solution: Redraw the tabline after clearing the screen.
Files: src/screen.c

Patch 7.0.108 (extra)

Problem: Amiga: Compilation problem.

Solution: Have mch_mkdir() return a failure flag. (Willy Catteau)
Files: src/os_amiga.c, src/proto/os_amiga.pro

Patch 7.0.109

Problem: Lisp indenting is confused by escaped quotes in strings. (Dorai Sitaram)

Solution: Check for backslash inside strings. (Sergey Khorev)
Files: src/misc1.c

Patch 7.0.110

Problem: Amiga: Compilation problems when not using libnix.

Solution: Change a few #ifdefs. (Willy Catteau)
Files: src/memfile.c

Patch 7.0.111

Problem: The gzip plugin can't handle filenames with single quotes.

Solution: Add and use the shellescape() function. (partly by Alexey Froloff)
Files: runtime/autoload/gzip.vim, runtime/doc/eval.txt, src/eval.c, src/mbyte.c, src/misc2.c, src/proto/misc2.pro

Patch 7.0.112

Problem: Python interface does not work with Python 2.5.

Solution: Change PyMem_DEL() to Py_DECREF(). (Sumner Hayes)
Files: src/if_python.c

Patch 7.0.113

Problem: Using **CTRL-L** in Insert completion when there is no current match may cause a crash. (Yukihiro Nakadaira)

Solution: Check for compl_leader to be NULL
Files: src/edit.c

Patch 7.0.114

Problem: When aborting an insert with **CTRL-C** an extra undo point is created in the GUI. (Yukihiro Nakadaira)

Solution: Call gotchars() only when advancing.
Files: src/getchar.c

Patch 7.0.115

Problem: When **'ignorecase'** is set, Insert mode completion only adds "foo" and not "Foo" when both are found.
A found match isn't displayed right away when **'completeopt'** does not have "menu" or "menuone".

Solution: Do not ignore case when checking if a completion match already exists. call `ins_compl_check_keys()` also when not using a popup menu. (Yukihiro Nakadaira)

Files: `src/edit.c`

Patch 7.0.116

Problem: 64 bit Windows version reports "32 bit" in the ":version" output. (M. Veerman)

Solution: Change the text for Win64.

Files: `src/version.c`

Patch 7.0.117

Problem: Using "extend" on a syntax item inside a region with "keepend", an intermediate item may be truncated.

When applying the "keepend" and there is an offset to the end pattern the highlighting of a contained item isn't adjusted.

Solution: Use the `seen_keepend` flag to remember when to apply the "keepend" flag. Adjust the keepend highlighting properly. (Ilya Bobir)

Files: `src/syntax.c`

Patch 7.0.118

Problem: `printf()` does not do zero padding for strings.

Solution: Do allow zero padding for strings.

Files: `src/message.c`

Patch 7.0.119

Problem: When going back from Insert to Normal mode the CursorHold event doesn't trigger. (Yakov Lerner)

Solution: Reset "did_cursorhold" when leaving Insert mode.

Files: `src/edit.c`

Patch 7.0.120

Problem: Crash when using **CTRL-R** = at the command line and entering "`getreg('=')`". (James Vega)

Solution: Avoid recursiveness of evaluating the = register.

Files: `src/ops.c`

Patch 7.0.121

Problem: GUI: Dragging the last status line doesn't work when there is a text tabline. (Markus Wolf)

Solution: Take the text tabline into account when deciding to start modeless selection.

Files: `src/gui.c`

Patch 7.0.122

Problem: GUI: When clearing after a bold, double-wide character half a character may be drawn.

Solution: Check for double-wide character and redraw it. (Yukihiro Nakadaira)

Files: `src/screen.c`

Patch 7.0.123

Problem: On SCO Openserver configure selects the wrong terminal library.
Solution: Put terminfo before the other libraries. (Roger Cornelius)
Also fix a small problem compiling on Mac without Darwin.
Files: src/configure.in, src/auto/configure

Patch 7.0.124

Problem: getwinvar() obtains a dictionary with window-local variables, but it's always for the current window.
Solution: Get the variables of the specified window. (Geoff Reedy)
Files: src/eval.c

Patch 7.0.125

Problem: When "autoselect" is in the '**clipboard**' option then the '<' and '>' marks are set while Visual mode is still active.
Solution: Don't set the '<' and '>' marks when yanking the selected area for the clipboard.
Files: src/normal.c

Patch 7.0.126

Problem: When '**formatexpr**' uses setline() and later internal formatting is used undo information is not correct. (Jiri Cerny, Benji Fisher)
Solution: Set ins_need_undo after using '**formatexpr**'.
Files: src/edit.c

Patch 7.0.127

Problem: Crash when swap file has invalid timestamp.
Solution: Check return value of ctime() for being NULL.
Files: src/memline.c

Patch 7.0.128

Problem: GUI: when closing gvim is cancelled because there is a changed buffer the screen isn't updated to show the changed buffer in the current window. (Krzysztof Kacprzak)
Solution: Redraw when closing gvim is cancelled.
Files: src/gui.c

Patch 7.0.129

Problem: GTK GUI: the GTK file dialog can't handle a relative path.
Solution: Make the initial directory a full path before passing it to GTK. (James Vega) Also postpone adding the default file name until after setting the directory.
Files: src/gui_gtk.c

Patch 7.0.130 (extra)

Problem: Win32: Trying to edit or write devices may cause Vim to get stuck.
Solution: Add the '**opendevic**e' option, default off. Disallow reading/writing from/to devices when it's off.
Also detect more devices by the full name starting with "\\.\\".
Files: runtime/doc/options.txt, src/fileio.c, src/option.c, src/option.h, src/os_win32.c

Patch 7.0.131

Problem: Win32: "vim -r" does not list all the swap files.
Solution: Also check for swap files starting with a dot.
Files: src/memline.c

Patch 7.0.132 (after 7.0.130)

Problem: Win32: Crash when Vim reads from stdin.
Solution: Only use mch_nodetype() when there is a file name.
Files: src/fileio.c

Patch 7.0.133

Problem: When searching included files messages are added to the history.
Solution: Set msg_hist_off for messages about scanning included files.
Set msg_silent to avoid message about wrapping around.
Files: src/edit.c, src/globals.h, src/message.c, src/search.c

Patch 7.0.134

Problem: Crash when comparing a recursively looped List or Dictionary.
Solution: Limit recursiveness for comparing to 1000.
Files: src/eval.c

Patch 7.0.135

Problem: Crash when garbage collecting list or dict with loop.
Solution: Don't use DEL_REFCOUNT but don't recurse into Lists and
Dictionaries when freeing them in the garbage collector.
Also add allocated Dictionaries to the list of Dictionaries to
avoid leaking memory.
Files: src/eval.c, src/proto/eval.pro, src/tag.c

Patch 7.0.136

Problem: Using "0" while matching parens are highlighted may not remove the
highlighting. (Ilya Bobir)
Solution: Also trigger CursorMoved when a line is inserted under the cursor.
Files: src/misc1.c

Patch 7.0.137

Problem: Configure check for big features is wrong.
Solution: Change "==" to "=". (Martti Kuparinen)
Files: src/auto/configure, src/configure.in

Patch 7.0.138 (extra)

Problem: Mac: modifiers don't work with function keys.
Solution: Use GetEventParameter() to obtain modifiers. (Nicolas Weber)
Files: src/gui_mac.c

Patch 7.0.139

Problem: Using **CTRL-PageUp** or **CTRL-PageDown** in Insert mode to go to another
tab page does not prepare for undo properly. (Stefano Zacchiroli)
Solution: Call start_arrow() before switching tab page.
Files: src/edit.c

Patch 7.0.140 (after 7.0.134)

Problem: Comparing recursively looped List or Dictionary doesn't work well.
Solution: Detect comparing a List or Dictionary with itself.
Files: src/eval.c

Patch 7.0.141

Problem: When pasting a whole line on the command line an extra CR is added literally.

Solution: Don't add the trailing CR when pasting with the mouse.

Files: src/ex_getln.c, src/proto/ops.pro, src/ops.c

Patch 7.0.142

Problem: Using the middle mouse button in Select mode to paste text results in an extra "y". (Kriton Kyrimis)

Solution: Let the middle mouse button replace the selected text with the contents of the clipboard.

Files: src/normal.c

Patch 7.0.143

Problem: Setting '**scroll**' to its default value was not handled correctly.

Solution: Compare the right field to PV_SCROLL.

Files: src/option.c

Patch 7.0.144

Problem: May compare two unrelated pointers when matching a pattern against a string. (Dominique Pelle)

Solution: Avoid calling reg_getline() when REG_MULTI is false.

Files: src/regexp.c

Patch 7.0.145 (after 7.0.142)

Problem: Compiler warning.

Solution: Add type cast.

Files: src/normal.c

Patch 7.0.146

Problem: When '**switchbuf**' is set to "usetab" and the current tab has only a quickfix window, jumping to an error always opens a new window. Also, when the buffer is open in another tab page it's not found.

Solution: Check for the "split" value of '**switchbuf**' properly. Search in other tab pages for the desired buffer. (Yegappan Lakshmanan)

Files: src/buffer.c, src/quickfix.c

Patch 7.0.147

Problem: When creating a session file and there are several tab pages and some windows have a local directory a short file name may be used when it's not valid. (Marius Roets)
A session with multiple tab pages may result in "No Name" buffers. (Bill McCarthy)

Solution: Don't enter tab pages when going through the list, only use a pointer to the first window in each tab page.

Use "tabedit" instead of "tabnew | edit" when possible.

Files: src/ex_docmd.c

Patch 7.0.148

Problem: When doing "call a.xyz()" and "xyz" does not exist in dictionary "a" there is no error message. (Yegappan Lakshmanan)

Solution: Add the error message.

Files: src/eval.c

Patch 7.0.149

Problem: When resizing a window that shows "~" lines the text sometimes jumps down.
Solution: Remove code that uses "~" lines in some situations. Fix the computation of the screen line of the cursor. Also set w_skipcol to handle very long lines.
Files: src/misc1.c, src/window.c

Patch 7.0.150

Problem: When resizing the Vim window scrollbinding doesn't work. (Yakov Lerner)
Solution: Do scrollbinding in set_shellsize().
Files: src/term.c

Patch 7.0.151

Problem: Buttons in file dialog are not according to Gnome guidelines.
Solution: Swap Cancel and Open buttons. (Stefano Zacchiroli)
Files: src/gui_gtk.c

Patch 7.0.152

Problem: Crash when using lesstif 2.
Solution: Fill in the extension field. (Ben Hutchings)
Files: src/gui_xmew.c

Patch 7.0.153

Problem: When using cscope and opening the temp file fails Vim crashes. (Kaya Bekiroglu)
Solution: Check for NULL pointer returned from mch_open().
Files: src/if_cscope.c

Patch 7.0.154

Problem: When '**foldnestmax**' is negative Vim can hang. (James Vega)
Solution: Avoid the fold level becoming negative.
Files: src/fold.c, src/syntax.c

Patch 7.0.155

Problem: When getchar() returns a mouse button click there is no way to get the mouse coordinates.
Solution: Add v:mouse_win, v:mouse_lnum and v:mouse_col.
Files: runtime/doc/eval.txt, src/eval.c, src/vim.h

Patch 7.0.156 (extra)

Problem: Vim doesn't compile for Amiga OS 4.
Solution: Various changes for Amiga OS4. (Peter Bengtsson)
Files: src/feature.h, src/mbyte.c, src/memfile.c, src/memline.c, src/os_amiga.c, src/os_amiga.h, src/pty.c

Patch 7.0.157

Problem: When a function is used recursively the profiling information is invalid. (Mikolaj Machowski)
Solution: Put the start time on the stack instead of in the function.
Files: src/eval.c

Patch 7.0.158

Problem: In a C file with ":set foldmethod=syntax", typing {<CR> on the last line results in the cursor being in a closed fold. (Gautam Iyer)

Solution: Open fold after inserting a new line.

Files: src/edit.c

Patch 7.0.159

Problem: When there is an I/O error in the swap file the cause of the error cannot be seen.

Solution: Use PERROR() instead of EMSG() where possible.

Files: src/memfile.c

Patch 7.0.160

Problem: ":@a" echoes the command, Vi doesn't do that.

Solution: Set the silent flag in the typeahead buffer to avoid echoing the command.

Files: src/ex_docmd.c, src/normal.c, src/ops.c, src/proto/ops.pro

Patch 7.0.161

Problem: Win32: Tab pages line popup menu isn't using the right encoding. (Yongwei Wu)

Solution: Convert the text when necessary. Also fixes the Find/Replace dialog title. (Yegappan Lakshmanan)

Files: src/gui_w48.c

Patch 7.0.162

Problem: "vim -o a b" when file "a" triggers the ATTENTION dialog, selecting "Quit" exits Vim instead of editing "b" only. When file "b" triggers the ATTENTION dialog selecting "Quit" or "Abort" results in editing file "a" in that window.

Solution: When selecting "Abort" exit Vim. When selecting "Quit" close the window. Also avoid hit-enter prompt when selecting Abort.

Files: src/buffer.c, src/main.c

Patch 7.0.163

Problem: Can't retrieve the position of a sign after it was set.

Solution: Add the netbeans interface getAnno command. (Xavier de Gaye)

Files: runtime/doc/netbeans.txt, src/netbeans.c

Patch 7.0.164

Problem: ":redir @+" doesn't work.

Solution: Accept "@+" just like "@*". (Yegappan Lakshmanan)

Files: src/ex_docmd.c

Patch 7.0.165

Problem: Using **CTRL-L** at the search prompt adds a "/" and other characters without escaping, causing the pattern not to match.

Solution: Escape special characters with a backslash.

Files: src/ex_getln.c

Patch 7.0.166

Problem: Crash in cscope code when connection could not be opened. (Kaya Bekiroglu)

Solution: Check for the file descriptor to be NULL.
Files: src/if_cscope.c

Patch 7.0.167

Problem: ":function" redefining a dict function doesn't work properly.
(Richard Emberson)

Solution: Allow a function name to be a number when it's a function
reference.

Files: src/eval.c

Patch 7.0.168

Problem: Using uninitialized memory and memory leak. (Dominique Pelle)

Solution: Use alloc_clear() instead of alloc() for w_lines. Free
b_ml.ml_stack after recovery.

Files: src/memline.c, src/window.c

Patch 7.0.169

Problem: With a Visual block selection, with the cursor in the left upper
corner, pressing "I" doesn't remove the highlighting. (Guopeng
Wen)

Solution: When checking if redrawing is needed also check if Visual
selection is still active.

Files: src/screen.c

Patch 7.0.170 (extra)

Problem: Win32: Using "gvim --remote-tab foo" when gvim is minimized while
it previously was maximized, un-maximizing doesn't work properly.
And the labels are not displayed properly when 'encoding' is
utf-8.

Solution: When minimized check for SW_SHOWMINIMIZED. When updating the tab
pages line use TCM_SETITEMW instead of TCM_INSERTITEMW. (Liu
Yubao)

Files: src/gui_w48.c

Patch 7.0.171 (extra)

Problem: VMS: A file name with multiple paths is written in the wrong file.

Solution: Get the actually used file name. (Zoltan Arpadffy)
Also add info to the :version command about compilation.

Files: src/Make_vms.mms, src/buffer.c, src/os_unix.c, src/version.c

Patch 7.0.172

Problem: Crash when recovering and quitting at the "press-enter" prompt.

Solution: Check for "msg_list" to be NULL. (Liu Yubao)

Files: src/ex_eval.c

Patch 7.0.173

Problem: ":call f().TT()" doesn't work. (Richard Emberson)

Solution: When a function returns a Dictionary or another composite continue
evaluating what follows.

Files: src/eval.c

Patch 7.0.174

Problem: ":mksession" doesn't restore window layout correctly in tab pages
other than the current one. (Zhibin He)

Solution: Use the correct topframe for producing the window layout commands.
Files: src/ex_docmd.c

Patch 7.0.175

Problem: The result of tr() is missing the terminating NUL. (Ingo Karkat)
Solution: Add the NUL.
Files: src/eval.c

Patch 7.0.176

Problem: ":emenu" isn't executed directly, causing the encryption key prompt to fail. (Life Jazzer)
Solution: Fix wrong #ifdef.
Files: src/menu.c

Patch 7.0.177

Problem: When the press-enter prompt gets a character from a non-remappable mapping, it's put back in the typeahead buffer as remappable, which may cause an endless loop.
Solution: Restore the non-remappable flag and the silent flag when putting a char back in the typeahead buffer.
Files: src/getchar.c, src/message.c, src/normal.c

Patch 7.0.178

Problem: When 'enc' is "utf-8" and 'ignorecase' is set the result of ":echo ("\xe4" == "\xe4")" varies.
Solution: In mb_strnicmp() avoid looking past NUL bytes.
Files: src/mbyte.c

Patch 7.0.179

Problem: Using ":recover" or "vim -r" without a swapfile crashes Vim.
Solution: Check for "buf" to be unequal NULL. (Yukihiro Nakadaira)
Files: src/memline.c

Patch 7.0.180 (extra, after 7.0.171)

Problem: VMS: build failed. Problem with swapfiles.
Solution: Add "compiled_arch". Always expand path and pass it to buf_modname(). (Zoltan Arpadffy)
Files: src/globals.h, src/memline.c, src/os_unix.c, runtime/menu.vim

Patch 7.0.181

Problem: When reloading a file that starts with an empty line, the reloaded buffer has an extra empty line at the end. (Motty Lentzitzky)
Solution: Delete all lines, don't use bufempty().
Files: src/fileio.c

Patch 7.0.182

Problem: When using a mix of undo and "g-" it may no longer be possible to go to every point in the undo tree. (Andy Wokula)
Solution: Correctly update pointers in the undo tree.
Files: src/undo.c

Patch 7.0.183

Problem: Crash in ":let" when redirecting to a variable that's being displayed. (Thomas Link)

Solution: When redirecting to a variable only do the assignment when stopping redirection to avoid that setting the variable causes a freed string to be accessed.
Files: src/eval.c

Patch 7.0.184

Problem: When the cscope program is called "mlcscope" the Cscope interface doesn't work.
Solution: Accept "\S*cscope:" instead of "cscope:". (Frodak D. Baksik)
Files: src/if_cscope.c

Patch 7.0.185

Problem: Multi-byte characters in a message are displayed with attributes from what comes before it.
Solution: Don't use the attributes for a multibyte character. Do use attributes for special characters. (Yukihiro Nakadaira)
Files: src/message.c

Patch 7.0.186

Problem: Get an ml_get error when 'encoding' is "utf-8" and searching for "/_s*/e" in an empty buffer. (Andrew Maykov)
Solution: Don't try getting the line just below the last line.
Files: src/search.c

Patch 7.0.187

Problem: Can't source a remote script properly.
Solution: Add the SourceCmd event. (Charles Campbell)
Files: runtime/doc/autocmd.txt, src/ex_cmds2.c, src/fileio.c, src/vim.h

Patch 7.0.188 (after 7.0.186)

Problem: Warning for wrong pointer type.
Solution: Add a type cast.
Files: src/search.c

Patch 7.0.189

Problem: Translated message about finding matches is truncated. (Yukihiro Nakadaira)
Solution: Enlarge the buffer. Also use vim_snprintf().
Files: src/edit.c

Patch 7.0.190

Problem: "syntax spell default" results in an error message.
Solution: Change 4 to 7 for STRNICMP(). (Raul Nunez de Arenas Coronado)
Files: src/syntax.c

Patch 7.0.191

Problem: The items used by getqflist() and setqflist() don't match.
Solution: Support the "bufnum" item for setqflist(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c

Patch 7.0.192

Problem: When 'swapfile' is switched off in an empty file it is possible that not all blocks are loaded into memory, causing ml_get errors later.

Solution: Rename "dont_release" to "mf_dont_release" and also use it to avoid using the cached line and locked block.
Files: src/globals.h, src/memfile.c, src/memline.c

Patch 7.0.193

Problem: Using --remote or --remote-tab with an argument that matches 'wildignore' causes a crash.
Solution: Check the argument count before using ARGVLIST[0].
Files: src/ex_cmds.c

Patch 7.0.194

Problem: Once an ml_get error is given redrawing part of the screen may cause it again, resulting in an endless loop.
Solution: Don't give the error message for a recursive call.
Files: src/memline.c

Patch 7.0.195

Problem: When a buffer is modified and 'autowriteall' is set, ":quit" results in an endless loop when there is a conversion error while writing. (Nikolai Weibull)
Solution: Make autowrite() return FAIL if the buffer is still changed after writing it.
/* put the cursor on the last char, for 'tw' formatting */
Files: src/ex_cmds2.c

Patch 7.0.196

Problem: When using ":vert ball" the computation of the mouse pointer position may be off by one column. (Stefan Karlsson)
Solution: Recompute the frame width when moving the vertical separator from one window to another.
Files: src/window.c

Patch 7.0.197 (extra)

Problem: Win32: Compiling with EXITFREE doesn't work.
Solution: Adjust a few #ifdefs. (Alexei Alexandrof)
Files: src/misc2.c, src/os_mswin.c

Patch 7.0.198 (extra)

Problem: Win32: Compiler warnings. No need to generate gvim.exe.mnf.
Solution: Add type casts. Use "*" for processorArchitecture. (George Reilly)
Files: src/Make_mvc.mak, src/eval.c, src/gvim.exe.mnf, src/misc2.c

Patch 7.0.199

Problem: When using multibyte characters the combination of completion and formatting may result in a wrong cursor position.
Solution: Don't decrement the cursor column, use dec_cursor(). (Yukihiro Nakadaira) Also check for the column to be zero.
Files: src/edit.c

Patch 7.0.200

Problem: Memory leaks when out of memory.
Solution: Free the memory.
Files: src/edit.c, src/diff.c

Patch 7.0.201

Problem: Message for ":diffput" about buffer not being in diff mode may be wrong.
Solution: Check for buffer in diff mode but not modifiable.
Files: src/diff.c

Patch 7.0.202

Problem: Problems on Tandem systems while compiling and at runtime.
Solution: Recognize root uid is 65535. Check select() return value for it not being supported. Avoid wrong function prototypes. Mention use of -lfloss. (Matthew Woehlke)
Files: src/Makefile, src/ex_cmds.c, src/fileio.c, src/main.c, src/osdef1.h.in, src/osdef2.h.in, src/os_unix.c, src/pty.c, src/vim.h

Patch 7.0.203

Problem: 0x80 characters in a register are not handled correctly for the "@" command.
Solution: Escape CSI and 0x80 characters. (Yukihiro Nakadaira)
Files: src/ops.c

Patch 7.0.204

Problem: Cscope: Parsing matches for listing isn't done properly.
Solution: Check for line number being found. (Yu Zhao)
Files: src/if_cscope.c

Patch 7.0.205 (after 7.0.203)

Problem: Can't compile.
Solution: Always include the vim_strsave_escape_csi function.
Files: src/getchar.c

Patch 7.0.206 (after 7.0.058)

Problem: Some characters of the "gb18030" encoding are not handled properly.
Solution: Do not use "cp936" as an alias for "gb18030" encoding. Instead initialize 'encoding' to "cp936".
Files: src/mbyte.c, src/option.c

Patch 7.0.207

Problem: After patch 2.0.203 CSI and K_SPECIAL characters are escaped when recorded and then again when the register is executed.
Solution: Remove escaping before putting the recorded characters in a register. (Yukihiro Nakadaira)
Files: src/getchar.c, src/ops.c, src/proto/getchar.pro

Patch 7.0.208 (after 7.0.171 and 7.0.180)

Problem: VMS: changes to path handling cause more trouble than they solve.
Solution: Revert changes.
Files: src/buffer.c, src/memline.c, src/os_unix.c

Patch 7.0.209

Problem: When replacing a line through Python the cursor may end up beyond the end of the line.
Solution: Check the cursor column after replacing the line.

Files: src/if_python.c

Patch 7.0.210

Problem: ":cbuffer" and ":lbuffer" always fail when the buffer is modified.
(Gary Johnson)

Solution: Support adding a !. (Yegappan Lakshmanan)

Files: runtime/doc/quickfix.txt, src/ex_cmds.h

Patch 7.0.211

Problem: With ":set cindent noai bs=0" using **CTRL-U** in Insert mode will delete auto-indent. After ":set ai" it doesn't.

Solution: Also check '**cindent**' being set. (Ryan Lortie)

Files: src/edit.c

Patch 7.0.212

Problem: The GUI can't be terminated with SIGTERM. (Mark Logan)

Solution: Use the signal protection in the GUI as in the console, allow signals when waiting for 100 msec or longer.

Files: src/ui.c

Patch 7.0.213

Problem: When '**spellfile**' has two regions that use the same sound folding using "z=" will cause memory to be freed twice. (Mark Woodward)

Solution: Clear the hashtable properly so that the items are only freed once.

Files: src/spell.c

Patch 7.0.214

Problem: When using `<f-args>` in a user command it's not possible to have an argument end in '\ '.

Solution: Change the handling of backslashes. (Yakov Lerner)

Files: runtime/doc/map.txt, src/ex_docmd.c

Patch 7.0.215 (extra)

Problem: Mac: Scrollbar size isn't set. Context menu has disabled useless Help entry. Call to MoreMasterPointers() is ignored.

Solution: Call SetControlViewSize() in gui_mch_set_scrollbar_thumb(). Use kCMHelpItemRemoveHelp for ContextualMenuSelect(). Remove call to MoreMasterPointers(). (Nicolas Weber)

Files: src/gui_mac.c

Patch 7.0.216

Problem: ":tab wincmd]" does not open a tab page. (Tony Mechelynck)

Solution: Copy the cmdmod.tab value to postponed_split_tab and use it.

Files: src/globals.h, src/ex_docmd.c, src/if_cscope.c, src/window.c

Patch 7.0.217

Problem: This hangs when pressing "n": ":%s/\n/,r/gc". (Ori Avtalion)

Solution: Set "skip_match" to advance to the next line.

Files: src/ex_cmds.c

Patch 7.0.218

Problem: "%B" in '**statusline**' always shows zero in Insert mode. (DervishD)

Solution: Remove the exception for Insert mode, check the column for being valid instead.

Files: src/buffer.c

Patch 7.0.219

Problem: When using the 'editexisting.vim' script and a file is being edited in another tab page the window is split. The "+123" argument is not used.

Solution: Make the tab page with the file the current tab page. Set v:swapcommand when starting up to the first "+123" or "-c" command line argument.

Files: runtime/macros/editexisting.vim, src/main.c

Patch 7.0.220

Problem: Crash when using winnr('#') in a new tab page. (Andy Wokula)

Solution: Check for not finding the window.

Files: src/eval.c

Patch 7.0.221

Problem: finddir() uses 'path' by default, where "." means relative to the current file. But it works relative to the current directory. (Tye Zdrojewski)

Solution: Add the current buffer name to find_file_in_path_option() for the relative file name.

Files: runtime/doc/eval.txt, src/eval.c

Patch 7.0.222

Problem: Perl indenting using 'cindent' works almost right.

Solution: Recognize '#' to start a comment. (Alex Manoussakis) Added '#' flag in 'cinoptions'.

Files: runtime/doc/indent.txt, src/misc1.c

Patch 7.0.223

Problem: Unprintable characters in completion text mess up the popup menu. (Gombault Damien)

Solution: Use strtrans() to make the text printable.

Files: src/charset.c, src/popupmnu.c

Patch 7.0.224

Problem: When expanding "##" spaces are escaped twice. (Pavol Juhas)

Solution: Don't escape the spaces that separate arguments.

Files: src/eval.c, src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 7.0.225

Problem: When using setline() in an InsertEnter autocommand and doing "A" the cursor ends up on the last byte in the line. (Yukihiro Nakadaira)

Solution: Only adjust the column when using setline() for the cursor line. Move it back to the head byte if necessary.

Files: src/eval.c, src/misc2.c

Patch 7.0.226

Problem: Display flickering when updating signs through the netbeans interface. (Xavier de Gaye)

Solution: Remove the redraw_later(CLEAR) call.

Files: src/netbeans.c

Patch 7.0.227

Problem: Crash when closing a window in the GUI. (Charles Campbell)
Solution: Don't call out_flush() from win_free().
Files: src/window.c

Patch 7.0.228

Problem: Cygwin: problem with symlink to DOS style path.
Solution: Invoke cygwin_conv_to_posix_path(). (Luca Masini)
Files: src/os_unix.c

Patch 7.0.229

Problem: When '**pastetoggle**' starts with Esc then pressing Esc in Insert mode will not time out. (Jeffery Small)
Solution: Use KL_PART_KEY instead of KL_PART_MAP, so that '**ttimeout**' applies to the '**pastetoggle**' key.
Files: src/getchar.c

Patch 7.0.230

Problem: After using ":lcd" a script doesn't know how to restore the current directory.
Solution: Add the haslocaldir() function. (Bob Hiestand)
Files: runtime/doc/usr_41.txt, runtime/doc/eval.txt, src/eval.c

Patch 7.0.231

Problem: When recovering from a swap file the page size is likely to be different from the minimum. The block used for the first page then has a buffer of the wrong size, causing a crash when it's reused later. (Zephaniah Hull)
Solution: Reallocate the buffer when the page size changes. Also check that the page size is at least the minimum value.
Files: src/memline.c

Patch 7.0.232 (extra)

Problem: Mac: doesn't support GUI tab page labels.
Solution: Add GUI tab page labels. (Nicolas Weber)
Files: src/feature.h, src/gui.c, src/gui.h, src/gui_mac.c, src/proto/gui_mac.pro

Patch 7.0.233 (extra)

Problem: Mac: code formatted badly.
Solution: Fix code formatting
Files: src/gui_mac.c

Patch 7.0.234

Problem: It's possible to use feedkeys() from a modeline. That is a security issue, can be used for a trojan horse.
Solution: Disallow using feedkeys() in the sandbox.
Files: src/eval.c

Patch 7.0.235

Problem: It is possible to use writefile() in the sandbox.
Solution: Add a few more checks for the sandbox.
Files: src/eval.c

Patch 7.0.236

Problem: Linux 2.4 uses sysinfo() with a mem_unit field, which is not backwards compatible.

Solution: Add an autoconf check for sysinfo.mem_unit. Let mch_total_mem() return Kbyte to avoid overflow.

Files: src/auto/configure, src/configure.in, src/config.h.in, src/option.c, src/os_unix.c

Patch 7.0.237

Problem: For root it is recommended to not use 'modeline', but in not-compatible mode the default is on.

Solution: Let 'modeline' default to off for root.

Files: runtime/doc/options.txt, src/option.c

Patch 7.0.238

Problem: Crash when ":match" pattern runs into 'maxmempattern'. (Yakov Lerner)

Solution: Don't free the regexp program of match_hl.

Files: src/screen.c

Patch 7.0.239

Problem: When using local directories and tab pages ":mksession" uses a short file name when it shouldn't. Window-local options from a modeline may be applied to the wrong window. (Teemu Likonen)

Solution: Add the did_lcd flag, use the full path when it's set. Don't use window-local options from the modeline when using the current window for another buffer in ":doautoall".

Files: src/fileio.c, src/ex_docmd.c

Patch 7.0.240

Problem: Crash when splitting a window in the GUI. (opposite of 7.0.227)

Solution: Don't call out_flush() from win_alloc(). Also avoid this for win_delete(). Also block autocommands while the window structure is invalid.

Files: src/window.c

Patch 7.0.241

Problem: ":windo throw 'foo'" loops forever. (Andy Wokula)

Solution: Detect that win_goto() doesn't work.

Files: src/ex_cmds2.c

Patch 7.0.242 (extra)

Problem: Win32: Using "--register" in a Vim that does not support OLE causes a crash.

Solution: Don't use EMSG() but mch_errmsg(). Check p_go for being NULL. (partly by Michael Wookey)

Files: src/gui_w32.c

Patch 7.0.243 (extra)

Problem: Win32: When GvimExt is built with MSVC 2005 or later, the "Edit with vim" context menu doesn't appear in the Windows Explorer.

Solution: Embed the linker manifest file into the resources of GvimExt.dll. (Mathias Michaelis)

Files: src/GvimExt/Makefile

Fixes after Vim 7.1a BETA:

The extra archive had CVS directories included below "farsi" and "runtime/icons". CVS was missing the farsi icon files.

Fix compiling with Gnome 2.18, undefine bind_textdomain_codeset. (Daniel Drake)

Mac: "make install" didn't copy rgb.txt.

When editing a compressed file while there are folds caused "ml_get" errors and some lines could be missing. When decompressing failed option values were not restored.

Patch 7.1a.001

Problem: Crash when downloading a spell file. (Szabolcs Horvat)

Solution: Avoid that did_set_spelllang() is used recursively when a new window is opened for the download.
Also avoid wiping out the wrong buffer.

Files: runtime/autoload/spellfile.vim, src/buffer.c, src/ex_cmds.c, src/spell.c

Patch 7.1a.002 (extra)

Problem: Compilation error with MingW.

Solution: Check for LPT00LTIPTTEXT to be defined.

Files: src/gui_w32.c

Fixes after Vim 7.1b BETA:

Made the Mzscheme interface build both with old and new versions of Mzscheme, using an #ifdef. (Sergey Khorev)

Mzscheme interface didn't link, missing function. Changed order of libraries in the configure script.

Ruby interface didn't compile on Mac. Changed #ifdef. (Lily Ballard)

Patch 7.1b.001 (extra)

Problem: Random text in a source file. No idea how it got there.

Solution: Delete the text.

Files: src/gui_w32.c

Patch 7.1b.002

Problem: When '**maxmem**' is large there can be an overflow in computations. (Thomas Wiegner)

Solution: Use the same mechanism as in mch_total_mem(): first reduce the multiplier as much as possible.

Files: src/memfile.c

=====

VERSION 7.2

version-7.2 version7.2

This section is about improvements made between version 7.1 and 7.2.

This is mostly a bug-fix release. The main new feature is floating point support. [Float](#)

Changed

changed-7.2

Changed the command line buffer name from "command-line" to "[Command Line]".

Removed optional ! for ":caddexpr", ":cgetexpr", ":cgetfile", ":laddexpr", ":lgetexpr" and ":lgetfile". They are not needed. (Yegappan Lakshmanan)

An offset for syntax matches worked on bytes instead of characters. That is inconsistent and can easily be done wrong. Use character offsets now. (Yukihiro Nakadaira)

The FileChangedShellPost event was also given when a file didn't change. (John Little)

When the current line is long (doesn't fit) the popup menu can't be seen. Display it below the screen line instead of below the text line. (Francois Ingelrest)

Switched to autoconf version 2.62.

Moved including fcntl.h to vim.h and removed it from all .c files.

Introduce macro STRMOVE(d, s), like STRCPY() for overlapping strings. Use it instead of mch_memmove(p, p + x, STRLEN(p + x) + 1).

Removed the bulgarian.vim keymap file, two more standard ones replace it. (Boyko Bantchev)

Increased the maximum number of tag matches for command line completion from 200 to 300.

Renamed help file sql.txt to ft_sql.txt and ada.txt to ft_ada.txt.

Added

added-7.2

New syntax files:

- CUDA (Timothy B. Terriberry)
- Cdrdao config (Nikolai Weibull)
- Coco/R (Ashish Shukla)
- Denyhosts config (Nikolai Weibull)
- Dtrace script (Nicolas Weber)
- Git output, commit, config, rebase, send-email (Tim Pope)
- HASTE and HastePreProc (M. Tranchero)

- HamL (Tim Pope)
- Host conf (Nikolai Weibull)
- Linden script (Timo Frenay)
- MS messages (Kevin Locke)
- PDF (Tim Pope)
- ProMeLa (Maurizio Tranchero)
- Reva Foth (Ron Aaron)
- Sass (Tim Pope)
- Symbian meta-makefile, MMP (Ron Aaron)
- VOS CM macro (Andrew McGill)
- XBL (Doug Kearns)

New tutor files:

- Made UTF-8 versions of all the tutor files.
- Greek renamed from ".gr" to ".el" (Greek vs Greece).
- Esperanto (Dominique Pelle)
- Croatian (Paul B. Mahol)

New filetype plugins:

- Cdrdao config (Nikolai Weibull)
- Debian control files (Debian Vim maintainers)
- Denyhosts (Nikolai Weibull)
- Dos .ini file (Nikolai Weibull)
- Dtrace script (Nicolas Weber)
- FnameScript (Nikolai Weibull)
- Git, Git config, Git commit, Git rebase, Git send-email (Tim Pope)
- HamL (Tim Pope)
- Host conf (Nikolai Weibull)
- Host access (Nikolai Weibull)
- Logtalk (Paulo Moura)
- MS messages (Kevin Locke)
- NSIS script (Nikolai Weibull)
- PDF (Tim Pope)
- Reva Forth (Ron Aaron)
- Sass (Tim Pope)

New indent files:

- DTD (Nikolai Weibull)
- Dtrace script (Nicolas Weber)
- Erlang (Csaba Hoch)
- FrameScript (Nikolai Weibull)
- Git config (Tim Pope)
- HamL (Tim Pope)
- Logtalk (Paulo Moura)
- Sass (Tim Pope)
- Tiny Fugue (Christian J. Robinson)

New compiler plugins:

- RSpec (Tim Pope)

New keymap files:

- Croatian (Paul B. Mahol)
- Russian Dvorak (Serhiy Boiko)
- Ukrainian Dvorak (Serhiy Boiko)

Removed plain Bulgarian, "bds" and phonetic are sufficient.

Other new runtime files:

Esperanto menu and message translations. (Dominique Pelle)

Finnish menu and message translations. (Flammie Pirinen)

Brazilian Portuguese message translations. (Eduardo Dobay)

Added floating point support. [Float](#)

Added argument to mode() to return a bit more detail about the current mode.
(Ben Schmidt)

Added support for BSD console mouse: [sysmouse](#) . (Paul B. Mahol)

Added the "newtab" value for the '[switchbuf](#)' option. (partly by Yegappan Lakshmanan)

Improved error messages for the netbeans interface. (Philippe Fremy)

Added support for using xterm mouse codes for screen. (Micah Cowan)

Added support for cross compiling:

Adjusted configure.in and added INSTALLcross.txt. (Marc Haisenko) Fixed mistakes in configure.in after that.

Don't use /usr/local/include and /usr/local/lib in configure. (Philip Prindeville)

For cross compiling the Cygwin version on Unix, change VIM.TLB to vim.tlb in src/vim.rc. (Tsuneo Nakagawa)

Added v:searchforward variable: What direction we're searching in. (Yakov Lerner)

Fixed

[fixed-7.2](#)

Patch 7.1.001

Problem: Still can't build with Gnome libraries.

Solution: Fix typo in bind_textdomain_codeset. (Mike Kelly)

Files: src/gui_gtk.c, src/gui_gtk_x11.c

Patch 7.1.002

Problem: Oracle Pro*C/C++ files are not detected.

Solution: Add the missing star. (Micah J. Cowan)

Files: runtime/filetype.vim

Patch 7.1.003 (extra)

Problem: The "Tear off this menu" message appears in the message history when using a menu. (Yongwei Wu)

Solution: Disable message history when displaying the menu tip.

Files: src/gui_w32.c

Patch 7.1.004

Problem: Crash when doing ":next directory". (Raphael Finkel)

Solution: Do not use "buf", it may be invalid after autocommands.
Files: src/ex_cmds.c

Patch 7.1.005

Problem: "cit" used on <foo></foo> deletes <foo>. Should not delete anything and start insertion, like "ci" does on "". (Michal Bozon)

Solution: Handle an empty object specifically. Made it work consistent for various text objects.

Files: src/search.c

Patch 7.1.006

Problem: Resetting 'modified' in a StdinReadPost autocommand doesn't work.

Solution: Set 'modified' before the autocommands instead of after it.

Files: src/buffer.c

Patch 7.1.007 (extra)

Problem: Mac: Context menu doesn't work on Intel Macs.

Scrollbars are not dimmed when Vim is not the active application.

Solution: Remove the test whether context menus are supported. They are always there in OS/X. Handle the dimming. (Nicolas Weber)

Files: src/gui_mac.c, src/gui.h

Patch 7.1.008

Problem: getfsize() returns a negative number for very big files.

Solution: Check for overflow and return -2.

Files: runtime/doc/eval.txt, src/eval.c

Patch 7.1.009

Problem: In diff mode, displaying the difference between a tab and spaces is not highlighted correctly.

Solution: Only change highlighting at the end of displaying a tab.

Files: src/screen.c

Patch 7.1.010

Problem: The Gnome session file doesn't restore tab pages.

Solution: Add SSOP_TABPAGES to the session flags. (Matias D'Ambrosio)

Files: src/gui_gtk_x11.c

Patch 7.1.011

Problem: Possible buffer overflow when \$VIMRUNTIME is very long. (Victor Stinner)

Solution: Use vim_snprintf().

Files: src/main.c

Patch 7.1.012

Problem: ":let &shiftwidth = 'asdf'" doesn't produce an error message.

Solution: Check for a string argument. (Chris Lubinski)

Files: src/option.c

Patch 7.1.013

Problem: ":syn include" only loads the first file, while it is documented as doing the equivalent of ":runtime!".

Solution: Change the argument to source_runtime(). (James Vega)

Files: src/syntax.c

Patch 7.1.014

Problem: Crash when doing C indenting. (Chris Monson)

Solution: Obtain the current line again after invoking cin_islabel().

Files: src/edit.c

Patch 7.1.015

Problem: MzScheme interface: current-library-collection-paths produces no list. Interface doesn't build on a Mac.

Solution: Use a list instead of a pair. (Bernhard Fisseni) Use "--framework" argument for MZSCHEME_LIBS in configure.

Files: src/configure.in, src/if_mzsch.c, src/auto/configure

Patch 7.1.016 (after patch 7.1.012)

Problem: Error message about setting 'diff' to a string.

Solution: Don't pass an empty string to set_option_value() when setting 'diff'.

Files: src/quickfix.c, src/popupmnu.c

Patch 7.1.017

Problem: ":confirm w" does give a prompt when 'readonly' is set, but not when the file permissions are read-only. (Michael Schaap)

Solution: Provide a dialog in both situations. (Chris Lubinski)

Files: src/ex_cmds.c, src/fileio.c, src/proto/fileio.pro

Patch 7.1.018

Problem: When 'virtualedit' is set a "p" of a block just past the end of the line inserts before the cursor. (Engelke)

Solution: Check for the cursor being just after the line (Chris Lubinski)

Files: src/ops.c

Patch 7.1.019

Problem: ":py" asks for an argument, ":py asd" then gives the error that ":py" isn't implemented. Should already happen for ":py".

Solution: Compare with ex_script_ni. (Chris Lubinski)

Files: src/ex_docmd.c

Patch 7.1.020

Problem: Reading from uninitialized memory when using a dialog. (Dominique Pelle)

Solution: In msg_show_console_dialog() append a NUL after every appended character.

Files: src/message.c

Patch 7.1.021 (after 7.1.015)

Problem: Mzscheme interface doesn't compile on Win32.

Solution: Fix the problem that 7.1.015 fixed in a better way. (Sergey Khorev)

Files: src/if_mzsch.c

Patch 7.1.022

Problem: When setting 'keymap' twice the b:keymap_name variable isn't set. (Milan Berta)

Solution: Don't unlet b:keymap_name for ":loadkeymap". (Martin Toft)

Files: src/digraph.c

Patch 7.1.023

Problem: "dw" in a line with one character deletes the line. Vi and nvi don't do this. (Kjell Arne Rekaa)

Solution: Check for one-character words especially.

Files: src/search.c

Patch 7.1.024

Problem: Using a pointer that has become invalid. (Chris Monson)

Solution: Obtain the line pointer again after we looked at another line.

Files: src/search.c

Patch 7.1.025

Problem: search() and searchpos() don't use match under cursor at start of line when using 'bc' flags. (Viktor Kojouharov)

Solution: Don't go to the previous line when the 'c' flag is present. Also fix that "j" doesn't move the cursor to the right column.

Files: src/eval.c, src/search.c

Patch 7.1.026

Problem: "[p" doesn't work in Visual mode. (David Brown)

Solution: Use checkclearop() instead of checkclearopq().

Files: src/normal.c

Patch 7.1.027

Problem: On Sun systems opening /dev/fd/N doesn't work, and they are used by process substitutions.

Solution: Allow opening specific character special files for Sun systems. (Gary Johnson)

Files: src/fileio.c, src/os_unix.h

Patch 7.1.028

Problem: Can't use last search pattern for ":sort". (Brian McKee)

Solution: When the pattern is empty use the last search pattern. (Martin Toft)

Files: runtime/doc/change.txt, src/ex_cmds.c

Patch 7.1.029 (after 7.1.019)

Problem: Can't compile when all interfaces are used. (Taylor Venable)

Solution: Only check for ex_script_ni when it's defined.

Files: src/ex_docmd.c

Patch 7.1.030

Problem: The "vimtutor" shell script checks for "vim6" but not for "vim7". (Christian Robinson)

Solution: Check for more versions, but prefer using "vim".

Files: src/vimtutor

Patch 7.1.031

Problem: virtcol([123, '\$']) doesn't work. (Michael Schaap)

Solution: When '\$' is used for the column number get the last column.

Files: runtime/doc/eval.txt, src/eval.c

Patch 7.1.032

Problem: Potential crash when editing a command line. (Chris Monson)
Solution: Check the position to avoid access before the start of an array.
Files: src/ex_getln.c

Patch 7.1.033

Problem: A buffer is marked modified when it was first deleted and then added again using a ":next" command. (John Mullin)
Solution: When checking if a buffer is modified use the BF_NEVERLOADED flag.
Files: src/option.c

Patch 7.1.034

Problem: Win64: A few compiler warnings. Problems with optimizer.
Solution: Use int instead of size_t. Disable the optimizer in one function. (George V. Reilly)
Files: src/eval.c, src/spell.c

Patch 7.1.035

Problem: After ":s./&/#" all listed lines have a line number. (Yakov Lerner)
Solution: Reset the line number flag when not using the "&" flag.
Files: src/ex_cmds.c

Patch 7.1.036

Problem: Completing ":echohl" argument should include "None". (Ori Avtalion) ":match" should have "none" too.
Solution: Add flags to use expand_highlight(). Also fix that when disabling FEAT_CMDL_COMPL compilation fails. (Chris Lubinski)
Files: src/eval.c, src/ex_docmd.c, src/ex_getln.c, src/proto/syntax.pro, src/syntax.c

Patch 7.1.037

Problem: strcpy() used for overlapping strings. (Chris Monson)
Solution: Use mch_memmove() instead.
Files: src/option.c

Patch 7.1.038

Problem: When '**expandtab**' is set then a Tab copied for '**copyindent**' is expanded to spaces, even when '**preserveindent**' is set. (Alexei Alexandrov)
Solution: Remove the check for '**expandtab**'. Also fix that ">>" doesn't obey '**preserveindent**'. (Chris Lubinski)
Files: src/misc1.c

Patch 7.1.039

Problem: A tag in a help file that starts with "help-tags" and contains a percent sign may make Vim crash. (Ulf Harnhammar)
Solution: Use puts() instead of fprintf().
Files: src/ex_cmds.c

Patch 7.1.040

Problem: ":match" only supports three matches.
Solution: Add functions clearmatches(), getmatches(), matchadd(), matchdelete() and setmatches(). Changed the data structures for

this. A small bug in syntax.c is fixed, so newly created highlight groups can have their name resolved correctly from their ID. (Martin Toft)

Files: runtime/doc/eval.txt, runtime/doc/pattern.txt,
runtime/doc/usr_41.txt, src/eval.c, src/ex_docmd.c,
src/proto/window.pro, src/screen.c, src/structs.h, src/syntax.c,
src/testdir/Makefile, src/testdir/test63.in,
src/testdir/test63.ok, src/window.c

Patch 7.1.041 (extra, after 7.1.040)

Problem: Some changes for patch 7.1.040 are in extra files.

Solution: Update the extra files.

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.1.042 (after 7.1.040)

Problem: Internal error when using matchadd(). (David Larson)

Solution: Check the third argument to be present before using the fourth argument. (Martin Toft)

Files: src/eval.c

Patch 7.1.043

Problem: In Ex mode using **CTRL-D** twice may cause a crash. Cursor isn't positioned properly after **CTRL-D**.

Solution: Set prev_char properly. Position the cursor correctly. (Antony Scriven)

Files: src/ex_getln.c

Patch 7.1.044

Problem: In Insert mode 0 **CTRL-T** deletes all indent, it should add indent. (Gautam Iyer)

Solution: Check for **CTRL-D** typed.

Files: src/edit.c

Patch 7.1.045

Problem: Unnecessary screen redrawing. (Jjgod Jiang)

Solution: Reset "must_redraw" after clearing the screen.

Files: src/screen.c

Patch 7.1.046

Problem: ":s" command removes combining characters. (Ron Aaron)

Solution: Copy composing characters individually. (Chris Lubinski)

Files: src/regexp.c

Patch 7.1.047

Problem: vim_regcomp() called with invalid argument. (Xiaozhou Liu)

Solution: Change TRUE to RE_MAGIC + RE_STRING.

Files: src/ex_eval.c

Patch 7.1.048

Problem: The matchparen plugin doesn't update the match when scrolling with the mouse wheel. (Ilya Bobir)

Solution: Set the match highlighting for text that can be scrolled into the viewable area without moving the cursor. (Chris Lubinski)

Files: runtime/plugin/matchparen.vim

Patch 7.1.049

Problem: Cannot compile GTK2 version with Hangul input feature.

Solution: Don't define FEAT_XFONTSET when using GTK2.

Files: src/feature.h

Patch 7.1.050

Problem: Possible crash when using C++ indenting. (Chris Monson)

Solution: Keep the line pointer to the line to compare with. Avoid going past the end of line.

Files: src/misc1.c

Patch 7.1.051

Problem: Accessing uninitialized memory when finding spell suggestions.

Solution: Don't try swapping characters at the end of a word.

Files: src/spell.c

Patch 7.1.052

Problem: When creating a new match not all fields are initialized, which may lead to unpredictable results.

Solution: Initialise rmm_ic and rmm_maxcol.

Files: src/window.c

Patch 7.1.053

Problem: Accessing uninitialized memory when giving a message.

Solution: Check going the length before checking for a NUL byte.

Files: src/message.c

Patch 7.1.054

Problem: Accessing uninitialized memory when displaying the fold column.

Solution: Add a NUL to the extra array. (Dominique Pelle). Also do this in a couple of other situations.

Files: src/screen.c

Patch 7.1.055

Problem: Using strcpy() with arguments that overlap.

Solution: Use mch_memmove() instead.

Files: src/buffer.c, src/charset.c, src/eval.c, src/ex_getln.c, src/misc1.c, src/regexp.c, src/termlib.c

Patch 7.1.056

Problem: More prompt does not behave correctly after scrolling back. (Randall W. Morris)

Solution: Avoid lines_left becomes negative. (Chris Lubinski) Don't check mp_last when deciding to show the more prompt. (Martin Toft)

Files: src/message.c

Patch 7.1.057

Problem: Problem with CursorHoldI when using "r" in Visual mode (Max Dyckhoff)

Solution: Ignore CursorHold(I) when getting a second character for a Normal mode command. Also abort the "r" command in Visual when a special key is typed.

Files: src/normal.c

Patch 7.1.058

Problem: When **'rightleft'** is set the completion menu is positioned wrong.
(Baha-Eddine MOKADEM)

Solution: Fix the completion menu. (Martin Toft)

Files: src/popupmnu.c, src/proto/search.pro, src/search.c

Patch 7.1.059

Problem: When in Ex mode and doing "g/^/vi" and then pressing **CTRL-C** Vim hangs and beeps. (Antony Scriven)

Solution: Clear "got_int" in the main loop to avoid the hang. When typing **CTRL-C** twice in a row abort the ":g" command. This is Vi compatible.

Files: src/main.c

Patch 7.1.060

Problem: Splitting quickfix window messes up window layout. (Marius Gedminas)

Solution: Compute the window size in a smarter way. (Martin Toft)

Files: src/window.c

Patch 7.1.061

Problem: Win32: When **'encoding'** is "latin1" **'ignorecase'** doesn't work for characters with umlaut. (Joachim Hofmann)

Solution: Do not use islower()/isupper()/tolower()/toupper() but our own functions. (Chris Lubinski)

Files: src/mbyte.c, src/regexp.c, src/vim.h

Patch 7.1.062 (after 7.1.038)

Problem: Indents of C comments can be wrong. (John Mullin)

Solution: Adjust ind_len. (Chris Lubinski)

Files: src/misc1.c

Patch 7.1.063 (after 7.1.040)

Problem: Warning for uninitialized variable.

Solution: Initialise it to NULL.

Files: src/ex_docmd.c

Patch 7.1.064

Problem: On Interix some files appear not to exist.

Solution: Remove the top bit from st_mode. (Ligesh)

Files: src/os_unix.c

Patch 7.1.065 (extra)

Problem: Win32: Compilation problem for newer version of w32api.

Solution: Only define __IID_DEFINED__ when needed. (Chris Sutcliffe)

Files: src/Make_ming.mak, src/iid_ole.c

Patch 7.1.066

Problem: When **'bomb'** is set or reset the file should be considered modified. (Tony Mechelynck)

Solution: Handle like **'endofline'**. (Martin Toft)

Files: src/buffer.c, src/fileio.c, src/option.c, src/structs.h

Patch 7.1.067

Problem: **'thesaurus'** doesn't work when **'infercase'** is set. (Mohsin)
Solution: Don't copy the characters being completed but check the case and apply it to the suggested word. Also fix that the first word in the thesaurus line is not used. (Martin Toft)
Files: src/edit.c

Patch 7.1.068

Problem: When **'equalalways'** is set and splitting a window, it's possible that another small window gets bigger.
Solution: Only equalize window sizes when after a split the windows are smaller than another window. (Martin Toft)
Files: runtime/doc/options.txt, runtime/doc/windows.txt, src/window.c

Patch 7.1.069

Problem: GTK GUI: When using confirm() without a default button there still is a default choice.
Solution: Ignore Enter and Space when there is no default button. (Chris Lubinski)
Files: src/gui_gtk.c

Patch 7.1.070 (extra)

Problem: Win32 GUI: When using confirm() without a default button there still is a default choice.
Solution: Set focus on something else than a button. (Chris Lubinski)
Files: src/gui_w32.c

Patch 7.1.071 (after 7.1.040)

Problem: Regexp patterns are not tested.
Solution: Add a basic test, to be expanded later.
Also add (commented-out) support for valgrind.
Files: src/testdir/Makefile, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.1.072 (extra, after 7.1.041 and 7.1.071)

Problem: Some changes for patch 7.1.071 are in extra files.
Solution: Update the extra files. Also fix a few warnings from the DOS test makefile.
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.1.073 (after 7.1.062)

Problem: Wrong cursor position and crash when **'preserveindent'** is set. (Charles Campbell)
Solution: Handle the situation that we start without indent. (Chris Lubinski)
Files: src/misc1.c

Patch 7.1.074

Problem: Crash when calling string() on a recursively nested List.
Solution: Check result value for being NULL. (Yukihiro Nakadaira)
Files: src/eval.c

Patch 7.1.075

Problem: ":let v:statusmsg" reads memory already freed.
Solution: Don't set v:statusmsg when listing it.
Files: src/eval.c

Patch 7.1.076

Problem: Another strcpy() with overlapping arguments.
Solution: Use mch_memmove(). (Dominique Pelle) And another one.
Files: src/ex_docmd.c, src/normal.c

Patch 7.1.077

Problem: Using "can_spell" without initializing it. (Dominique Pelle)
Solution: Set a default for get_syntax_attr().
Files: src/syntax.c

Patch 7.1.078

Problem: Dropping a file name on gvim that contains a CSI byte doesn't work when editing the command line.
Solution: Escape the CSI byte when inserting in the input buffer. (Yukihiro Nakadaira)
Files: src/gui.c, src/ui.c

Patch 7.1.079

Problem: When the locale is "C" and 'encoding' is "latin1" then the "@" character in 'isfname', 'isprint', etc. doesn't pick up accented characters.
Solution: Instead of isalpha() use MB_ISLOWER() and MB_ISUPPER().
Files: src/charset.c, src/macros.h

Patch 7.1.080 (extra)

Problem: Compiler warnings for using "const char *" for "char *".
Solution: Add type casts. (Chris Sutcliffe)
Files: src/GvimExt/gvimext.cpp

Patch 7.1.081

Problem: Command line completion for a shell command: "cat </tmp/file<Tab>" doesn't work.
Solution: Start the file name at any character that can't be in a file name. (Martin Toft)
Files: src/ex_docmd.c

Patch 7.1.082

Problem: After a ":split" the matchparen highlighting isn't there.
Solution: Install a WinEnter autocommand. Also fixes that after ":NoMatchParen" only the current window is updated. (Martin Toft)
Files: runtime/doc/pi_paren.txt, runtime/plugin/matchparen.vim

Patch 7.1.083 (after 7.1.081)

Problem: Command line completion doesn't work with wildcards.
Solution: Add vim_isfilec_or_wc() and use it. (Martin Toft)
Files: src/charset.c, src/proto/charset.pro, src/ex_docmd.c

Patch 7.1.084

Problem: Using the "-nb" argument twice causes netbeans not to get fileOpened events.

Solution: Change "&" to "&&". (Xavier de Gaye)
Files: src/ex_cmds.c

Patch 7.1.085

Problem: ":e fold.c" then ":sp fold.c" results in folds of original window to disappear. (Akita Noek)

Solution: Invoke foldUpdateAll() for all windows of the changed buffer. (Martin Toft)

Files: src/ex_cmds.c

Patch 7.1.086

Problem: Crash when using specific Python syntax highlighting. (Quirk)

Solution: Check for a negative index, coming from a keyword match at the start of a line from a saved state.

Files: src/syntax.c

Patch 7.1.087

Problem: Reading past ":cscope find" command. Writing past end of a buffer.

Solution: Check length of the argument before using the pattern. Use vim_strncpy(). (Dominique Pelle)

Files: if_cscope.c

Patch 7.1.088 (extra)

Problem: The coordinates used by ":winpos" differ from what getwinposx() and getwinposy() return.

Solution: Use MoveWindowStructure() instead of MoveWindow(). (Michael Henry)

Files: src/gui_mac.c

Patch 7.1.089

Problem: ":let loaded_getscriptPlugin" doesn't clear to eol, result is "#1in".

Solution: Clear to the end of the screen after displaying the first variable value.

Files: src/eval.c

Patch 7.1.090

Problem: Compiler warning on Mac OS X 10.5.

Solution: Don't redeclare sigaltstack(). (Hisashi T Fujinaka)

Files: src/os_unix.c

Patch 7.1.091 (extra)

Problem: Win32: Can't embed Vim inside another application.

Solution: Add the --windowid argument. (Nageshwar)

Files: runtime/doc/gui_w32.txt, runtime/doc/starting.txt, runtime/doc/vi_diff.txt, src/globals.h, src/gui_w32.c, src/main.c

Patch 7.1.092 (extra, after 7.1.088)

Problem: Wrong arguments for MoveWindowStructure().

Solution: Remove "TRUE". (Michael Henry)

Files: src/gui_mac.c

Patch 7.1.093

Problem: Reading past end of a screen line when determining cell width. (Dominique Pelle)

Solution: Add an argument to mb_off2cells() for the maximum offset.
Files: src/globals.h, src/gui.c, src/mbyte.c, src/proto/mbyte.pro,
src/screen.c

Patch 7.1.094

Problem: When checking if syntax highlighting is present, looking in the current buffer instead of the specified one.

Solution: Use "buf" instead of "curbuf".

Files: src/syntax.c

Patch 7.1.095

Problem: The FocusLost and FocusGained autocommands are triggered asynchronously in the GUI. This may cause arbitrary problems.

Solution: Put the focus event in the input buffer and handle it when ready for it.

Files: src/eval.c, src/getchar.c, src/gui.c, src/gui_gtk_x11.c,
src/keymap.h

Patch 7.1.096

Problem: Reading past end of a string when resizing Vim. (Dominique Pelle)

Solution: Check the string pointer before getting the char it points to.

Files: src/message.c

Patch 7.1.097

Problem: ":setlocal stl=%!1+1" does not work.

Solution: Adjust check for pointer. (Politz)

Files: src/option.c

Patch 7.1.098

Problem: ":call s:var()" doesn't work if "s:var" is a Funcref. (Andy Wokula)

Solution: Before converting "s:" into a script ID, check if it is a Funcref.

Files: src/eval.c

Patch 7.1.099

Problem: When the **'keymap'** and **'paste'** options have a non-default value, ":mkexrc" and ":mksession" do not correctly set the options.

Solution: Set the options with side effects before other options.

Files: src/option.c

Patch 7.1.100

Problem: Win32: Executing cscope doesn't always work properly.

Solution: Use another way to invoke cscope. (Mike Williams)

Files: src/if_cscope.c, src/if_cscope.h, src/main.c,
src/proto/if_cscope.pro

Patch 7.1.101

Problem: Ruby: The Buffer.line= method does not work.

Solution: Add the "self" argument to set_current_line(). (Jonathan Hankins)

Files: src/if_ruby.c

Patch 7.1.102

Problem: Perl interface doesn't compile with new version of Perl.

Solution: Add two variables to the dynamic library loading. (Suresh Govindachar)

Files: src/if_perl.xs

Patch 7.1.103

Problem: Using "dw" with the cursor past the end of the last line (using CTRL-\ **CTRL-O** from Insert mode) deletes a character. (Tim Chase)

Solution: Don't move the cursor back when the movement failed.

Files: src/normal.c

Patch 7.1.104 (after 7.1.095)

Problem: When '**lazyredraw**' is set a focus event causes redraw to be postponed until a key is pressed.

Solution: Instead of not returning from vgetc() when a focus event is encountered return K_IGNORE. Add plain_vgetc() for when the caller doesn't want to get K_IGNORE.

Files: src/digraph.c, src/edit.c, src/ex_cmds.c, src/ex_getln.c, src/getchar.c, src/normal.c, src/proto/getchar.pro, src/window.c

Patch 7.1.105

Problem: Internal error when using "0 ? {'a': 1} : {}". (A.Politz)

Solution: When parsing a dictionary value without using the value, don't try obtaining the key name.

Files: src/eval.c

Patch 7.1.106

Problem: ":messages" doesn't quit listing on ":".

Solution: Break the loop when "got_int" is set.

Files: src/message.c

Patch 7.1.107

Problem: When doing a block selection and using "s" to change the text, while triggering auto-indenting, causes the wrong text to be repeated in other lines. (Adri Verhoef)

Solution: Compute the change of indent and compensate for that.

Files: src/ops.c

Patch 7.1.108 (after 7.1.100)

Problem: Win32: Compilation problems in Cscope code. (Jeff Lanzarotta)

Solution: Use (long) instead of (intptr_t) when it's not defined.

Files: src/if_cscope.c

Patch 7.1.109

Problem: GTK: when there are many tab pages, clicking on the arrow left of the labels moves to the next tab page on the right. (Simeon Bird)

Solution: Check the X coordinate of the click and pass -1 as value for the left arrow.

Files: src/gui_gtk_x11.c, src/term.c

Patch 7.1.110 (after 7.1.102)

Problem: Win32: Still compilation problems with Perl.

Solution: Change the #ifdefs. (Suresh Govindachar)

Files: src/if_perl.xs

Patch 7.1.111

Problem: When using ":vimgrep" with the "j" flag folds from another buffer

may be displayed. (A.Politz)
Solution: When not jumping to another buffer update the folds.
Files: src/quickfix.c

Patch 7.1.112

Problem: Using input() with a wrong argument may crash Vim. (A.Politz)
Solution: Init the input() return value to NULL.
Files: src/eval.c

Patch 7.1.113

Problem: Using map() to go over an empty list causes memory to be freed twice. (A.Politz)
Solution: Don't clear the typeval in restore_vimvar().
Files: src/eval.c

Patch 7.1.114

Problem: Memory leak in getmatches().
Solution: Don't increment the refcount twice.
Files: src/eval.c

Patch 7.1.115 (after 7.1.105)

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Init variable to NULL.
Files: src/eval.c

Patch 7.1.116

Problem: Cannot display Unicode characters above 0x10000.
Solution: Remove the replacement with a question mark when UNICODE16 is not defined. (partly by Nicolas Weber)
Files: src/screen.c

Patch 7.1.117

Problem: Can't check whether Vim was compiled with Gnome. (Tony Mechelynck)
Solution: Add gui_gnome to the has() list.
Files: src/eval.c

Patch 7.1.118 (after 7.1.107)

Problem: Compiler warning for Visual C compiler.
Solution: Add typecast. (Mike Williams)
Files: src/ops.c

Patch 7.1.119

Problem: Crash when **'cmdheight'** set to very large value. (A.Politz)
Solution: Limit **'cmdheight'** to **'lines'** minus one. Store right value of **'cmdheight'** when running out of room.
Files: src/option.c, src/window.c

Patch 7.1.120

Problem: Can't properly check memory leaks while running tests.
Solution: Add an argument to garbagecollect(). Delete functions and variables in the test scripts.
Files: runtime/doc/eval.txt src/eval.c, src/globals.h, src/main.c, src/testdir/Makefile, src/testdir/test14.in, src/testdir/test26.in, src/testdir/test34.in,

src/testdir/test45.in, src/testdir/test47.in,
src/testdir/test49.in, src/testdir/test55.in,
src/testdir/test56.in, src/testdir/test58.in,
src/testdir/test59.in, src/testdir/test60.in,
src/testdir/test60.vim, src/testdir/test62.in,
src/testdir/test63.in, src/testdir/test64.in,

Patch 7.1.121

Problem: Using ":cd %:h" when editing a file in the current directory results in an error message for using an empty string.
Solution: When "%:h" results in an empty string use ".".
Files: src/eval.c

Patch 7.1.122

Problem: Mac: building Vim.app fails. Using wrong architecture.
Solution: Use line continuation for the gui_bundle dependency. Detect the system architecture with "uname -a".
Files: src/main.aap

Patch 7.1.123

Problem: Win32: ":edit foo ~ foo" expands "~".
Solution: Change the call to expand_env().
Files: src/ex_docmd.c, src/misc1.c, src/proto/misc1.pro, src/option.c

Patch 7.1.124 (extra)

Problem: Mac: When dropping a file on Vim.app that is already in the buffer list (from .viminfo) results in editing an empty, unnamed buffer. (Axel Kielhorn) Also: warning for unused variable.
Solution: Move to the buffer of the first argument. Delete unused variable.
Files: src/gui_mac.c

Patch 7.1.125

Problem: The TermResponse autocommand event is not always triggered. (Aron Griffis)
Solution: When unblocking autocommands check if v:termresponse changed and trigger the event then.
Files: src/buffer.c, src/diff.c, src/ex_getln.c, src/fileio.c, src/globals.h, src/misc2.c, src/proto/fileio.pro, src/window.c

Patch 7.1.126 (extra)

Problem: ":vimgrep */*" fails when a BufRead autocommand changes directory. (Bernhard Kuhn)
Solution: Change back to the original directory after loading a file. Also: use shorten_fname1() to avoid duplicating code.
Files: src/buffer.c, src/ex_docmd.c, src/fileio.c, src/gui_gtk.c, src/gui_w48.c, src/proto/ex_docmd.pro, src/proto/fileio.pro, src/quickfix.c

Patch 7.1.127

Problem: Memory leak when doing cmdline completion. (Dominique Pelle)
Solution: Free "orig" argument of ExpandOne() when it's not used.
Files: src/ex_getln.c

Patch 7.1.128 (extra)

Problem: Build problems with new version of Cygwin.
Solution: Remove `-D__IID_DEFINED__`, like with MingW. (Guopeng Wen)
Files: `src/Make_cyg.mak`

Patch 7.1.129 (extra)
Problem: Win32: Can't get the user name when it is longer than 15 characters.
Solution: Use `UNLEN` instead of `MAX_COMPUTERNAME_LENGTH`. (Alexei Alexandrov)
Files: `src/os_win32.c`

Patch 7.1.130
Problem: Crash with specific order of undo and redo. (A.Politz)
Solution: Clear and adjust pointers properly. Add `u_check()` for debugging.
Files: `src/undo.c`, `src/structs.h`

Patch 7.1.131
Problem: `":mksession"` always adds `":setlocal autoread"`. (Christian J. Robinson)
Solution: Skip boolean global/local option using global value.
Files: `src/option.c`

Patch 7.1.132
Problem: `getpos("'>")` may return a negative column number for a Linewise selection. (A.Politz)
Solution: Don't add one to `MAXCOL`.
Files: `src/eval.c`

Patch 7.1.133 (after 7.1.126)
Problem: `shorten_fname1()` linked when it's not needed.
Solution: Add `#ifdef`.
Files: `src/fileio.c`

Patch 7.1.134 (extra)
Problem: Win32: Can't build with VC8
Solution: Detect the MSVC version instead of using `NMAKE_VER`. (Mike Williams)
Files: `src/Make_mvc.mak`

Patch 7.1.135
Problem: Win32: When editing a file `c:\tmp\foo` and `c:\tmp\foo` we have two buffers for the same file. (Suresh Govindachar)
Solution: Invoke `FullName_save()` when a path contains `"/"` or `"\"`.
Files: `src/buffer.c`

Patch 7.1.136
Problem: Memory leak when using Ruby syntax highlighting. (Dominique Pelle)
Solution: Free the contained-in list.
Files: `src/syntax.c`

Patch 7.1.137
Problem: Build failure when using `EXITFREE`. (Dominique Pelle)
Solution: Add an `#ifdef` around using `clip_exclude_prog`.
Files: `src/misc2.c`

Patch 7.1.138

Problem: The Perl Msg() function doesn't stop when "q" is typed at the more prompt. (Hari Krishna Dara)
Solution: Check got_int.
Files: src/if_perl.xs

Patch 7.1.139

Problem: When using marker folding and ending Insert mode with **CTRL-C** the current fold is truncated. (Fred Kater)
Solution: Ignore got_int while updating folds.
Files: src/fold.c

Patch 7.1.140

Problem: v:count is set only after typing a non-digit, that makes it difficult to make a nice mapping.
Solution: Set v:count while still typing the count.
Files: src/normal.c

Patch 7.1.141

Problem: GTK: -geom argument doesn't support a negative offset.
Solution: Compute position from the right/lower corner.
Files: src/gui_gtk_x11.c

Patch 7.1.142

Problem: ":redir @A>" doesn't work.
Solution: Ignore the extra ">" also when appending. (James Vega)
Files: src/ex_docmd.c

Patch 7.1.143

Problem: Uninitialized memory read when diffing three files. (Dominique Pelle)
Solution: Remove "+ !notset" so that we don't use fields that were not computed.
Files: src/diff.c

Patch 7.1.144

Problem: After ":diffup" cursor can be in the wrong position.
Solution: Force recomputing the cursor position.
Files: src/diff.c

Patch 7.1.145

Problem: Insert mode completion: When using the popup menu, after completing a word and typing a non-word character Vim is still completing the same word, following **CTRL-N** doesn't work.
Insert mode Completion: When using **CTRL-X** 0 and there is only "struct." before the cursor, typing one char to reduce the matches, then BS completion stops.
Solution: When typing a character that is not part of the item being completed, stop complete mode. For whole line completion also accept a space. For file name completion stop at a path separator.
For omni completion stay in completion mode even if completing with empty string.
Files: src/edit.c

Patch 7.1.146 (extra)

Problem: VMS: Files with a very rare record organization (VFC) cannot be properly written by Vim.

On older VAX systems mms runs into a syntax error.

Solution: Check for this special situation. Do not wrap a comment, make it one long line. (Zoltan Arpadffy)

Files: src/fileio.c, src/Make_vms.mms

Patch 7.1.147 (after 7.1.127)

Problem: Freeing memory already freed when completing user name. (Meino Cramer)

Solution: Use a flag to remember if "orig" needs to be freed.

Files: src/ex_getln.c

Patch 7.1.148

Problem: Some types are not found by configure.

Solution: Test for the sys/types.h header file. (Sean Boudreau)

Files: src/configure.in, src/auto/configure

Patch 7.1.149

Problem: GTK GUI: When the completion popup menu is used scrolling another window by the scrollbar is OK, but using the scroll wheel it behaves like `<Enter>`.

Solution: Ignore K_MOUSEDOWN and K_MOUSEUP. Fix redrawing the popup menu.

Files: src/edit.c, src/gui.c

Patch 7.1.150

Problem: When **'clipboard'** has "unnamed" using "p" in Visual mode doesn't work correctly. (Jianrong Yu)

Solution: When **'clipboard'** has "unnamed" also obtain the selection when getting the default register.

Files: src/ops.c

Patch 7.1.151

Problem: Using whole line completion with **'ignorecase'** and **'infercase'** set and the line is empty get an lalloc(0) error.

Solution: Don't try changing case for an empty match. (Matthew Wozniski)

Files: src/edit.c

Patch 7.1.152

Problem: Display problem when **'hls'** and **'cursorcolumn'** are set and searching for "\$". (John Mullin) Also when scrolling horizontally when **'wrap'** is off.

Solution: Keep track of the column where highlighting was set. Check the column offset when skipping characters.

Files: src/screen.c

Patch 7.1.153

Problem: Compiler warnings on SGI. Undefined XpmAllocColor (Charles Campbell)

Solution: Add type casts. Init st_dev and st_ino separately. Don't use type casts for vim_snprintf() when HAVE_STDARG_H is defined. Define XpmAllocColor when needed.

Files: src/eval.c, src/ex_cmds.c, src/fileio.c, src/misc2.c,
src/gui_xmebw.c

Patch 7.1.154

Problem: Compiler warning for signed/unsigned compare.
Solution: Add type cast.
Files: src/screen.c

Patch 7.1.155

Problem: Crash when **'undolevels'** is 0 and repeating "udd". (James Vega)
Solution: When there is only one branch use u_freeheader() to delete it.
Files: src/undo.c

Patch 7.1.156

Problem: Overlapping arguments for strcpy() when expanding command line variables.
Solution: Use mch_memmove() instead of STRCPY(). Also fix a few typos. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.1.157

Problem: In Ex mode, ":" gives an error at end-of-file. (Michael Hordijk)
Solution: Only give an error for an empty line, not for a comment.
Files: src/ex_docmd.c

Patch 7.1.158 (extra)

Problem: Win32 console: When **'encoding'** is "utf-8" and typing Alt-y the result is wrong. Win32 GUI: Alt-y results in "u" when **'encoding'** is "cp1250" (Lukas Cerman)
Solution: For utf-8 don't set the 7th bit in a byte, convert to the correct byte sequence. For cp1250, when conversion to **'encoding'** results in the 7th bit not set, set the 7th bit after conversion.
Files: src/os_win32.c, src/gui_w48.c

Patch 7.1.159

Problem: strcpy() has overlapping arguments.
Solution: Use mch_memmove() instead. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.1.160

Problem: When a focus autocommand is defined, getting or losing focus causes the hit-enter prompt to be redrawn. (Bjorn Winckler)
Solution: Overwrite the last line.
Files: src/message.c

Patch 7.1.161

Problem: Compilation errors with tiny features and EXITFREE.
Solution: Add #ifdefs. (Dominique Pelle)
Files: src/edit.c, src/misc2.c

Patch 7.1.162

Problem: Crash when using a modifier before "while" or "for". (A.Politz)
Solution: Skip modifiers when checking for a loop command.
Files: src/proto/ex_docmd.pro, src/ex_docmd.c, src/ex_eval.c

Patch 7.1.163

Problem: Warning for the unknown option **'bufsecret'**.
Solution: Remove the lines .vim that use this option. (Andy Wokula)
Files: runtime/menu.vim

Patch 7.1.164

Problem: Reading past end of regexp pattern. (Dominique Pelle)
Solution: Use utf_ptr2len().
Files: src/regexp.c

Patch 7.1.165

Problem: Crash related to getting X window ID. (Dominique Pelle)
Solution: Don't trust the window ID that we got in the past, check it every time.
Files: src/os_unix.c

Patch 7.1.166

Problem: Memory leak for using "gp" in Visual mode.
Solution: Free memory in put_register(). (Dominique Pelle)
Files: src/ops.c

Patch 7.1.167

Problem: Xxd crashes when using "xxd -b -c 110". (Debian bug 452789)
Solution: Allocate more memory. Fix check for maximum number of columns.
Files: src/xxd/xxd.c

Patch 7.1.168 (extra)

Problem: Win32 GUI: Since patch 7.1.095, when the Vim window does not have focus, clicking in it doesn't position the cursor. (Juergen Kraemer)
Solution: Don't reset s_button_pending just after receiving focus.
Files: src/gui_w48.c

Patch 7.1.169

Problem: Using uninitialized variable when system() fails. (Dominique Pelle)
Solution: Let system() return an empty string when it fails.
Files: src/eval.c

Patch 7.1.170

Problem: Valgrind warning for overlapping arguments for strcpy().
Solution: Use mch_memmove() instead. (Dominique Pelle)
Files: src/getchar.c

Patch 7.1.171

Problem: Reading one byte before allocated memory.
Solution: Check index not to become negative. (Dominique Pelle)
Files: src/ex_getln.c

Patch 7.1.172

Problem: When **'buftype'** is "acwrite" Vim still checks if the file or directory exists before overwriting.
Solution: Don't check for overwriting when the buffer name is not a file

name.
Files: src/ex_cmds.c

Patch 7.1.173

Problem: Accessing freed memory. (Dominique Pelle)
Solution: Don't call reg_getline() to check if a line is the first in the file.
Files: src/regexp.c

Patch 7.1.174

Problem: Writing NUL past end of a buffer.
Solution: Copy one byte less when using strncat(). (Dominique Pelle)
Files: src/ex_cmds.c, src/ex_docmd.c,

Patch 7.1.175

Problem: <BS> doesn't work with some combination of 'sts', 'linebreak' and 'backspace'. (Francois Ingelrest)
Solution: When adding white space results in not moving back delete one character.
Files: src/edit.c

Patch 7.1.176

Problem: Building with Aap fails when the "compiledby" argument contains '<' or '>' characters. (Alex Yeh)
Solution: Change how quoting is done in the Aap recipe.
Files: src/main.aap

Patch 7.1.177

Problem: Freeing memory twice when in debug mode while reading a script.
Solution: Ignore script input while in debug mode.
Files: src/ex_cmds2.c, src/getchar.c, src/globals.h

Patch 7.1.178

Problem: "%" doesn't work on "/* comment // comment */".
Solution: Don't handle the "//" in "*/" as a C++ comment. (Markus Heidelberg)
Files: src/search.c

Patch 7.1.179

Problem: Need to check for TCL 8.5.
Solution: Adjust configure script. (Alexey Froloff)
Files: src/configure.in, src/auto/configure

Patch 7.1.180

Problem: Regexp patterns not tested sufficiently.
Solution: Add more checks to the regexp test.
Files: src/testdir/test64.in, src/testdir/test64.ok

Patch 7.1.181

Problem: Accessing uninitialized memory in Farsi mode. (Dominique Pelle)
Solution: Only invoke lrF_sub() when there is something to do.
Files: src/ex_cmds.c

Patch 7.1.182

Problem: When using tab pages and an argument list the session file may contain wrong "next" commands. (Alexander Bluem)
Solution: Use "argu" commands and only when needed.
Files: src/ex_docmd.c

Patch 7.1.183

Problem: "Internal error" for ":echo matchstr('a', 'a\[&'])" (Mitanu Paul)
Solution: Inside "\%[]" detect \&, \| and \) as an error.
Files: src/regexp.c

Patch 7.1.184

Problem: Crash when deleting backwards over a line break in Insert mode.
Solution: Don't advance the cursor when it's already on the NUL after a line. (Matthew Wozniski)
Files: src/normal.c

Patch 7.1.185

Problem: Using "gR" with a multibyte encoding and typing a CR pushes characters onto the replace stack incorrectly, resulting in BS putting back the wrong characters. (Paul B. Mahol)
Solution: Push multibyte characters onto the replace stack in reverse byte order. Add replace_push_mb().
Files: src/edit.c, src/misc1.c, src/proto/edit.pro

Patch 7.1.186

Problem: "expand('<afile>')" returns a bogus value after changing directory. (Dave Fishburn)
Solution: Copy "autocmd_fname" to allocated memory and expand to full filename. Shorten the path when expanding <afile>.
Files: src/ex_docmd.c, src/fileio.c

Patch 7.1.187

Problem: Win32 GUI: Custom completion using system() no longer works after patch 7.1.104. (Erik Falor)
Solution: Loop when safe_vgetc() returns K_IGNORE.
Files: src/ex_getln.c

Patch 7.1.188

Problem: When 'showmode' is off the message for changing a readonly file is given in the second column instead of the first. (Payl B. Mahol)
Solution: Put the W10 message in the first column.
Files: src/edit.c

Patch 7.1.189 (after 7.1.104)

Problem: Patch 7.1.104 was incomplete.
Solution: Also call plain_vgetc() in ask_yesno().
Files: src/misc1.c

Patch 7.1.190

Problem: Cursor after end-of-line: "iA sentence.<Esc>)"
Solution: Move cursor back and make motion inclusive.
Files: src/normal.c

Patch 7.1.191

Problem: Win32 GUI: after patch 7.1.168 there is still a problem when clicking in a scrollbar. (Juergen Jottkaerr)
Solution: Don't check the input buffer when dragging the scrollbar.
Files: src/gui.c

Patch 7.1.192

Problem: With Visual block selection, "s" and typing something, **CTRL-C** doesn't stop Vim from repeating the replacement in other lines, like happens for "I".
Solution: Check for "got_int" to be set.
Files: src/ops.c

Patch 7.1.193

Problem: Some Vim 5.x digraphs are missing in Vim 7, even though the character pairs are not used. (Philippe de Muyter)
Solution: Add those Vim 5.x digraphs that don't conflict with others.
Files: src/digraph.c

Patch 7.1.194

Problem: ":echo glob('~/{})" results in /home/user//.
Solution: Don't add a slash if there already is one.
Files: src/os_unix.c

Patch 7.1.195

Problem: '0 mark doesn't work for "~/foo ~ foo".
Solution: Don't expand the whole file name, only "~/".
Files: src/mark.c

Patch 7.1.196 (extra)

Problem: Win32 GUI: "\n" in a tooltip doesn't cause a line break. (Erik Falor)
Solution: Use the TTM_SETMAXTIPWIDTH message.
Files: src/gui_w32.c

Patch 7.1.197

Problem: Mac: "make install" doesn't work when prefix defined.
Solution: Pass different arguments to "make installruntime". (Jjgod Jiang)
Files: src/Makefile

Patch 7.1.198

Problem: Hang when using ":s/\n//gn". (Burak Gorkemli)
Solution: Set "skip_match".
Files: src/ex_cmds.c

Patch 7.1.199

Problem: Can't do command line completion for a specific file name extension.
Solution: When the pattern ends in "\$" don't add a star for completion and remove the "\$" before matching with file names.
Files: runtime/doc/cmdline.txt, src/ex_getln.c

Patch 7.1.200 (after 7.1.177 and 7.1.182)

Problem: Compiler warnings for uninitialized variables.

Solution: Init variables.
Files: src/ex_cmds2.c, src/ex_docmd.c

Patch 7.1.201

Problem: When reading stdin '**fenc**' and '**ff**' are not set.
Solution: Set the options after reading stdin. (Ben Schmidt)
Files: src/fileio.c

Patch 7.1.202

Problem: Incomplete utf-8 byte sequence is not checked for validity.
Solution: Check the bytes that are present for being valid. (Ben Schmidt)
Files: src/mbyte.c

Patch 7.1.203

Problem: When '**virtualedit**' is "onemore" then "99|" works but ":normal 99|" doesn't. (Andy Wokula)
Solution: Check for "onemore" flag in check_cursor_col().
Files: src/misc2.c

Patch 7.1.204 (extra)

Problem: Win32: Using the example at '**balloonexpr**' the balloon disappears after four seconds and then comes back again. Also moves the mouse pointer a little bit. (Yongwei Wu)
Solution: Set the autopop time to 30 seconds (the max value). (Sergey Khorev) Move the mouse two pixels forward and one back to end up in the same position (really!).
Files: src/gui_w32.c

Patch 7.1.205

Problem: Can't get the operator in an ":omap".
Solution: Add the "v:operator" variable. (Ben Schmidt)
Files: runtime/doc/eval.txt, src/eval.c, src/normal.c, src/vim.h

Patch 7.1.206

Problem: Compiler warnings when using MODIFIED_BY.
Solution: Add type casts. (Ben Schmidt)
Files: src/version.c

Patch 7.1.207

Problem: Netbeans: "remove" cannot delete one line.
Solution: Remove partial lines and whole lines properly. Avoid a memory leak. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.1.208

Problem: On Alpha get an unaligned access error.
Solution: Store the dictitem pointer before using it. (Matthew Luckie)
Files: src/eval.c

Patch 7.1.209

Problem: GTK: When using the netrw plugin and doing ":gui" Vim hangs.
Solution: Stop getting a selection after three seconds. This is a hack.
Files: src/gui_gtk_x11.c

Patch 7.1.210

Problem: Listing mapping for 0xdb fails when **'encoding'** is utf-8. (Tony Mechelynck)
Solution: Recognize K_SPECIAL KS_EXTRA KE_CSI as a CSI byte.
Files: src/mbyte.c

Patch 7.1.211

Problem: The matchparen plugin may take an unexpected amount of time, so that it looks like Vim hangs.
Solution: Add a timeout to searchpair(), searchpairpos(), search() and searchpos(). Use half a second timeout in the plugin.
Files: runtime/doc/eval.txt, runtime/plugin/matchparen.vim, src/edit.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/normal.c, src/proto/eval.pro, src/proto/ex_cmds2.pro, src/proto/search.pro, src/search.c

Patch 7.1.212

Problem: Accessing a byte before a line.
Solution: Check that the column is 1 or more. (Dominique Pelle)
Files: src/edit.c

Patch 7.1.213

Problem: A ":tabedit" command that results in the "swap file exists" dialog and selecting "abort" doesn't close the new tab. (Al Budden)
Solution: Pass "old_curwin" to do_exedit().
Files: src/ex_docmd.c

Patch 7.1.214

Problem: ":!s/g\n\zs1//\" deletes characters from the first line. (A Politz)
Solution: Start replacing in the line where the match starts.
Files: src/ex_cmds.c

Patch 7.1.215

Problem: It is difficult to figure out what syntax items are nested at a certain position.
Solution: Add the synstack() function.
Files: runtime/doc/eval.txt, src/eval.c, src/proto/syntax.pro, src/syntax.c

Patch 7.1.216

Problem: Variants of --remote-tab are not mentioned for "vim --help".
Solution: Display optional -wait and -silent.
Files: src/main.c

Patch 7.1.217

Problem: The "help-tags" tag may be missing from runtime/doc/tags when it was generated during "make install".
Solution: Add the "+t" argument to ":helptags" to force adding the tag.
Files: runtime/doc/Makefile, runtime/doc/various.txt, src/ex_cmds.c, src/ex_cmds.h

Patch 7.1.218

Problem: A syntax region without a "keepend", containing a region with "extend" could be truncated at the end of the containing region.

Solution: Do not call `syn_update_ends()` when there are no keepend items.
Files: `src/syntax.c`

Patch 7.1.219 (after 7.1.215)

Problem: `synstack()` returns situation after the current character, can't see the state for a one-character region.

Solution: Don't update ending states in the requested column.

Files: `runtime/doc/eval.txt`, `src/eval.c`, `src/hardcopy.c`,
`src/proto/syntax.pro`, `src/screen.c`, `src/spell.c`, `src/syntax.c`

Patch 7.1.220

Problem: When a `)` or word movement command moves the cursor back from the end of the line it may end up on the trail byte of a multibyte character. It's also moved back when it isn't needed.

Solution: Add the `adjust_cursor()` function.

Files: `src/normal.c`

Patch 7.1.221

Problem: When inserting a `"`, triggering the `matchparen` plugin, the following highlighting may be messed up.

Solution: Before triggering the `CursorMovedI` autocommands update the display to update the stored syntax stacks for the change.

Files: `src/edit.c`

Patch 7.1.222 (after 7.1.217)

Problem: Wildcards in argument of `:"helptags"` are not expanded. (Marcel Svitalsky)

Solution: Expand wildcards in the directory name.

Files: `src/ex_cmds.c`

Patch 7.1.223

Problem: `glob()` doesn't work properly when `'shell'` is `"sh"` or `"bash"` and the expanded name contains spaces, `'~'`, single quotes and other special characters. (Adri Verhoef, Charles Campbell)

Solution: For Posix shells define a `vimglob()` function to list the matches instead of using `"echo"` directly.

Files: `src/os_unix.c`

Patch 7.1.224

Problem: When using `"vim -F -o file1 file2"` only one window is right-to-left. Same for `"-H"`. (Ben Schmidt)

Solution: use `set_option_value()` to set `'rightleft'`.

Files: `src/main.c`

Patch 7.1.225

Problem: Using uninitialized value when `XGetWMNormalHints()` fails.

Solution: Check the return value. (Dominique Pelle)

Files: `src/os_unix.c`

Patch 7.1.226

Problem: Command line completion doesn't work when a file name contains a `'&'` character.

Solution: Accept all characters in a file name, except ones that end a command or white space.

Files: src/ex_docmd.c

Patch 7.1.227

Problem: Hang in syntax HL when moving over a ")". (Dominique Pelle)

Solution: Avoid storing a syntax state in the wrong position in the list of remembered states.

Files: src/syntax.c

Patch 7.1.228

Problem: When '**foldmethod**' is "indent" and a fold is created with ">>" it can't be closed with "zc". (Daniel Shahaf)

Solution: Reset the "small" flag of a fold when adding a line to it.

Files: src/fold.c

Patch 7.1.229

Problem: A fold is closed when it shouldn't when '**foldmethod**' is "indent" and backspacing a non-white character so that the indent increases.

Solution: Keep the fold open after backspacing a character.

Files: src/edit.c

Patch 7.1.230

Problem: Memory leak when executing SourceCmd autocommands.

Solution: Free the memory. (Dominique Pelle)

Files: src/ex_cmds2.c

Patch 7.1.231

Problem: When shifting lines the change is acted upon multiple times.

Solution: Don't have shift_line() call changed_bytes.

Files: src/edit.c, src/ops.c, src/proto/edit.pro, src/proto/ops.pro

Patch 7.1.232 (after 7.1.207 and 7.1.211)

Problem: Compiler warnings with MSVC.

Solution: Add type casts. (Mike Williams)

Files: src/ex_cmds2.c, src/netbeans.c

Patch 7.1.233

Problem: Crash when doing Insert mode completion for a user defined command. (Yegappan Lakshmanan)

Solution: Don't use the non-existing command line.

Files: src/ex_getln.c

Patch 7.1.234

Problem: When diff'ing three files the third one isn't displayed correctly. (Gary Johnson)

Solution: Compute the size of diff blocks correctly when merging blocks. Compute filler lines correctly when scrolling.

Files: src/diff.c

Patch 7.1.235

Problem: Pattern matching is slow when using a lot of simple patterns.

Solution: Avoid allocating memory by not freeing it when it's not so much. (Alexei Alexandrov)

Files: src/regexp.c

Patch 7.1.236

Problem: When using **'incsearch'** and **'hlsearch'** a complicated pattern may make Vim hang until **CTRL-C** is pressed.

Solution: Add the **'redrawtime'** option.

Files: runtime/doc/options.txt, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/gui.c, src/misc1.c, src/normal.c, src/option.c, src/quickfix.c, src/regexp.c, src/proto/regexp.pro, src/proto/search.pro, src/search.c, src/screen.c, src/option.h, src/spell.c, src/structs.h, src/syntax.c, src/tag.c, src/vim.h

Patch 7.1.237

Problem: Compiler warning on an Alpha processor in Motif code.

Solution: Change a typecast. (Adri Verhoef)

Files: src/gui_motif.c

Patch 7.1.238

Problem: Using the 'c' flag with searchpair() may cause it to fail. Using the 'r' flag doesn't work when **'wrapscan'** is set. (A.Politz)

Solution: Only use the 'c' flag for the first search, not for repeating. When using 'r' imply 'W'. (Antony Scriven)

Files: src/eval.c

Patch 7.1.239 (after 7.1.233)

Problem: Compiler warning for sprintf() argument.

Solution: Add a typecast. (Nico Weber)

Files: src/ex_getln.c

Patch 7.1.240

Problem: When "gUe" turns a German sharp s into SS the operation stops before the end of the word. Latin2 has the same sharp s but it's not changed to SS there.

Solution: Make sure all the characters are operated upon. Detect the sharp s in latin2. Also fixes that changing case of a multibyte character that changes the byte count doesn't always work.

Files: src/ops.c

Patch 7.1.241

Problem: Focus change events not always ignored. (Erik Falor)

Solution: Ignore K_IGNORE in Insert mode in a few more places.

Files: src/edit.c

Patch 7.1.242 (after 7.1.005)

Problem: "cib" doesn't work properly on "(x)". (Tim Pope)

Solution: Use ltoreq() instead of lt(). Also fix "ciT" on "<a>x".

Files: src/search.c

Patch 7.1.243 (after 7.1.240)

Problem: "U" doesn't work on all text in Visual mode. (Adri Verhoef)

Solution: Loop over all the lines to be changed. Add tests for this.

Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.1.244

Problem: GUI may have part of the command line cut off.

Solution: Don't round the number of lines up, always round down.
(Tony Houghton, Scott Dillard)
Files: src/gui.c

Patch 7.1.245

Problem: Pressing CTRL-\ three times causes Vim to quit. (Ranganath Rao).
Also for f CTRL-\ CTRL-\
Solution: When going to cooked mode in mch_delay() set a flag to ignore
SIGQUIT.
Files: src/os_unix.c

Patch 7.1.246

Problem: Configure hangs when the man pager is something strange. (lorien)
Solution: Set MANPAGER and PAGER to "cat". (Micah Cowan)
Files: src/auto/configure, src/configure.in

Patch 7.1.247

Problem: When using Netbeans backspacing in Insert mode skips a character
now and then. (Ankit Jain)
Solution: Avoid calling netbeans_removed(), it frees the line pointer.
(partly by Dominique Pelle).
Files: src/misc1.c

Patch 7.1.248

Problem: Can't set the '"' mark. Can't know if setpos() was successful.
Solution: Allow setting the '"' mark with setpos(). Have setpos() return a
value indicating success/failure.
Files: runtime/doc/eval.txt, src/eval.c, src/mark.c

Patch 7.1.249

Problem: After "U" the cursor can be past end of line. (Adri Verhoef)
Solution: Adjust the cursor position in u_undoline().
Files: src/undo.c

Patch 7.1.250

Problem: ":setglobal fenc=anything" gives an error message in a buffer
where 'modifiable' is off. (Ben Schmidt)
Solution: Don't give an error if 'modifiable' doesn't matter.
Files: src/option.c

Patch 7.1.251

Problem: Using freed memory when spell checking enabled.
Solution: Obtain the current line again after calling spell_move_to().
(Dominique Pelle)
Files: src/screen.c

Patch 7.1.252 (after 7.1.243)

Problem: Test 39 fails when the environment has a utf-8 locale. (Dominique
Pelle)
Solution: Force 'encoding' to be latin1.
Files: src/testdir/test39.in

Patch 7.1.253

Problem: ":sort" doesn't work in a one line file. (Patrick Texier)

Solution: Don't sort if there is only one line. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.1.254

Problem: Tests 49 and 55 fail when the locale is French.
Solution: Using C messages for test 49. Filter the error message in test 55 such that it works when the number is halfway the message.
Files: src/testdir/test49.in, src/testdir/test55.in

Patch 7.1.255

Problem: Vim doesn't support utf-32. (Yongwei Wu)
Solution: Add aliases for utf-32, it's the same as ucs-4.
Files: src/mbyte.c

Patch 7.1.256

Problem: findfile() also returns directories.
Solution: Cleanup the code for finding files and directories in a list of directories. Remove the ugly global ff_search_ctx.
Files: src/eval.c, src/misc2.c, src/vim.h, src/tag.c

Patch 7.1.257

Problem: Configure can't always find the Tcl header files.
Solution: Also look in /usr/local/include/tcl\$tclver and /usr/include/tcl\$tclver (James Vega)
Files: src/auto/configure, src/configure.in

Patch 7.1.258

Problem: Crash when doing "d/\n/e" and **'virtualedit'** is "all". (Andy Wokula)
Solution: Avoid that the column becomes negative. Also fixes other problems with the end of a pattern match is in column zero. (A.Politz)
Files: src/search.c

Patch 7.1.259

Problem: Cursor is in the wrong position when **'rightleft'** is set, **'encoding'** is "utf-8" and on an illegal byte. (Dominique Pelle)
Solution: Only put the cursor in the first column when actually on a double-wide character. (Yukihiro Nakadaira)
Files: src/screen.c

Patch 7.1.260

Problem: Cursor positioning problem after ^@ wrapping halfway when **'encoding'** is utf-8.
Solution: Only count a position for printable characters. (partly by Yukihiro Nakadaira)
Files: src/charset.c

Patch 7.1.261

Problem: When a 2 byte BOM is detected Vim uses UCS-2, which doesn't work for UTF-16 text. (Tony Mechelynck)
Solution: Default to UTF-16.
Files: src/fileio.c, src/testdir/test42.ok

Patch 7.1.262

Problem: Can't get the process ID of Vim.

Solution: Implement getpid().
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.1.263

Problem: The filetype can consist of two dot separated names. This works for syntax and ftplugin, but not for indent. (Brett Stahlman)
Solution: Use split() and loop over each dot separated name.
Files: runtime/indent.vim

Patch 7.1.264

Problem: Crash when indenting lines. (Dominique Pelle)
Solution: Set the cursor column when changing the cursor line.
Files: src/ops.c, src/misc1.c

Patch 7.1.265

Problem: When '**isfname**' contains a space, cmdline completion can hang. (James Vega)
Solution: Reset the "len" variable.
Files: src/ex_docmd.c

Patch 7.1.266

Problem: When the version string returned by the terminal contains unexpected characters, it is used as typed input. (James Vega)
Solution: Assume the escape sequence ends in a letter.
Files: src/term.c

Patch 7.1.267

Problem: When changing folds cursor may be positioned in the wrong place.
Solution: Call changed_window_setting_win() instead of changed_window_setting().
Files: src/fold.c

Patch 7.1.268

Problem: Always shows "+" at end of screen line with: ":set listchars=eol:\$,extends:+ nowrap list cursorline" (Gary Johnson)
Solution: Check for lcs_eol_one instead of lcs_eol.
Files: src/screen.c

Patch 7.1.269

Problem: The matchparen plugin has an arbitrary limit for the number of lines to look for a match.
Solution: Rely on the searchpair() timeout.
Files: runtime/plugin/matchparen.vim

Patch 7.1.270

Problem: "?:foo?" matches in current line since patch 7.1.025. (A.Politz)
Solution: Remove the SEARCH_START flag.
Files: src/ex_docmd.c, src/search.c

Patch 7.1.271

Problem: In a Vim build without autocommands, checking a file that was changed externally causes the current buffer to be changed unexpectedly. (Karsten Hopp)
Solution: Store "curbuf" instead of "buf".

Files: src/fileio.c

Patch 7.1.272

Problem: The special buffer name [Location List] is not used for a buffer displayed in another tab page.

Solution: Use FOR_ALL_TAB_WINDOWS instead of FOR_ALL_WINDOWS. (Hiroaki Nishihara)

Files: src/buffer.c

Patch 7.1.273

Problem: When profiling on Linux Vim exits early. (Liu Yubao)

Solution: When profiling don't exit on SIGPROF.

Files: src/Makefile, src/os_unix.c

Patch 7.1.274 (after 7.1.272)

Problem: Compiler warning for optimized build.

Solution: Init win to NULL.

Files: src/buffer.c

Patch 7.1.275 (extra)

Problem: Mac: ATSUI and '**antialias**' don't work properly together.

Solution: Fix this and the input method. (Jjgod Jiang)

Files: src/vim.h, src/gui_mac.c

Patch 7.1.276

Problem: "gw" uses '**formatexpr**', even though the docs say it doesn't.

Solution: Don't use '**formatexpr**' for "gw".

Files: src/vim.h, src/edit.c, src/ops.c, src/proto/ops.pro

Patch 7.1.277

Problem: Default for '**paragraphs**' misses some items (Colin Watson)

Solution: Add TP, HP, Pp, Lp and It to '**paragraphs**'. (James Vega)

Files: runtime/doc/options.txt, src/option.c

Patch 7.1.278 (extra, after 7.1.275)

Problem: Build failure when USE_CARBONKEYHANDLER is not defined.

Solution: Remove #ifdef.

Files: src/gui_mac.c

Patch 7.1.279

Problem: When using cscope temporary files are left behind.

Solution: Send the quit command to cscope and give it two seconds to exit nicely before killing it. (partly by Dominique Pelle)

Files: src/if_cscope.c

Patch 7.1.280 (after 7.1.275)

Problem: Mac: build problems when not using multibyte feature. (Nicholas Stallard)

Solution: Don't define USE_IM_CONTROL when not using multibyte.

Files: src/vim.h

Patch 7.1.281 (after 7.1.279)

Problem: sa.sa_mask is not initialized. Cscope may not exit.

Solution: Use sigemptyset(). Use SIGKILL instead of SIGTERM. (Dominique

Pelle)
Files: src/if_cscope.c

Patch 7.1.282 (extra)

Problem: Win64: Edit with Vim context menu isn't installed correctly.
Compiler warnings and a few other things.

Solution: Add [and] to entry of class name. Use UINT_PTR instead of UINT.
And fixes for other things. (George V. Reilly)

Files: src/GvimExt/Makefile, src/dosinst.c, src/if_ole.cpp, src/if_ole.h,
src/if_ole.idl, src/INSTALLpc.txt, src/Make_mvc.mak,
src/os_win32.c,

Patch 7.1.283

Problem: Non-extra part for 7.1.282.

Solution: Various changes.

Files: src/ex_docmd.c, src/globals.h, src/if_cscope.c, src/main.c,
src/mark.c, src/netbeans.c, src/popupmnu.c, src/vim.h,
src/window.c

Patch 7.1.284

Problem: Compiler warnings for functions without prototype.

Solution: Add the function prototypes. (Patrick Texier)

Files: src/eval.c, src/quickfix.c

Patch 7.1.285 (extra)

Problem: Mac: dialog hotkeys don't work.

Solution: Add hotkey support. (Dan Sandler)

Files: src/gui_mac.c

Patch 7.1.286 (after 7.1.103)

Problem: "w" at the end of the buffer moves the cursor past the end of the
line. (Markus Heidelberg)

Solution: Move the cursor back from the NUL when it was moved forward.

Files: src/normal.c

Patch 7.1.287

Problem: Crash when reversing a list after using it. (Andy Wokula)

Solution: Update the pointer to the last used element. (Dominique Pelle)

Files: src/eval.c

Patch 7.1.288 (after 7.1.281)

Problem: Cscope still leaves behind temp files when using gvim.

Solution: When getting the ECHILD error loop for a while until cscope exits.
(Dominique Pelle)

Files: if_cscope.c

Patch 7.1.289

Problem: When EXITFREE is defined and 'acd' is set freed memory is used.
(Dominique Pelle)

Solution: Reset p_acd before freeing all buffers.

Files: src/misc2.c

Patch 7.1.290

Problem: Reading bytes that were not written when spell checking and a line

has a very large indent.
Solution: Don't copy the start of the next line when it only contains spaces. (Dominique Pelle)
Files: src/spell.c

Patch 7.1.291 (after 7.1.288)
Problem: Compiler warning.
Solution: Change 50 to 50L.
Files: src/if_cscope.c

Patch 7.1.292
Problem: When using a pattern with "\@<=" the submatches can be wrong. (Brett Stahlman)
Solution: Save the submatches when attempting a look-behind match.
Files: src/regexp.c

Patch 7.1.293
Problem: Spell checking considers super- and subscript characters as word characters.
Solution: Recognize the Unicode super and subscript characters.
Files: src/spell.c

Patch 7.1.294
Problem: Leaking memory when executing a shell command.
Solution: Free memory when not able to save for undo. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.1.295
Problem: Vimtutor only works with vim, not gvim.
Solution: Add the -g flag to vimtutor. (Dominique Pelle) Add gvimtutor.
Files: src/Makefile, src/gvimtutor, src/vimtutor, runtime/doc/vimtutor.1

Patch 7.1.296
Problem: SELinux is not supported.
Solution: Detect the selinux library and use mch_copy_sec(). (James Vega)
Files: src/auto/configure, src/config.h.in, src/configure.in, src/fileio.c, src/memfile.c, src/os_unix.c, src/proto/os_unix.pro

Patch 7.1.297
Problem: When using the search/replace dialog the parenmatch highlighting can be wrong. (Tim Duncan)
Solution: In the GUI redraw function invoke the CursorMoved autocmd.
Files: src/gui.c

Patch 7.1.298 (after 7.1.295)
Problem: src/gvimtutor is not distributed.
Solution: Add it to the list of distributed files.
Files: Filelist

Patch 7.1.299
Problem: Filetype detection doesn't work properly for file names ending in a part that is ignored and contain a space or other special characters.
Solution: Escape the special characters using the new fnameescape function.

Files: runtime/doc/eval.txt, runtime/filetype.vim, src/eval.c,
src/ex_getln.c, src/proto/ex_getln.pro, src/vim.h

Patch 7.1.300

Problem: Value of asmsyntax argument isn't checked for valid characters.
Solution: Only accepts letters and digits.
Files: runtime/filetype.vim

Patch 7.1.301

Problem: When the "File/Save" menu is used in Insert mode, a tab page label
is not updated to remove the "+".
Solution: Call draw_tabline() from showruler(). (Bjorn Winckler)
Files: src/screen.c

Patch 7.1.302 (after 7.1.299)

Problem: Compilation error on MS-Windows.
Solution: Don't use xp_shell when it's not defined.
Files: src/ex_getln.c

Patch 7.1.303 (after 7.1.302)

Problem: Compilation error on MS-Windows, again.
Solution: Declare p.
Files: src/ex_getln.c

Patch 7.1.304

Problem: Shortpath_for_invalid_fname() does not work correctly and is
unnecessary complex.
Solution: Clean up shortpath_for_invalid_fname(). (mostly by Yegappan
Lakshmanan)
Files: src/eval.c

Patch 7.1.305

Problem: Editing a compressed file with special characters in the name
doesn't work properly.
Solution: Escape special characters.
Files: runtime/autoload/gzip.vim

Patch 7.1.306

Problem: Some Unicode characters are handled like word characters while
they are symbols.
Solution: Adjust the table for Unicode classification.
Files: src/mbyte.c

Patch 7.1.307

Problem: Many warnings when compiling with Python 2.5.
Solution: Use ssize_t instead of int for some types. (James Vega)
Files: src/if_python.c

Patch 7.1.308

Problem: When in readonly mode ":options" produces an error.
Solution: Reset '**readonly**'. (Gary Johnson)
Files: runtime/optwin.vim

Patch 7.1.309

Problem: Installing and testing with a shadow directory doesn't work.
(James Vega)
Solution: Add "po" to the list of directories to link. Also link the Vim
scripts in testdir. And a few more small fixes.
Files: src/Makefile

Patch 7.1.310

Problem: Incomplete utf-8 byte sequence at end of the file is not detected.
Accessing memory that wasn't written.
Solution: Check the last bytes in the buffer for being a valid utf-8
character. (mostly by Ben Schmidt)
Also fix that the reported line number of the error was wrong.
Files: src/fileio.c

Patch 7.1.311

Problem: Compiler warning for missing sentinel in X code.
Solution: Change 0 to NULL. (Markus Heidelberg)
Files: src/mbyte.c

Patch 7.1.312

Problem: The .po files have mistakes in error numbers.
Solution: Search for these mistakes in the check script. (Dominique Pelle)
Files: src/po/check.vim

Patch 7.1.313

Problem: When the netbeans interface setModified call is used the status
lines and window title are not updated.
Solution: Redraw the status lines and title. (Philippe Fremy)
Files: src/netbeans.c

Patch 7.1.314

Problem: The value of '**pastetoggle**' is written to the session file without
any escaping. (Randall Hansen)
Solution: Use put_escstr(). (Ben Schmidt)
Files: src/option.c

Patch 7.1.315

Problem: Crash with specific search pattern using look-behind match.
(Andreas Politz)
Solution: Also save the value of "need_clear_subexpr".
Files: src/regexp.c

Patch 7.1.316

Problem: When '**cscopetag**' is set ":tag" gives an error message instead of
going to the next tag in the tag stack.
Solution: Don't call do_cstag() when there is no argument. (Mark Goldman)
Files: src/ex_docmd.c

Patch 7.1.317

Problem: Compiler warnings in Motif calls.
Solution: Change zero to NULL. (Dominique Pelle)
Files: src/gui_motif.c

Patch 7.1.318

Problem: Memory leak when closing xsmc connection. Crash on exit when using Lesstif.
Solution: Don't close the X display to work around a Lesstif bug. Free clientid. Also fix a leak for Motif and Athena. (Dominique Pelle)
Files: src/gui_x11.c, src/os_unix.c

Patch 7.1.319

Problem: When a register has an illegal utf-8 sequence, pasting it on the command line causes an illegal memory access.
Solution: Use mb_cptr2char_adv(). (Dominique Pelle)
Files: src/ex_getln.c

Patch 7.1.320 (extra)

Problem: Win64: Warnings while compiling Python interface.
Solution: Use PyInt in more places. Also update version message for the console. (George Reilly)
Files: src/if_python.c, src/version.c

Patch 7.1.321 (extra)

Problem: Win32 / Win64: Install file is outdated.
Solution: Update the text for recent compiler. (George Reilly)
Files: src/INSTALLpc.txt

Patch 7.1.322

Problem: Can't get start of Visual area in an `<expr>` mapping.
Solution: Add the 'v' argument to getpos().
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.1.323

Problem: Test 19 fails with some termcaps. (Dominique Pelle)
Solution: Set the t_kb and t_kD termcap values.
Files: src/testdir/test19.in, src/testdir/test38.in

Patch 7.1.324

Problem: File name path length on Unix is limited to 1024.
Solution: Use PATH_MAX when it's more than 1000.
Files: src/os_unix.h

Patch 7.1.325

Problem: When editing a command line that's longer than available space in the window, the characters at the end are in reverse order.
Solution: Increment the insert position even when the command line doesn't fit. (Ingo Karkat)
Files: src/ex_getln.c

Patch 7.1.326

Problem: ":s!from!to!" works, but ":smagic!from!to!" doesn't. It sees the "!" as a flag to the command. Same for ":snomagic". (Johan Spetz)
Solution: When checking for a forced command also ignore ":smagic" and ":snomagic". (Ian Kelling)
Files: src/ex_docmd.c

Patch 7.1.327

Problem: The GUI tutor is installed when there is no GUI version.

Solution: Only install gvimtutor when building a GUI version.
Files: src/Makefile

Patch 7.1.328

Problem: Crash when using Cygwin and non-posix path name in tags file.
Solution: Use separate buffer for posix path. (Ben Schmidt)
Files: src/os_unix.c

Patch 7.1.329

Problem: When the popup menu is removed a column of cells, the right half of double-wide characters, may not be redrawn.
Solution: Check if the right half of a character needs to be redrawn. (Yukihiro Nakadaira)
Files: src/screen.c

Patch 7.1.330

Problem: Reading uninitialized memory when using Del in replace mode.
Solution: Use utfc_ptr2len_len() instead of mb_ptr2len(). (Dominique Pelle)
Files: src/misc1.c

Warning for missing sentinel in gui_xmldlg.c. (Dominique Pelle)

A search offset from the end of a match didn't work properly for multibyte characters. (Yukihiro Nakadaira)

When displaying the value of 'key' don't show "*****" when the value is empty. (Ben Schmidt)

Internal error when compiled with EXITFREE and using the nerd_tree plugin. Set last_msg_hist to NULL when history becomes empty. Call free_all_functions() after garbage collection. (Dominique Pelle)

GTK with XIM: <S-Space> does not work. (Yukihiro Nakadaira)

Some shells do not support "echo -n", which breaks glob(). Use "echo" instead of "echo -n \$1; echo". (Gary Johnson)

"echo 22,44" printed "22" on top of the command, the error messages caused the rest not to be cleared. Added the need_clr_eos flag.

Netbeans events are handled while updating the screen, causing a crash. Change the moment when events are handled. Rename nb_parse_messages() to netbeans_parse_messages(). (Xavier de Gaye)

Test 11 was broken after patch 7.1.186 on Win32 console. (Daniel Shahaf)
Use shellescape() on the file name.

IM was turned off in im_preedit_end_cb() for no good reason. (Takuhiro Nishioka)

A corrupted spell file could cause Vim to use lots of memory. Better detection for running into the end of the file. (idea from James Vega)

Mac: Included a patch to make it build with GTK. Moved language init to mac_lang_init() function. (Ben Schmidt)

Problem with **'wildmenu'** after ":", up/down arrows don't work. (Erik Falor)

Fix configure.in to avoid "implicitly declared" warnings when running configure.

Fixed a memory leak when redefining a keymap. (Dominique Pelle)

Setting **'pastetoggle'** to "jj" didn't work.

'ic' and **'smartcase'** don't work properly when using \%V in a search pattern. (Kana Natsuno)

Patch 7.2a.001

Problem: On some systems X11/Xlib.h exists (from X11-dev package) but X11/Intrinsic.h does not (in Xt-dev package). This breaks the build. Also, on Solaris 9 sys/ptem.h isn't found.

Solution: Have configure only accept X11 when X11/Intrinsic.h exists. Check for sys/ptem.h while including sys/stream.h. (Vladimir Marek)

Files: src/auto/configure, src/configure.in

Patch 7.2a.002

Problem: getbufvar(N, "") gets the dictionary of the current buffer instead of buffer N.

Solution: Set curbuf before calling find_var_in_ht(). (Kana Natsuno)

Files: src/eval.c

Patch 7.2a.003

Problem: Leaking memory when using ":file name" and using access control lists.

Solution: Invoke mch_free_acl() in vim_rename(). (Dominique Pelle)

Files: src/fileio.c

Patch 7.2a.004

Problem: Some systems can't get spell files by ftp.

Solution: Use http when it looks like it's possible. (James Vega)

Files: runtime/autoload/spellfile.vim

Patch 7.2a.005

Problem: A few error messages use confusing names. Misspelling.

Solution: Change "dissallows" to "disallows". (Dominique Pelle) Change "number" to "Number".

Files: src/eval.c, src/fileio.c

Patch 7.2a.006

Problem: Reading past NUL in a string.

Solution: Check for invalid utf-8 byte sequence. (Dominique Pelle)

Files: src/charset.c

Patch 7.2a.007

Problem: ":let v = 1.2.3" was OK in Vim 7.1, now it gives an error.

Solution: Don't look for a floating point number after the "." operator.
Files: src/eval.c

Patch 7.2a.008

Problem: printf("%g", 1) doesn't work.
Solution: Convert Number to Float when needed.
Files: src/message.c

Patch 7.2a.009

Problem: cygwin_conv_to_posix_path() does not specify buffer size.
Solution: Use new Cygwin function: cygwin_conv_path(). (Corinna Vinschen)
Files: src/main.c, src/os_unix.c

Patch 7.2a.010

Problem: When a file name has an illegal byte sequence Vim may read uninitialised memory.
Solution: Don't use UTF_COMPOSINGLIKE() on an illegal byte. In msg_outtrans_len_attr() use char2cells() instead of ptr2cells(). In utf_ptr2char() don't check second byte when first byte is illegal. (Dominique Pelle)
Files: src/mbyte.c, src/message.c

Patch 7.2a.011

Problem: The Edit/Startup Settings menu doesn't work.
Solution: Expand environment variables. (Ben Schmidt)
Files: runtime/menu.vim

Patch 7.2a.012

Problem: Compiler warnings for casting int to pointer.
Solution: Add cast to long in between. (Martin Toft)
Files: src/gui_gtk_x11.c

Patch 7.2a.013

Problem: shellescape() does not escape "%" and "#" characters.
Solution: Add find_cmdline_var() and use it when the second argument to shellescape() is non-zero.
Files: runtime/doc/eval.txt, src/eval.c, src/ex_docmd.c, src/proto/ex_docmd.pro, src/proto/misc2.pro, src/misc2.c

Patch 7.2a.014

Problem: Problem with % in message.
Solution: Put % in single quotes.
Files: src/eval.c

Patch 7.2a.015 (after 7.2a.010)

Problem: Misaligned messages.
Solution: Compute length of unprintable chars correctly.
Files: src/message.c

Patch 7.2a.016

Problem: Using **CTRL-W v** in the quickfix window results in two quickfix windows, which is not allowed. ":tab split" should be allowed to open a new quickfix window in another tab.
Solution: For **CTRL-W v** instead of splitting the window open a new one.

When using `:"tab"` do allow splitting the quickfix window (was already included in patch 7.2a.013).

Files: `src/window.c`

Patch 7.2a.017

Problem: `:"doautoall"` executes autocommands for all buffers instead of just for loaded buffers.

Solution: Change `"curbuf"` to `"buf"`.

Files: `src/fileio.c`

Patch 7.2a.018

Problem: Compiler warnings when compiling with Gnome. (Tony Mechelynck)

Solution: Add type casts.

Files: `src/gui_gtk_x11.c`

Patch 7.2a.019

Problem: `:"let &g:tw = 44"` sets the local option value. (Cyril Slobin)

Solution: Use `get_varp_scope()` instead of `get_varp()`. (Ian Kelling)

Files: `src/option.c`

There is no way to avoid adding `/usr/local/{include|lib}` to the build commands. Add the `--with-local-dir` argument to configure. (Michael Haubenwallner)

When using **CTRL-D** after `:"help"`, the number of matches could be thousands. Restrict to `TAG_MANY` to avoid this taking too long. (Ian Kelling)

The popup menu could be placed at a weird location. Caused by `w_wcol` computed by `curs_columns()`. (Dominique Pelle)

Overlapping `STRCPY()` arguments when using `%r` item in `'errorformat'`. Use `STRMOVE()` instead. (Ralf Wildenhues)

Mac: On Leopard `gvim`, when using the mouse wheel nothing would happen until another event occurs, such as moving the mouse. Then the recorded scrolling would take place all at once. (Eckehard Berns)

Solution for cursor color not reflecting IM status for GTK 2. Add `preedit_is_active` flag. (SungHyun Nam)

`filereadable()` can hang on a FIFO on Linux. Use `open()` instead of `fopen()`, with `O_NONBLOCK`. (suggested by Lars Kotthoff)

Included patch to support Perl 5.10. (Yasuhiro Matsumoto)

When files are dropped on `gvim` while the screen is being updated, ignore the drop command to avoid freeing memory that is being used.

In a terminal, when drawing the popup menu over double-wide characters, half characters may not be cleared properly. (Yukihiro Nakadaira)

The `#ifdef` for including `"vimio.h"` was inconsistent. In a few files it depended on `MSWIN`, which isn't defined until later.

Patch 7.2b.001

Problem: Compilation problem: mb_fix_col() missing with multibyte feature but without GUI or clipboard.
Solution: Remove #ifdef.
Files: src/mbyte.c

Patch 7.2b.002

Problem: Compiler warnings for signed/unsigned mismatch.
Solution: Add type casts.
Files: src/screen.c

Patch 7.2b.003

Problem: Still a compilation problem, check_col() and check_row() missing.
Solution: Add FEAT_MBYTE to the #if.
Files: src/ui.c

Patch 7.2b.004

Problem: Trying to free memory for a static string when using ":helpgrep". (George Reilly)
Solution: Set '**cpo**' to empty_option instead of an empty string. Also for searchpair() and substitute().
Files: src/quickfix.c, src/eval.c

Patch 7.2b.005

Problem: The special character "!" isn't handled properly in shellescape(). (Jan Minar)
Solution: Escape "!" when using a "csh" like shell and with shellescape(s, 1). Twice for both. Also escape <NL>.
Files: src/misc2.c

Patch 7.2b.006

Problem: Reading past end of string when reading info from tags line.
Solution: Break the loop when encountering a NUL. (Dominique Pelle)
Files: src/tag.c

Patch 7.2b.007

Problem: Part of a message cannot be translated.
Solution: Put _() around the message.
Files: src/search.c

Patch 7.2b.008

Problem: A few filetypes are not detected or not detected properly.
Solution: Add filetype detection patterns. (Nikolai Weibull)
Files: runtime/filetype.vim

Patch 7.2b.009

Problem: Reading past end of screen line. (Epicurus)
Solution: Avoid going past the value of Columns.
Files: src/screen.c

Patch 7.2b.010

Problem: ":mksession" doesn't work for ":map , foo", ":sunmap ,". (Ethan Mallove)
Solution: Check for "nxo", "nso" and other strange mapping combinations.

Files: src/getchar.c

Patch 7.2b.011

Problem: Configure for TCL ends up with include file in compiler command.
(Richard Hogg)

Solution: Delete items from \$TCL_DEFS that do not start with a dash.

Files: src/auto/configure, src/configure.in

Patch 7.2b.012

Problem: Build failure with +multi_byte but without +diff.

Solution: Add #ifdef. (Patrick Texier)

Files: src/main.c

Patch 7.2b.013

Problem: Build fails with tiny features and Perl. (Dominique Pelle)

Solution: Define missing functions. Also when compiling Python.

Files: src/if_perl.xs, src/if_python.c

Patch 7.2b.014

Problem: Configure uses an unsafe temp file to store commands.

Solution: Create the temp file in local directory.

Files: src/auto/configure, src/configure.in

Patch 7.2b.015

Problem: Build fails on Mac when using Aap.

Solution: Fix typo in configure script.

Files: src/auto/configure, src/configure.in

Patch 7.2b.016

Problem: Build fails with normal features but without +autocmd.

Solution: Fix #ifdefs. (Ian Kelling)

Files: src/eval.c, src/ex_cmds.c, src/quickfix.c, src/option.c,
src/ex_docmd.c

Patch 7.2b.017

Problem: "vim -O foo foo" results in only one window. (Zdenek Sekera)

Solution: Handle result of ATTENTION prompt properly. (Ian Kelling)

Files: src/main.c

Patch 7.2b.018

Problem: When doing command line completion on a file name for a csh-like
shell argument a '!' character isn't escaped properly.

Solution: Add another backslash.

Files: src/ex_getln.c, src/misc2.c, src/proto/misc2.pro, src/screen.c

Patch 7.2b.019 (extra)

Problem: Win32: Various compiler warnings.

Solution: Use __w64 attribute. Comment-out unused parameters. Adjust a few
#ifdefs. (George Reilly)

Files: src/gui_w48.c, src/GvimExt/gvimext.cpp, src/Make_mvc.mak,
src/os_mswin.c, src/os_win32.c, src/vim.h

Patch 7.2b.020

Problem: ":sort n" doesn't handle negative numbers. (James Vega)

Solution: Include '-' in the number.
Files: src/charset.c, src/ex_cmds.c

Patch 7.2b.021

Problem: Reloading doesn't read the BOM correctly. (Steve Gardner)
Solution: Accept utf-8 BOM when specified file encoding is utf-8.
Files: src/fileio.c

Patch 7.2b.022

Problem: When using ":normal" while updating the status line the count of an operator is lost. (Dominique Pelle)
Solution: Save and restore "opcount".
Files: src/ex_docmd.c, src/globals.h, src/normal.c

Patch 7.2b.023

Problem: Crash when using the result of synstack(0,0). (Matt Wozniski)
Solution: Check for v_list to be NULL in a few more places.
Files: src/eval.c

Patch 7.2b.024

Problem: Using ":gui" while the netrw plugin is active causes a delay in updating the display.
Solution: Don't check for terminal codes when starting the GUI.
Files: src/term.c

Patch 7.2b.025

Problem: When the CursorHold event triggers a pending count is lost. (Juergen Kraemer)
Solution: Save the counts and restore them.
Files: src/normal.c, src/structs.h

Patch 7.2b.026

Problem: The GTK 2 file chooser causes the ~/.recently-used.xbel file to be written over and over again. This may cause a significant slowdown. (Guido Berhoerster)
Solution: Don't use the GTK 2 file chooser.
Files: src/gui_gtk.c

Patch 7.2b.027

Problem: Memory leak for Python, Perl, etc. script command with end marker.
Solution: Free the memory of the end marker. (Andy Kittner)
Files: src/ex_getln.c

Patch 7.2b.028

Problem: Reading uninitialized memory when doing ":gui -f". (Dominique Pelle)
Solution: Don't position the cursor when the screen size is invalid.
Files: src/gui.c

Patch 7.2b.029

Problem: ":help a" doesn't jump to "a" tag in docs. (Tony Mechelynck)
Solution: Get all tags and throw away more than TAG_MANY after sorting. When there is no argument find matches for "help" to avoid a long delay.

Files: src/ex_cmds.c, src/ex_getln.c

Patch 7.2b.030

Problem: When changing the value of t_Co from 8 to 16 the Visual highlighting keeps both reverse and a background color.
Solution: Remove the attribute when setting the default highlight color. (Markus Heidelberg)
Files: src/syntax.c

Error when cancelling completion menu and auto-formatting. (fixed by Ian Kelling)

Patch 7.2c.001

Problem: ":let x=[''] | let x += x" causes hang. (Matt Wozniski)
Solution: Only insert elements up to the original length of the List.
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.2c.002

Problem: fnameescape() doesn't handle a leading '+' or '>'. (Jan Minar)
Solution: Escape a leading '+' and '>'. And a single '-'.
Files: runtime/doc/eval.txt, src/ex_getln.c

Patch 7.2c.003

Problem: Searching for "foo\[bar]\+" gives a "Corrupted regexp program" error. (Joachim Hofmann)
Solution: Mark the \[%[] item as not being simple.
Files: src/regexp.c

On Vista access to system directories is virtualized. (Michael Mutschler)
Adjusted the manifest file to avoid this. (George Reilly)

Memory leak when using **CTRL-C** to cancel listing the jump list. (Dominique Pelle)

Mac: Could not build with Perl interface.

=====

VERSION 7.3

version-7.3 version7.3

This section is about improvements made between version 7.2 and 7.3.

This release has hundreds of bug fixes and there are a few new features. The most notable new features are:

Persistent undo

new-persistent-undo

Store undo information in a file. Can undo to before when the file was read, also for unloaded buffers. See [undo-persistence](#) (partly by Jordan Lewis)

Added the ":earlier 1f" and ":later 1f" commands.
Added file save counter to undo information.
Added the [undotree\(\)](#) and [undofile\(\)](#) functions.

Also added the **'undoreload'** option. This makes it possible to save the current text when reloading the buffer, so that the reload can be undone.

More encryption

[new-more-encryption](#)

Support for Blowfish encryption. Added the **'cryptmethod'** option. Mostly by Mohsin Ahmed.

Also encrypt the text in the swap file and the undo file.

Conceal text

[new-conceal](#)

Added the **+conceal** feature. (Vince Negri)
This allows hiding stretches of text, based on syntax highlighting. It also allows replacing a stretch of text by a character **:syn-cchar**. The **'conceallevel'** option specifies what happens with text matching a syntax item that has the conceal attribute. The **'concealcursor'** option specifies what happens in the cursor line.

The help files conceal characters used to mark tags and examples.

Added the **synconcealed()** function and use it for **:TOhtml**. (Benjamin Fritz)

Added the **'cursorbind'** option, keeps the cursor in two windows with the same text in sync.

Lua interface

[new-lua](#)

Added the **Lua** interface. (Luis Carvalho)

Python3 interface

[new-python3](#)

Added the Python3 interface. It exists next to Python 2.x, both can be used at the same time. See **python3** (Roland Puntaier)

Changed

[changed-7.3](#)

The MS-Windows installer no longer requires the user to type anything in the console windows. The installer now also works on 64 bit systems, including the "Edit with Vim" context menu.

The gvim executable is 32 bits, the installed gvimext.dll is either a 32 or 64 bit version. (mostly by George Reilly)

Made the DOS installer work with more compilers.

The MS-Windows big gvim is now built with Python 2.7 and 3.1.2, Perl 5.12 and Ruby 1.9.1. You need the matching .dll files to use them.

The extra and language files are no longer distributed separately. The source files for all systems are included in one distribution.

After using `":recover"` or recovering a file in another way, `":x"` and `"ZZ"` didn't save what you see. This could result in work being lost. Now the text after recovery is compared to the original file contents. When they differ the buffer is marked as modified.

When Vim is exiting because of a deadly signal, when `v:dying` is 2 or more, `VimLeavePre`, `VimLeave`, `BufWinLeave` and `BufUnload` autocommands are not executed.

Removed support for GTK 1. It was no longer maintained and required a lot of `#ifdefs` in the source code. GTK 2 should be available for every system. (James Vega)

It is no longer allowed to set the `'encoding'` option from a modeline. It would corrupt the text. (Patrick Texier)

Renamed `runtime/spell/fixdup` to `runtime/spell/fixdup.vim`.

Removed obsolete Mac code.

Updated spell files for Ubuntu locale names.

Switched from `autoconf 2.63` to `2.65`.

Removed `Mupad indent` and `ftplugin` files, they are not useful.

The maximum number of messages remembered in the history is now 200 (was 100).

Added added-7.3

Added the `'relativenumber'` option. (Markus Heidelberg)

Added the `'colorcolumn'` option: highlight one or more columns in a window. E.g. to highlight the column after `'textwidth'`. (partly by Gregor Uhlenheuer)

Added support for NetBeans in a terminal. Added `:nstart` and `:nbclose`. (Xavier de Gaye)

More floating point functions: `acos()`, `asin()`, `atan2()`, `cosh()`, `exp()`, `fmod()`, `log()`, `sinh()`, `tan()`, `tanh()`. (Bill McCarthy)

Added the `gettabvar()` and `settabvar()` functions. (Yegappan Lakshmanan)

Added the `strchars()`, `strwidth()` and `strdisplaywidth()` functions.

Support `GDK_SUPER_MASK` for GTK on Mac. (Stephan Schulz)

Made CTRL and ALT modifier work for mouse wheel. (Benjamin Haskell)

Added support for horizontal scroll wheel. (Bjorn Winckler)

When the buffer is in diff mode, have :TOhtml create HTML to show the diff side-by-side. (Christian Brabandt)

Various improvements to ":TOhtml" and the 2html.vim script. (Benjamin Fritz)

Add the 'L' item to '**cinoptions**'. (Manuel Konig)

Improve Javascript indenting. Add "J" flag to '**cinoptions**'. (Hari Kumar G)

Mac: Support disabling antialias. (LC Mi)

Mac: Add clipboard support in the Mac console. (Bjorn Winckler)

Make it possible to drag a tab page label to another position. (Paul B. Mahol)

Better implementation of creating the Color Scheme menu. (Juergen Kraemer)

In Visual mode with '**showcmd**' display the number of bytes and characters.

Allow synIDattr() getting GUI attributes when built without GUI. (Matt Wozniski)

Support completion for ":find". Added test 73. (Nazri Ramliy)

Command line completion for :ownsyntax and :setfiletype. (Dominique Pelle)

Command line completion for :lmap and :lunmap.

Support syntax and filetype completion for user commands. (Christian Brabandt)

Avoid use of the GTK main_loop() so that the GtkFileChooser can be used. (James Vega)

When '**formatexpr**' evaluates to non-zero fall back to internal formatting, also for "gq". (James Vega)

Support :browse for commands that use an error file argument. (Lech Lorens)

Support wide file names in gvimext. (Szabolcs Horvat)

Improve test for joining lines. (Milan Vancura)

Make joining a range of lines much faster. (Milan Vancura)

Add patch to improve support of z/OS (OS/390). (Ralf Schandl)

Added the helphelp.txt file. Moved text from various.txt to it.

Added "q" item for '**statusline**'. Added [w:quickfix_title](#) . (Lech Lorens)

Various improvements for VMS. (Zoltan Arpadffy)

New syntax files:

Haskell Cabal build file (Vincent Berthoux)
ChaiScript (Jason Turner)
Cucumber (Tim Pope)
Dascript (Dominique Pelle)
Fantom (Kamil Toman)
Liquid (Tim Pope)
Markdown (Tim Pope)
wavefront's obj file (Vincent Berthoux)
Perl 6 (Andy Lester)
SDC - Synopsys Design Constraints (Maurizio Tranchero)
SVG - Scalable Vector Graphics (Vincent Berthoux)
task data (John Florian)
task 42 edit (John Florian)

New filetype plugins:

Cucumber (Tim Pope)
Liquid (Tim Pope)
Logcheck (Debian)
Markdown (Tim Pope)
Perl 6 (Andy Lester)
Quickfix window (Lech Lorens)
Tcl (Robert L Hicks)

New indent plugins:

CUDA (Bram Moolenaar)
ChaiScript (Jason Turner)
Cucumber (Tim Pope)
LifeLines (Patrick Texier)
Liquid (Tim Pope)
Mail (Bram Moolenaar)
Perl 6 (Andy Lester)

Other new runtime files:

Breton spell file (Dominique Pelle)
Dvorak keymap (Ashish Shukla)
Korean translations. (SungHyun Nam)
Python 3 completion (Aaron Griffin)
Serbian menu translations (Aleksandar Jelenak)
Tetum spell files
Tutor Bairish (Sepp Hell)
Tutor in Esperanto. (Dominique Pellé)
Tutor in Portuguese.
Norwegian Tutor now also available as tutor.nb

Removed the Mupad runtime files, they were not maintained.

Fixed

fixed-7.3

Patch 7.2.001

Problem: Mac: pseudo-ttys don't work properly on Leopard, resulting in the shell not to have a prompt, **CTRL-C** not working, etc.
Solution: Don't use SVR4 compatible ptys, even though they are detected. (Ben Schmidt)
Files: src/pty.c

Patch 7.2.002

Problem: Leaking memory when displaying menus.
Solution: Free allocated memory. (Dominique Pelle)
Files: src/menu.c

Patch 7.2.003

Problem: Typo in translated message. Message not translated.
Solution: Correct spelling. Add _(). (Dominique Pelle)
Files: src/spell.c, src/version.c

Patch 7.2.004

Problem: Cscope help message is not translated.
Solution: Put it in _(). (Dominique Pelle)
Files: src/if_cscope.c, src/if_cscope.h

Patch 7.2.005

Problem: A few problems when profiling. Using flag pointer instead of flag value. Allocating zero bytes. Not freeing used memory.
Solution: Remove wrong '&' characters. Skip dumping when there is nothing to dump. Free used memory. (Dominique Pelle)
Files: src/eval.c

Patch 7.2.006

Problem: HTML files are not recognized by contents.
Solution: Add a rule to the scripts file. (Nico Weber)
Files: runtime/scripts.vim

Patch 7.2.007 (extra)

Problem: Minor issues for VMS.
Solution: Minor fixes for VMS. Add float support. (Zoltan Arpadffy)
Files: runtime/doc/os_vms.txt, src/os_vms_conf.h, src/Make_vms.mms, src/testdir/Make_vms.mms, src/testdir/test30.in, src/testdir/test54.in

Patch 7.2.008

Problem: With a BufHidden autocommand that invokes ":bunload" the window count for a buffer can be wrong. (Bob Hiestand)
Solution: Don't call enter_buffer() when already in that buffer.
Files: src/buffer.c

Patch 7.2.009

Problem: Can't compile with Perl 5.10 on MS-Windows. (Cesar Romani)
Solution: Add the Perl_sv_free2 function for dynamic loading. (Dan Sharp)
Files: src/if_perl.xs

Patch 7.2.010

Problem: When using "K" in Visual mode not all characters are properly

escaped. (Ben Schmidt)
Solution: Use a function with the functionality of shellescape(). (Jan Minar)
Files: src/mbyte.c, src/misc2.c, src/normal.c

Patch 7.2.011

Problem: Get an error when inserting a float value from the expression register.
Solution: Convert the Float to a String automatically in the same place where a List would be converted to a String.
Files: src/eval.c

Patch 7.2.012

Problem: Compiler warnings when building with startup timing.
Solution: Add type casts.
Files: src/ex_cmds2.c

Patch 7.2.013

Problem: While waiting for the X selection Vim consumes a lot of CPU time and hangs until a response is received.
Solution: Sleep a bit when the selection event hasn't been received yet. Time out after a couple of seconds to avoid a hang when the selection owner isn't responding.
Files: src/ui.c

Patch 7.2.014

Problem: synstack() doesn't work in an empty line.
Solution: Accept column zero as a valid position.
Files: src/eval.c

Patch 7.2.015

Problem: "make all test install" doesn't stop when the test fails. (Daniel Shahaf)
Solution: When test.log contains failures exit with non-zero status.
Files: src/testdir/Makefile

Patch 7.2.016

Problem: The pattern being completed may be in freed memory when the command line is being reallocated. (Dominique Pelle)
Solution: Keep a pointer to the expand_T in the command line structure. Don't use <S-Tab> as **CTRL-P** when there are no results. Clear the completion when using a command line from the history.
Files: src/ex_getln.c

Patch 7.2.017

Problem: strlen() used on text that may not end in a NUL. (Dominique Pelle)
Pasting a very big selection doesn't work.
Solution: Use the length passed to the XtSelectionCallbackProc() function. After getting the SelectionNotify event continue dispatching events until the callback is actually called. Also dispatch the PropertyNotify event.
Files: src/ui.c

Patch 7.2.018

Problem: Memory leak when substitute is aborted.
Solution: Free the buffer allocated for the new text. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.2.019

Problem: Completion of ":noautocmd" doesn't work and exists(":noautocmd") returns zero. (Ben Fritz)
Solution: Add "noautocmd" to the list of modifiers and commands.
Files: src/ex_cmds.h, src/ex_docmd.c

Patch 7.2.020

Problem: Starting the GUI when the executable starts with 'k', but the KDE version no longer exists.
Solution: Don't have "kvim" start the GUI.
Files: src/main.c

Patch 7.2.021

Problem: When executing autocommands getting the full file name may be slow. (David Kotchan)
Solution: Postpone calling FullName_save() until autocmd_fname is used.
Files: src/ex_docmd.c, src/fileio.c, src/globals.h

Patch 7.2.022 (extra)

Problem: Testing is not possible when compiling with MingW.
Solution: Add a MingW specific test Makefile. (Bill McCarthy)
Files: Filelist, src/testdir/Make_ming.mak

Patch 7.2.023

Problem: '**cursorcolumn**' is in the wrong place in a closed fold when the display is shifted left. (Gary Johnson)
Solution: Subtract w_skipcol or w_leftcol when needed.
Files: src/screen.c

Patch 7.2.024

Problem: It's possible to set '**history**' to a negative value and that causes an out-of-memory error.
Solution: Check that '**history**' has a positive value. (Doug Kearns)
Files: src/option.c

Patch 7.2.025

Problem: When a CursorHold event invokes system() it is retrigged over and over again.
Solution: Don't reset did_cursorhold when getting K_IGNORE.
Files: src/normal.c

Patch 7.2.026 (after 7.2.010)

Problem: "K" doesn't use the length of the identifier but uses the rest of the line.
Solution: Copy the desired number of characters first.
Files: src/normal.c

Patch 7.2.027

Problem: Can use cscope commands in the sandbox.
Solution: Disallow them, they might not be safe.

Files: src/ex_cmds.h

Patch 7.2.028

Problem: Confusing error message for missing ().

Solution: Change "braces" to "parentheses". (Gary Johnson)

Files: src/eval.c

Patch 7.2.029

Problem: No completion for ":doautoall".

Solution: Complete ":doautoall" like ":doautocmd". (Doug Kearns)

Files: src/ex_docmd.c

Patch 7.2.030 (after 7.2.027)

Problem: Can't compile.

Solution: Remove prematurely added ex_oldfiles.

Files: src/ex_cmds.h

Patch 7.2.031

Problem: Information in the viminfo file about previously edited files is not available to the user. There is no way to get a complete list of files edited in previous Vim sessions.

Solution: Add v:oldfiles and fill it with the list of old file names when first reading the viminfo file. Add the ":oldfiles" command, ":browse oldfiles" and the "#<123" special file name. Increase the default value for '**viminfo**' from '20' to '100'.

Files: runtime/doc/cmdline.txt, runtime/doc/eval.txt, runtime/doc/starting.txt, runtime/doc/usr_21.txt, src/eval.c, src/ex_cmds.c, src/ex_cmds.h, src/ex_docmd.c, src/feature.h, src/fileio.c, src/main.c, src/mark.c, src/misc1.c, src/proto/eval.pro, src/proto/ex_cmds.pro, src/proto/mark.pro, src/option.c, src/structs.h, src/vim.h

Patch 7.2.032 (after 7.2.031)

Problem: Can't build with EXITFREE defined. (Dominique Pelle)

Solution: Change vv_string to vv_str.

Files: src/eval.c

Patch 7.2.033

Problem: When detecting a little endian BOM "ucs-2le" is used, but the text might be "utf-16le".

Solution: Default to "utf-16le", it also works for "ucs-2le". (Jia Yanwei)

Files: src/fileio.c, src/testdir/test42.ok

Patch 7.2.034

Problem: Memory leak in spell info when deleting buffer.

Solution: Free the memory. (Dominique Pelle)

Files: src/buffer.c

Patch 7.2.035

Problem: Mismatches between alloc/malloc, free/vim_free, realloc/vim_realloc.

Solution: Use the right function. (Dominique Pelle)

Files: src/gui_x11.c, src/mbyte.c, src/misc2.c, src/os_unix.c

Patch 7.2.036 (extra)

Problem: Mismatches between alloc/malloc, free/vim_free, realloc/vim_realloc.
Solution: Use the right function. (Dominique Pelle)
Files: src/gui_riscos.c, src/gui_w48.c, src/mbyte.c, src/os_vms.c, src/os_w32exe.c, src/os_win16.c

Patch 7.2.037

Problem: Double free with GTK 1 and compiled with EXITFREE.
Solution: Don't close display. (Dominique Pelle)
Files: src/os_unix.c

Patch 7.2.038

Problem: Overlapping arguments to memcpy().
Solution: Use mch_memmove(). (Dominique Pelle)
Files: src/if_xcmdsrv.c

Patch 7.2.039

Problem: Accessing freed memory on exit when EXITFREE is defined.
Solution: Call hash_init() on the v: hash table.
Files: src/eval.c

Patch 7.2.040

Problem: When using ":e ++ff=dos fname" and the file contains a NL without a CR before it and '**ffs**' contains "unix" then the fileformat becomes unix.
Solution: Ignore '**ffs**' when using the ++ff argument. (Ben Schmidt)
Also remove unreachable code.
Files: src/fileio.c

Patch 7.2.041

Problem: In diff mode, when using two tabs, each with two diffed buffers, editing a buffer of the other tab messes up the diff. (Matt Mzyzik)
Solution: Only copy options from a window where the buffer was edited that doesn't have '**diff**' set or is for the current tab page.
Also fix that window options for a buffer are stored with the wrong window.
Files: src/buffer.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/if_sniff.c, src/main.c, src/netbeans.c, src/normal.c, src/popupmnu.c, src/proto/buffer.pro, src/proto/ex_cmds.pro, src/quickfix.c, src/window.c

Patch 7.2.042

Problem: When using winrestview() in a BufWinEnter autocommand the window is scrolled anyway. (Matt Zyzik)
Solution: Don't recompute topline when above '**scrolloff**' from the bottom.
Don't always put the cursor halfway when entering a buffer. Add "w_topline_was_set".
Files: src/buffer.c, src/move.c, src/structs.h

Patch 7.2.043

Problem: VMS: Too many characters are escaped in filename and shell commands.

Solution: Escape fewer characters. (Zoltan Arpadffy)
Files: src/vim.h

Patch 7.2.044

Problem: Crash because of STRCPY() being over protective of the destination size. (Dominique Pelle)

Solution: Add -D_FORTIFY_SOURCE=1 to CFLAGS. Use an intermediate variable for the pointer to avoid a warning.

Files: src/auto/configure, src/configure.in, src/eval.c

Patch 7.2.045

Problem: The Python interface has an empty entry in sys.path.

Solution: Filter out the empty entry. (idea from James Vega)

Files: src/if_python.c

Patch 7.2.046

Problem: Wrong check for filling buffer with encoding. (Danek Duvall)

Solution: Remove pointers. (Dominique Pelle)

Files: src/mbyte.c

Patch 7.2.047

Problem: Starting Vim with the -nb argument while it's not supported causes the other side to hang.

Solution: When -nb is used while it's not supported exit Vim. (Xavier de Gaye)

Files: src/main.c, src/vim.h

Patch 7.2.048

Problem: v:prevcount is changed too often. Counts are not multiplied when setting v:count.

Solution: Set v:prevcount properly. Multiply counts. (idea by Ben Schmidt)

Files: src/eval.c, src/normal.c, src/proto/eval.pro

Patch 7.2.049 (extra)

Problem: Win32: the clipboard doesn't support UTF-16.

Solution: Change UCS-2 support to UTF-16 support. (Jia Yanwei)

Files: src/gui_w32.c, src/gui_w48.c, src/mbyte.c, src/misc1.c, src/os_mswin.c, src/os_win32.c, src/proto/os_mswin.pro

Patch 7.2.050

Problem: Warnings for not checking return value of fwrite(). (Chip Campbell)

Solution: Use the return value.

Files: src/spell.c

Patch 7.2.051

Problem: Can't avoid **'wildignore'** and **'suffixes'** for glob() and globpath().

Solution: Add an extra argument to these functions. (Ingo Karkat)

Files: src/eval.c, src/ex_getln.c, src/proto/ex_getln.pro, runtime/doc/eval.txt, runtime/doc/options.txt

Patch 7.2.052

Problem: synIDattr() doesn't support "sp" for special color.

Solution: Recognize "sp" and "sp#". (Matt Wozniski)

Files: runtime/doc/eval.txt, src/eval.c

Patch 7.2.053

Problem: Crash when using WorkShop command ":ws foo". (Dominique Pelle)
Solution: Avoid using a NULL pointer.
Files: src/workshop.c

Patch 7.2.054

Problem: Compilation warnings for format in getchar.c.
Solution: Use fputs() instead of fprintf(). (Dominique Pelle)
Files: src/getchar.c

Patch 7.2.055

Problem: Various compiler warnings with strict checking.
Solution: Avoid the warnings by using return values and renaming.
Files: src/diff.c, src/eval.c, src/ex_cmds.c, src/ex_docmd.c,
src/fileio.c, src/fold.c, src/globals.h, src/gui.c,
src/gui_at_sb.c, src/gui_gtk_x11.c, src/gui_xmdl.c,
src/gui_xmew.c, src/main.c, src/mbyte.c, src/message.c,
src/netbeans.c, src/option.c, src/os_unix.c, src/spell.c,
src/ui.c, src/window.c

Patch 7.2.056 (after 7.2.050)

Problem: Tests 58 and 59 fail.
Solution: Don't invoke fwrite() with a zero length. (Dominique Pelle)
Files: src/spell.c

Patch 7.2.057 (after 7.2.056)

Problem: Combination of int and size_t may not work.
Solution: Use size_t for variable.
Files: src/spell.c

Patch 7.2.058

Problem: Can't add a patch name to the ":version" output.
Solution: Add the extra_patches array.
Files: src/version.c

Patch 7.2.059

Problem: Diff display is not always updated.
Solution: Update the display more often.
Files: src/diff.c

Patch 7.2.060

Problem: When a spell files has many compound rules it may take a very long time making the list of suggestions. Displaying also can be slow when there are misspelled words.
Can't parse some Hunspell .aff files.
Solution: Check if a compounding can possibly work before trying a combination, if the compound rules don't contain wildcards.
Implement using CHECKCOMPOUNDPATTERN.
Ignore COMPOUNDRULES. Ignore a comment after most items.
Accept ONLYINCOMPOUND as an alias for NEEDCOMPOUND.
Accept FORBIDDENWORD as an alias for BAD.
Files: runtime/doc/spell.txt, src/spell.c

Patch 7.2.061

Problem: Can't create a funcref for an autoload function without loading the script first. (Marc Weber)
Solution: Accept autoload functions that don't exist yet in function().
Files: src/eval.c

Patch 7.2.062

Problem: "[Scratch]" is not translated.
Solution: Mark the string for translation. (Dominique Pelle)
Files: src/buffer.c

Patch 7.2.063

Problem: Warning for NULL argument of Perl_sys_init3().
Solution: Use Perl_sys_init() instead. (partly by Dominique Pelle)
Files: src/if_perl.xs

Patch 7.2.064

Problem: Screen update bug when repeating "~" on a Visual block and the last line doesn't change.
Solution: Keep track of changes for all lines. (Moritz Orbach)
Files: src/ops.c

Patch 7.2.065

Problem: GTK GUI: the cursor disappears when doing ":vsp" and the Vim window is maximized. (Dominique Pelle, Denis Smolyar)
Solution: Don't change "Columns" back to an old value at a wrong moment. Do change "Rows" when it should not be a problem.
Files: src/gui.c

Patch 7.2.066

Problem: It's not easy to see whether 'encoding' is a multibyte encoding.
Solution: Add has('multi_byte_encoding').
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.2.067

Problem: Session file can't load extra file when the path contains special characters.
Solution: Escape the file name. (Lech Lorens)
Files: src/ex_docmd.c

Patch 7.2.068

Problem: Emacs tags file lines can be too long, resulting in an error message. (James Vega)
Solution: Ignore lines with errors if they are too long.
Files: src/tag.c

Patch 7.2.069 (after 7.2.060)

Problem: Compiler warning for storing size_t in int.
Solution: Add type cast.
Files: src/spell.c

Patch 7.2.070

Problem: Crash when a function returns a:000. (Matt Wozniski)
Solution: Don't put the function struct on the stack, allocate it. Free it

only when nothing in it is used.
Files: src/eval.c

Patch 7.2.071 (extra)

Problem: Win32: Handling netbeans events while Vim is busy updating the screen may cause a crash.

Solution: Like with GTK, only handle netbeans messages in the main loop. (Xavier de Gaye)

Files: src/gui_w48.c, src/netbeans.c

Patch 7.2.072 (extra)

Problem: Compiler warning in Sniff code.

Solution: Use return value of pipe(). (Dominique Pelle)

Files: src/if_sniff.c

Patch 7.2.073

Problem: ":set <xHome>" has the same output as ":set <Home>". (Matt Wozniski)

Solution: Don't translate "x" keys to its alternative for ":set".

Files: src/gui_mac.c, src/misc2.c, src/option.c, src/proto/misc2.pro

Patch 7.2.074 (extra, after 7.2.073)

Problem: ":set <xHome>" has the same output as ":set <Home>". (Matt Wozniski)

Solution: Don't translate "x" keys to its alternative for ":set".

Files: src/gui_mac.c

Patch 7.2.075 (after 7.2.058)

Problem: Explanation about making a diff for extra_patches is unclear.

Solution: Adjust comment.

Files: src/version.c

Patch 7.2.076

Problem: rename(from, to) deletes the file if "from" and "to" are not equal but still refer to the same file. E.g., on a FAT32 filesystem under Unix.

Solution: Go through another file name.

Files: src/fileio.c

Patch 7.2.077 (after 7.2.076)

Problem: rename(from, to) doesn't work if "from" and "to" differ only in case on a system that ignores case in file names.

Solution: Go through another file name.

Files: src/fileio.c

Patch 7.2.078

Problem: When deleting a fold that is specified with markers the cursor position may be wrong. Folds may not be displayed properly after a delete. Wrong fold may be deleted.

Solution: Fix the problems. (mostly by Lech Lorens)

Files: src/fold.c

Patch 7.2.079

Problem: "killed" netbeans events are not handled correctly.

Solution: A "killed" netbeans event is sent when the buffer is deleted or wiped out (in this case, the netbeans annotations in this buffer have been removed). A user can still remove a sign with the command ":sign unplace" and this does not trigger a "killed" event. (Xavier de Gaye)

Files: runtime/doc/netbeans.txt, src/buffer.c, src/globals.h, src/netbeans.c, src/proto/netbeans.pro

Patch 7.2.080

Problem: When typing a composing character just after starting completion may access memory before its allocation point. (Dominique Pelle)

Solution: Don't delete before the completion start column. Add extra checks for the offset not being negative.

Files: src/edit.c

Patch 7.2.081

Problem: Compiler warning for floating point overflow on VAX.

Solution: For VAX use a smaller number. (Zoltan Arpadffy)

Files: src/message.c

Patch 7.2.082

Problem: When 'ff' is "mac" then "ga" on a ^J shows 0x0d instead of 0x0a. (Andy Wokula)

Solution: Use NL for this situation. (Lech Lorens)

Files: src/ex_cmds.c

Patch 7.2.083

Problem: ":tag" does not return to the right tag entry from the tag stack.

Solution: Don't change the current match when there is no argument. (Erik Falor)

Files: src/tag.c

Patch 7.2.084

Problem: Recursive structures are not handled properly in Python vim.eval().

Solution: Keep track of references in a better way. (Yukihiro Nakadaira)

Files: src/if_python.c

Patch 7.2.085

Problem: ":set <M-b>=<Esc>b" does not work when 'encoding' is utf-8.

Solution: Put the <M-b> character in the input buffer as valid utf-8. (partly by Matt Wozniski)

Files: src/term.c

Patch 7.2.086

Problem: Using ":diffget 1" in buffer 1 corrupts the text.

Solution: Don't do anything when source and destination of ":diffget" or ":diffput" is the same buffer. (Dominique Pelle)

Files: src/diff.c

Patch 7.2.087

Problem: Adding URL to 'path' doesn't work to edit a file.

Solution: Skip simplify_filename() for URLs. (Matt Wozniski)

Files: src/misc2.c

Patch 7.2.088 (extra)

Problem: OpenClipboard() may fail when another application is using the clipboard.

Solution: Retry OpenClipboard() a few times. (Jianrong Yu)

Files: src/os_mswin.c

Patch 7.2.089 (extra)

Problem: Win32: crash when using Ultramon buttons.

Solution: Don't use a WM_OLE message of zero size. (Ray Megal)

Files: src/if_ole.cpp, src/gui_w48.c

Patch 7.2.090

Problem: User command containing 0x80 in multibyte character does not work properly. (Yasuhiro Matsumoto)

Solution: Undo replacement of K_SPECIAL and CSI characters when executing the command.

Files: src/ex_docmd.c

Patch 7.2.091

Problem: ":cs help" output is not aligned for some languages.

Solution: Compute character size instead of byte size. (Dominique Pelle)

Files: src/if_cscode.c

Patch 7.2.092

Problem: Some error messages are not translated.

Solution: Add _() around the messages. (Dominique Pelle)

Files: src/eval.c

Patch 7.2.093 (extra)

Problem: Win32: inputdialog() and find/replace dialogs can't handle multibyte text.

Solution: Use the wide version of dialog functions when available. (Yanwei Jia)

Files: src/gui_w32.c, src/gui_w48.c

Patch 7.2.094

Problem: Compiler warning for signed/unsigned compare.

Solution: Add type cast. Also fix a few typos.

Files: src/edit.c

Patch 7.2.095

Problem: With Visual selection, "r" and then **CTRL-C** Visual mode is stopped but the highlighting is not removed.

Solution: Call reset_VIsual().

Files: src/normal.c

Patch 7.2.096

Problem: After ":number" the "Press Enter" message may be on the wrong screen, if switching screens for shell commands.

Solution: Reset info_message. (James Vega)

Files: src/ex_cmds.c

Patch 7.2.097

Problem: `!xterm&` doesn't work when `'shell'` is `"bash"`.
Solution: Ignore SIGHUP after calling `setsid()`. (Simon Schubert)
Files: `src/os_unix.c`

Patch 7.2.098

Problem: Warning for signed/unsigned pointer.
Solution: Add type cast.
Files: `src/eval.c`

Patch 7.2.099

Problem: Changing GUI options causes an unnecessary redraw when the GUI isn't active.
Solution: Avoid the redraw. (Lech Lorens)
Files: `src/option.c`

Patch 7.2.100

Problem: When using `!source` on a FIFO or something else that can't rewind the first three bytes are skipped.
Solution: Instead of rewinding read the first line and detect a BOM in that. (mostly by James Vega)
Files: `src/ex_cmds2.c`

Patch 7.2.101 (extra)

Problem: MSVC version not recognized.
Solution: Add the version number to the list. (Zhong Zhang)
Files: `src/Make_mvc.mak`

Patch 7.2.102 (after 7.2.100)

Problem: When `'encoding'` is `"utf-8"` a BOM at the start of a Vim script is not removed. (Tony Mechelynck)
Solution: When no conversion is taking place make a copy of the line without the BOM.
Files: `src/ex_cmds2.c`

Patch 7.2.103

Problem: When `'bomb'` is changed the window title is updated to show/hide a `"+"`, but the tab page label isn't. (Patrick Texier)
Solution: Set `"redraw_tabline"` in most places where `"need_maketitle"` is set. (partly by Lech Lorens)
Files: `src/option.c`

Patch 7.2.104

Problem: When using `!saveas bar.c` the tab label isn't updated right away.
Solution: Set `redraw_tabline`. (Francois Ingelrest)
Files: `src/ex_cmds.c`

Patch 7.2.105

Problem: Modeline setting for `'foldmethod'` overrules diff options. (Ingo Karkat)
Solution: Don't set `'foldmethod'` and `'wrap'` from a modeline when `'diff'` is on.
Files: `src/option.c`

Patch 7.2.106

Problem: Endless loop when using "]" in HTML when there are no misspellings. (Ingo Karkat)
Solution: Break the search loop. Also fix pointer alignment for systems with pointers larger than int.
Files: src/spell.c

Patch 7.2.107

Problem: When using a GUI dialog and ":echo" commands the messages are deleted after the dialog. (Vincent Birebent)
Solution: Don't call msg_end_prompt() since there was no prompt.
Files: src/message.c

Patch 7.2.108 (after 7.2.105)

Problem: Can't build without the diff feature.
Solution: Add #ifdef.
Files: src/option.c

Patch 7.2.109

Problem: '**langmap**' does not work for multibyte characters.
Solution: Add a list of mapped multibyte characters. (based on work by Konstantin Korikov, Agathoklis Hatzimanikas)
Files: runtime/doc/options.txt, src/edit.c, src/getchar.c, src/macros.h, src/normal.c, src/option.c, src/proto/option.pro, src/window.c

Patch 7.2.110

Problem: Compiler warning for unused variable.
Solution: Init the variable.
Files: src/ex_docmd.c

Patch 7.2.111

Problem: When using Visual block mode with '**cursorcolumn**' it's unclear what is selected.
Solution: Don't use '**cursorcolumn**' highlighting inside the Visual selection. (idea by Dominique Pelle)
Files: src/screen.c

Patch 7.2.112

Problem: Cursor invisible in Visual mode when '**number**' is set and cursor in first column. (Matti Niemenmaa, Renato Alves)
Solution: Check that vcol_prev is smaller than vcol.
Files: src/screen.c

Patch 7.2.113

Problem: Crash for substitute() call using submatch(1) while there is no such submatch. (Yukihiro Nakadaira)
Solution: Also check the start of the submatch is set, it can be NULL when an attempted match didn't work out.
Files: src/regexp.c

Patch 7.2.114

Problem: Using wrong printf format.
Solution: Use "%ld" instead of "%d". (Dominique Pelle)
Files: src/netbeans.c

Patch 7.2.115

Problem: Some debugging code is never used.
Solution: Remove nbtrace() and nbprt(). (Dominique Pelle)
Files: src/nbdebug.c, src/nbdebug.h

Patch 7.2.116

Problem: Not all memory is freed when EXITFREE is defined.
Solution: Free allocated memory on exit. (Dominique Pelle)
Files: src/ex_docmd.c, src/gui_gtk_x11.c, src/misc2.c, src/search.c, src/tag.c

Patch 7.2.117

Problem: Location list incorrectly labelled "Quickfix List".
Solution: Break out of both loops for finding window for location list buffer. (Lech Lorens)
Files: src/buffer.c, src/quickfix.c, src/screen.c

Patch 7.2.118

Problem: <PageUp> at the more prompt only does half a page.
Solution: Make <PageUp> go up a whole page. Also make 'f' go a page forward, but not quit the more prompt. (Markus Heidelberg)
Files: src/message.c

Patch 7.2.119

Problem: Status line is redrawn too often.
Solution: Check ScreenLinesUC[] properly. (Yukihiro Nakadaira)
Files: src/screen.c

Patch 7.2.120

Problem: When opening the quickfix window or splitting the window and setting the location list, the location list is copied and then deleted, which is inefficient.
Solution: Don't copy the location list when not needed. (Lech Lorens)
Files: src/quickfix.c, src/vim.h, src/window.c

Patch 7.2.121

Problem: In gvim "!grep a *.c" spews out a lot of text that can't be stopped with **CTRL-C**.
Solution: When looping to read and show text, do check for typed characters every two seconds.
Files: src/os_unix.c

Patch 7.2.122

Problem: Invalid memory access when the VimResized autocommand changes **'columns'** and/or **'lines'**.
Solution: After VimResized check for changed values. (Dominique Pelle)
Files: src/screen.c

Patch 7.2.123

Problem: Typing 'q' at more prompt for ":map" output still displays another line, causing another more prompt. (Markus Heidelberg)
Solution: Quit listing maps when 'q' typed.
Files: src/getchar.c

Patch 7.2.124

Problem: Typing 'q' at more prompt for ":tselect" output still displays more lines, causing another more prompt. (Markus Heidelberg)
Solution: Quit listing tags when 'q' typed.
Files: src/tag.c

Patch 7.2.125

Problem: Leaking memory when reading XPM bitmap for a sign.
Solution: Don't allocate the memory twice. (Dominique Pelle)
Files: src/gui_x11.c

Patch 7.2.126

Problem: When EXITFREE is defined signs are not freed.
Solution: Free all signs on exit. Also free keymaps. (Dominique Pelle)
Files: src/misc2.c, src/ex_cmds.c, src/proto/ex_cmds.pro

Patch 7.2.127

Problem: When listing mappings and a wrapping line causes the more prompt, after typing 'q' there can be another more prompt. (Markus Heidelberg)
Solution: Set "lines_left" to allow more lines to be displayed.
Files: src/message.c

Patch 7.2.128 (after 7.2.055)

Problem: Using ":lcd" makes session files not work.
Solution: Compare return value of mch_chdir() properly. (Andreas Bernauer)
Files: src/ex_docmd.c

Patch 7.2.129

Problem: When opening a command window from input() it uses the search history.
Solution: Use get_cmdline_type(). (James Vega)
Files: src/ex_getln.c

Patch 7.2.130

Problem: Vim may hang until **CTRL-C** is typed when using **CTRL-Z**.
Solution: Avoid using pause(). Also use "volatile" for variables used in signal functions. (Dominique Pelle)
Files: src/auto/configure, src/configure.in, src/config.h.in, src/globals.h, src/os_unix.c

Patch 7.2.131

Problem: When '**keymap**' is cleared may still use the cursor highlighting for when it's enabled.
Solution: Reset '**iminsert**' and '**imsearch**'. (partly by Dominique Pelle)
Also avoid ":setlocal" for these options have a global effect.
Files: src/option.c

Patch 7.2.132

Problem: When changing directory during a SwapExists autocmd freed memory may be accessed. (Dominique Pelle)
Solution: Add the allbuf_lock flag.
Files: src/ex_getln.c, src/globals.h, src/fileio.c, src/proto/ex_getln.pro

Patch 7.2.133

Problem: ":diffoff!" changes settings in windows not in diff mode.
Solution: Only change settings in other windows when '**diff**' is set, always do it for the current window. (Lech Lorens)
Files: src/diff.c

Patch 7.2.134

Problem: Warning for discarding "const" from pointer.
Solution: Don't pass const pointer to mch_memmove().
Files: src/fileio.c

Patch 7.2.135

Problem: Memory leak when redefining user command with complete argument.
Solution: Free the old complete argument. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.2.136 (after 7.2.132)

Problem: ":cd" is still possible in a SwapExists autocmd.
Solution: Check the allbuf_lock flag in ex_cd().
Files: src/ex_docmd.c

Patch 7.2.137

Problem: When '**virtualedit**' is set, a left shift of a blockwise selection that starts and ends inside a tab shifts too much. (Helmut Stiegler)
Solution: Redo the block left shift code. (Lech Lorens)
Files: src/ops.c, src/testdir/Makefile, src/testdir/test66.in, src/testdir/test66.ok

Patch 7.2.138 (extra part of 7.2.137)

Problem: See 7.2.137.
Solution: See 7.2.137.
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.2.139

Problem: Crash when '**virtualedit**' is "all". (James Vega)
Solution: Avoid overflow when column is MAXCOL. (Dominique Pelle)
Files: src/misc2.c

Patch 7.2.140

Problem: Diff highlighting isn't displayed before the Visual area if it starts at the cursor position. (Markus Heidelberg)
Solution: Also check fromcol_prev.
Files: src/screen.c

Patch 7.2.141

Problem: When redrawing a character for bold spill this causes the next character to be redrawn as well.
Solution: Only redraw one extra character. (Yukihiro Nakadaira)
Files: src/screen.c

Patch 7.2.142

Problem: Motif and Athena balloons don't use tooltip colors.
Solution: Set the colors. (Matt Wozniski)
Files: src/gui_beval.c

Patch 7.2.143

Problem: No command line completion for ":cscope" command.
Solution: Add the completion for ":cscope". (Dominique Pelle)
Files: src/ex_docmd.c, src/ex_getln.c, src/if_cscope.c,
src/proto/if_cscope.pro, src/vim.h

Patch 7.2.144

Problem: When '**t_Co**' is set to the value it already had the color scheme is reloaded anyway.
Solution: Only load the colorscheme when the t_Co value changes. (Dominique Pelle)
Files: src/option.c

Patch 7.2.145

Problem: White space in ":cscope find" is not ignored.
Solution: Ignore the white space, but not when the leading white space is useful for the argument.
Files: runtime/doc/if_scop.txt, src/if_cscope.c

Patch 7.2.146

Problem: v:warningmsg isn't used for all warnings.
Solution: Set v:warningmsg for relevant warnings. (Ingo Karkat)
Files: src/fileio.c, src/misc1.c, src/option.c

Patch 7.2.147

Problem: When compiled as small version and '**number**' is on the cursor is displayed in the wrong position after a tab. (James Vega)
Solution: Don't increment vcol when still displaying the line number.
Files: src/screen.c

Patch 7.2.148

Problem: When searching for "\$" while '**hlsearch**' is set, highlighting the character after the line does not work in the cursor column. Also highlighting for Visual mode after the line end when this isn't needed. (Markus Heidelberg)
Solution: Only compare the cursor column in the cursor line. Only highlight for Visual selection after the last character when it's needed to see where the Visual selection ends.
Files: src/screen.c

Patch 7.2.149

Problem: Using return value of function that doesn't return a value results in reading uninitialized memory.
Solution: Set the default to return zero. Make cursor() return -1 on failure. Let complete() return an empty string in case of an error. (partly by Dominique Pelle)
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.2.150 (extra)

Problem: Can't use tab pages from VisVim.
Solution: Add tab page support to VisVim. (Adam Slater)
Files: src/VisVim/Commands.cpp, src/VisVim/Resource.h,
src/VisVim/VisVim.rc

Patch 7.2.151

Problem: ":hist a" doesn't work like ":hist all" as the docs suggest.
Solution: Make ":hist a" and ":hist al" work. (Dominique Pelle)
Files: src/ex_getln.c

Patch 7.2.152

Problem: When using "silent echo x" inside ":redir" a next echo may start halfway the line. (Tony Mechelynck, Dennis Benzinger)
Solution: Reset msg_col after redirecting silently.
Files: src/ex_docmd.c, src/message.c, src/proto/message.pro

Patch 7.2.153

Problem: Memory leak for ":recover empty_dir/".
Solution: Free files[] when it becomes empty. (Dominique Pelle)
Files: src/memline.c

Patch 7.2.154 (after 7.2.132)

Problem: ":cd" is still possible in a SwapExists autocmd.
Solution: Set allbuf_lock in do_swapexists().
Files: src/memline.c

Patch 7.2.155

Problem: Memory leak in ":function /pat".
Solution: Free the memory. (Dominique Pelle)
Files: src/eval.c

Patch 7.2.156 (after 7.2.143)

Problem: No completion for :scscope and :lcscope commands.
Solution: Implement the completion. (Dominique Pelle)
Files: src/if_cscope.c, src/ex_docmd.c, src/proto/if_cscope.pro

Patch 7.2.157

Problem: Illegal memory access when searching in path.
Solution: Avoid looking at a byte after end of a string. (Dominique Pelle)
Files: src/search.c

Patch 7.2.158

Problem: Warnings from VisualC compiler.
Solution: Add type casts. (George Reilly)
Files: src/ops.c

Patch 7.2.159

Problem: When \$x_includes ends up being "NONE" configure fails.
Solution: Check for \$x_includes not to be "NONE" (Rainer)
Files: src/auto/configure, src/configure.in

Patch 7.2.160

Problem: Search pattern not freed on exit when 'rightleft' set.
Solution: Free mr_pattern_allocated.

Files: src/search.c

Patch 7.2.161

Problem: Folds messed up in other tab page. (Vlad Irnov)

Solution: Instead of going over all windows in current tab page go over all windows in all tab pages. Also free memory for location lists in other tab pages when exiting. (Lech Lorens)

Files: src/fileio.c, src/mark.c, src/misc1.c, src/misc2.c

Patch 7.2.162

Problem: The quickfix window may get wrong filetype.

Solution: Do not detect the filetype for the quickfix window. (Lech Lorens)

Files: src/quickfix.c

Patch 7.2.163

Problem: The command line window may get folding.

Solution: Default to no/manual folding. (Lech Lorens)

Files: src/ex_getln.c

Patch 7.2.164

Problem: When '**showbreak**' is set the size of the Visual block may be reported wrong. (Eduardo Daudt Flach)

Solution: Temporarily make '**sbr**' empty.

Files: src/normal.c, src/ops.c

Patch 7.2.165

Problem: The argument for the FuncUndefined autocmd event is expanded like a file name.

Solution: Don't try expanding it. (Wang Xu)

Files: src/fileio.c

Patch 7.2.166

Problem: No completion for ":sign" command.

Solution: Add ":sign" completion. (Dominique Pelle)

Files: src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/vim.h, src/proto/ex_cmds.pro

Patch 7.2.167

Problem: Splint doesn't work well for checking the code.

Solution: Add splint arguments in the Makefile. Exclude some code from splint that it can't handle. Tune splint arguments to give reasonable errors. Add a filter for removing false warnings from splint output. Many small changes to avoid warnings. More to follow...

Files: Filelist, src/Makefile, src/buffer.c, src/charset.c, src/cleanlint.vim, src/digraph.c, src/edit.c, src/ex_cmds.c, src/globals.h, src/ops.c, src/os_unix.c, src/os_unix.h, src/proto/buffer.pro, src/proto/edit.pro, src/screen.c, src/structs.h

Patch 7.2.168

Problem: When no ctags program can be found, "make tags" attempts to execute the first C file.

Solution: Default to "ctags" when no ctags program can be found.

Files: src/configure.in, src/auto/configure

Patch 7.2.169

Problem: Splint complains about a lot of things.

Solution: Add type casts, #ifdefs and other changes to avoid warnings. Change colnr_T from unsigned to int. Avoids mistakes with subtracting columns.

Files: src/cleanlint.vim, src/diff.c, src/edit.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/proto/ex_cmds.pro, src/proto/spell.pro, src/quickfix.c, src/spell.c, src/structs.h, src/term.h, src/vim.h

Patch 7.2.170

Problem: Using b_dev while it was not set. (Dominique Pelle)

Solution: Add the b_dev_valid flag.

Files: src/buffer.c, src/fileio.c, src/structs.h

Patch 7.2.171 (after 7.2.169)

Problem: Compiler warnings. (Tony Mechelynck)

Solution: Add function prototype. (Patrick Texier) Init variable.

Files: src/ex_cmds.c

Patch 7.2.172 (extra)

Problem: Compiler warning.

Solution: Adjust function prototype. (Patrick Texier)

Files: src/os_mswin.c

Patch 7.2.173

Problem: Without lint there is no check for unused function arguments.

Solution: Use gcc -Wunused-parameter instead of lint. For a few files add attributes to arguments that are known not to be used.

Files: src/auto/configure, src/buffer.c, src/charset.c, src/diff.c, src/configure.in, src/config.h.in, src/edit.c, src/ex_cmds.c, src/ex_cmds2.c, src/version.c, src/vim.h

Patch 7.2.174

Problem: Too many warnings from gcc -Wextra.

Solution: Change initializer. Add UNUSED. Add type casts.

Files: src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, getchar.c, globals.h, main.c, memline.c, message.c, src/misc1.c, src/move.c, src/normal.c, src/option.c, src/os_unix.c, src/os_unix.h, src/regexp.c, src/search.c, src/tag.c

Patch 7.2.175

Problem: Compiler warning in OpenBSD.

Solution: Add type cast for NULL. (Dasn)

Files: src/if_cscope.c

Patch 7.2.176

Problem: Exceptions for splint are not useful.

Solution: Remove the S_SPLINT_S ifdefs.

Files: src/edit.c, src/ex_cmds.c, src/ex_docmd.c, src/os_unix.c, src/os_unix.h, src/os_unixx.h, src/structs.h, src/term.h

Patch 7.2.177

Problem: Compiler warnings when using -Wextra
Solution: Add UNUSED and type casts.
Files: src/eval.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c,
src/fileio.c, src/hardcopy.c, src/if_cscope.c, src/if_xcmdsrv.c,
src/farsi.c, src/mark.c, src/menu.c

Patch 7.2.178

Problem: Using negative value for device number might not work.
Solution: Use a separate flag for whether ffv_dev was set.
Files: src/misc2.c

Patch 7.2.179

Problem: Using negative value for device number might not work.
Solution: Use a separate flag for whether sn_dev was set.
Files: src/ex_cmds2.c

Patch 7.2.180

Problem: Some more compiler warnings when using gcc -Wextra.
Solution: Add UNUSED and type casts.
Files: src/buffer.c, src/ex_cmds.c, src/macros.h, src/main.c,
src/menu.c, src/message.c, src/misc1.c, src/mbyte.c,
src/normal.c, src/option.c, src/os_unix.c, src/quickfix.c,
src/screen.c, src/search.c, src/spell.c, src/syntax.c, src/tag.c,
src/term.c, src/ui.c

Patch 7.2.181

Problem: Some more compiler warnings when using gcc -Wextra.
Solution: Add UNUSED and type casts.
Files: src/if_mzsch.c, src/gui.c, src/gui_gtk.c, src/gui_gtk_x11.c,
src/gui_gtk_f.c, src/gui_beval.c, src/netbeans.c

Patch 7.2.182 (after 7.2.181)

Problem: Compilation problems after previous patch for Motif. Gvim with
GTK crashes on startup.
Solution: Add comma. Init form structure to zeroes.
Files: src/netbeans.c, src/gui_gtk_f.c

Patch 7.2.183

Problem: Configure problem for sys/sysctl.h on OpenBSD. (Dasn)
Solution: Add separate check for this header file. Also switch to newer
version of autoconf.
Files: src/auto/configure, src/configure.in

Patch 7.2.184

Problem: Some more compiler warnings when using gcc -Wextra.
Solution: Add UNUSED and type casts. Autoconf check for wchar_t.
Files: src/auto/configure, src/config.h.in, src/configure.in,
src/gui_athena.c, src/gui_x11.c, src/gui.c, src/gui_beval.c,
src/gui_at_sb.c, src/gui_at_fs.c, src/gui_motif.c,
src/gui_xmdl.c, src/gui_xmebw.c, src/if_python.c, src/window.c,
src/workshop.c

Patch 7.2.185

Problem: Some more compiler warnings when using gcc -Wextra.
Solution: Add UNUSED and type casts.
Files: src/Makefile, src/if_tlc.c, src/if_ruby.c

Patch 7.2.186

Problem: Some more compiler warnings when using gcc -Wextra.
Solution: Now with the intended if_tcl.c changes.
Files: src/if_tcl.c

Patch 7.2.187 (after 7.2.186)

Problem: Doesn't build with older versions of TCL. (Yongwei Wu)
Solution: Add #ifdefs. (Dominique Pelle)
Files: src/if_tcl.c

Patch 7.2.188

Problem: Crash with specific use of function calls. (Meikel Brandmeyer)
Solution: Make sure the items referenced by a function call are not freed twice. (based on patch from Nico Weber)
Files: src/eval.c

Patch 7.2.189

Problem: Possible hang for deleting auto-indent. (Dominique Pelle)
Solution: Make sure the position is not beyond the end of the line.
Files: src/edit.c

Patch 7.2.190

Problem: The register executed by @@ isn't restored.
Solution: Mark the executable register in the viminfo file.
Files: src/ops.c

Patch 7.2.191

Problem: Mzscheme interface doesn't work on Ubuntu.
Solution: Change autoconf rules. Define missing macro. Some changes to avoid gcc warnings. Remove per-buffer namespace. (Sergey Khorev)
Files: runtime/doc/if_mzsch.txt, src/Makefile, src/Make_ming.mak, src/Make_mvc.mak, src/auto/configure, src/configure.in, src/config.mk.in, src/eval.c, src/if_mzsch.c, src/if_mzsch.h, src/main.c, src/proto/if_mzsch.pro

Patch 7.2.192 (after 7.2.188)

Problem: Still a crash in the garbage collector for a very rare situation.
Solution: Make sure current_copyID is always incremented correctly. (Kent Sibilev)
Files: src/eval.c

Patch 7.2.193

Problem: Warning for uninitialized values.
Solution: Initialize all the struct items.
Files: src/eval.c

Patch 7.2.194 (extra)

Problem: MSVC: rem commands are echoed.
Solution: Add commands to switch off echo. (Wang Xu)

Files: src/msvc2008.bat

Patch 7.2.195

Problem: Leaking memory for the command Vim was started with.

Solution: Remember the pointer and free it.

Files: src/gui_gtk_x11.c

Patch 7.2.196 (after 7.2.167)

Problem: Turns out splint doesn't work well enough to be usable.

Solution: Remove splint support.

Files: Filelist, src/cleanlint.vim

Patch 7.2.197

Problem: Warning for uninitialized values.

Solution: Initialize all the struct items of typebuf.

Files: src/globals.h

Patch 7.2.198

Problem: Size of buffer used for tgetent() may be too small.

Solution: Use the largest known size everywhere.

Files: src/vim.h

Patch 7.2.199

Problem: Strange character in comment.

Solution: Change to "message". (Yongwei Wu)

Files: src/term.c

Patch 7.2.200

Problem: Reading past end of string when navigating the menu bar or resizing the window.

Solution: Add and use mb_ptr2len_len(). (partly by Dominique Pelle)
Also add mb_ptr2cells_len() to prevent more trouble.

Files: src/gui_gtk_x11.c, src/os_unix.c, src/globals.h, src/mbyte.c,
src/proto/mbyte.pro

Patch 7.2.201

Problem: Cannot copy/paste HTML to/from Firefox via the clipboard.

Solution: Implement this for GTK. Add the "html" value to **'clipboard'**.

Files: runtime/doc/options.txt, src/globals.h, src/gui_gtk_x11.c,
src/mbyte.c, src/proto/mbyte.pro, src/option.c

Patch 7.2.202

Problem: BufWipeout autocommand that edits another buffer causes problems.

Solution: Check for the situation, give an error and quit the operation.

Files: src/fileio.c

Patch 7.2.203

Problem: When reloading a buffer or doing anything else with a buffer that is not displayed in a visible window, autocommands may be applied to the current window, folds messed up, etc.

Solution: Instead of using the current window for the hidden buffer use a special window, splitting the current one temporarily.

Files: src/fileio.c, src/globals.h, src/gui.c, src/if_perl.xs,
src/progo/gui.pro, src/proto/window.pro, src/screen.c,

src/structs.h, src/window.c

Patch 7.2.204 (extra)

Problem: Win32: Can't build with Visual Studio 2010 beta 1.
Solution: Fix the makefile. (George Reilly)
Files: src/Make_mvc.mak

Patch 7.2.205 (extra)

Problem: Win32: No support for High DPI awareness.
Solution: Fix the manifest file. (George Reilly)
Files: src/Make_mvc.mak, src/gvim.exe.mnf

Patch 7.2.206

Problem: Win32: Can't build netbeans interface with Visual Studio 2010.
Solution: Undefine ECONNREFUSED. (George Reilly)
Files: src/netbeans.c

Patch 7.2.207

Problem: Using freed memory with ":redrawstatus" when it works recursively.
Solution: Prevent recursively updating the status line. (partly by Dominique Pelle)
Files: src/screen.c

Patch 7.2.208

Problem: "set novice" gives an error message, it should be ignored.
Solution: Don't see "no" in "novice" as unsetting an option. (Patrick Texier)
Files: src/option.c

Patch 7.2.209

Problem: For xxd setmode() is undefined on Cygwin.
Solution: Include io.h. (Dominique Pelle)
Files: src/xxd/xxd.c

Patch 7.2.210

Problem: When a file that is being edited has its timestamp updated outside of Vim and ":checktime" is used still get a warning when writing the file. (Matt Mueller)
Solution: Store the timestamp in b_mtime_read when the timestamp is the only thing that changed.
Files: src/fileio.c

Patch 7.2.211

Problem: Memory leak when expanding a series of file names.
Solution: Use ga_clear_strings() instead of ga_clear().
Files: src/misc1.c

Patch 7.2.212 (extra)

Problem: Warnings for redefining SIG macros.
Solution: Don't define them if already defined. (Bjorn Winckler)
Files: src/os_mac.h

Patch 7.2.213

Problem: Warning for using vsprintf().

Solution: Use vim_vsnprintf().
Files: src/netbeans.c

Patch 7.2.214

Problem: Crash with complete function for user command. (Andy Wokula)
Solution: Avoid using a NULL pointer (Dominique Pelle)
Files: src/ex_getln.c

Patch 7.2.215

Problem: ml_get error when using ":vimgrep".
Solution: Load the memfile for the hidden buffer before putting it in a window. Correct the order of splitting the window and filling the window and buffer with data.
Files: src/fileio.c, src/proto/window.pro, src/quickfix.c, src/window.c

Patch 7.2.216

Problem: Two error messages have the same number E812.
Solution: Give one message a different number.
Files: runtime/doc/autocmd.txt, runtime/doc/if_mzsch.txt, src/if_mzsch.c

Patch 7.2.217

Problem: Running tests with valgrind doesn't work as advertised.
Solution: Fix the line in the Makefile.
Files: src/testdir/Makefile

Patch 7.2.218

Problem: Cannot build GTK with hangul_input feature. (Dominique Pelle)
Solution: Adjust #ifdef. (SungHyun Nam)
Files: src/gui.c

Patch 7.2.219 (extra)

Problem: Photon GUI is outdated.
Solution: Updates for QNX 6.4.0. (Sean Boudreau)
Files: src/gui_photon.c

Patch 7.2.220 (after 7.2.215)

Problem: a BufEnter autocommand that changes directory causes problems. (Ajit Thakkar)
Solution: Disable autocommands when opening a hidden buffer in a window.
Files: src/fileio.c

Patch 7.2.221

Problem: X cut_buffer0 text is used as-is, it may be in the wrong encoding.
Solution: Convert between '**enc**' and latin1. (James Vega)
Files: src/gui_gtk_x11.c, src/message.c, src/ops.c, src/proto/ui.pro, src/ui.c

Patch 7.2.222

Problem: ":mksession" doesn't work properly with '**acd**' set.
Solution: Make it work. (Yakov Lerner)
Files: src/ex_docmd.c

Patch 7.2.223

Problem: When a script is run with ":silent" it is not able to give warning

messages.
Solution: Add the ":unsilent" command.
Files: runtime/doc/various.txt, src/ex_cmds.h, src/ex_docmd.c

Patch 7.2.224

Problem: Crash when using '**completefunc**'. (Ingo Karkat)
Solution: Disallow entering edit() recursively when doing completion.
Files: src/edit.c

Patch 7.2.225

Problem: When using ":normal" a saved character may be executed.
Solution: Also store old_char when saving typeahead.
Files: src/getchar.c, src/structs.h

Patch 7.2.226

Problem: ml_get error after deleting the last line. (Xavier de Gaye)
Solution: When adjusting marks a callback may be invoked. Adjust the cursor position before invoking deleted_lines_mark().
Files: src/ex_cmds.c, src/ex_docmd.c, src/if_mzsch.c, src/if_python.c, src/if_perl.xs, src/misc1.c

Patch 7.2.227

Problem: When using ":cd" in a script there is no way to track this.
Solution: Display the directory when '**verbose**' is 5 or higher.
Files: src/ex_docmd.c

Patch 7.2.228

Problem: Cscope is limited to 8 connections.
Solution: Allocated the connection array to handle any number of connections. (Dominique Pelle)
Files: runtime/doc/if_cscop.txt, src/if_cscope.h, src/if_cscope.c

Patch 7.2.229

Problem: Warning for shadowed variable.
Solution: Rename "wait" to "wait_time".
Files: src/os_unix.c

Patch 7.2.230

Problem: A few old lint-style ARGUSED comments.
Solution: Change to the new UNUSED style.
Files: src/getchar.c

Patch 7.2.231

Problem: Warning for unreachable code.
Solution: Add #ifdef.
Files: src/if_perl.xs

Patch 7.2.232

Problem: Cannot debug problems with being in a wrong directory.
Solution: When '**verbose**' is 5 or higher report directory changes.
Files: src/os_unix.c, src/os_unix.h, src/proto/os_unix.pro

Patch 7.2.233 (extra part of 7.2.232)

Problem: Cannot debug problems with being in a wrong directory.

Solution: When **'verbose'** is 5 or higher report directory changes.
Files: src/os_msdos.c, src/os_mswin.c, src/os_riscos.c, src/os_mac.h

Patch 7.2.234

Problem: It is not possible to ignore file names without a suffix.
Solution: Use an empty entry in **'suffixes'** for file names without a dot.
Files: runtime/doc/cmdline.txt, src/misc1.c

Patch 7.2.235

Problem: Using **CTRL-O z=** in Insert mode has a delay before redrawing.
Solution: Reset msg_didout and msg_scroll.
Files: src/misc1.c, src/spell.c

Patch 7.2.236

Problem: Mac: Compiling with Ruby doesn't always work.
Solution: In configure filter out the --arch argument (Bjorn Winckler)
Files: src/configure.in, src/auto/configure

Patch 7.2.237

Problem: Crash on exit when window icon not set.
Solution: Copy terminal name when using it for the icon name.
Files: src/os_unix.c

Patch 7.2.238

Problem: Leaking memory when setting term to "builtin_dumb".
Solution: Free memory when resetting term option t_Co.
Files: src/option.c, src/proto/option.pro, src/term.c

Patch 7.2.239

Problem: Using :diffpatch twice or when patching fails causes memory corruption and/or a crash. (Bryan Venteicher)
Solution: Detect missing output file. Avoid using non-existing buffer.
Files: src/diff.c

Patch 7.2.240

Problem: Crash when using find/replace dialog repeatedly. (Michiel Hartsuiker)
Solution: Avoid doing the operation while busy or recursively. Also refuse replace when text is locked.
Files: src/gui.c

Patch 7.2.241

Problem: When using a combination of ":bufdo" and "doautoall" we may end up in the wrong directory. (Ajit Thakkar)
Crash when triggering an autocommand in ":vimgrep". (Yukihiro Nakadaira)
Solution: Clear w_localdir and globaldir when using the aucmd_win.
Use a separate flag to decide aucmd_win needs to be restored.
Files: src/fileio.c, src/globals.h, src/structs.h

Patch 7.2.242

Problem: Setting **'lazyredraw'** causes the cursor column to be recomputed. (Tom Link)
Solution: Only recompute the cursor column for a boolean option if changes

the cursor position.
Files: src/option.c

Patch 7.2.243

Problem: Memory leak when using :vimgrep and resizing. (Dominique Pelle)
Solution: Free memory for aucmd_win when resizing and don't allocate it twice.
Files: src/screen.c

Patch 7.2.244

Problem: When 'enc' is utf-8 and 'fenc' is latin1, writing a non-latin1 character gives a conversion error without any hint what is wrong.
Solution: When known add the line number to the error message.
Files: src/fileio.c

Patch 7.2.245

Problem: When 'enc' is "utf-16" and 'fenc' is "utf-8" writing a file does conversion while none should be done. (Yukihiro Nakadaira) When 'fenc' is empty the file is written as utf-8 instead of utf-16.
Solution: Do proper comparison of encodings, taking into account that all Unicode values for 'enc' use utf-8 internally.
Files: src/fileio.c

Patch 7.2.246

Problem: Cscope home page link is wrong.
Solution: Update the URL. (Sergey Khorev)
Files: runtime/doc/if_cscop.txt

Patch 7.2.247

Problem: Mzscheme interface minor problem.
Solution: Better error message when build fails. (Sergey Khorev)
Files: src/if_mzsch.c

Patch 7.2.248 (extra)

Problem: Mzscheme interface building minor problems.
Solution: Update Win32 makefiles. (Sergey Khorev)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak

Patch 7.2.249

Problem: The script to check .po files can't handle '%' in plural forms.
Solution: Remove "Plural-Forms:" from the checked string.
Files: src/po/check.vim

Patch 7.2.250 (extra)

Problem: Possible buffer overflow.
Solution: Compute the remaining space. (Dominique Pelle)
Files: src/GvimExt/gvimext.cpp

Patch 7.2.251 (after 7.2.044)

Problem: Compiler adds invalid memory bounds check.
Solution: Remove _FORTIFY_SOURCE=2 from CFLAGS. (Dominique Pelle)
Files: src/auto/configure, src/configure.in

Patch 7.2.252

Problem: When using a multibyte **'enc'** the **'iskeyword'** option cannot contain characters above 128.
Solution: Use `mb_ptr2char_adv()`.
Files: `src/charset.c`

Patch 7.2.253

Problem: Netbeans interface: `getLength` always uses current buffer.
Solution: Use `ml_get_buf()` instead of `ml_get()`. (Xavier de Gaye)
Files: `src/netbeans.c`

Patch 7.2.254

Problem: Compiler warning for assigning `size_t` to `int`.
Solution: Use `size_t` for the variable. (George Reilly)
Files: `src/fileio.c`

Patch 7.2.255 (after 7.2.242)

Problem: Setting **'rightleft'**, **'linebreak'** and **'wrap'** may cause cursor to be in wrong place.
Solution: Recompute the cursor column for these options.
Files: `src/option.c`

Patch 7.2.256

Problem: When **'guifont'** was not set GTK font dialog doesn't have a default. (Andreas Metzler)
Solution: Set default to `DEFAULT_FONT`. (James Vega)
Files: `src/gui_gtk_x11.c`

Patch 7.2.257

Problem: With GTK 2.17 lots of assertion error messages.
Solution: Remove check for static gravity. (Sebastian Droege)
Files: `src/gui_gtk_f.c`

Patch 7.2.258

Problem: `v:beval_col` and `b:beval_text` are wrong in UTF-8 text. (Tony Mechelynck)
Solution: Use byte number instead of character number for the column.
Files: `src/ui.c`

Patch 7.2.259

Problem: `exists()` doesn't work properly for an empty aucmd group.
Solution: Change how `au_exists()` handles a missing pattern. Also add a test for this. (Bob Hiestand)
Files: `src/fileio.c`, `src/testdir/Makefile`, `src/testdir/test67.in`, `src/testdir/test67.ok`

Patch 7.2.260 (extra part of 7.2.259)

Problem: `exists()` doesn't work properly for empty aucmd group.
Solution: Change how `au_exists()` handles a missing pattern. Also add a test for this. (Bob Hiestand)
Files: `src/testdir/Make_amiga.mak`, `src/testdir/Make_dos.mak`, `src/testdir/Make_ming.mak`, `src/testdir/Make_os2.mak`, `src/testdir/Make_vms.mms`

Patch 7.2.261

Problem: When deleting lines with a specific folding configuration E38 may appear. (Shahaf)
Solution: When adjusting nested folds for deleted lines take into account that they don't start at the top of the enclosing fold.
Files: src/fold.c

Patch 7.2.262

Problem: When using custom completion for a user command the pattern string goes beyond the cursor position. (Hari Krishna Dara)
Solution: Truncate the string at the cursor position.
Files: src/ex_getln.c, src/structs.h

Patch 7.2.263

Problem: GTK2: when using the `-geom` argument with an offset from the right edge and the size is smaller than the default, the Vim window is not positioned properly.
Solution: Use another function to set the size. (Vitaly Minko)
Files: src/gui_gtk_x11.c

Patch 7.2.264

Problem: GTK2: When the Vim window is maximized setting `'columns'` or `'lines'` doesn't work.
Solution: Unmaximize the window before setting the size. (Vitaly Minko)
Files: src/gui.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro

Patch 7.2.265

Problem: When using `":silent broken"` inside try/catch silency may persist. (dr-dr xp)
Solution: Set `msg_silent` when there is an error and it's bigger than the saved value.
Files: src/ex_docmd.c

Patch 7.2.266

Problem: When an expression abbreviation is triggered, the typed character is unknown.
Solution: Make the typed character available in `v:char`.
Files: runtime/doc/map.txt, src/eval.c, src/getchar.c, src/ops.c, src/proto/eval.pro

Patch 7.2.267

Problem: Crash for narrow window and double-width character.
Solution: Check for zero width. (Taro Muraoka)
Files: src/charset.c

Patch 7.2.268

Problem: Crash when using Python to set cursor beyond end of line. (winterTTr)
Solution: Check the column to be valid.
Files: src/if_python.c

Patch 7.2.269

Problem: Many people struggle to find out why Vim startup is slow.
Solution: Add the `--startuptime` command line flag.
Files: runtime/doc/starting.txt, src/globals.h, src/feature.h,

src/main.c, src/macros.h

Patch 7.2.270

Problem: Using ":@" when the c register contains a CR causes the rest to be executed later. (Dexter Douglas)
Solution: Don't check for typeahead to start with ':', keep executing commands until all added typeahead has been used.
Files: src/ex_docmd.c

Patch 7.2.271

Problem: Using freed memory in Motif GUI version when making a choice.
Solution: Free memory only after using it. (Dominique Pelle)
Files: src/gui_xmdl.c

Patch 7.2.272

Problem: ".svz" is not recognized as a swap file. (David M. Besonen)
Solution: Accept .s[uvw][a-z] as a swap file name extension.
Files: src/memline.c

Patch 7.2.273

Problem: Crash with redir to unknown array. (Christian Brabandt)
Solution: Don't assign the redir result when there was an error.
Files: src/eval.c

Patch 7.2.274

Problem: Syntax folding doesn't work properly when adding a comment.
Solution: Fix it and add a test. (Lech Lorens)
Files: src/fold.c, src/testdir/test45.in, src/testdir/test45.ok

Patch 7.2.275

Problem: Warning for unused argument and comparing signed and unsigned.
Solution: Add type cast.
Files: src/memline.c

Patch 7.2.276

Problem: Crash when setting 'isprint' to a small bullet. (Raul Coronado)
Solution: Check for the character to be < 256. Also make it possible to specify a range of multibyte characters. (Lech Lorens)
Files: src/charset.c

Patch 7.2.277

Problem: **CTRL-Y** in a diff'ed window may move the cursor outside of the window. (Lech Lorens)
Solution: Limit the number of filler lines to the height of the window. Don't reset filler lines to zero for an empty buffer.
Files: src/move.c

Patch 7.2.278

Problem: Using magic number in the folding code.
Solution: Use the defined MAX_LEVEL.
Files: src/fold.c

Patch 7.2.279

Problem: Invalid memory read with visual mode "r". (Dominique Pelle)

Solution: Make sure the cursor position is valid. Don't check the cursor position but the position being used. And make sure we get the right line.

Files: src/misc2.c, src/ops.c

Patch 7.2.280

Problem: A redraw in a custom statusline with %! may cause a crash. (Yukihiro Nakadaira)

Solution: Make a copy of '**statusline**'. Also fix typo in function name redraw_custom_statusline. (partly by Dominique Pelle)

Files: src/screen.c

Patch 7.2.281

Problem: '**cursorcolumn**' highlighting is wrong in diff mode.

Solution: Adjust the column computation. (Lech Lorens)

Files: src/screen.c

Patch 7.2.282

Problem: A fold can't be closed.

Solution: Initialize fd_small to MAYBE. (Lech Lorens)

Files: src/fold.c

Patch 7.2.283

Problem: Changing font while the window is maximized doesn't keep the window maximized.

Solution: Recompute number of lines and columns after changing font. (James Vega)

Files: src/gui_gtk_x11.c

Patch 7.2.284

Problem: When editing the same buffer in two windows, one with folding, display may be wrong after changes.

Solution: Call set_topline() to take care of side effects. (Lech Lorens)

Files: src/misc1.c

Patch 7.2.285 (after 7.2.169)

Problem: **CTRL-U** in Insert mode also deletes indent. (Andrey Voropaev)

Solution: Fix mistake made in patch 7.2.169.

Files: src/edit.c

Patch 7.2.286 (after 7.2.269)

Problem: The "--startuptime=<file>" argument is not consistent with other arguments.

Solution: Use "--startuptime <file>". Added the +startuptime feature.

Files: runtime/doc/eval.txt, runtime/doc/starting.txt, runtime/doc/various.txt, src/eval.c, src/main.c, src/version.c

Patch 7.2.287

Problem: Warning from gcc 3.4 about uninitialized variable.

Solution: Move assignment outside of #ifdef.

Files: src/if_perl.xs

Patch 7.2.288

Problem: Python 2.6 pyconfig.h redefines macros.

Solution: Undefine the macros before including pyconfig.h.
Files: src/if_python.c

Patch 7.2.289

Problem: Checking wrong struct member.
Solution: Change tb_buf to tb_noremap. (Dominique Pelle)
Files: src/getchar.c

Patch 7.2.290

Problem: Not freeing memory from ":lmap", ":xmap" and ":menutranslate".
Solution: Free the memory when exiting. (Dominique Pelle)
Files: src/misc2.c

Patch 7.2.291

Problem: Reading uninitialised memory in arabic mode.
Solution: Use utfc_ptr2char_len() rather than utfc_ptr2char(). (Dominique Pelle)
Files: src/screen.c

Patch 7.2.292

Problem: Block right-shift doesn't work properly with multibyte encoding and '**list**' set.
Solution: Add the missing "else". (Lech Lorens)
Files: src/ops.c

Patch 7.2.293

Problem: When setting '**comments**' option it may be used in a wrong way.
Solution: Don't increment after skipping over digits. (Yukihiro Nakadaira)
Files: src/misc1.c

Patch 7.2.294

Problem: When using TEMPDIRS dir name could get too long.
Solution: Overwrite tail instead of appending each time. Use mkdtemp() when available. (James Vega)
Files: src/auto/configure, src/config.h.in, src/configure.in, src/fileio.c

Patch 7.2.295

Problem: When using map() on a List the index is not known.
Solution: Set v:key to the index. (Hari Krishna Dara)
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.2.296

Problem: Help message about startuptime is wrong. (Dominique Pelle)
Solution: Remove the equal sign.
Files: src/main.c

Patch 7.2.297

Problem: Reading freed memory when writing ":reg" output to a register. (Dominique Pelle)
Solution: Skip the register being written to.
Files: src/ops.c

Patch 7.2.298

Problem: ":vimgrep" crashes when there is an autocommand that sets a

window-local variable.
Solution: Initialize the w: hashtab for re-use. (Yukihiro Nakadaira)
Files: src/fileio.c

Patch 7.2.299

Problem: Crash when comment middle is longer than start.
Solution: Fix size computation. (Lech Lorens)
Files: src/misc1.c

Patch 7.2.300

Problem: Vim doesn't close file descriptors when forking and executing another command, e.g., ":shell".
Solution: Use FD_CLOEXEC when available. (James Vega)
Files: auto/configure, src/config.h.in, src/configure.in, src/ex_cmdds2.c, src/fileio.c, src/memfile.c, src/memline.c

Patch 7.2.301

Problem: Formatting is wrong when 'tw' is set to a small value.
Solution: Fix it and add tests. Also fix behavior of "1" in 'fo'. (Yukihiro Nakadaira)
Files: src/edit.c, src/testdir/Makefile, src/testdir/test68.in, src/testdir/test68.ok, src/testdir/test69.in, src/testdir/test69,ok

Patch 7.2.302 (extra part of 7.2.301)

Problem: Formatting wrong with small 'tw' value.
Solution: Add build rules for tests.
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.2.303 (after 7.2.294)

Problem: Can't build on MS-Windows.
Solution: Add #ifdef around vim_settempdir(). (James Vega)
Files: src/fileio.c

Patch 7.2.304

Problem: Compiler warning for bad pointer cast.
Solution: Use another variable for int pointer.
Files: src/ops.c

Patch 7.2.305

Problem: Recursively redrawing causes a memory leak. (Dominique Pelle)
Solution: Disallow recursive screen updating.
Files: src/screen.c

Patch 7.2.306

Problem: shellescape("10%%", 1) only escapes first %_. (Christian Brabandt)
Solution: Don't copy the character after the escaped one.
Files: src/misc2.c

Patch 7.2.307

Problem: Crash with a very long syntax match statement. (Guy Gur Ari)
Solution: When the offset does not fit in the two bytes available give an

error instead of continuing with invalid pointers.
Files: src/regexp.c

Patch 7.2.308

Problem: When using a regexp in the "\=" expression of a substitute command, submatch() returns empty strings for further lines. (Clockwork Jam)

Solution: Save and restore the line number and line count when calling reg_getline().

Files: src/regexp.c

Patch 7.2.309 (after 7.2.308)

Problem: Warning for missing function prototype. (Patrick Texier)

Solution: Add the prototype.

Files: src/regexp.c

Patch 7.2.310

Problem: When a filetype plugin in ~/.vim/ftdetect uses ":setfiletype" and the file starts with a "# comment" it gets "conf" filetype.

Solution: Check for "conf" filetype after using ftdetect plugins.

Files: runtime/filetype.vim

Patch 7.2.311

Problem: Can't compile with FreeMiNT.

Solution: Change #ifdef for limits.h. (Alan Hourihane)

Files: src/fileio.c

Patch 7.2.312

Problem: iconv() returns an invalid character sequence when conversion fails. It should return an empty string. (Yongwei Wu)

Solution: Be more strict about invalid characters in the input.

Files: src/mbyte.c

Patch 7.2.313

Problem: Command line completion doesn't work after "%:h" and similar.

Solution: Expand these items before doing the completion.

Files: src/ex_getln.c, src/misc1.c, src/proto/misc1.pro

Patch 7.2.314

Problem: Missing function in small build.

Solution: Always include concat_str.

Files: src/misc1.c

Patch 7.2.315

Problem: Python libs can't be found on 64 bit system.

Solution: Add lib64 to the list of directories. (Michael Henry)

Files: src/auto/configure, src/configure.in

Patch 7.2.316

Problem: May get multiple _FORTIFY_SOURCE arguments. (Tony Mechelynck)

Solution: First remove all these arguments and then add the one we want. (Dominique Pelle)

Files: src/auto/configure, src/configure.in

Patch 7.2.317

Problem: Memory leak when adding a highlight group with unprintable characters, resulting in E669.
Solution: Free the memory. And fix a few typos. (Dominique Pelle)
Files: src/syntax.c

Patch 7.2.318

Problem: Wrong locale value breaks floating point numbers for gvim.
Solution: Set the locale again after doing GUI inits. (Dominique Pelle)
Files: src/main.c

Patch 7.2.319

Problem: Motif: accessing freed memory when cancelling font dialog.
Solution: Destroy the widget only after accessing it. (Dominique Pelle)
Files: src/gui_xmdlg.c

Patch 7.2.320

Problem: Unused function in Mzscheme interface.
Solution: Remove the function and what depends on it. (Dominique Pelle)
Files: src/if_mzsch.c, src/proto/if_mzsch.pro

Patch 7.2.321

Problem: histadd() and searching with "*" fails to add entry to history when it is empty.
Solution: Initialize the history. (Lech Lorens)
Files: src/eval.c, src/normal.c

Patch 7.2.322

Problem: Wrong indenting in virtual replace mode with **CTRL-Y** below a short line.
Solution: Check for character to be NUL. (suggested by Lech Lorens)
Files: src/edit.c

Patch 7.2.323 (extra)

Problem: Balloon evaluation crashes on Win64.
Solution: Change pointer types. (Sergey Khorev)
Files: src/gui_w32.c

Patch 7.2.324

Problem: A negative column argument in setpos() may cause a crash.
Solution: Check for invalid column number. (James Vega)
Files: src/eval.c, src/misc2.c

Patch 7.2.325

Problem: A stray "w" in the startup vimrc file causes the edited file to be replaced with an empty file. (Stone Kang).
Solution: Do not write a buffer when it has never been loaded.
Files: src/fileio.c

Patch 7.2.326

Problem: Win32: \$HOME doesn't work when %HOMEPATH% is not defined.
Solution: Use "\" for %HOMEPATH% when it is not defined.
Files: src/misc1.c

Patch 7.2.327

Problem: Unused functions in Workshop.
Solution: Add "#if 0" and minor cleanup. (Dominique Pelle)
Files: src/workshop.c, src/integration.c, src/integration.h

Patch 7.2.328

Problem: has("win64") does not return 1 on 64 bit MS-Windows version.
Solution: Also check for _WIN64 besides WIN64.
Files: src/eval.c

Patch 7.2.329

Problem: "g_" doesn't position cursor correctly when in Visual mode and '**selection**' is "exclusive". (Ben Fritz)
Solution: Call adjust_for_sel().
Files: src/normal.c

Patch 7.2.330

Problem: Tables for Unicode case operators are outdated.
Solution: Add a Vim script for generating the tables. Include tables for Unicode 5.2.
Files: runtime/tools/README.txt, runtime/tools/unicode.vim, src/mbyte.c

Patch 7.2.331

Problem: Can't interrupt "echo list" for a very long list.
Solution: Call line_breakcheck() in list_join().
Files: src/eval.c

Patch 7.2.332

Problem: Crash when spell correcting triggers an autocommand that reloads the buffer.
Solution: Make a copy of the line to be modified. (Dominique Pelle)
Files: src/spell.c

Patch 7.2.333

Problem: Warnings from static code analysis.
Solution: Small changes to various lines. (Dominique Pelle)
Files: src/buffer.c, src/edit.c, src/ex_getln.c, src/fileio.c, src/if_cscope.c, src/netbeans.c, src/ops.c, src/quickfix.c, src/syntax.c, src/ui.c

Patch 7.2.334

Problem: Postponing keys in Netbeans interface does not work properly.
Solution: Store the key string instead of the number. Avoid an infinite loop. (Mostly by Xavier de Gaye)
Files: src/netbeans.c, src/proto/netbeans.pro

Patch 7.2.335

Problem: The **CTRL-]** command escapes too many characters.
Solution: Use a different list of characters to be escaped. (Sergey Khorev)
Files: src/normal.c

Patch 7.2.336

Problem: MzScheme interface can't evaluate an expression.
Solution: Add mzeval(). (Sergey Khorev)

Files: runtime/doc/eval.txt, runtime/doc/if_mzsch.txt,
runtime/doc/usr_41.txt, src/eval.c, src/if_mzsch.c,
src/proto/eval.pro, src/proto/if_mzsch.pro,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Makefile, src/testdir/main.aap, src/testdir/test1.in,
src/testdir/test70.in, src/testdir/test70.ok

Patch 7.2.337

Problem: The :compiler command doesn't function properly when invoked in a function.

Solution: Add "g:" before "current_compiler". (Yukihiro Nakadaira)

Files: src/ex_cmds2.c

Patch 7.2.338 (after 7.2.300)

Problem: Part of FD_CLOEXEC change is missing.

Solution: Include source file skipped because of typo.

Files: src/ex_cmds2.c

Patch 7.2.339 (after 7.2.269)

Problem: Part of --startuptime patch is missing.

Solution: Add check for time_fd.

Files: src/ex_cmds2.c

Patch 7.2.340

Problem: Gcc warning for condition that can never be true. (James Vega)

Solution: Use start_lvl instead flp->lvl.

Files: src/fold.c

Patch 7.2.341

Problem: Popup menu wraps to next line when double-wide character doesn't fit. (Jiang Ma)

Solution: Display a ">" instead. (Dominique Pelle)

Files: src/screen.c

Patch 7.2.342

Problem: Popup menu displayed wrong in '**rightleft**' mode when there are multibyte characters.

Solution: Adjust the column computations. (Dominique Pelle)

Files: src/popupmnu.c

Patch 7.2.343 (after 7.2.338)

Problem: Can't compile on Win32.

Solution: Insert the missing '|'.

Files: src/ex_cmds2.c

Patch 7.2.344 (after 7.2.343)

Problem: Can't compile on some systems

Solution: Move the #ifdef outside of the mch_open macro. (Patrick Texier)

Files: src/ex_cmds2.c

Patch 7.2.345

Problem: Tab line is not updated when the value of '**bt**' is changed.

Solution: Call redraw_titles(). (Lech Lorens)

Files: src/option.c

Patch 7.2.346

Problem: Repeating a command with @: causes a mapping to be applied twice.
Solution: Do not remap characters inserted in the typeahead buffer. (Kana Natsuno)
Files: src/ops.c

Patch 7.2.347

Problem: Crash when executing <expr> mapping redefines that same mapping.
Solution: Save the values used before evaluating the expression.
Files: src/getchar.c

Patch 7.2.348 (after 7.2.330)

Problem: Unicode double-width characters are not up-to date.
Solution: Produce the double-width table like the others.
Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 7.2.349

Problem: **CTRL-W** gf doesn't put the new tab in the same place as "tab split" and "gf". (Tony Mechelynck)
Solution: Store the tab number in cmdmod.tab.
Files: src/window.c

Patch 7.2.350

Problem: Win32: When changing font the window may jump from the secondary to the primary screen. (Michael Wookey)
Solution: When the screen position was negative don't correct it to zero.
Files: src/gui.c

Patch 7.2.351 (after 7.2.347)

Problem: Can't build with some compilers.
Solution: Move the #ifdef outside of a macro. Cleanup the code.
Files: src/getchar.c

Patch 7.2.352 (extra)

Problem: Win64: Vim doesn't work when cross-compiled with MingW libraries.
Solution: Always return TRUE for the WM_NCCREATE message. (Andy Kittner)
Files: src/gui_w48.c

Patch 7.2.353

Problem: No command line completion for ":profile".
Solution: Complete the subcommand and file name.
Files: src/ex_docmd.c, src/ex_cmds2.c, src/ex_getln.c, src/proto/ex_cmds2.pro, src/vim.h

Patch 7.2.354

Problem: Japanese single-width double-byte characters not handled correctly.
Solution: Put 0x8e in ScreenLines[] and the second byte in ScreenLines2[].
(partly by Kikuchan)
Files: src/screen.c

Patch 7.2.355

Problem: Computing the cursor column in validate_cursor_col() is wrong when line numbers are used and 'n' is not in '**coptions**', causing the

popup menu to be positioned wrong.
Solution: Correctly use the offset. (partly by Dominique Pelle)
Files: src/move.c

Patch 7.2.356

Problem: When **'foldmethod'** is changed not all folds are closed as expected.
Solution: In foldUpdate() correct the start position and reset fd_flags when w_foldinvalid is set. (Lech Lorens)
Files: src/fold.c

Patch 7.2.357

Problem: When changing **'fileformat'** from/to "mac" and there is a CR in the text the display is wrong.
Solution: Redraw the text when **'fileformat'** is changed. (Ben Schmidt)
Files: src/option.c

Patch 7.2.358

Problem: Compiler warnings on VMS. (Zoltan Arpadffy)
Solution: Pass array itself instead its address. Return a value.
Files: src/gui_gtk_x11.c, src/os_unix.c

Patch 7.2.359

Problem: Crash when using the Netbeans join command.
Solution: Make sure the ml_flush_line() function is not used recursively. (Xavier de Gaye)
Files: src/memline.c

Patch 7.2.360

Problem: Ruby on MS-Windows: can't use sockets.
Solution: Call NtInitialize() during initialization. (Ariya Mizutani)
Files: src/if_ruby.c

Patch 7.2.361

Problem: Ruby 1.9 is not supported.
Solution: Add Ruby 1.9 support. (Masaki Suketa)
Files: src/Makefile, src/auto/configure, src/configure.in, src/if_ruby.c

Patch 7.2.362 (extra, after 7.2.352)

Problem: Win64: Vim doesn't work when cross-compiled with MingW libraries.
Solution: Instead of handling WM_NCCREATE, create wide text area window class if the parent window iw side. (Sergey Khorev)
Files: src/gui_w32.c, src/gui_w48.c

Patch 7.2.363

Problem: Can't dynamically load Perl 5.10.
Solution: Add the function Perl_croak_xs_usage. (Sergey Khorev)
Files: src/if_perl.xs

Patch 7.2.364 (extra)

Problem: Can't build gvimext.dll on Win 7 x64 using MinGW (John Marriott)
Solution: Check if _MSC_VER is defined. (Andy Kittner)
Files: src/GvimExt/gvimext.h

Patch 7.2.365 (extra)

Problem: MS-Windows with MingW: "File->Save As" does not work. (John Marriott)
Solution: Correctly fill in structure size. (Andy Kittner)
Files: src/gui_w48.c

Patch 7.2.366

Problem: **CTRL-B** doesn't go back to the first line of the buffer.
Solution: Avoid an overflow when adding MAXCOL.
Files: src/move.c

Patch 7.2.367

Problem: "xxd -r -p" doesn't work as documented.
Solution: Skip white space. (James Vega)
Files: src/xxd/xxd.c

Patch 7.2.368 (after 7.2.361)

Problem: Ruby interface: Appending line doesn't work. (Michael Henry)
Solution: Reverse check for NULL line. (James Vega)
Files: src/if_ruby.c

Patch 7.2.369

Problem: Error message is not easy to understand.
Solution: Add quotes. (SungHyun Nam)
Files: src/ex_cmds2.c

Patch 7.2.370 (after 7.2.356)

Problem: A redraw may cause folds to be closed.
Solution: Revert part of the previous patch. Add a test. (Lech Lorens)
Files: src/diff.c, src/fold.c, src/option.c, src/testdir/test45.in, src/testdir/test45.ok

Patch 7.2.371

Problem: Build problems on Tandem NonStop.
Solution: A few changes to #ifdefs (Joachim Schmitz)
Files: src/auto/configure, src/configure.in, src/config.h.in, src/vim.h, src/if_cscope.c, src/osdef1.h.in, src/tag.c

Patch 7.2.372 (extra)

Problem: Cross-compiling GvimExt and xxd doesn't work.
Solution: Change the build files. (Markus Heidelberg)
Files: src/INSTALLpc.txt, src/GvimExt/Make_ming.mak, src/Make_cyg.mak, src/Make_ming.mak, src/xxd/Make_cyg.mak

Patch 7.2.373

Problem: Gcc 4.5 adds more error messages. (Chris Indy)
Solution: Update default **'errorformat'**.
Files: src/option.h

Patch 7.2.374

Problem: Ruby eval() doesn't understand Vim types.
Solution: Add the vim_to_ruby() function. (George Gensure)
Files: src/eval.c, src/if_ruby.c

Patch 7.2.375

Problem: ml_get errors when using ":bprevious" in a BufEnter autocmd.
(Dominique Pelle)
Solution: Clear w_valid when entering another buffer.
Files: src/buffer.c

Patch 7.2.376

Problem: ml_get error when using SiSU syntax. (Nathan Thomas)
Solution: If the match ends below the last line move it to the end of the
last line.
Files: src/syntax.c

Patch 7.2.377 (extra, after 7.2.372)

Problem: Misplaced assignment. Duplicate build line for gvimext.dll.
Solution: Move setting CROSS_COMPILE to before ifneq. Remove the wrong
build line. (Markus Heidelberg)
Files: src/Make_ming.mak

Patch 7.2.378

Problem: C function declaration indented too much. (Rui)
Solution: Don't see a line containing { or } as a type. (Matt Wozniski)
Files: src/misc1.c

Patch 7.2.379

Problem: 'eventignore' is set to an invalid value inside ":doau". (Antony
Scriven)
Solution: Don't include the leading comma when the option was empty.
Files: src/fileio.c

Patch 7.2.380 (after 7.2.363)

Problem: Perl interface builds with 5.10.1 but not with 5.10.0.
Solution: Change the #ifdefs. (Sergey Khorev)
Files: src/if_perl.xs

Patch 7.2.381

Problem: No completion for :behave.
Solution: Add :behave completion. Minor related fixes. (Dominique Pelle)
Files: src/ex_docmd.c, src/ex_getln.c, src/proto/ex_docmd.pro, src/vim.h

Patch 7.2.382

Problem: Accessing freed memory when closing the cmdline window when
'bufhide' is set to "wipe".
Solution: Check if the buffer still exists before invoking close_buffer()
(Dominique Pelle)
Files: src/ex_getln.c

Patch 7.2.383

Problem: Vim doesn't build cleanly with MSVC 2010.
Solution: Change a few types. (George Reilly)
Files: src/ex_cmds2.c, src/if_python.c, src/syntax.c

Patch 7.2.384 (extra)

Problem: Vim doesn't build properly with MSVC 2010.
Solution: Add the nmake version to the build file. (George Reilly)
Files: src/Make_mvc.mak, src/testdir/Make_dos.mak

Patch 7.2.385

Problem: When in the command line window dragging status line only works for last-but-one window. (Jean Johner)
Solution: Remove the code that disallows this.
Files: src/ui.c

Patch 7.2.386

Problem: Focus hack for KDE 3.1 causes problems for other window managers.
Solution: Remove the hack. (forwarded by Joel Bradshaw)
Files: src/gui_gtk.c

Patch 7.2.387

Problem: Ruby with MingW still doesn't build all versions.
Solution: More #ifdefs for the Ruby code. (Sergey Khorev)
Files: src/if_ruby.c

Patch 7.2.388 (extra part of 7.2.387)

Problem: Ruby with MingW still doesn't build all versions.
Solution: Different approach to build file. (Sergey Khorev)
Files: src/Make_ming.mak

Patch 7.2.389

Problem: synIDattr() cannot return the font.
Solution: Support the "font" argument. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/eval.c, src/syntax.c

Patch 7.2.390

Problem: In some situations the popup menu can be displayed wrong.
Solution: Remove the popup menu if the cursor moved. (Lech Lorens)
Files: src/edit.c

Patch 7.2.391

Problem: Internal alloc(0) error when doing "**CTRL-V** \$ c". (Martti Kuparinen)
Solution: Fix computations in getvcol(). (partly by Lech Lorens)
Files: src/charset.c, src/memline.c

Patch 7.2.392

Problem: Netbeans hangs reading from a socket at the maximum block size.
Solution: Use select() or poll(). (Xavier de Gaye)
Files: src/vim.h, src/os_unixx.h, src/if_xcmdsrv.c, src/netbeans.c

Patch 7.2.393

Problem: Mac: Can't build with different Xcode developer tools directory.
Solution: make "Developer" directory name configurable. (Rainer Muller)
Files: src/configure.in, src/auto/configure

Patch 7.2.394

Problem: .lzma and .xz files are not supported.
Solution: Recognize .lzma and .xz files so that they can be edited.
Files: runtime/plugin/gzip.vim

Patch 7.2.395

Problem: In help CTRL=] on g?g? escapes the ?, causing it to fail. (Tony

Mechelynck)
Solution: Don't escape ? for a help command. (Sergey Khorev)
Files: src/normal.c

Patch 7.2.396

Problem: Get E38 errors. (Dasn)
Solution: Set cursor to line 1 instead of 0. (Dominique Pelle)
Files: src/popupmnu.c

Patch 7.2.397

Problem: Redundant check for w_lines_valid.
Solution: Remove the if. (Lech Lorens)
Files: src/fold.c

Patch 7.2.398

Problem: When moving windows the cursor ends up in the wrong line.
Solution: Set the window width and height properly. (Lech Lorens)
Files: src/window.c

Patch 7.2.399 (extra, after 7.2.388)

Problem: Cannot compile on MingW.
Solution: Move ifneq to separate line. (Vlad Sandrini, Dominique Pelle)
Files: src/Make_ming.mak

Patch 7.2.400 (after 7.2.387)

Problem: Dynamic Ruby is not initialised properly for version 1.9.1.
Ruby cannot create strings from NULL.
Solution: Cleanup #ifdefs. Handle NULL like an empty string. Add
ruby_init_stack. (Sergey Khorev)
Files: src/if_ruby.c

Patch 7.2.401

Problem: ":e dir<Tab>" with 'wildmode' set to "list" doesn't highlight
directory names with a space. (Alexandre Provencio)
Solution: Remove the backslash before checking if the name is a directory.
(Dominique Pelle)
Files: src/ex_getln.c

Patch 7.2.402

Problem: This gives a #705 error: let X = function('haslocaldir')
let X = function('getcwd')
Solution: Don't give E705 when the name is found in the hashtab. (Sergey
Khorev)
Files: src/eval.c

Patch 7.2.403 (after 7.2.400)

Problem: Compiler warning for pointer type. (Tony Mechelynck)
Solution: Move type cast to the right place.
Files: src/if_ruby.c

Patch 7.2.404

Problem: Pointers for composing characters are not properly initialized.
Solution: Compute the size of the pointer, not what it points to. (Yukihiro
Nakadaira)

Files: src/screen.c

Patch 7.2.405

Problem: When built with small features the matching text is not highlighted for ":s/pat/repl/c".

Solution: Remove the #ifdef for IncSearch. (James Vega)

Files: src/syntax.c

Patch 7.2.406

Problem: Patch 7.2.119 introduces uninit mem read. (Dominique Pelle)

Solution: Only used ScreenLinesC when ScreenLinesUC is not zero. (Yukihiro Nakadaira) Also clear ScreenLinesC when allocating.

Files: src/screen.c

Patch 7.2.407

Problem: When using an expression in ":s" backslashes in the result are dropped. (Sergey Goldgaber, Christian Brabandt)

Solution: Double backslashes.

Files: src/regexp.c

Patch 7.2.408

Problem: With ":g/the/s/foo/bar/" the '[' and ']' marks can be set to a line that was not changed.

Solution: Only set '[' and ']' marks when a substitution was done.

Files: src/ex_cmds.c

Patch 7.2.409

Problem: Summary of number of substitutes is incorrect for ":folddo". (Jean Johner)

Solution: Reset sub_nsubs and sub_nlines in global_exe().

Files: src/ex_cmds.c

Patch 7.2.410

Problem: Highlighting directories for completion doesn't work properly.

Solution: Don't halve backslashes when not needed, expanded "~/". (Dominique Pelle)

Files: src/ex_getln.c

Patch 7.2.411

Problem: When parsing '**cin**' a comma isn't skipped properly.

Solution: Skip the comma. (Lech Lorens)

Files: src/misc1.c

Patch 7.2.412

Problem: [or] followed by mouse click doesn't work.

Solution: Reverse check for key being a mouse event. (Dominique Pelle)

Files: src/normal.c

Patch 7.2.413

Problem: Large file support is incorrect.

Solution: Add AC_SYS_LARGEFILE to configure. (James Vega)

Files: src/configure.in, src/config.h.in, src/auto/configure

Patch 7.2.414

Problem: CTRL-K <space> <space> does not produce 0xa0 as expected. (Tony Mechelynck)
Solution: Remove the Unicode range 0xe000 - 0xefff from digraphs, these are not valid characters.
Files: src/digraph.c

Patch 7.2.415

Problem: Win32: Can't open a remote file when starting Vim.
Solution: Don't invoke cygwin_conv_path() for URLs. (Tomoya Adachi)
Files: src/main.c

Patch 7.2.416

Problem: Logtalk.dict is not installed.
Solution: Add it to the install target. (Markus Heidelberg)
Files: src/Makefile

Patch 7.2.417

Problem: When 'shell' has an argument with a slash then 'shellpipe' is not set properly. (Britton Kerin)
Solution: Assume there are no spaces in the path, arguments follow.
Files: src/option.c

Patch 7.2.418

Problem: Vim tries to set the background or foreground color in a terminal to -1. (Graywh) Happens with ":hi Normal ctermbg=NONE".
Solution: When resetting the foreground or background color don't set the color, let the clear screen code do that.
Files: src/syntax.c

Patch 7.2.419

Problem: Memory leak in Motif when clicking on "Search Vim Help".
Solution: Free string returned by XmTextGetString(). (Dominique Pelle)
Files: src/gui_motif.c

Patch 7.2.420

Problem: ":argedit" does not accept "++enc=utf8" as documented. (Dominique Pelle)
Solution: Add the ARGOPT flag to ":argedit".
Files: src/ex_cmds.h

Patch 7.2.421

Problem: Folds are sometimes not updated properly and there is no way to force an update.
Solution: Make "zx" and "zX" recompute folds (suggested by Christian Brabandt)
Files: src/normal.c

Patch 7.2.422

Problem: May get E763 when using spell dictionaries.
Solution: Avoid utf-8 case folded character to be truncated to 8 bits and differ from latin1. (Dominique Pelle)
Files: src/spell.c

Patch 7.2.423

Problem: Crash when assigning s: to variable. (Yukihiro Nakadaira)
Solution: Make ga_scripts contain pointer to scriptvar_T instead of scriptvar_T itself. (Dominique Pelle)
Files: src/eval.c

Patch 7.2.424

Problem: ":colorscheme" without an argument doesn't do anything.
Solution: Make it echo the current color scheme name. (partly by Christian Brabandt)
Files: runtime/doc/syntax.txt, src/ex_cmds.h, src/ex_docmd.c

Patch 7.2.425

Problem: Some compilers complain about fourth EX() argument.
Solution: Add cast to long_u.
Files: src/ex_cmds.h

Patch 7.2.426

Problem: Commas in '**langmap**' are not always handled correctly.
Solution: Require commas to be backslash escaped. (James Vega)
Files: src/option.c

Patch 7.2.427

Problem: The swapfile is created using the destination of a symlink, but recovery doesn't follow symlinks.
Solution: When recovering, resolve symlinks. (James Vega)
Files: src/memline.c

Patch 7.2.428

Problem: Using setqflist([]) to clear the error list doesn't work properly.
Solution: Set qf_nonevalid to TRUE when appropriate. (Christian Brabandt)
Files: src/quickfix.c

Patch 7.2.429

Problem: A file that exists but access is denied may result in a "new file" message. E.g. when its directory is unreadable.
Solution: Specifically check for ENOENT to decide a file doesn't exist. (partly by James Vega)
Files: src/fileio.c

Patch 7.2.430

Problem: The ++bad argument is handled wrong, resulting in an invalid memory access.
Solution: Use the bad_char field only for the replacement character, add bad_char_idx to store the position. (Dominique Pelle)
Files: src/eval.c, src/ex_cmds.h, src/ex_docmd.c

Patch 7.2.431

Problem: ":amenu" moves the cursor when in Insert mode.
Solution: Use CTRL-\ **CTRL-O** instead of **CTRL-O**. (Christian Brabandt)
Files: src/menu.c

Patch 7.2.432

Problem: When menus are translated they can only be found by the translated name. That makes ":emenu" difficult to use.

Solution: Store the untranslated name and use it for completion and :emenu.
(Liang Peng (Bezetek James), Edward L. Fox)
Files: src/menu.c, src/structs.h

Patch 7.2.433

Problem: Can't use cscope with QuickFixCmdPre and QuickFixCmdPost.
Solution: Add cscope support for these autocmd events. (Bryan Venteicher)
Files: runtime/doc/autocmd.txt, src/if_cscope.c

Patch 7.2.434 (after 7.2.432)

Problem: Compilation fails without the multi-lang feature.
Solution: Add #ifdefs. (John Marriott)
Files: src/menu.c

Patch 7.2.435 (after 7.2.430)

Problem: Crash when using bad_char_idx uninitialized. (Patrick Texier)
Solution: Don't use bad_char_idx, reproduce the ++bad argument from bad_char.
Files: src/eval.c, src/ex_cmds.h, src/ex_docmd.c

Patch 7.2.436

Problem: Reproducible crash in syntax HL. (George Reilly, Dominique Pelle)
Solution: Make sst_stacksize an int instead of short. (Dominique Pelle)
Files: src/structs.h

Patch 7.2.437 (after 7.2.407)

Problem: When "\\n" appears in the expression result the \n doesn't result
in a line break. (Andy Wokula)
Solution: Also replace a \n after a backslash into \r.
Files: src/regexp.c

Patch 7.2.438 (after 7.2.427)

Problem: "vim -r" crashes.
Solution: Don't use NULL pointer argument.
Files: src/memline.c

Patch 7.2.439

Problem: Invalid memory access when doing thesaurus completion and
'infercase' is set.
Solution: Use the minimal length of completed word and replacement.
(Dominique Pelle)
Files: src/edit.c

Patch 7.2.440

Problem: Calling a function through a funcref, where the function deletes
the funcref, leads to an invalid memory access.
Solution: Make a copy of the function name. (Lech Lorens)
Files: src/eval.c, src/testdir/test34.in, src/testdir/test34.ok

Patch 7.2.441

Problem: When using ":earlier" undo information may be wrong.
Solution: When changing alternate branches also adjust b_u_oldhead.
Files: src/undo.c

Patch 7.2.442 (after 7.2.201)

Problem: Copy/paste with OpenOffice doesn't work.
Solution: Do not offer the HTML target when it is not supported. (James Vega)
Files: src/gui_gtk_x11.c, src/option.c, src/proto/gui_gtk_x11.pro

Patch 7.2.443

Problem: Using taglist() on a tag file with duplicate fields generates an internal error. (Peter Odding)
Solution: Check for duplicate field names.
Files: src/eval.c, src/proto/eval.pro, src/tag.c

Patch 7.2.444 (after 7.2.442)

Problem: Can't build with GTK 1, gtk_selection_clear_targets() is not available. (Patrick Texier)
Solution: Don't change the targets for GTK 1, set them once.
Files: src/gui_gtk_x11.c, src/option.c

Patch 7.2.445

Problem: Crash when using undo/redo and a FileChangedRO autocmd event that reloads the buffer. (Dominique Pelle)
Solution: Do not allow autocommands while performing an undo or redo.
Files: src/misc1.c, src/undo.c

Patch 7.2.446

Problem: Crash in GUI when closing the last window in a tabpage. (ryo7000)
Solution: Remove the tabpage from the list before freeing the window.
Files: src/window.c

When writing a file, switching tab pages and selecting a word the file write message would be displayed again. This happened in Insert mode and with `'cmdheight'` set to 2.

When using `":lang"` to set a locale that uses a comma for decimal separator and using GTK floating point numbers stop working. Use `gtk_disable_setlocale()`. (James Vega)

`"g8"` didn't produce the right value on a NUL. (Dominique Pelle)

Use `BASEMODLIBS` instead of `MODLIBS` for Python configuration to pick up the right compiler flags. (Michael Bienia)

Window title was not updated after dropping a file on Vim. (Hari G)

`synstack()` did not return anything when just past the end of the line. Useful when using the cursor position in Insert mode.

When entering a digraph or special character after a line that fits the window the `'?'` or `'^'` on the next line is not redrawn. (Ian Kelling)

Composing characters in `:s` substitute text were dropped.

`exists()` was causing an autoload script to be loaded.

Filter out `-pthread` for `cproto`.

Make **CTRL-L** in command line mode respect '**ignorecase**' and '**smartcase**'. (Martin Toft)

Spell menu moved the cursor, causing Copy not to work. Spell replacement didn't work in '**compatible**' mode.

Various small fixes from Dominique Pelle.

Fix that :mksession may generate "2argu" even though there is no such argument. (Peter Odding)

Fixes for time in clipboard request. Also fix ownership. (David Fries)

Fixed completion of file names with '%' and '*'.

Fixed MSVC makefile use of /Wp64 flag.

Correct use of long instead of off_t for file size. (James Vega)

Add a few #ifdefs to exclude functions that are not used. (Dominique Pelle)

Remove old and unused method to allocate memory for undo.

Fix definition of UINT_PTR for 64 bit systems.

Some versions of Ruby redefine rb_str_new2 to rb_str_new_cstr.

Window title not updated after file dropped.

Fixed crash for ":find" completion, might also happen in other path expansion usage.

When '**searchhl**' causes a hang make **CTRL-C** disable '**searchhl**'.

When resetting both '**title**' and '**icon**' the title would be set after a shell command.

Reset '**title**' and '**icon**' in test47 to avoid the xterm title getting messed up.

Fix for compiler warning about function prototype in pty.c.

Added '**window**' to the options window.

Fixed: errors for allocating zero bytes when profiling an empty function.

Remove -arch flag from build flags for Perl. (Bjorn Wickler)

Fix '**autochdir**' not showing up in :options window. (Dominique Pelle)

Fix: test 69 didn't work on MS-Windows. Test 72 beeped too often.

Avoid illegal memory access in spell suggestion. (Dominique Pelle)

Fix: crash in spell checking with a 0x300 character.

Avoid that running tests changes viminfo.

Fix: changing case of a character removed combining characters.
Fixed: **CTRL-R** in Insert mode doesn't insert composing characters.

Added the WOW64 flag to OLE registration, for 64 bit Windows systems.

Various fixes for coverity warnings.

Fix compile warnings, esp. for 64-bit systems. (Mike Williams)

Fix: :redir to a dictionary that is changed before ":redir END" causes a memory access error.

Fix: terminal title not properly restored when there are multibyte characters. (partly by James Vega)

Set **'wrapscan'** when checking the .po files. (Mike Williams)

Win32: Put quotes around the gvim.exe path for the "Open with" menu entry.

On MS-Windows sometimes files with number 4913 or higher are left behind.

'suffixesadd' was used for finding tags file.

Removed unused code.

Improved positioning of combining characters in GTK.

Made test 11 pass when there is no gzip program. (John Beckett)

Changed readfile() to ignore byte order marks, unless in binary mode.

On MS-Windows completion of shell commands didn't work.

An unprintable multibyte character at the start of the screen line caused the following text to be drawn at the wrong position.

Got ml_get errors when using undo with **'virtualedit'**.

Call gui_mch_update() before triggering GuiEnter autocmd. (Ron Aaron)

Unix "make install" installed a few Amiga .info files.

Disallow setting **'ambiwidth'** to "double" when **'listchars'** or **'fillchars'** contains a character that would become double width.

Set **'wrapscan'** when checking the .po files. (Mike Williams)

Fixed: using expression in command line may cause a crash.

Avoid warnings from the clang compiler. (Dominique Pelle)

Fix: Include wchar.h in charset.c for towupper().

Fixed: Using ":read file" in an empty buffer when **'compatible'** is set caused an error. Was caused by patch 7.2.132.

Make the references to features in the help more consistent. (Sylvain Hitier)

VERSION 7.4

[version-7.4](#) [version7.4](#) [vim-7.4](#)

This section is about improvements made between version 7.3 and 7.4.

This release has hundreds of bug fixes and there are a few new features. The most notable new features are:

- New regexp engine [new-regexp-engine](#)
- A more pythonic Python interface [better-python-interface](#)

New regexp engine

[new-regexp-engine](#)

What is now called the "old" regexp engine uses a backtracking algorithm. It tries to match the pattern with the text in one way, and when that fails it goes back and tries another way. This works fine for simple patterns, but complex patterns can be very slow on longer text.

The new engine uses a state machine. It tries all possible alternatives at the current character and stores the possible states of the pattern. This is a bit slower for simple patterns, but much faster for complex patterns and long text.

Most notably, syntax highlighting for Javascript and XML files with long lines is now working fine. Previously Vim could get stuck.

More information here: [two-engines](#)

Better Python interface

[better-python-interface](#)

Added [python-bindeval](#) function. Unlike [python-eval](#) this one returns [python-Dictionary](#), [python-List](#) and [python-Function](#) objects for dictionaries lists and functions respectively in place of their Python built-in equivalents (or None if we are talking about function references).

For simple types this function returns Python built-in types and not only Python ``str()`` like [python-eval](#) does. On Python 3 it will return ``bytes()`` objects in place of ``str()`` ones avoiding possibility of UnicodeDecodeError.

Interface of new objects mimics standard Python ``dict()`` and ``list()`` interfaces to some extent. Extent will be improved in the future.

Added special [python-vars](#) objects also available for [python-buffer](#) and [python-window](#). They ease access to Vim script variables from Python.

Now you no longer need to alter ``sys.path`` to import your module: special hooks are responsible for importing from `{rtp}/python2`, `{rtp}/python3` and `{rtp}/pythonx` directories (for Python 2, Python 3 and both respectively). See [python-special-path](#) .

Added possibility to work with `tabpage` s through `python-tabpage` object.

Added automatic conversion of Vim errors and exceptions to Python exceptions.

Changed the behavior of the `python-buffers` object: it now uses buffer numbers as keys in place of the index of the buffer in the internal buffer list. This should not break anything as the only way to get this index was iterating over `python-buffers` .

Added `:pydo` and `:py3do` commands.

Added the `pyeval()` and `py3eval()` functions.

Now in all places which previously accepted ``str()`` objects, ``str()`` and ``unicode()`` (Python 2) or ``bytes()`` and ``str()`` (Python 3) are accepted.

`python-window` has gained `.col`` and `.row`` attributes that are currently the only way to get internal window positions.

Added or fixed support for ``dir()`` in Vim Python objects.

Changed

[changed-7.4](#)

Old Python versions (≤ 2.2) are no longer supported. Building with them did not work anyway.

Options:

Added ability to automatically save the selection into the system clipboard when using non-GUI version of Vim (autoselectplus in `'clipboard'`). Also added ability to use the system clipboard as default register (previously only primary selection could be used). (Ivan Krasilnikov, Christian Brabandt, Bram Moolenaar)

Added a special `'shiftwidth'` value that makes `'sw'` follow `'tabstop'`. As indenting via `'indentexpr'` became tricky `shiftwidth()` function was added. Also added equivalent special value to `'softtabstop'` option. (Christian Brabandt, so8res)

Show absolute number in number column when `'relativenumber'` option is on. Now there are four combinations with `'number'` and `'relativenumber'`. (Christian Brabandt)

Commands:

`:diffoff` now saves the local values of some settings and restores them in place of blindly resetting them to the defaults. (Christian Brabandt)

Other:

Lua interface now also uses userdata bound to Vim structures. (Taro Muraoka, Luis Carvalho)

`glob()` and autocommand patterns used to work with the undocumented "`{n,m}`" item from a regexp. "`{`" is now used for a literal "`{`", as this is normal in shell file patterns. Now used "`\\{n,m}`" to get "`{n,m}`" in the regexp pattern.

Added

added-7.4

Various syntax, indent and other plugins were added.

Added support for [Lists](#) and [Dictionaries](#) in `viminfo`. (Christian Brabandt)

Functions:

Bitwise functions: `and()`, `or()`, `invert()`, `xor()`.

Added `luaeval()` function. (Taro Muraoka, Luis Carvalho)

Added `sha256()` function. (Tyru, Hirohito Higashi)

Added `wildmenumode()` function. (Christian Brabandt)

Debugging functions: `screenattr()`, `screenchar()`, `screencol()`, `screenrow()`. (Simon Ruderich, Bram Moolenaar)

Added ability to use [Dictionary-function](#)s for `sort()`ing, via optional third argument. (Nikolay Pavlov)

Added special `expand()` argument that expands to the current line number.

Made it possible to force `char2nr()` to always give unicode codepoints regardless of current encoding. (Yasuhiro Matsumoto)

Made it possible for functions generating file list generate [List](#) and not NL-separated string. (e.g. `glob()`, `expand()`) (Christian Brabandt)

Functions that obtain variables from the specific window, tabpage or buffer scope dictionary can now return specified default value in place of empty string in case variable is not found. (`gettabvar()`, `getwinvar()`, `getbufvar()`) (Shougo Matsushita, Hirohito Higashi)

Autocommands:

Added [InsertCharPre](#) event launched before inserting character. (Jakson A. Aquino)

Added [CompleteDone](#) event launched after finishing completion in insert mode. (idea by Florian Klein)

Added `QuitPre` event launched when commands that can either close Vim or only some window(s) are launched.

Added `TextChanged` and `TextChangedI` events launched when text is changed.

Commands:

`:syntime` command useful for debugging.

Made it possible to remove all signs from the current buffer using `:sign-unplace .` (Christian Brabandt)

Added `:language` autocompletion. (Dominique Pelle)

Added more `:command-complete` completion types: `:behave` suboptions, color schemes, compilers, `:cscope` suboptions, files from `'path'`, `:history` suboptions, locale names, `:syntime` suboptions, user names. (Dominique Pelle)

Added `:map-nowait` creating mapping which when having lhs that is the prefix of another mapping's lhs will not allow Vim to wait for user to type more characters to resolve ambiguity, forcing Vim to take the shorter alternative: one with `<nowait>`.

Options:

Made it possible to ignore case when completing: `'wildignorecase'`.

Added ability to delete comment leader when using `J` by ``j`` flag in `'formatoptions'` (`fo-table`). (Lech Lorens)

Added ability to control indentation inside namespaces: `cino-N`. (Konstantin Lepa)

Added ability to control alignment inside ``if`` condition separately from alignment inside function arguments: `cino-k`. (Lech Lorens)

Other:

Improved support for `cmd.exe`. (Ben Fritz, Bram Moolenaar)

Added `v:windowid` variable containing current window number in GUI Vim. (Christian J. Robinson, Lech Lorens)

Added rxvt-unicode and SGR mouse support. (Yiding Jia, Hayaki Saito)

All changes in 7.4

fixed-7.4

Patch 7.3.001

Problem: When editing `"src/main.c"` and `'path'` set to `"/proto"`, `":find e<C-D"` shows `./proto/eval.pro` instead of `eval.pro`.

Solution: Check for path separator when comparing names. (Nazri Ramliy)

Files: `src/misc1.c`

Patch 7.3.002

Problem: ":find" completion doesn't work when halfway an environment variable. (Dominique Pelle)
Solution: Only use in-path completion when expanding file names. (Nazri Ramliy)
Files: src/ex_docmd.c

Patch 7.3.003

Problem: Crash with specific BufWritePost autocmd. (Peter Odding)
Solution: Don't free the quickfix title twice. (Lech Lorens)
Files: src/quickfix.c

Patch 7.3.004

Problem: Crash when using very long regexp. (Peter Odding)
Solution: Reset reg_toolong. (Carlo Teubner)
Files: src/regexp.c

Patch 7.3.005

Problem: Crash when using undotree(). (Christian Brabandt)
Solution: Increase the list reference count. Add a test for undotree() (Lech Lorens)
Files: src/eval.c, src/testdir/Makefile, src/testdir/test61.in

Patch 7.3.006

Problem: Can't build some multibyte code with C89.
Solution: Move code to after declarations. (Joachim Schmitz)
Files: src/mbyte.c, src/spell.c

Patch 7.3.007

Problem: Python code defines global "buffer". Re-implements a grow-array.
Solution: Use a grow-array instead of coding the same functionality. Handle out-of-memory situation properly.
Files: src/if_py_both.h

Patch 7.3.008

Problem: **'cursorbind'** is kept in places where **'scrollbind'** is reset.
Solution: Reset **'cursorbind'**.
Files: src/buffer.c, src/diff.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/if_cscope.c, src/macros.h, src/quickfix.c, src/search.c, src/tag.c, src/window.c

Patch 7.3.009

Problem: Win32: Crash on Windows when using a bad argument for strftime(). (Christian Brabandt)
Solution: Use the bad_param_handler(). (Mike Williams)
Files: src/os_win32.c

Patch 7.3.010

Problem: Mac GUI: Missing break statements.
Solution: Add the break statements. (Dominique Pelle)
Files: src/gui_mac.c

Patch 7.3.011

Problem: X11 clipboard doesn't work in Athena/Motif GUI. First selection after a shell command doesn't work.
Solution: When using the GUI use XtLastTimestampProcessed() instead of changing a property. (partly by Toni Ronkko)
When executing a shell command disown the selection.
Files: src/ui.c, src/os_unix.c

Patch 7.3.012

Problem: Problems building with MingW.
Solution: Adjust the MingW makefiles. (Jon Maken)
Files: src/Make_ming.mak, src/GvimExt/Make_ming.mak

Patch 7.3.013

Problem: Dynamic loading with Ruby doesn't work for 1.9.2.
Solution: Handle rb_str2cstr differently. Also support dynamic loading on Unix. (Jon Maken)
Files: src/if_ruby.c

Patch 7.3.014

Problem: Ending a line in a backslash inside an ":append" or ":insert" command in Ex mode doesn't work properly. (Ray Frush)
Solution: Halve the number of backslashes, only insert a NUL after an odd number of backslashes.
Files: src/ex_getln.c

Patch 7.3.015

Problem: Test is using error message that no longer exists.
Solution: Change E106 to E121. (Dominique Pelle)
Files: src/testdir/test49.vim

Patch 7.3.016

Problem: Netbeans doesn't work under Athena.
Solution: Support Athena, just like Motif. (Xavier de Gaye)
Files: runtime/doc/netbeans.txt, src/gui.c, src/main.c, src/netbeans.c

Patch 7.3.017

Problem: smatch reports errors.
Solution: Fix the reported errors. (Dominique Pelle)
Files: src/spell.c, src/syntax.c

Patch 7.3.018 (after 7.3.012)

Problem: Missing argument to windres in MingW makefiles.
Solution: Add the argument that was wrapped in the patch. (Jon Maken)
Files: src/Make_ming.mak, src/GvimExt/Make_ming.mak

Patch 7.3.019

Problem: ":nstart" can fail silently.
Solution: Give an error when netbeans is not supported by the GUI. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.3.020

Problem: Cursor position wrong when joining multiple lines and **'formatoptions'** contains "a". (Moshe Kamensky)

Solution: Adjust cursor position for skipped indent. (Carlo Teubner)
Files: src/ops.c, src/testdir/test68.in, src/testdir/test68.ok

Patch 7.3.021

Problem: Conflict for defining Boolean in Mac header files.
Solution: Define NO_X11_INCLUDES. (Rainer Muller)
Files: src/os_macosx.m, src/vim.h

Patch 7.3.022

Problem: When opening a new window the '**spellcapcheck**' option is cleared.
Solution: Copy the correct option value. (Christian Brabandt)
Files: src/option.c

Patch 7.3.023

Problem: External program may hang when it tries to write to the tty.
Solution: Don't close the slave tty until after the child exits. (Nikola Knezevic)
Files: src/os_unix.c

Patch 7.3.024

Problem: Named signs do not use a negative number as intended.
Solution: Fix the numbering of named signs. (Xavier de Gaye)
Files: src/ex_cmds.c

Patch 7.3.025

Problem: ":mksession" does not square brackets escape file name properly.
Solution: Improve escaping of file names. (partly by Peter Odding)
Files: src/ex_docmd.c

Patch 7.3.026

Problem: **CTRL-]** in a help file doesn't always work. (Tony Mechelynck)
Solution: Don't escape special characters. (Carlo Teubner)
Files: src/normal.c

Patch 7.3.027

Problem: Opening a file on a network share is very slow.
Solution: When fixing file name case append "*" to directory, server and network share names. (David Anderson, John Beckett)
Files: src/os_win32.c

Patch 7.3.028 (after 7.3.024)

Problem: Signs don't show up. (Charles Campbell)
Solution: Don't use negative numbers. Also assign a number to signs that have a name of all digits to avoid using a sign number twice.
Files: src/ex_cmds.c

Patch 7.3.029

Problem: ":sort n" sorts lines without a number as number zero. (Beeyawned)
Solution: Make lines without a number sort before lines with a number. Also fix sorting negative numbers.
Files: src/ex_cmds.c, src/testdir/test57.in, src/testdir/test57.ok

Patch 7.3.030

Problem: Cannot store Dict and List in viminfo file.

Solution: Add support for this. (Christian Brabandt)
Files: runtime/doc/options.txt, src/eval.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/main.aap, src/testdir/test74.in,
src/testdir/test74.ok

Patch 7.3.031

Problem: Can't pass the X window ID to another application.
Solution: Add v:windowid. (Christian J. Robinson, Lech Lorens)
Files: runtime/doc/eval.txt, src/eval.c, src/gui.c, src/vim.h,
src/os_unix.c

Patch 7.3.032

Problem: maparg() doesn't return the flags, such as <buffer>, <script>,
<silent>. These are needed to save and restore a mapping.
Solution: Improve maparg(). (also by Christian Brabandt)
Files: runtime/doc/eval.txt, src/eval.c, src/getchar.c, src/gui_w48.c,
src/message.c, src/proto/getchar.pro, src/proto/message.pro,
src/structs.h src/testdir/test75.in, src/testdir/test75.ok

Patch 7.3.033 (after 7.3.032)

Problem: Can't build without FEAT_LOCALMAP.
Solution: Add an #ifdef. (John Marriott)
Files: src/getchar.c

Patch 7.3.034

Problem: Win32: may be loading .dll from the wrong directory.
Solution: Go to the Vim executable directory when opening a library.
Files: src/gui_w32.c, src/if_lua.c, src/if_mzsch.c, src/if_perl.xs,
src/if_python.c, src/if_python3.c, src/if_ruby.c, src/mbyte.c,
src/os_mswin.c, src/os_win32.c, src/proto/os_win32.pro

Patch 7.3.035 (after 7.3.034)

Problem: Stray semicolon after if statement. (Hari G)
Solution: Remove the semicolon.
Files: src/os_win32.c

Patch 7.3.036

Problem: Win32 GUI: When building without menus, the font for dialogs and
tab page headers also changes.
Solution: Define USE_SYSMENU_FONT always. (Harig G.)
Files: src/gui_w32.c

Patch 7.3.037

Problem: Compiler warnings for loss of data. (Mike Williams)
Solution: Add type casts.
Files: src/if_py_both.h, src/getchar.c, src/os_win32.c

Patch 7.3.038

Problem: v:windowid isn't set on MS-Windows.
Solution: Set it to the window handle. (Chris Sutcliffe)
Files: runtime/doc/eval.txt, src/gui_w32.c

Patch 7.3.039

Problem: Crash when using skk.vim plugin.
Solution: Get length of expression evaluation result only after checking for NULL. (Noriaki Yagi, Dominique Pelle)
Files: src/ex_getln.c

Patch 7.3.040

Problem: Comparing strings while ignoring case goes beyond end of the string when there are illegal bytes. (Dominique Pelle)
Solution: Explicitly check for illegal bytes.
Files: src/mbyte.c

Patch 7.3.041

Problem: Compiler warning for accessing mediumVersion. (Tony Mechelynck)
Solution: Use the pointer instead of the array itself. (Dominique Pelle)
Files: src/version.c

Patch 7.3.042

Problem: No spell highlighting when re-using an empty buffer.
Solution: Clear the spell checking info only when clearing the options for a buffer. (James Vega)
Files: src/buffer.c

Patch 7.3.043

Problem: Can't load Ruby dynamically on Unix.
Solution: Adjust the configure script. (James Vega)
Files: src/Makefile, src/config.h.in, src/configure.in, src/auto/configure, src/if_ruby.c

Patch 7.3.044

Problem: The preview window opened by the popup menu is larger than specified with '**previewheight**'. (Benjamin Haskell)
Solution: Use '**previewheight**' if it's set and smaller.
Files: src/popupmnu.c

Patch 7.3.045

Problem: Compiler warning for uninitialized variable.
Solution: Initialize the variable always.
Files: src/getchar.c

Patch 7.3.046 (after 7.3.043)

Problem: Can't build Ruby on MS-Windows.
Solution: Add #ifdef, don't use WIN3264 before including vim.h.
Files: src/if_ruby.c

Patch 7.3.047 (after 7.3.032)

Problem: Missing makefile updates for test 75.
Solution: Update the makefiles.
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Makefile, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.048

Problem: ":earlier lf" doesn't work after loading undo file.

Solution: Set `b_u_save_nr_cur` when loading an undo file. (Christian Brabandt)
Fix only showing time in `":undolist"`
Files: `src/undo.c`

Patch 7.3.049

Problem: PLT has rebranded their Scheme to Racket.
Solution: Add support for Racket 5.x. (Sergey Khorev)
Files: `src/Make_cyg.mak`, `src/Make_ming.mak`, `src/Make_mvc.mak`,
`src/auto/configure`, `src/configure.in`, `src/if_mzsch.c`

Patch 7.3.050

Problem: The link script is clumsy.
Solution: Use the `--as-needed` linker option if available. (Kirill A. Shutemov)
Files: `src/Makefile`, `src/auto/configure`, `src/config.mk.in`,
`src/configure.in`, `src/link.sh`

Patch 7.3.051

Problem: Crash when `$PATH` is empty.
Solution: Check for `vim_getenv()` returning NULL. (Yasuhiro Matsumoto)
Files: `src/ex_getln.c`, `src/os_win32.c`

Patch 7.3.052

Problem: When `'completefunc'` opens a new window all kinds of errors follow. (Xavier Deguillard)
Solution: When `'completefunc'` goes to another window or buffer and when it deletes text abort completion. Add a test for `'completefunc'`.
Files: `src/edit.c`, `src/testdir/Make_amiga.mak`, `src/testdir/Make_dos.mak`,
`src/testdir/Make_ming.mak`, `src/testdir/Make_os2.mak`,
`src/testdir/Make_vms.mms`, `src/testdir/Makefile`,
`src/testdir/test76.in`, `src/testdir/test76.ok`

Patch 7.3.053

Problem: `complete()` function doesn't reset complete direction. Can't use an empty string in the list of matches.
Solution: Set `compl_direction` to FORWARD. Add "empty" key to allow empty words. (Kikuchan)
Files: `src/edit.c`

Patch 7.3.054

Problem: Can define a user command for `:Print`, but it doesn't work. (Aaron Thoma)
Solution: Let user command `:Print` overrule the builtin command (Christian Brabandt) Disallow `:X` and `:Next` as a user defined command.
Files: `src/ex_docmd.c`

Patch 7.3.055

Problem: Recursively nested lists and dictionaries cause a near-endless loop when comparing them with a copy. (ZyX)
Solution: Limit recursiveness in a way that non-recursive structures can still be nested very deep.
Files: `src/eval.c`, `src/testdir/test55.in`, `src/testdir/test55.ok`

Patch 7.3.056

Problem: "getline" argument in do_cmdline() shadows global.
Solution: Rename the argument.
Files: src/ex_docmd.c

Patch 7.3.057

Problem: Segfault with command line abbreviation. (Randy Morris)
Solution: Don't retrigger the abbreviation when abandoning the command line.
Continue editing the command line after the error.
Files: src/ex_getln.c

Patch 7.3.058

Problem: Error "code converter not found" when loading Ruby script.
Solution: Load Gem module. (Yasuhiro Matsumoto)
Files: src/if_ruby.c

Patch 7.3.059

Problem: Netbeans: Problem with recursively handling messages for Athena
and Motif.
Solution: Call netbeans_parse_messages() in the main loop, like it's done
for GTK. (Xavier de Gaye)
Files: src/gui_x11.c, src/netbeans.c

Patch 7.3.060

Problem: Netbeans: crash when socket is disconnected unexpectedly.
Solution: Don't cleanup when a read fails, put a message in the queue and
disconnect later. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.3.061

Problem: Remote ":drop" does not respect 'autochdir'. (Peter Odding)
Solution: Don't restore the directory when 'autochdir' is set. (Benjamin
Fritz)
Files: src/main.c

Patch 7.3.062

Problem: Python doesn't work properly when installed in another directory
than expected.
Solution: Figure out home directory in configure and use Py_SetPythonHome()
at runtime. (Roland Puntaier)
Files: src/configure.in, src/auto/configure, src/if_python.c,
src/if_python3.c

Patch 7.3.063

Problem: Win32: Running a filter command makes Vim lose focus.
Solution: Use SW_SHOWMINNOACTIVE instead of SW_SHOWMINIMIZED. (Hong Xu)
Files: src/os_win32.c

Patch 7.3.064

Problem: Win32: ":dis +" shows nothing, but "+p does insert text.
Solution: Display the * register, since that's what will be inserted.
(Christian Brabandt)
Files: src/globals.h, src/ops.c

Patch 7.3.065

Problem: Can't get current line number in a source file.
Solution: Add the <slnum> item, similar to <sfile>.
Files: src/ex_docmd.c

Patch 7.3.066

Problem: Crash when changing to another window while in a :vimgrep command.
(Christian Brabandt)
Solution: When wiping out the dummy before, remove it from aucmd_win.
Files: src/quickfix.c

Patch 7.3.067 (after 7.3.058)

Problem: Ruby: Init_prelude is not always available.
Solution: Remove use of Init_prelude. (Yasuhiro Matsumoto)
Files: src/if_ruby.c

Patch 7.3.068

Problem: Using freed memory when doing ":saveas" and an autocommand sets
'autochdir'. (Kevin Klement)
Solution: Get the value of fname again after executing autocommands.
Files: src/ex_cmds.c

Patch 7.3.069

Problem: GTK: pressing Enter in inputdialog() doesn't work like clicking OK
as documented.
Solution: call gtk_entry_set_activates_default(). (Britton Kerin)
Files: src/gui_gtk.c

Patch 7.3.070

Problem: Can set environment variables in the sandbox, could be abused.
Solution: Disallow it.
Files: src/eval.c

Patch 7.3.071

Problem: Editing a file in a window that's in diff mode resets 'diff'
but not cursor binding.
Solution: Reset cursor binding in two more places.
Files: src/quickfix.c, src/option.c

Patch 7.3.072

Problem: Can't complete file names while ignoring case.
Solution: Add 'wildignorecase'.
Files: src/ex_docmd.c, src/ex_getln.c, src/misc1.c, src/option.c,
src/option.h, src/vim.h, src/runtime/options.txt

Patch 7.3.073

Problem: Double free memory when netbeans command follows DETACH.
Solution: Only free the node when owned. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.3.074

Problem: Can't use the "+ register like "*" for yank and put.
Solution: Add "unnamedplus" to the 'clipboard' option. (Ivan Krasilnikov)
Files: runtime/doc/options.txt, src/eval.c, src/globals.h, src/ops.c,

src/option.c

Patch 7.3.075 (after 7.3.072)

Problem: Missing part of **'wildignorecase'**

Solution: Also adjust expand()

Files: src/eval.c

Patch 7.3.076

Problem: Clang warnings for dead code.

Solution: Remove it. (Carlo Teubner)

Files: src/gui_gtk.c, src/if_ruby.c, src/misc2.c, src/netbeans.c,
src/spell.c

Patch 7.3.077

Problem: When updating crypt of swapfile fails there is no error message.
(Carlo Teubner)

Solution: Add the error message.

Files: src/memline.c

Patch 7.3.078

Problem: Warning for unused variable.

Solution: Adjust #ifdefs.

Files: src/ops.c

Patch 7.3.079

Problem: Duplicate lines in makefile.

Solution: Remove the lines. (Hong Xu)

Files: src/Make_mvc.mak

Patch 7.3.080

Problem: Spell doesn't work on VMS.

Solution: Use different file names. (Zoltan Bartos, Zoltan Arpadffy)

Files: src/spell.c

Patch 7.3.081

Problem: Non-printable characters in **'statusline'** cause trouble. (ZyX)

Solution: Use transstr(). (partly by Caio Ariede)

Files: src/screen.c

Patch 7.3.082

Problem: Leaking file descriptor when hostname doesn't exist.

Solution: Remove old debugging lines.

Files: src/netbeans.c

Patch 7.3.083

Problem: When a read() or write() is interrupted by a signal it fails.

Solution: Add read_eintr() and write_eintr().

Files: src/fileio.c, src/proto/fileio.pro, src/memfile.c, src/memline.c,
src/os_unix.c, src/undo.c, src/vim.h

Patch 7.3.084

Problem: When splitting the window, the new one scrolls with the cursor at
the top.

Solution: Compute w_fraction before setting the new height.

Files: src/window.c

Patch 7.3.085 (after 7.3.083)

Problem: Inconsistency with preproc symbols. void * computation.

Solution: Include vimio.h from vim.h. Add type cast.

Files: src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/fileio.c,
src/if_cscope.c, src/if_sniff.c, src/main.c, src/memfile.c,
src/memline.c, src/netbeans.c, src/os_msdos.c, src/os_mswin.c,
src/os_win16.c, src/os_win32.c, src/spell.c, src/tag.c,
src/undo.c, src/vim.h

Patch 7.3.086

Problem: When using a mapping with an expression and there was no count, v:count has the value of the previous command. (ZyX)

Solution: Also set v:count and v:count1 before getting the character that could be a command or a count.

Files: src/normal.c

Patch 7.3.087

Problem: EINTR is not always defined.

Solution: Include errno.h in vim.h.

Files: src/if_cscope.c, src/if_tcl.c, src/integration.c, src/memline.c,
src/os_mswin.c, src/os_win16.c, src/os_win32.c, src/vim.h,
src/workshop.c

Patch 7.3.088

Problem: Ruby can't load Gems sometimes, may cause a crash.

Solution: Undefine off_t. Use ruby_process_options(). (Yasuhiro Matsumoto)

Files: src/if_ruby.c

Patch 7.3.089

Problem: Compiler warning on 64 bit MS-Windows.

Solution: Add type cast. (Mike Williams)

Files: src/netbeans.c

Patch 7.3.090

Problem: Wrong help text for Cscope.

Solution: Adjust the help text for "t". (Dominique Pelle)

Files: src/if_cscope.c

Patch 7.3.091

Problem: "vim -w foo" writes special key codes for removed escape sequences. (Josh Triplett)

Solution: Don't write K_IGNORE codes.

Files: src/getchar.c, src/misc1.c, src/term.c, src/vim.h

Patch 7.3.092

Problem: Resizing the window when exiting.

Solution: Don't resize when exiting.

Files: src/term.c

Patch 7.3.093

Problem: New DLL dependencies in MingW with gcc 4.5.0.

Solution: Add STATIC_STDCPLUS, LDFLAGS and split up WINDRES. (Guopeng Wen)

Files: src/GvimExt/Make_ming.mak, src/Make_ming.mak

Patch 7.3.094

Problem: Using abs() requires type cast to int.
Solution: Use labs() so that the value remains long. (Hong Xu)
Files: src/screen.c

Patch 7.3.095

Problem: Win32: In Chinese tear-off menu doesn't work. (Weasley)
Solution: Use menu_name_equal(). (Alex Jakushev)
Files: src/menu.c

Patch 7.3.096

Problem: "gvim -nb" is not interruptible. Leaking file descriptor on netbeans connection error.
Solution: Check for **CTRL-C** typed. Free file descriptor. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.3.097

Problem: Using ":call" inside "if 0" does not see that a function returns a Dict and gives error for "." as string concatenation.
Solution: Use eval0() to skip over the expression. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.098

Problem: Function that ignores error still causes called_emsg to be set. E.g. when expand() fails the status line is disabled.
Solution: Move check for emsg_not_now() up. (James Vega)
Files: src/message.c

Patch 7.3.099

Problem: Crash when splitting a window with zero height. (Yukihiro Nakadaira)
Solution: Don't set the fraction in a window with zero height.
Files: src/window.c

Patch 7.3.100

Problem: When using :normal v:count isn't set.
Solution: Call normal_cmd() with toplevel set to TRUE.
Files: src/ex_docmd.c

Patch 7.3.101

Problem: ino_t defined with wrong size.
Solution: Move including auto/config.h before other includes. (Marius Geminas)
Files: src/if_ruby.c, src/if_lua.c

Patch 7.3.102

Problem: When using ":make", typing the next command and then getting the "reload" prompt the next command is (partly) eaten by the reload prompt.
Solution: Accept ':' as a special character at the reload prompt to accept the default choice and execute the command.
Files: src/eval.c, src/fileio.c, src/gui.c, src/gui_xmdlg.c,

src/memline.c, src/message.c, src/proto/message.pro,
src/gui_athena.c, src/gui_gtk.c, src/gui_mac.c, src/gui_motif.c,
src/gui_photon.c, src/gui_w16.c, src/gui_w32.c, src/os_mswin.c
src/proto/gui_athena.pro, src/proto/gui_gtk.pro,
src/proto/gui_mac.pro, src/proto/gui_motif.pro,
src/proto/gui_photon.pro, src/proto/gui_w16.pro,
src/proto/gui_w32.pro

Patch 7.3.103

Problem: Changing **'fileformat'** and then using `":w"` in an empty file sets the **'modified'** option.

Solution: In `unchanged()` don't ignore **'ff'** for an empty file.

Files: src/misc1.c, src/option.c, src/proto/option.pro, src/undo.c

Patch 7.3.104

Problem: Conceal: using Tab for cchar causes problems. (ZyX)

Solution: Do not accept a control character for cchar.

Files: src/syntax.c

Patch 7.3.105

Problem: Can't get the value of `"b:changedtick"` with `getbufvar()`.

Solution: Make it work. (Christian Brabandt)

Files: src/eval.c

Patch 7.3.106

Problem: When **'cursorbind'** is set another window may scroll unexpectedly when **'scrollbind'** is also set. (Xavier Wang)

Solution: Don't call `update_topleftine()` if **'scrollbind'** is set.

Files: src/move.c

Patch 7.3.107

Problem: Year number for `:undolist` can be confused with month or day.

Solution: Change `"%y"` to `"%Y"`.

Files: src/undo.c

Patch 7.3.108

Problem: Useless check for NULL when calling `vim_free()`.

Solution: Remove the check. (Dominique Pelle)

Files: src/eval.c, src/ex_cmds.c, src/os_win32.c

Patch 7.3.109

Problem: Processing new Esperanto spell file fails and crashes Vim. (Dominique Pelle)

Solution: When running out of memory give an error. Handle `'?'` in `COMPOUNDRULE` properly.

Files: src/spell.c

Patch 7.3.110

Problem: The `"nbsp"` item in **'listchars'** isn't used for `":list"`.

Solution: Make it work. (Christian Brabandt)

Files: src/message.c

Patch 7.3.111 (after 7.3.100)

Problem: Executing a `:normal` command in **'statusline'** evaluation causes the

cursor to move. (Dominique Pelle)
Solution: When updating the cursor for **'cursorbind'** allow the cursor beyond the end of the line. When evaluating **'statusline'** temporarily reset **'cursorbind'**.

Files: src/move.c, src/screen.c

Patch 7.3.112

Problem: Setting **'statusline'** to "%!'asdf%" reads uninitialized memory.

Solution: Check for NUL after %.

Files: src/buffer.c

Patch 7.3.113

Problem: Windows: Fall back directory for creating temp file is wrong.

Solution: Use "." instead of empty string. (Hong Xu)

Files: src/fileio.c

Patch 7.3.114

Problem: Potential problem in initialization when giving an error message early.

Solution: Initialize **'verbosefile'** empty. (Ben Schmidt)

Files: src/option.h

Patch 7.3.115

Problem: Vim can crash when tmpnam() returns NULL.

Solution: Check for NULL. (Hong Xu)

Files: src/fileio.c

Patch 7.3.116

Problem: **'cursorline'** is displayed too short when there are concealed characters and **'list'** is set. (Dennis Preiser)

Solution: Check for **'cursorline'** when **'list'** is set. (Christian Brabandt)

Files: src/screen.c

Patch 7.3.117

Problem: On some systems --as-needed does not work, because the "tinfo" library is included indirectly from "ncurses". (Charles Campbell)

Solution: In configure prefer using "tinfo" instead of "ncurses".

Files: src/configure.in, src/auto/configure

Patch 7.3.118

Problem: Ruby uses SIGVTALARM which makes Vim exit. (Alec Tica)

Solution: Ignore SIGVTALARM. (Dominique Pelle)

Files: src/os_unix.c

Patch 7.3.119

Problem: Build problem on Mac. (Nicholas Stallard)

Solution: Use "extern" instead of "EXTERN" for p_vfile.

Files: src/option.h

Patch 7.3.120

Problem: The message for an existing swap file is too long to fit in a 25 line terminal.

Solution: Make the message shorter. (Chad Miller)

Files: src/memline.c

Patch 7.3.121

Problem: Complicated **'statusline'** causes a crash. (Christian Brabandt)
Solution: Check that the number of items is not too big.
Files: src/buffer.c

Patch 7.3.122

Problem: Having auto/config.mk in the repository causes problems.
Solution: Remove auto/config.mk from the distribution. In the toplevel Makefile copy it from the "dist" file.
Files: Makefile, src/Makefile, src/auto/config.mk

Patch 7.3.123

Problem: ml_get error when executing register being recorded into, deleting lines and **'conceallevel'** is set. (ZyX)
Solution: Don't redraw a line for concealing when it doesn't exist.
Files: src/main.c

Patch 7.3.124

Problem: When writing a file in binary mode it may be missing the final EOL if a file previously read was missing the EOL. (Kevin Goodsell)
Solution: Move the write_no_eol_lnum into the buffer struct.
Files: src/structs.h, src/fileio.c, src/globals.h, src/os_unix.c

Patch 7.3.125

Problem: MSVC: Problem with quotes in link argument.
Solution: Escape backslashes and quotes. (Weasley)
Files: src/Make_mvc.mak

Patch 7.3.126

Problem: Compiler warning for signed pointer.
Solution: Use unsigned int argument for sscanf().
Files: src/blowfish.c

Patch 7.3.127

Problem: Compiler complains about comma.
Solution: Remove comma after last enum element.
Files: src/ex_cmds2.c

Patch 7.3.128

Problem: Another compiler warning for signed pointer.
Solution: Use unsigned int argument for sscanf().
Files: src/mark.c

Patch 7.3.129

Problem: Using integer like a boolean.
Solution: Nicer check for integer being non-zero.
Files: src/tag.c

Patch 7.3.130

Problem: Variable misplaced in #ifdef.
Solution: Move clipboard_event_time outside of #ifdef.
Files: src/gui_gtk_x11.c

Patch 7.3.131

Problem: Including errno.h too often.
Solution: Don't include errno.h in Unix header file.
Files: src/os_unix.h

Patch 7.3.132

Problem: C++ style comments.
Solution: Change to C comments.
Files: src/if_python3.c

Patch 7.3.133

Problem: When using encryption it's not clear what method was used.
Solution: In the file message show "blowfish" when using blowfish.
Files: src/fileio.c

Patch 7.3.134

Problem: Drag-n-drop doesn't work in KDE Dolphin.
Solution: Add GDK_ACTION_MOVE flag. (Florian Degner)
Files: src/gui_gtk_x11.c

Patch 7.3.135

Problem: When there is no previous substitute pattern, the previous search pattern is used. The other way around doesn't work.
Solution: When there is no previous search pattern, use the previous substitute pattern if possible. (Christian Brabandt)
Files: src/search.c

Patch 7.3.136

Problem: Duplicate include of assert.h.
Solution: Remove it.
Files: src/if_cscope.c

Patch 7.3.137 (after 7.3.091)

Problem: When '**lazyredraw**' is set the screen may not be updated. (Ivan Krasilnikov)
Solution: Call update_screen() before waiting for input.
Files: src/misc1.c, src/getchar.c

Patch 7.3.138

Problem: ":com" changes the multibyte text of :echo. (Dimitar Dimitrov)
Solution: Search for K_SPECIAL as a byte, not a character. (Ben Schmidt)
Files: src/ex_docmd.c

Patch 7.3.139 (after 7.3.137)

Problem: When '**lazyredraw**' is set ":ver" output can't be read.
Solution: Don't redraw the screen when at a prompt or command line.
Files: src/getchar.c, src/message.c, src/misc1.c

Patch 7.3.140

Problem: Crash when drawing the "\$" at end-of-line for list mode just after the window border and '**cursorline**' is set.
Solution: Don't check for '**cursorline**'. (Quentin Carbonneaux)
Files: src/screen.c

Patch 7.3.141

Problem: When a key code is not set get a confusing error message.
Solution: Change the error message to say the key code is not set.
Files: src/option.c, runtime/doc/options.txt

Patch 7.3.142

Problem: Python stdout doesn't have a flush() method, causing an import to fail.
Solution: Add a dummy flush() method. (Tobias Columbus)
Files: src/if_py_both.h

Patch 7.3.143

Problem: Memfile is not tested sufficiently. Looking up blocks in a memfile is slow when there are many blocks.
Solution: Add high level test and unittest. Adjust the number of hash buckets to the number of blocks. (Ivan Krasilnikov)
Files: Filelist, src/Makefile, src/main.c, src/memfile.c, src/memfile_test.c src/structs.h src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mak, src/testdir/Makefile, src/testdir/test77.in, src/testdir/test77.ok

Patch 7.3.144

Problem: Crash with ":python help(dir)". (Kearn Holliday)
Solution: Fix the way the type is set on objects. (Tobias Columbus)
Files: src/if_python.c

Patch 7.3.145 (after 7.3.144)

Problem: Can't build with Python dynamically loading.
Solution: Add dll_PyType_Ready.
Files: src/if_python.c

Patch 7.3.146

Problem: It's possible to assign to a read-only member of a dict.
It's possible to create a global variable "0". (ZyX)
It's possible to add a v: variable with ":let v:.name = 1".
Solution: Add check for dict item being read-only.
Check the name of g: variables.
Disallow adding v: variables.
Files: src/eval.c

Patch 7.3.147 (after 7.3.143)

Problem: Can't build on HP-UX.
Solution: Remove an unnecessary backslash. (John Marriott)
Files: src/Makefile

Patch 7.3.148

Problem: A syntax file with a huge number of items or clusters causes weird behavior, a hang or a crash. (Yukihiro Nakadaira)
Solution: Check running out of IDs. (partly by Ben Schmidt)
Files: src/syntax.c

Patch 7.3.149

Problem: The cursor disappears after the processing of the 'setDot'

netbeans command when vim runs in a terminal.
Solution: Show the cursor after a screen update. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.3.150

Problem: readline() does not return the last line when the NL is missing.
(Hong Xu)
Solution: When at the end of the file Also check for a previous line.
Files: src/eval.c

Patch 7.3.151 (after 7.3.074)

Problem: When "unnamedplus" is in '**clipboard**' the selection is sometimes
also copied to the star register.
Solution: Avoid copy to the star register when undesired. (James Vega)
Files: src/ops.c

Patch 7.3.152

Problem: Xxd does not check for errors from library functions.
Solution: Add error checks. (Florian Zumbiehl)
Files: src/xxd/xxd.c

Patch 7.3.153 (after 7.3.152)

Problem: Compiler warning for ambiguous else, missing prototype.
Solution: Add braces. (Dominique Pelle) Add prototype for die().
Files: src/xxd/xxd.c

Patch 7.3.154 (after 7.3.148)

Problem: Can't compile with tiny features. (Tony Mechelynck)
Solution: Move #define outside of #ifdef.
Files: src/syntax.c

Patch 7.3.155

Problem: Crash when using map(), filter() and remove() on v:. (ZyX)
Also for extend(). (Yukihiro Nakadaira)
Solution: Mark v: as locked. Also correct locking error messages.
Files: src/eval.c

Patch 7.3.156

Problem: Tty names possibly left unterminated.
Solution: Use vim_strncpy() instead of strncpy().
Files: src/pty.c

Patch 7.3.157

Problem: Superfluous assignment.
Solution: Remove assignment.
Files: src/misc1.c

Patch 7.3.158

Problem: Might use uninitialized memory in C indenting.
Solution: Init arrays to empty.
Files: src/misc1.c

Patch 7.3.159

Problem: Using uninitialized pointer when out of memory.

Solution: Check for NULL return value.
Files: src/mbyte.c

Patch 7.3.160

Problem: Unsafe string copying.
Solution: Use vim_strncpy() instead of strcpy(). Use vim_strcat() instead of strcat().
Files: src/buffer.c, src/ex_docmd.c, src/hardcopy.c, src/menu.c, src/misc1.c, src/misc2.c, src/proto/misc2.pro, src/netbeans.c, src/os_unix.c, src/spell.c, src/syntax.c, src/tag.c

Patch 7.3.161

Problem: Items on the stack may be too big.
Solution: Make items static or allocate them.
Files: src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/fileio.c, src/hardcopy.c, src/quickfix.c, src/main.c, src/netbeans.c, src/spell.c, src/tag.c, src/vim.h, src/xxd/xxd.c

Patch 7.3.162

Problem: No error message when assigning to a list with an index out of range. (Yukihiro Nakadaira)
Solution: Add the error message.
Files: src/eval.c

Patch 7.3.163

Problem: For the default of 'shellpipe' "mksh" and "pdksh" are not recognized.
Solution: Recognize these shell names.
Files: src/option.c

Patch 7.3.164

Problem: C-indenting: a preprocessor statement confuses detection of a function declaration.
Solution: Ignore preprocessor lines. (Lech Lorens) Also recognize the style to put a comma before the argument name.
Files: src/misc1.c, testdir/test3.in, testdir/test3.ok

Patch 7.3.165

Problem: ":find" completion does not escape spaces in a directory name. (Isz)
Solution: Add backslashes for EXPAND_FILES_IN_PATH. (Carlo Teubner)
Files: src/ex_getln.c

Patch 7.3.166

Problem: Buffer on the stack may be too big
Solution: Allocate the space.
Files: src/option.c

Patch 7.3.167

Problem: When using the internal grep QuickFixCmdPost is not triggered. (Yukihiro Nakadaira)
Solution: Change the place where autocommands are triggered.
Files: src/quickfix.c

Patch 7.3.168

Problem: When the second argument of input() contains a CR the text up to that is used without asking the user. (Yasuhiro Matsumoto)
Solution: Change CR, NL and ESC in the text to a space.
Files: src/getchar.c

Patch 7.3.169

Problem: Freeing memory already freed, warning from static code analyzer.
Solution: Initialize pointers to NULL, correct use of "mustfree". (partly by Dominique Pelle)
Files: src/mis1.c

Patch 7.3.170

Problem: VMS Makefile for testing was not updated for test77.
Solution: Add test77 to the Makefile.
Files: src/testdir/Make_vms.mms

Patch 7.3.171

Problem: When the clipboard isn't supported: ":yank*" gives a confusing error message.
Solution: Specifically mention that the register name is invalid. (Jean-Rene David)
Files: runtime/doc/change.txt, src/ex_docmd.c, src/globals.h

Patch 7.3.172

Problem: MS-Windows: rename() might delete the file if the name differs but it's actually the same file.
Solution: Use the file handle to check if it's the same file. (Yukihiro Nakadaira)
Files: src/if_cscope.c, src/fileio.c, src/os_win32.c, src/proto/os_win32.pro, src/vim.h

Patch 7.3.173

Problem: After using setqflist() to make the quickfix list empty ":cwindow" may open the window anyway. Also after ":vimgrep".
Solution: Correctly check whether the list is empty. (Ingo Karkat)
Files: src/quickfix.c

Patch 7.3.174

Problem: When Exuberant ctags binary is exctags it's not found.
Solution: Add configure check for exctags. (Hong Xu)
Files: src/configure.in, src/auto/configure

Patch 7.3.175

Problem: When '**colorcolumn**' is set locally to a window, ":new" opens a window with the same highlighting but '**colorcolumn**' is empty. (Tyru)
Solution: Call check_colorcolumn() after clearing and copying options. (Christian Brabandt)
Files: src/buffer.c

Patch 7.3.176

Problem: Ruby linking doesn't work properly on Mac OS X.
Solution: Fix the configure check for Ruby. (Bjorn Winckler)

Files: src/configure.in, src/auto/configure

Patch 7.3.177

Problem: MS-Windows: mkdir() doesn't work properly when **'encoding'** is "utf-8".

Solution: Convert to utf-16. (Yukihiro Nakadaira)

Files: src/os_win32.c, src/os_win32.h, src/proto/os_win32.pro

Patch 7.3.178

Problem: C-indent doesn't handle code right after { correctly.

Solution: Fix detecting unterminated line. (Lech Lorens)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.179

Problem: C-indent doesn't handle colon in string correctly.

Solution: Skip the string. (Lech Lorens)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.180

Problem: When both a middle part of **'comments'** matches and an end part, the middle part was used erroneously.

Solution: After finding the middle part match continue looking for a better end part match. (partly by Lech Lorens)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.181

Problem: When repeating the insert of **CTRL-V** or a digraph the display may not be updated correctly.

Solution: Only call edit_unputchar() after edit_putchar(). (Lech Lorens)

Files: src/edit.c

Patch 7.3.182 (after 7.3.180)

Problem: Compiler warning for uninitialized variable.

Solution: Add dummy initializer.

Files: src/misc1.c

Patch 7.3.183 (after 7.3.174)

Problem: When Exuberant ctags binary is exuberant-ctags it's not found.

Solution: Add configure check for exuberant-ctags.

Files: src/configure.in, src/auto/configure

Patch 7.3.184

Problem: Static code analysis errors in riscOS.

Solution: Make buffer size bigger. (Dominique Pelle)

Files: src/gui_riscos.c

Patch 7.3.185

Problem: ":windo g/pattern/q" closes windows and reports "N more lines". (Tim Chase)

Solution: Remember what buffer ":global" started in. (Jean-Rene David)

Files: src/ex_cmds.c

Patch 7.3.186

Problem: When **'clipboard'** contains "unnamed" or "unnamedplus" the value of

`v:register` is wrong for operators without a specific register.
Solution: Adjust the register according to '`clipboard`'. (Ingo Karkat)
Files: `src/normal.c`

Patch 7.3.187

Problem: The RISC OS port has obvious errors and is not being maintained.
Solution: Remove the RISC OS files and code.
Files: `src/ascii.h`, `src/eval.c`, `src/ex_cmds.c`, `src/ex_cmds2.c`,
`src/ex_docmd.c`, `src/fileio.c`, `src/globals.h`, `src/gui.c`, `src/gui.h`,
`src/main.c`, `src/memfile.c`, `src/memline.c`, `src/misc1.c`,
`src/proto.h`, `src/quickfix.c`, `src/search.c`, `src/structs.h`,
`src/term.c`, `src/termlib.c`, `src/version.c`, `src/vim.h`,
`src/gui_riscos.h`, `src/os_riscos.h`, `src/gui_riscos.c`,
`src/os_riscos.c`, `runtime/doc/os_risc.txt`

Patch 7.3.188

Problem: More RISC OS files to remove.
Solution: Remove them. Update the file list.
Files: `src/proto/gui_riscos.pro`, `src/proto/os_riscos.pro`, `Filelist`

Patch 7.3.189 (after 7.3.186)

Problem: Can't build without `+clipboard` feature. (Christian Ebert)
Solution: Add the missing `#ifdef`.
Files: `src/normal.c`

Patch 7.3.190

Problem: When there is a `"containedin"` syntax argument highlighting may be wrong. (Radek)
Solution: Reset `current_next_list`. (Ben Schmidt)
Files: `src/syntax.c`

Patch 7.3.191

Problem: Still some RISC OS stuff to remove.
Solution: Remove files and lines. (Hong Xu)
Remove the '`osfiletype`' option code.
Files: `README_extra.txt`, `src/Make_ro.mak`, `src/INSTALL`, `src/Makefile`,
`src/buffer.c`, `src/eval.c`, `src/feature.h`, `src/option.c`,
`src/option.h`, `src/structs.h`, `src/version.c`, `src/pty.c`, `Filelist`

Patch 7.3.192

Problem: Ex command `":s/ \?/ /g"` splits multibyte characters into bytes. (Dominique Pelle)
Solution: Advance over whole character instead of one byte.
Files: `src/ex_cmds.c`

Patch 7.3.193

Problem: In the command line window `":close"` doesn't work properly. (Tony Mechelynck)
Solution: Use `Ctrl_C` instead of `K_IGNORE` for `cmdwin_result`. (Jean-Rene David)
Files: `src/ex_docmd.c`, `src/ex_getln.c`

Patch 7.3.194

Problem: When `"b"` is a symlink to directory `"a"`, `resolve("b/")` doesn't

result in "a/". (ZyX)
Solution: Remove the trailing slash. (Jean-Rene David)
Files: src/eval.c

Patch 7.3.195

Problem: "}" else" causes following lines to be indented too much. (Rouben Rostamian)
Solution: Better detection for the "else". (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.196

Problem: Can't intercept a character that is going to be inserted.
Solution: Add the InsertCharPre autocommand event. (Jakson A. Aquino)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, runtime/doc/map.txt, src/edit.c, src/eval.c, src/fileio.c, src/vim.h

Patch 7.3.197

Problem: When a QuickfixCmdPost event removes all errors, Vim still tries to jump to the first error, resulting in E42.
Solution: Get the number of error after the autocmd event. (Mike Lundy)
Files: src/quickfix.c

Patch 7.3.198

Problem: No completion for ":lang".
Solution: Get locales to complete from. (Dominique Pelle)
Files: src/eval.c, src/ex_cmds2.c, src/ex_getln.c, src/proto/ex_cmds2.pro, src/proto/ex_getln.pro, src/vim.h

Patch 7.3.199

Problem: MS-Windows: Compilation problem of OLE with MingW compiler.
Solution: Put #ifdef around declarations. (Guopeng Wen)
Files: src/if_ole.h

Patch 7.3.200 (after 7.3.198)

Problem: **CTRL-D** doesn't complete :lang.
Solution: Add the missing part of the change. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.3.201 (after 7.3.195)

Problem: "}" else" still causes following lines to be indented too much.
Solution: Better detection for the "else" block. (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.202

Problem: Cannot influence the indent inside a namespace.
Solution: Add the "N" '**cin**' parameter. (Konstantin Lepa)
Files: runtime/doc/indent.txt, src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.203

Problem: MS-Windows: Can't run an external command without a console window.
Solution: Support "!:start /b cmd". (Xaizek)
Files: runtime/doc/os_win32.txt, src/os_win32.c

Patch 7.3.204 (after 7.3.201)

Problem: Compiler warning.
Solution: Add type cast. (Mike Williams)
Files: src/misc1.c

Patch 7.3.205

Problem: Syntax "extend" doesn't work correctly.
Solution: Avoid calling check_state_ends() recursively (Ben Schmidt)
Files: src/syntax.c

Patch 7.3.206

Problem: 64bit MS-Windows compiler warning.
Solution: Use HandleToLong() instead of type cast. (Mike Williams)
Files: src/gui_w32.c

Patch 7.3.207

Problem: Can't compile with MSVC with pentium4 and 64 bit.
Solution: Only use SSE2 for 32 bit. (Mike Williams)
Files: src/Make_mvc.mak

Patch 7.3.208

Problem: Early terminated if statement.
Solution: Remove the semicolon. (Lech Lorens)
Files: src/gui_mac.c

Patch 7.3.209

Problem: MSVC Install instructions point to wrong batch file.
Solution: Add a batch file for use with MSVC 10.
Files: src/msvc2010.bat, src/INSTALLpc.txt, Filelist

Patch 7.3.210

Problem: Can't always find the file when using cscope.
Solution: Add the '**cscoperelative**' option. (Raghavendra D Prabhu)
Files: runtime/doc/if_cscop.txt, runtime/doc/options.txt, src/if_cscope.c

Patch 7.3.211 (after 7.3.210)

Problem: Compiler warning.
Solution: Add type cast.
Files: src/if_cscope.c

Patch 7.3.212

Problem: With Python 3.2 ":py3" fails.
Solution: Move PyEval_InitThreads() to after Py_Initialize(). (Roland Punturier) Check abiflags in configure. (Andreas Behr)
Files: src/if_python3.c, src/auto/configure, src/configure.in

Patch 7.3.213

Problem: Javascript object literal is not indented correctly.
Solution: Make a special case for when "J1" is in '**cin**'. (Luc Deschenaux)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.214

Problem: The text displayed by ":z-" isn't exactly like old Vi.
Solution: Add one to the start line number. (ChangZhuo Chen)
Files: src/ex_cmds.c

Patch 7.3.215 (after 7.3.210)

Problem: Wrong file names in previous patch. (Toothpik)
Solution: Include the option changes.
Files: src/option.c, src/option.h

Patch 7.3.216

Problem: When recovering a file a range of lines is missing. (Charles Jie)
Solution: Reset the index when advancing to the next pointer block. Add a test to verify recovery works.
Files: src/memline.c, src/testdir/test78.in, src/testdir/test78.ok, src/testdir/Makefile, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.217

Problem: Inside an "if" a ":wincmd" causes problems.
Solution: When skipping commands let ":wincmd" skip over its argument.
Files: src/ex_docmd.c

Patch 7.3.218 (after 7.3.212)

Problem: Tiny configuration problem with Python 3.
Solution: Add abiflags in one more place. (Andreas Behr)
Files: src/auto/configure, src/configure.in

Patch 7.3.219

Problem: Can't compile with GTK on Mac.
Solution: Add some #ifdef trickery. (Ben Schmidt)
Files: src/os_mac_conv.c, src/os_macosx.m, src/vim.h

Patch 7.3.220

Problem: Python 3: vim.error is a '**str**' instead of an 'Exception' object, so '**except**' or '**raise**' it causes a 'SystemError' exception. Buffer objects do not support slice assignment. When exchanging text between Vim and Python, multibyte texts become garbage or cause Unicode Exceptions, etc. 'py3file' tries to read in the file as Unicode, sometimes causes UnicodeDecodeException
Solution: Fix the problems. (lilydjwg)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.221

Problem: Text from the clipboard is sometimes handled as linewise, but not consistently.
Solution: Assume the text is linewise when it ends in a CR or NL.
Files: src/gui_gtk_x11.c, src/gui_mac.c, src/ops.c, src/os_msdos.c, src/os_mswin.c, src/os_qnx.c, src/ui.c

Patch 7.3.222

Problem: Warning for building GvimExt.
Solution: Comment-out the DESCRIPTION line. (Mike Williams)

Files: src/GvimExt/gvimext.def, src/GvimExt/gvimext_ming.def

Patch 7.3.223

Problem: MingW cross compilation doesn't work with tiny features.
Solution: Move acp_to_enc(), enc_to_utf16() and utf16_to_enc() outside of "#ifdef CLIPBOARD". Fix typo in makefile.
Files: src/Make_ming.mak, src/os_mswin.c

Patch 7.3.224

Problem: Can't pass dict to sort function.
Solution: Add the optional {dict} argument to sort(). (ZyX)
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.3.225

Problem: Using "\n" in a substitute inside ":s" does not result in a line break.
Solution: Change behavior inside vim_regexec_nl(). Add tests. (Motoya Kurotsu)
Files: src/regexp.c, src/testdir/test79.in, src/testdir/test79.ok, src/testdir/test80.in, src/testdir/test80.ok, src/testdir/Makefile, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.226

Problem: On a 64 bit system "syn sync fromstart" is very slow. (Bjorn Steinbrink)
Solution: Store the state when starting to parse from the first line.
Files: src/syntax.c

Patch 7.3.227 (after 7.3.221)

Problem: Mac OS doesn't have the linewise clipboard fix.
Solution: Also change the Mac OS file. (Bjorn Winckler)
Files: src/os_macosx.m

Patch 7.3.228

Problem: "2gj" does not always move to the correct position.
Solution: Get length of line after moving to a next line. (James Vega)
Files: src/normal.c

Patch 7.3.229

Problem: Using fork() makes gvim crash on Mac when build with CoreFoundation.
Solution: Disallow fork() when __APPLE__ is defined. (Hisashi T Fujinaka)
Files: src/gui.c

Patch 7.3.230

Problem: ":wundo" and ":rundo" don't unescape their argument. (Aaron Thoma)
Solution: Use FILE1 instead of XFILE.
Files: src/ex_cmds.h

Patch 7.3.231

Problem: Runtime file patches failed.

Solution: Redo the patches made against the patched files instead of the files in the mercurial repository.
Files: runtime/doc/indent.txt, runtime/doc/os_win32.txt

Patch 7.3.232

Problem: Python doesn't compile without +multi_byte
Solution: Use "latin1" when MULTI_BYTE is not defined.
Files: src/if_py_both.h

Patch 7.3.233

Problem: ":scriptnames" and ":breaklist" show long file names.
Solution: Shorten to use "~/\" when possible. (Jean-Rene David)
Files: src/ex_cmds2.c

Patch 7.3.234

Problem: With GTK menu may be popping down.
Solution: Use event time instead of GDK_CURRENT_TIME. (Hong Xu)
Files: src/gui.c, src/gui.h, src/gui_gtk.c, src/gui_gtk_x11.c

Patch 7.3.235

Problem: ";" gets stuck on a "t" command, it's not useful.
Solution: Add the ';' flag in '**cpo**'. (Christian Brabandt)
Files: runtime/doc/motion.txt, runtime/doc/options.txt, src/option.h,
src/search.c src/testdir/test81.in, src/testdir/test81.ok,
src/testdir/Makefile, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.236 (after 7.3.232)

Problem: Python 3 doesn't compile without +multi_byte
Solution: Use "latin1" when MULTI_BYTE is not defined. (lilydjwg)
Files: src/if_python3.c

Patch 7.3.237

Problem: "filetype" completion doesn't work on Windows. (Yue Wu)
Solution: Don't use a glob pattern for the directories, use a list of directories. (Dominique Pelle)
Files: src/ex_getln.c

Patch 7.3.238

Problem: Compiler warning for conversion.
Solution: Add type cast. (Mike Williams)
Files: src/ex_getln.c

Patch 7.3.239

Problem: Python corrects the cursor column without taking '**virtualedit**' into account. (lilydjwg)
Solution: Call check_cursor_col_win().
Files: src/if_py_both.h, src/mbyte.c, src/misc2.c, src/normal.c,
src/proto/mbyte.pro, src/proto/misc2.pro

Patch 7.3.240

Problem: External commands can't use pipes on MS-Windows.
Solution: Implement pipes and use them when '**shelltemp**' isn't set. (Vincent

Berthoux)
Files: src/eval.c, src/ex_cmds.c, src/misc2.c, src/os_unix.c,
src/os_win32.c, src/proto/misc2.pro, src/ui.c

Patch 7.3.241

Problem: Using **CTRL-R CTRL-W** on the command line may insert only part of the word.

Solution: Use the cursor position instead of assuming it is at the end of the command. (Tyru)

Files: src/ex_getln.c

Patch 7.3.242

Problem: Illegal memory access in after_pathsep().

Solution: Check that the pointer is not at the start of the file name. (Dominique Pelle)

Files: src/misc2.c

Patch 7.3.243

Problem: Illegal memory access in readline().

Solution: Swap the conditions. (Dominique Pelle)

Files: src/eval.c

Patch 7.3.244

Problem: MS-Windows: Build problem with old compiler. (John Beckett)

Solution: Only use HandleToLong() when available. (Mike Williams)

Files: src/gui_w32.c

Patch 7.3.245

Problem: Python 3.2 libraries not correctly detected.

Solution: Add the suffix to the library name. (Niclas Zeising)

Files: src/auto/configure, src/configure.in

Patch 7.3.246 (after 7.3.235)

Problem: Repeating "f4" in "4444" skips one 4.

Solution: Check the t_cmd flag. (Christian Brabandt)

Files: src/search.c

Patch 7.3.247

Problem: Running tests changes the users viminfo file. Test for patch 7.3.246 missing.

Solution: Add "nviminfo" to the **'viminfo'** option. Include the test.

Files: src/testdir/test78.in, src/testdir/test81.in

Patch 7.3.248

Problem: PC Install instructions missing install instructions.

Solution: Step-by-step explanation. (Michael Soyka)

Files: src/INSTALLpc.txt

Patch 7.3.249

Problem: Wrong indenting for array initializer.

Solution: Detect '}' in a better way. (Lech Lorens)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.250

Problem: Python: Errors in Unicode characters not handled nicely.
Solution: Add the surrogateescape error handler. (lilydjwg)
Files: src/if_python3.c

Patch 7.3.251

Problem: "gH" deletes the current line, except when it's the last line.
Solution: Set the "include" flag to indicate the last line is to be deleted.
Files: src/normal.c, src/ops.c

Patch 7.3.252 (after 7.3.247)

Problem: Tests fail. (David Northfield)
Solution: Add missing update for .ok file.
Files: src/testdir/test81.ok

Patch 7.3.253

Problem: "echo 'abc' > ''" returns 0 or 1, depending on 'ignorecase'. Checks in mb_strnicmp() for illegal and truncated bytes are wrong. Should not assume that byte length is equal before case folding.
Solution: Add utf_safe_read_char_adv() and utf_strnicmp(). Add a test for this. (Ivan Krasilnikov)
Files: src/mbyte.c src/testdir/test82.in, src/testdir/test82.ok, src/testdir/Makefile, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.254

Problem: The coladd field is not reset when setting the line number for a ":call" command.
Solution: Reset it.
Files: src/eval.c

Patch 7.3.255

Problem: When editing a file such as "File[2010-08-15].vim" an E16 error is given. (Manuel Stol)
Solution: Don't give an error for failing to compile the regexp.
Files: src/ex_docmd.c, src/misc1.c, src/vim.h

Patch 7.3.256

Problem: Javascript indenting not sufficiently tested.
Solution: Add more tests. (Luc Deschenaux) Mark the lines that are indented wrong.
Files: src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.257

Problem: Not all completions are available to user commands.
Solution: Add "color", "compiler", "file_in_path" and "locale". (Dominique Pelle)
Files: src/ex_docmd.c, runtime/doc/map.txt

Patch 7.3.258

Problem: MS-Windows: The edit with existing vim context menu entries can be unwanted.

Solution: Let a registry entry disable them. (Jerome Vuarand)
Files: src/GvimExt/gvimext.cpp

Patch 7.3.259

Problem: Equivalence classes only work for latin characters.
Solution: Add the Unicode equivalence characters. (Dominique Pelle)
Files: runtime/doc/pattern.txt, src/regexp.c, src/testdir/test44.in,
src/testdir/test44.ok

Patch 7.3.260

Problem: CursorHold triggers on an incomplete mapping. (Will Gray)
Solution: Don't trigger CursorHold when there is typeahead.
Files: src/fileio.c

Patch 7.3.261

Problem: G++ error message erroneously recognized as error.
Solution: Ignore "In file included from" line also when it ends in a colon.
(Fernando Castillo)
Files: src/option.h

Patch 7.3.262

Problem: Photon code style doesn't match Vim style.
Solution: Clean up some of it. (Elias Diem)
Files: src/gui_photon.c

Patch 7.3.263

Problem: Perl and Tcl have a few code style problems.
Solution: Clean it up. (Elias Diem)
Files: src/if_perl.xs, src/if_tcl.c

Patch 7.3.264

Problem: When the current directory name contains wildcard characters, such as "foo[with]bar", the tags file can't be found. (Jeremy Erickson)
Solution: When searching for matching files also match without expanding wildcards. This is a bit of a hack.
Files: src/vim.h, src/misc1.c, src/misc2.c

Patch 7.3.265

Problem: When storing a pattern in search history there is no proper check for the separator character.
Solution: Pass the separator character to in_history(). (Taro Muraoka)
Files: src/ex_getln.c

Patch 7.3.266

Problem: In gvim with iBus typing space in Insert mode doesn't work.
Solution: Clear xim_expected_char after checking it.
Files: src/mbyte.c

Patch 7.3.267

Problem: Ruby on Mac OS X 10.7 may crash.
Solution: Avoid alloc(0). (Bjorn Winckler)
Files: src/if_ruby.c

Patch 7.3.268

Problem: Vim freezes when executing an external command with zsh.
Solution: Use O_NOCTTY both in the master and slave. (Bjorn Winckler)
Files: src/os_unix.c

Patch 7.3.269

Problem: **'shellcmdflag'** only works with one flag.
Solution: Split into multiple arguments. (Gary Johnson)
Files: src/os_unix.c

Patch 7.3.270

Problem: Illegal memory access.
Solution: Swap conditions. (Dominique Pelle)
Files: src/ops.c

Patch 7.3.271

Problem: Code not following Vim coding style.
Solution: Fix the style. (Elias Diem)
Files: src/gui_photon.c

Patch 7.3.272

Problem: ":put =list" does not add an empty line for a trailing empty item.
Solution: Add a trailing NL when turning a list into a string.
Files: src/eval.c

Patch 7.3.273

Problem: A BOM in an error file is seen as text. (Aleksey Baibarin)
Solution: Remove the BOM from the text before evaluating. (idea by Christian Brabandt)
Files: src/quickfix.c, src/mbyte.c, src/proto/mbyte.pro, src/testdir/test10.in

Patch 7.3.274

Problem: With concealed characters tabs do not have the right size.
Solution: Use VCOL_HLC instead of vcol. (Eiichi Sato)
Files: src/screen.c

Patch 7.3.275

Problem: MS-Windows: When using a black background some screen updates cause the window to flicker.
Solution: Add WS_CLIPCHILDREN to CreateWindow(). (René Aguirre)
Files: src/gui_w32.c

Patch 7.3.276

Problem: GvimExt sets \$LANG in the wrong way.
Solution: Save the environment and use it for gvim. (Yasuhiro Matsumoto)
Files: src/GvimExt/gvimext.cpp

Patch 7.3.277

Problem: MS-Windows: some characters do not show in dialogs.
Solution: Use the wide methods when available. (Yanwei Jia)
Files: src/gui_w32.c, src/gui_w48.c, src/os_mswin.c, src/os_win32.c, src/os_win32.h

Patch 7.3.278

Problem: Passing the file name to open in VisVim doesn't work.
Solution: Adjust the index and check for end of buffer. (Jiri Sedlak)
Files: src/VisVim/Commands.cpp

Patch 7.3.279

Problem: With GTK, when gvim is full-screen and a tab is opened and using a specific monitor configuration the window is too big.
Solution: Adjust the window size like on MS-Windows. (Yukihiro Nakadaira)
Files: src/gui.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro

Patch 7.3.280

Problem: ":lmake" does not update the quickfix window title.
Solution: Update the title. (Lech Lorens)
Files: src/quickfix.c, src/testdir/test10.in, src/testdir/test10.ok

Patch 7.3.281

Problem: After using "expand('%:8')" the buffer name is changed.
Solution: Make a copy of the file name before shortening it.
Files: src/eval.c

Patch 7.3.282

Problem: When using input() and :echo in a loop the displayed text is incorrect. (Benjamin Fritz)
Solution: Only restore the cursor position when there is a command line. (Ben Schmidt)
Files: src/ex_getln.c

Patch 7.3.283

Problem: An expression mapping with a multibyte character containing a 0x80 byte gets messed up. (ZyX)
Solution: Unescape the expression before evaluating it (Yukihiro Nakadaira)
Files: src/getchar.c

Patch 7.3.284

Problem: The str2special() function doesn't handle multibyte characters properly.
Solution: Recognize multibyte characters. (partly by Vladimir Vichniakov)
Files: src/getchar.c, src/message.c, src/misc2.c

Patch 7.3.285 (after 7.3.284)

Problem: Mapping <Char-123> no longer works.
Solution: Properly check for "char-". Add a test for it.
Files: src/misc2.c, src/testdir/test75.in, src/testdir/test75.ok

Patch 7.3.286

Problem: Crash when using "zd" on a large number of folds. (Sam King)
Solution: Recompute pointer after reallocating array. Move fewer entries when making room.
Files: src/fold.c

Patch 7.3.287

Problem: Can't compile with MSVC and tiny options.

Solution: Move variables and #ifdefs. (Sergey Khorev)
Files: src/os_win32.c

Patch 7.3.288

Problem: has('python') may give an error message for not being able to load the library after using python3.

Solution: Only give the error when the verbose argument is true.

Files: src/if_python.c, src/if_python3.c

Patch 7.3.289

Problem: Complete function isn't called when the leader changed.

Solution: Call ins_compl_restart() when the leader changed. (Taro Muraoka)

Files: src/edit.c

Patch 7.3.290

Problem: When a BufWriteCmd autocommand resets **'modified'** this doesn't change older buffer states to be marked as **'modified'** like ":write" does. (Yukihiro Nakadaira)

Solution: When the BufWriteCmd resets **'modified'** then adjust the undo information like ":write" does.

Files: src/fileio.c

Patch 7.3.291

Problem: Configure doesn't work properly with Python3.

Solution: Put -ldl before \$LDLFLAGS. Add PY3_NO_RTLD_GLOBAL. (Roland Puntaier)

Files: src/config.h.in, src/auto/configure, src/configure.in

Patch 7.3.292

Problem: Crash when using fold markers and selecting a visual block that includes a folded line and goes to end of line. (Sam Lidder)

Solution: Check for the column to be MAXCOL. (James Vega)

Files: src/screen.c

Patch 7.3.293

Problem: MSVC compiler has a problem with non-ASCII characters.

Solution: Avoid non-ASCII characters. (Hong Xu)

Files: src/ascii.h, src/spell.c

Patch 7.3.294 (after 7.3.289)

Problem: Patch 289 causes more problems than it solves.

Solution: Revert the patch until a better solution is found.

Files: src/edit.c

Patch 7.3.295

Problem: When filtering text with an external command Vim may not read all the output.

Solution: When select() is interrupted loop and try again. (James Vega)

Files: src/os_unix.c

Patch 7.3.296

Problem: When writing to an external command a zombie process may be left behind.

Solution: Wait on the process. (James Vega)

Files: src/os_unix.c

Patch 7.3.297

Problem: Can't load Perl 5.14 dynamically.
Solution: Add code in #ifdefs. (Charles Cooper)
Files: if_perl.xs

Patch 7.3.298

Problem: Built-in colors are different from rgb.txt.
Solution: Adjust the color values. (Benjamin Haskell)
Files: src/gui_photon.c, src/gui_w48.c

Patch 7.3.299

Problem: Source code not in Vim style.
Solution: Adjust the style. (Elias Diem)
Files: src/gui_photon.c

Patch 7.3.300

Problem: Python doesn't parse multibyte argument correctly.
Solution: Use "t" instead of "s". (lilydjwg)
Files: src/if_py_both.h

Patch 7.3.301

Problem: When '**smartindent**' and '**copyindent**' are set a Tab is used even though '**expandtab**' is set.
Solution: Do not insert Tabs. Add a test. (Christian Brabandt)
Files: src/misc1.c, src/testdir/test19.in, src/testdir/test19.ok

Patch 7.3.302 (after 7.3.301)

Problem: Test 19 fails without '**smartindent**' and +eval.
Solution: Don't use ":exe". Source small.vim.
Files: src/testdir/test19.in

Patch 7.3.303 (after 7.3.296)

Problem: Compilation error.
Solution: Correct return type from int to pid_t. (Danek Duvall)
Files: src/os_unix.c

Patch 7.3.304

Problem: Strawberry Perl doesn't work on MS-Windows.
Solution: Use xsubpp if needed. (Yasuhiro Matsumoto)
Files: src/Make_ming.mak, src/Make_mvc.mak

Patch 7.3.305

Problem: Auto-loading a function while editing the command line causes scrolling up the display.
Solution: Don't set msg_scroll when defining a function and the user is not typing. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.306

Problem: When closing a window there is a chance that deleting a scrollbar triggers a GUI resize, which uses the window while it is not in a valid state.

Solution: Set the buffer pointer to NULL to be able to detect the invalid situation. Fix a few places that used the buffer pointer incorrectly.

Files: src/buffer.c, src/ex_cmds.c, src/term.c, src/window.c

Patch 7.3.307

Problem: Python 3 doesn't support slice assignment.

Solution: Implement slices. (Brett Overesch, Roland Puntaier)

Files: src/if_python3.c

Patch 7.3.308

Problem: Writing to '**verbosefile**' has problems, e.g. for :highlight.

Solution: Do not use a separate verbose_write() function but write with the same code that does redirecting. (Yasuhiro Matsumoto)

Files: src/message.c

Patch 7.3.309 (after 7.3.307)

Problem: Warnings for pointer types.

Solution: Change PySliceObject to PyObject.

Files: src/if_python3.c

Patch 7.3.310

Problem: Code not following Vim style.

Solution: Fix the style. (Elias Diem)

Files: src/gui_photon.c

Patch 7.3.311 (replaces 7.3.289)

Problem: Complete function isn't called when the leader changed.

Solution: Allow the complete function to return a dictionary with a flag that indicates ins_compl_restart() is to be called when the leader changes. (Taro Muraoka)

Files: runtime/insert.txt, src/edit.c, src/eval.c, src/proto/eval.pro

Patch 7.3.312 (after 7.3.306)

Problem: Can't compile with tiny features.

Solution: Add #ifdef around win_valid().

Files: src/buffer.c

Patch 7.3.313 (after 7.3.307)

Problem: One more warning when compiling with dynamic Python 3.

Solution: Change PySliceObject to PyObject.

Files: src/if_python3.c

Patch 7.3.314 (after 7.3.304)

Problem: Missing parenthesis.

Solution: Add it. (Benjamin R. Haskell)

Files: src/Make_mvc.mak

Patch 7.3.315

Problem: Opening a window before forking causes problems for GTK.

Solution: Fork first, create the window in the child and report back to the parent process whether it worked. If successful the parent exits, if unsuccessful the child exits and the parent continues in the terminal. (Tim Starling)

Files: src/gui.c

Patch 7.3.316 (after 7.3.306)

Problem: Crash when **'colorcolumn'** is set and closing buffer.

Solution: Check for w_buffer to be NULL. (Yasuhiro Matsumoto)

Files: src/option.c

Patch 7.3.317

Problem: Calling debug.debug() in Lua may cause Vim to hang.

Solution: Add a better debug method. (Rob Hoelz, Luis Carvalho)

Files: src/if_lua.c

Patch 7.3.318

Problem: "C" on the last line deletes that line if it's blank.

Solution: Only delete the last line for a delete operation. (James Vega)

Files: src/ops.c

Patch 7.3.319 (after 7.3.311)

Problem: Redobuff doesn't always include changes of the completion leader.

Solution: Insert backspaces as needed. (idea by Taro Muraoka)

Files: src/edit.c

Patch 7.3.320

Problem: When a 0xa0 character is in a sourced file the error message for unrecognized command does not show the problem.

Solution: Display 0xa0 as <a0>.

Files: src/ex_docmd.c

Patch 7.3.321

Problem: Code not following Vim style.

Solution: Fix the style. (Elias Diem)

Files: src/os_qnx.c

Patch 7.3.322

Problem: #ifdef for PDP_RETVAl doesn't work, INT_PTR can be a typedef.

Solution: Check the MSC version and 64 bit flags. (Sergiu Dotenco)

Files: src/os_mswin.c

Patch 7.3.323

Problem: The default **'errorformat'** does not ignore some "included from" lines.

Solution: Add a few more patterns. (Ben Boeckel)

Files: src/option.h

Patch 7.3.324 (after 7.3.237)

Problem: Completion for ":compiler" shows color scheme names.

Solution: Fix the directory name. (James Vega)

Files: src/ex_getln.c

Patch 7.3.325

Problem: A duplicated function argument gives an internal error.

Solution: Give a proper error message. (based on patch by Tyru)

Files: src/eval.c

Patch 7.3.326

Problem: MingW 4.6 no longer supports the -mno-cygwin option.
Solution: Split the Cygwin and MingW makefiles. (Matsushita Shougo)
Files: src/GvimExt/Make_cyg.mak, src/GvimExt/Make_ming.mak,
src/Make_cyg.mak, src/Make_ming.mak, src/xxd/Make_ming.mak,
Filelist

Patch 7.3.327

Problem: When jumping to a help tag a closed fold doesn't open.
Solution: Save and restore KeyTyped. (Yasuhiro Matsumoto)
Files: src/ex_cmds.c

Patch 7.3.328

Problem: When command line wraps the cursor may be displayed wrong when there are multibyte characters.
Solution: Position the cursor before drawing the text. (Yasuhiro Matsumoto)
Files: src/ex_getln.c

Patch 7.3.329

Problem: When skipping over code from ":for" to ":endfor" get an error for calling a dict function. (Yasuhiro Matsumoto)
Solution: Ignore errors when skipping over :call command.
Files: src/ex_docmd.c, src/eval.c

Patch 7.3.330

Problem: When longjmp() is invoked if the X server gives an error the state is not properly restored.
Solution: Reset vgetc_busy. (Yukihiro Nakadaira)
Files: src/main.c

Patch 7.3.331

Problem: "vit" selects wrong text when a tag name starts with the same text as an outer tag name. (Ben Fritz)
Solution: Add "\>" to the pattern to check for word boundary.
Files: src/search.c

Patch 7.3.332 (after 7.3.202)

Problem: Indent after "public:" is not increased in C++ code. (Lech Lorens)
Solution: Check for namespace after the regular checks. (partly by Martin Giesecking)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.333

Problem: Using "." to repeat a Visual delete counts the size in bytes, not characters. (Connor Lane Smith)
Solution: Store the virtual column numbers instead of byte positions.
Files: src/normal.c

Patch 7.3.334

Problem: Latest MingW about XSUBPP referencing itself. (Gongqian Li)
Solution: Rename the first use to XSUBPPTRY.
Files: src/Make_ming.mak

Patch 7.3.335

Problem: When **'imdisable'** is reset from an autocommand in Insert mode it doesn't take effect.
Solution: Call `im_set_active()` in Insert mode. (Taro Muraoka)
Files: `src/option.c`

Patch 7.3.336

Problem: When a tags file specifies an encoding different from **'enc'** it may hang and using a pattern doesn't work.
Solution: Convert the whole line. Continue reading the header after the SORT tag. Add `test83`. (Yukihiro Nakadaira)
Files: `src/tag.c`, `src/testdir/Make_amiga.mak`, `src/testdir/Make_dos.mak`, `src/testdir/Make_ming.mak`, `src/testdir/Make_os2.mak`, `src/testdir/Make_vms.mms`, `src/testdir/Makefile`, `src/testdir/test83-tags2`, `src/testdir/test83-tags3`, `src/testdir/test83.in`, `src/testdir/test83.ok`

Patch 7.3.337 (after 7.3.295)

Problem: Screen doesn't update after resizing the xterm until a character is typed.
Solution: When the select call is interrupted check `do_resize`. (Taylor Hedberg)
Files: `src/os_unix.c`

Patch 7.3.338

Problem: Using `getchar()` in an expression mapping doesn't work well.
Solution: Don't save and restore the typeahead. (James Vega)
Files: `src/getchar.c`, `src/testdir/test34.ok`

Patch 7.3.339

Problem: "make shadow" doesn't link all test files.
Solution: Add a line in `Makefile` and `Filelist`.
Files: `src/Makefile`, `Filelist`

Patch 7.3.340

Problem: When **'verbosefile'** is set `ftplugof.vim` can give an error.
Solution: Only remove filetypeplugin autocommands when they exist. (Yasuhiro Matsumoto)
Files: `runtime/ftplugof.vim`

Patch 7.3.341

Problem: Local help files are only listed in `help.txt`, not in translated help files.
Solution: Also find translated help files. (Yasuhiro Matsumoto)
Files: `src/ex_cmds.c`

Patch 7.3.342

Problem: Code not in Vim style.
Solution: Fix the style. (Elias Diem)
Files: `src/os_amiga.c`, `src/os_mac_conv.c`, `src/os_win16.c`

Patch 7.3.343

Problem: No mouse support for `urxvt`.
Solution: Implement `urxvt` mouse support, also for > 252 columns. (Yiding Jia)

Files: src/feature.h, src/keymap.h, src/option.h, src/os_unix.c,
src/term.c, src/version.c

Patch 7.3.344

Problem: Problem with GUI startup related to XInitThreads.
Solution: Use read() and write() instead of fputs() and fread(). (James Vega)
Files: src/gui.c

Patch 7.3.345

Problem: When switching language with ":lang" the window title doesn't change until later.
Solution: Update the window title right away. (Dominique Pelle)
Files: src/ex_cmds2.c

Patch 7.3.346

Problem: It's hard to test netbeans commands.
Solution: Process netbeans commands after :sleep. (Xavier de Gaye)
Files: runtime/doc/netbeans.txt, src/ex_docmd.c, src/netbeans.c

Patch 7.3.347

Problem: When dropping text from a browser on Vim it receives HTML even though "html" is excluded from '**clipboard**'. (Andrei Avk)
Solution: Fix the condition for TARGET_HTML.
Files: src/gui_gtk_x11.c

Patch 7.3.348

Problem: "call range(1, 947948399)" causes a crash. (ZyX)
Solution: Avoid a loop in the out of memory message.
Files: src/misc2.c

Patch 7.3.349

Problem: When running out of memory during startup trying to open a swapfile will loop forever.
Solution: Let findswapname() set dirp to NULL if out of memory.
Files: src/memline.c

Patch 7.3.350

Problem: Block of code after ":lua << EOF" may not work. (Paul Isambert)
Solution: Recognize the ":lua" command, skip to EOF.
Files: src/eval.c

Patch 7.3.351

Problem: Text formatting uses start of insert position when it should not. (Peter Wagenaar)
Solution: Do not use Insstart when intentionally formatting.
Files: src/edit.c

Patch 7.3.352

Problem: When completing methods dict functions and script-local functions get in the way.
Solution: Sort function names starting with "<" to the end. (Yasuhiro Matsumoto)
Files: src/ex_getln.c

Patch 7.3.353 (after 7.3.343)

Problem: Missing part of the urxvt patch.
Solution: Add the change in term.c
Files: src/term.c

Patch 7.3.354

Problem: ":set backspace+=eol" doesn't work when '**backspace**' has a backwards compatible value of 2.
Solution: Convert the number to a string. (Hirohito Higashi)
Files: src/option.c

Patch 7.3.355

Problem: GTK warnings when using netrw.vim. (Ivan Krasilnikov)
Solution: Do not remove the beval event handler twice.
Files: src/option.c

Patch 7.3.356

Problem: Using "o" with '**cindent**' set may freeze Vim. (lolilolicon)
Solution: Skip over {} correctly. (Hari G)
Files: src/misc1.c

Patch 7.3.357

Problem: Compiler warning in MS-Windows console build.
Solution: Adjust return type of PrintHookProc(). (Mike Williams)
Files: src/os_mswin.c

Patch 7.3.358 (after 7.3.353)

Problem: Mouse support doesn't work properly.
Solution: Add HMT_URXVT. (lilydjwg, James McCoy)
Files: src/term.c

Patch 7.3.359

Problem: Command line completion shows dict functions.
Solution: Skip dict functions for completion. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.360

Problem: Interrupting the load of an autoloader function may cause a crash.
Solution: Do not use the hashitem when not valid. (Yukihiro Nakadaira)
Files: src/eval.c

Patch 7.3.361

Problem: Accessing memory after it is freed when EXITFREE is defined.
Solution: Don't access curwin when firstwin is NULL. (Dominique Pelle)
Files: src/buffer.c

Patch 7.3.362

Problem: ml_get error when using ":g" with folded lines.
Solution: Adjust the line number for changed_lines(). (Christian Brabandt)
Files: src/ex_cmds.c

Patch 7.3.363

Problem: C indenting is wrong after #endif followed by a semicolon.

Solution: Add special handling for a semicolon in a line by itself. (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.364 (after 7.3.353)
Problem: Can't compile on HP-UX. (John Marriott)
Solution: Only use TTYM_URXVT when it is defined.
Files: src/term.c

Patch 7.3.365
Problem: Crash when using a large Unicode character in a file that has syntax highlighting. (ngollan)
Solution: Check for going past the end of the utf tables. (Dominique Pelle)
Files: src/mbyte.c

Patch 7.3.366
Problem: A tags file with an extremely long name causes errors.
Solution: Ignore tags that are too long. (Arno Renevier)
Files: src/tag.c

Patch 7.3.367
Problem: :wundo and :rundo use a wrong checksum.
Solution: Include the last line when computing the hash. (Christian Brabandt)
Files: src/undo.c

Patch 7.3.368
Problem: Gcc complains about redefining _FORTIFY_SOURCE.
Solution: Undefine it before redefining it.
Files: src/Makefile, src/configure.in, src/auto/configure

Patch 7.3.369
Problem: When compiled with Gnome get an error message when using --help.
Solution: Don't fork. (Ivan Krasilnikov)
Files: src/main.c

Patch 7.3.370
Problem: Compiler warns for unused variable in Lua interface.
Solution: Remove the variable.
Files: src/if_lua.c

Patch 7.3.371
Problem: Crash in autocomplete. (Greg Weber)
Solution: Check not going over allocated buffer size.
Files: src/misc2.c

Patch 7.3.372
Problem: When using a command line mapping to <Up> with file name completion to go one directory up, 'wildchar' is inserted. (Yasuhiro Matsumoto)
Solution: Set the KeyTyped flag.
Files: src/ex_getln.c

Patch 7.3.373 (after 7.3.366)
Problem: A tags file with an extremely long name may cause an infinite loop.

Solution: When encountering a long name switch to linear search.
Files: src/tag.c

Patch 7.3.374

Problem: ++encoding does not work properly.
Solution: Recognize ++encoding before ++enc. (Charles Cooper)
Files: src/ex_docmd.c

Patch 7.3.375

Problem: Duplicate return statement.
Solution: Remove the superfluous one. (Dominique Pelle)
Files: src/gui_mac.c

Patch 7.3.376

Problem: Win32: Toolbar repainting does not work when the mouse pointer hovers over a button.
Solution: Call DefWindowProc() when not handling an event. (Sergiu Dotenco)
Files: src/gui_w32.c

Patch 7.3.377

Problem: No support for bitwise AND, OR, XOR and invert.
Solution: Add and(), or(), invert() and xor() functions.
Files: src/eval.c, src/testdir/test49.in, src/testdir/test65.in, src/testdir/test65.ok, runtime/doc/eval.txt

Patch 7.3.378

Problem: When cross-compiling the check for uint32_t fails.
Solution: Only give a warning message. (Maksim Melnikau)
Files: src/configure.in, src/auto/configure

Patch 7.3.379

Problem: C-indenting wrong for static enum.
Solution: Skip over "static". (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.380

Problem: C-indenting wrong for a function header.
Solution: Skip to the start paren. (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.381

Problem: Configure silently skips interfaces that won't work.
Solution: Add the --enable-fail_if_missing argument. (Shlomi Fish)
Files: src/Makefile, src/configure.in, src/auto/configure

Patch 7.3.382 (after 7.3.376)

Problem: IME characters are inserted twice.
Solution: Do not call DefWindowProc() if the event was handled. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 7.3.383

Problem: For EBCDIC pound sign is defined as 't'.
Solution: Correctly define POUND.

Files: src/ascii.h

Patch 7.3.384

Problem: Mapping CTRL-K in Insert mode breaks CTRL-X CTRL-K for dictionary completion.

Solution: Add CTRL-K to the list of recognized keys. (James McCoy)

Files: src/edit.c

Patch 7.3.385

Problem: When using an expression mapping on the command line the cursor ends up in the wrong place. (Yasuhiro Matsumoto)

Solution: Save and restore msg_col and msg_row when evaluating the expression.

Files: src/getchar.

Patch 7.3.386

Problem: Test 83 fails when iconv does not support cp932. (raf)

Solution: Test if conversion works. (Yukihiro Nakadaira)

Files: src/testdir/test83.in

Patch 7.3.387 (after 7.3.386)

Problem: Test 83 may fail for some encodings.

Solution: Set 'encoding' to utf-8 earlier.

Files: src/testdir/test83.in

Patch 7.3.388

Problem: Crash on exit when EXITFREE is defined and using tiny features.

Solution: Check for NULL window pointer. (Dominique Pelle)

Files: src/buffer.c

Patch 7.3.389

Problem: After typing at a prompt the "MORE" message appears too soon.

Solution: reset lines_left in msg_end_prompt(). (Eswald)

Files: src/message.c

Patch 7.3.390

Problem: Using NULL buffer pointer in a window.

Solution: Check for w_buffer being NULL in more places. (Bjorn Winckler)

Files: src/ex_cmds.c, src/quickfix.c, src/window.c

Patch 7.3.391

Problem: Can't check if the XPM_W32 feature is enabled.

Solution: Add xpm_w32 to the list of features. (kat)

Files: src/eval.c

Patch 7.3.392

Problem: When setting 'undofile' while the file is already loaded but unchanged, try reading the undo file. (Andy Wokula)

Solution: Compute a checksum of the text when 'undofile' is set. (Christian Brabandt)

Files: src/option.c, src/testdir/test72.in, src/testdir/test72.ok

Patch 7.3.393

Problem: Win32: When resizing Vim it is always moved to the primary monitor

if the secondary monitor is on the left.
Solution: Use the nearest monitor. (Yukihiro Nakadaira)
Files: src/gui_w32.c

Patch 7.3.394

Problem: When placing a mark while starting up a screen redraw messes up the screen. (lith)
Solution: Don't redraw while still starting up. (Christian Brabandt)
Files: src/screen.c

Patch 7.3.395 (after 7.3.251)

Problem: "dv?bar" in the last line deletes too much and breaks undo.
Solution: Only adjust the cursor position when it's after the last line of the buffer. Add a test. (Christian Brabandt)
Files: src/ops.c, src/testdir/test43.in, src/testdir/test43.ok

Patch 7.3.396

Problem: After forcing an operator to be characterwise it can still become linewise when spanning whole lines.
Solution: Don't make the operator linewise when motion_force was set. (Christian Brabandt)
Files: src/ops.c

Patch 7.3.397

Problem: ":helpgrep" does not work properly when 'encoding' is not utf-8 or latin1.
Solution: Convert non-ascii lines to 'encoding'. (Yasuhiro Matsumoto)
Files: src/quickfix.c, src/spell.c, src/misc2.c, src/proto/misc2.pro

Patch 7.3.398

Problem: When creating more than 10 location lists and adding items one by one a previous location may be used. (Audrius Kazukauskas)
Solution: Clear the location list completely when adding the tenth one.
Files: src/quickfix.c

Patch 7.3.399

Problem: ":cd" doesn't work when the path contains wildcards. (Yukihiro Nakadaira)
Solution: Ignore wildcard errors when the EW_NOTWILD flag is used.
Files: src/misc1.c

Patch 7.3.400

Problem: Compiler warnings for shadowed variables.
Solution: Remove or rename the variables.
Files: src/charset.c, src/digraph.c, src/edit.c, src/eval.c, src/fold.c, src/getchar.c, src/message.c, src/misc2.c, src/move.c, src/netbeans.c, src/option.c, src/os_unix.c, src/screen.c, src/search.c, src/spell.c, src/syntax.c, src/tag.c, src/window.c

Patch 7.3.401

Problem: A couple more shadowed variables.
Solution: Rename the variables.
Files: src/netbeans.c

Patch 7.3.402

Problem: When jumping to the first error a line of the buffer is sometimes redrawn on top of the list of errors.
Solution: Do not call update_topline_redraw() if the display was scrolled up.
Files: src/quickfix.c

Patch 7.3.403

Problem: ":helpgrep" does not trigger QuickFixCmd* autocommands.
Solution: Trigger the autocommands. (Christian Brabandt)
Files: src/quickfix.c

Patch 7.3.404

Problem: When a complete function uses refresh "always" redo will not work properly.
Solution: Do not reset compl_leader when compl_opt_refresh_always is set. (Yasuhiro Matsumoto)
Files: src/edit.c

Patch 7.3.405

Problem: When xterm gets back the function keys it may delete the urxvt mouse termcap code.
Solution: Check for the whole code, not just the start. (Egmont Koblinger)
Files: src/keymap.h, src/misc2.c, src/term.c

Patch 7.3.406

Problem: Multi-byte characters in b:browsefilter are not handled correctly.
Solution: First use convert_filter() normally and then convert to wide characters. (Taro Muraoka)
Files: src/gui_w48.c

Patch 7.3.407

Problem: ":12verbose call F()" may duplicate text while trying to truncate. (Thinca)
Solution: Only truncate when there is not enough room. Also check the byte length of the buffer.
Files: src/buffer.c, src/eval.c, src/ex_getln.c, src/message.c, src/proto/message.pro

Patch 7.3.408 (after 7.3.406)

Problem: Missing declaration.
Solution: Add the declaration. (John Marriott)
Files: src/gui_w48.c

Patch 7.3.409

Problem: The license in pty.c is unclear.
Solution: Add a comment about the license.
Files: src/pty.c

Patch 7.3.410

Problem: Compiler error for // comment. (Joachim Schmitz)
Solution: Turn into /* comment */.
Files: src/message.c

Patch 7.3.411

Problem: Pasting in Visual mode using the "" register does not work. (John Beckett)
Solution: Detect that the write is overwriting the pasted register. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.412

Problem: Storing a float in a session file has an additional '&'.
Solution: Remove the '&'. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.413

Problem: Build warnings on MS-Windows.
Solution: Add type casts. (Mike Williams)
Files: src/ex_getln.c, src/message.c, src/term.c

Patch 7.3.414

Problem: Using **CTRL-A** on "000" drops the leading zero, while on "001" it doesn't.
Solution: Detect "000" as an octal number. (James McCoy)
Files: src/charset.c

Patch 7.3.415 (after 7.3.359)

Problem: Completion of functions stops once a dictionary is encountered. (James McCoy)
Solution: Return an empty string instead of NULL.
Files: src/eval.c

Patch 7.3.416 (after 7.3.415)

Problem: Compiler warning for wrong pointer.
Solution: Add type cast.
Files: src/eval.c

Patch 7.3.417 (after 7.3.395)

Problem: Test 43 fails with a tiny build.
Solution: Only run test 43 with at least a small build.
Files: src/testdir/test43.in

Patch 7.3.418

Problem: When a user complete function returns -1 an error message is given.
Solution: When -2 is returned stop completion silently. (Yasuhiro Matsumoto)
Files: src/edit.

Patch 7.3.419

Problem: DBCS encoding in a user command does not always work.
Solution: Skip over DBCS characters. (Yasuhiro Matsumoto)
Files: src/ex_docmd.c

Patch 7.3.420

Problem: "it" and "at" don't work properly with a dash in the tag name.
Solution: Require a space to match the tag name. (Christian Brabandt)
Files: src/search.c

Patch 7.3.421

Problem: Get E832 when setting **'undofile'** in vimrc and there is a file to be edited on the command line. (Toothpik)
Solution: Do not try reading the undo file for a file that wasn't loaded.
Files: src/option.c

Patch 7.3.422

Problem: Python 3 does not have `__members__`.
Solution: Add "name" and "number" in another way. (lilydjwg)
Files: src/if_py_both.h, src/if_python3.c

Patch 7.3.423

Problem: Small mistakes in comments, proto and indent.
Solution: Fix the mistakes.
Files: src/ex_cmds2.c, src/structs.h, src/ui.c, src/proto/ex_docmd.pro

Patch 7.3.424

Problem: Win16 version missing some functions.
Solution: Add #defines for the functions.
Files: src/gui_w16.c

Patch 7.3.425 (after 7.3.265)

Problem: Search history lines are duplicated. (Edwin Steiner)
Solution: Convert separator character from space to NUL.
Files: src/ex_getln.c

Patch 7.3.426

Problem: With '\$' in **'coptions'** the \$ is not displayed in the first column.
Solution: Use -1 instead of 0 as a special value. (Hideki Eiraku and Hirohito Higashi)
Files: src/edit.c, src/globals.h, src/move.c, src/screen.c, src/search.c

Patch 7.3.427

Problem: readfile() can be slow with long lines.
Solution: Use realloc() instead of alloc(). (John Little)
Files: src/eval.c

Patch 7.3.428

Problem: Win32: an xpm file without a mask crashes Vim.
Solution: Fail when the mask is missing. (Dave Bodenstab)
Files: src/xpm_w32.c

Patch 7.3.429

Problem: When **'coptions'** includes "E" "c0" in the first column is an error. The redo register is then set to the erroneous command.
Solution: Do not set the redo register if the command fails because of an empty region. (Hideki Eiraku)
Files: src/getchar.c, src/normal.c, src/proto/getchar.pro

Patch 7.3.430

Problem: When a custom filetype detection uses "augroup END" the conf filetype detection does not have the filetypepedetect group.

Solution: Always end the group and include filetypedetect in the conf
autocommand. (Lech Lorens)
Files: runtime/filetype.vim

Patch 7.3.431

Problem: Fetching a key at a prompt may be confused by escape sequences.
Especially when getting a prompt at a VimEnter autocommand.
(Alex Efros)
Solution: Properly handle escape sequences deleted by check_termcode().
Files: src/getchar.c, src/misc1.c, src/term.c, src/proto/term.pro

Patch 7.3.432

Problem: ACLs are not supported for ZFS or NFSv4 on Solaris.
Solution: Add configure check and code. (Danek Duvall)
Files: src/configure.in, src/auto/configure, src/config.h.in,
src/os_unix.c

Patch 7.3.433

Problem: Using continued lines in a Vim script can be slow.
Solution: Instead of reallocating for every line use a growarray. (Yasuhiro
Matsumoto)
Files: src/ex_cmds2.c

Patch 7.3.434

Problem: Using join() can be slow.
Solution: Compute the size of the result before allocation to avoid a lot of
allocations and copies. (Taro Muraoka)
Files: src/eval.c

Patch 7.3.435

Problem: Compiler warning for unused variable.
Solution: Move the variable inside #ifdef.
Files: src/ex_cmds2.c

Patch 7.3.436

Problem: Compiler warnings for types on Windows.
Solution: Add type casts. (Mike Williams)
Files: src/eval.c

Patch 7.3.437

Problem: Continue looping inside FOR_ALL_TAB_WINDOWS even when already done.
Solution: Use goto instead of break. (Hirohito Higashi)
Files: src/fileio.c, src/globals.h

Patch 7.3.438

Problem: There is no way to avoid ":doautoall" reading modelines.
Solution: Add the <nomodeline> argument. Adjust documentation.
Files: src/fileio.c, runtime/doc/autocmd.txt

Patch 7.3.439

Problem: Compiler warnings to size casts in Perl interface.
Solution: Use XS macros. (James McCoy)
Files: src/if_perl.xs, src/typemap

Patch 7.3.440

Problem: Vim does not support UTF8_STRING for the X selection.
Solution: Add UTF8_STRING atom support. (Alex Efros) Use it only when **'encoding'** is set to Unicode.
Files: src/ui.c

Patch 7.3.441

Problem: Newer versions of MzScheme (Racket) require earlier (trampolined) initialisation.
Solution: Call mzscheme_main() early in main(). (Sergey Khorev)
Files: src/Make_mvc.mak, src/if_mzsch.c, src/main.c, src/proto/if_mzsch.pro

Patch 7.3.442 (after 7.3.438)

Problem: Still read modelines for ":doautocmd".
Solution: Move check for <nomodeline> to separate function.
Files: src/fileio.c, src/ex_docmd.c, src/proto/fileio.pro, runtime/doc/autocmd.txt

Patch 7.3.443

Problem: MS-Windows: **'shcf'** and **'shellxquote'** defaults are not very good.
Solution: Make a better guess when **'shell'** is set to "cmd.exe". (Ben Fritz)
Files: src/option.c, runtime/doc/options.txt

Patch 7.3.444

Problem: ":all!" and ":sall!" give error E477, even though the documentation says these are valid commands.
Solution: Support the exclamation mark. (Hirohito Higashi)
Files: src/ex_cmds.h, src/testdir/test31.in, src/testdir/test31.ok

Patch 7.3.445 (after 7.3.443)

Problem: Can't properly escape commands for cmd.exe.
Solution: Default **'shellxquote'** to '('. Append ')' to make '(command)'. No need to use "/s" for **'shellcmdflag'**.
Files: src/misc2.c, src/option.c, src/os_win32.c

Patch 7.3.446 (after 7.3.445)

Problem: Win32: External commands with special characters don't work.
Solution: Add the **'shellxescape'** option.
Files: src/misc2.c, src/option.c, src/option.h, runtime/doc/options.txt

Patch 7.3.447 (after 7.3.446)

Problem: Win32: External commands with "start" do not work.
Solution: Unescape part of the command. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.3.448 (after 7.3.447)

Problem: Win32: Still a problem with "!start /b".
Solution: Escape only '|'. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.3.449

Problem: Crash when a BufWinLeave autocommand closes the only other window. (Daniel Hunt)

Solution: Abort closing a buffer when it becomes the only one.
Files: src/buffer.c, src/proto/buffer.pro, src/ex_cmds.c, src/ex_getln.c,
src/misc2.c, src/quickfix.c, src/window.c, src/proto/window.pro

Patch 7.3.450 (after 7.3.448)
Problem: Win32: Still a problem with "!start /b".
Solution: Fix pointer use. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.3.451
Problem: Tcl doesn't work on 64 MS-Windows.
Solution: Make it work. (Dave Bodenstab)
Files: src/Make_mvc.mak, src/if_tcl.c

Patch 7.3.452
Problem: Undo broken when pasting close to the last line. (Andrey Radev)
Solution: Use a flag to remember if the deleted included the last line.
(Christian Brabandt)
Files: src/ops.c

Patch 7.3.453
Problem: Pasting in the command line is slow.
Solution: Don't redraw if there is another character to read. (Dominique
Pelle)
Files: src/ex_getln.c

Patch 7.3.454
Problem: Re-allocating memory slows Vim down.
Solution: Use realloc() in ga_grow(). (Dominique Pelle)
Files: src/misc2.c

Patch 7.3.455
Problem: Using many continuation lines can be slow.
Solution: Adjust the reallocation size to the current length.
Files: src/ex_cmds2.c

Patch 7.3.456
Problem: ":tab drop file" has several problems, including moving the
current window and opening a new tab for a file that already has a
window.
Solution: Refactor ":tab drop" handling. (Hirohito Higashi)
Files: src/buffer.c, src/testdir/test62.in, src/testdir/test62.ok

Patch 7.3.457
Problem: When setting \$VIMRUNTIME later the directory for fetching
translated messages is not adjusted.
Solution: Put bindtextdomain() in vim_setenv().
Files: src/misc1.c

Patch 7.3.458
Problem: Crash when calling msg() during startup.
Solution: Don't use 'shortmess' when it is not set yet.
Files: src/option.c

Patch 7.3.459

Problem: Win32: Warnings for type conversion.
Solution: Add type casts. (Mike Williams)
Files: src/misc2.c, src/os_win32.c

Patch 7.3.460

Problem: Win32: UPX does not compress 64 bit binaries.
Solution: Mention and add the alternative: mpress. (Dave Bodenstab)
Files: src/INSTALLpc.txt, src/Make_ming.mak

Patch 7.3.461

Problem: The InsertCharPre autocommand event is not triggered during completion and when typing several characters quickly.
Solution: Also trigger InsertCharPre during completion. Do not read ahead when an InsertCharPre autocommand is defined. (Yasuhiro Matsumoto)
Files: src/edit.c, src/fileio.c, src/proto/fileio.pro

Patch 7.3.462

Problem: When using ":loadview" folds may be closed unexpectedly.
Solution: Take into account foldlevel. (Xavier de Gaye)
Files: src/fold.c

Patch 7.3.463

Problem: When using ":s///c" the cursor is moved away from the match. (Lawman)
Solution: Don't move the cursor when do_ask is set. (Christian Brabandt)
Files: src/ex_cmds.c

Patch 7.3.464

Problem: Compiler warning for sprintf.
Solution: Put the length in a variable. (Dominique Pelle)
Files: src/version.c

Patch 7.3.465

Problem: Cannot get file name with newline from glob().
Solution: Add argument to glob() and expand() to indicate they must return a list. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/eval.c, src/ex_getln.c, src/vim.h

Patch 7.3.466

Problem: Get ml_get error hen ":behave mswin" was used and selecting several lines. (A. Sinan Unur)
Solution: Adjust the end of the operation. (Christian Brabandt)
Files: src/ops.c

Patch 7.3.467

Problem: Cursor positioned wrong at the command line when regaining focus and using some input method.
Solution: Do not position the cursor in command line mode.
Files: src/mbyte.c

Patch 7.3.468

Problem: For some compilers the error file is not easily readable.
Solution: Use QuickFixCmdPre for more commands. (Marcin Szamotulski)

Files: runtime/doc/autocmd.txt, src/quickfix.c

Patch 7.3.469

Problem: Compiler warning for unused argument without some features.

Solution: Add UNUSED.

Files: src/buffer.c

Patch 7.3.470

Problem: Test 62 fails when compiled without GUI and X11.

Solution: Don't test :drop when it is not supported.

Files: src/testdir/test62.in

Patch 7.3.471

Problem: Can't abort listing placed signs.

Solution: Check "got_int". (Christian Brabandt)

Files: src/buffer.c, src/ex_cmds.c

Patch 7.3.472

Problem: Crash when using ":redraw" in a BufEnter autocommand and switching to another tab. (☒☒)

Solution: Move triggering the autocommands to after correcting the option values. Also check the row value to be out of bounds. (Christian Brabandt, Sergey Khorev)

Files: src/screen.c, src/window.c

Patch 7.3.473

Problem: **'cursorbind'** does not work correctly in combination with **'virtualedit'** set to "all".

Solution: Copy coladd. (Gary Johnson)

Files: src/move.c

Patch 7.3.474

Problem: Perl build with gcc 4 fails.

Solution: Remove XS() statements. (Yasuhiro Matsumoto)

Files: src/if_perl.xs

Patch 7.3.475

Problem: In a terminal with few colors the omnicomplete menu may be hard to see when using the default colors.

Solution: Use more explicit colors. (suggested by Alex Henrie)

Files: src/syntax.c

Patch 7.3.476

Problem: When selecting a block, using "\$" to include the end of each line and using "A" and typing a backspace strange things happen. (Yuangchen Xie)

Solution: Avoid using a negative length. (Christian Brabandt)

Files: src/ops.c

Patch 7.3.477

Problem: Using ":echo" to output enough lines to scroll, then using "j" and "k" at the more prompt, displays the command on top of the output. (Marcin Szamotulski)

Solution: Put the output below the command. (Christian Brabandt)

Files: src/eval.c

Patch 7.3.478

Problem: Memory leak using the ':rv!' command when reading dictionary or list global variables i.e. with '**viminfo**' containing !.

Solution: Free the typeval. (Dominique Pelle)

Files: src/eval.c

Patch 7.3.479

Problem: When '**cursorline**' is set the line number highlighting can't be set separately.

Solution: Add "CursorLineNr". (Howard Buchholz)

Files: src/option.c, src/screen.c, src/syntax.c, src/vim.h

Patch 7.3.480

Problem: When using ":qa" and there is a changed buffer picking the buffer to jump to is not very good.

Solution: Consider current and other tab pages. (Hirohito Higashi)

Files: src/ex_cmds2.c

Patch 7.3.481

Problem: Changing '**virtualedit**' in an operator function to "all" does not have the desired effect. (Aaron Bohannon)

Solution: Save, reset and restore virtual_op when executing an operator function.

Files: src/normal.c

Patch 7.3.482

Problem: With '**cursorbind**' set moving up/down does not always keep the same column.

Solution: Set curswant appropriately. (Gary Johnson)

Files: src/move.c

Patch 7.3.483 (after 7.3.477)

Problem: More prompt shows up too often.

Solution: Instead of adding a line break, only start a new line in the message history. (Christian Brabandt)

Files: src/eval.c, src/message.c, src/proto/message.pro

Patch 7.3.484

Problem: The -E and --echo-wid command line arguments are not mentioned in "vim --help".

Solution: Add the help lines. (Dominique Pelle)

Files: src/main.c

Patch 7.3.485

Problem: When building Vim LDFLAGS isn't passed on to building xxd.

Solution: Pass the LDFLAGS value. (James McCoy)

Files: src/Makefile

Patch 7.3.486

Problem: Build error with mingw64 on Windows 7.

Solution: Avoid the step of going through vimres.res. (Guopeng Wen)

Files: src/Make_ming.mak

Patch 7.3.487

Problem: When setting **'timeoutlen'** or **'ttimeoutlen'** the column for vertical movement is reset unnecessarily.

Solution: Do not set `w_set_curswant` for every option. Add a test for this. (Kana Natsuno) Add the `P_CURSWANT` flag for options.

Files: `src/option.c`, `src/testdir/test84.in`, `src/testdir/test84.ok`,
`src/testdir/Make_amiga.mak`, `src/testdir/Make_dos.mak`,
`src/testdir/Make_ming.mak`, `src/testdir/Make_os2.mak`,
`src/testdir/Make_vms.mms`, `src/testdir/Makefile`

Patch 7.3.488

Problem: `":help!"` in a help file does not work as documented.

Solution: When in a help file don't give an error message. (thinca)

Files: `src/ex_cmds.c`

Patch 7.3.489

Problem: **CTRL-]** in Insert mode does not expand abbreviation when used in a mapping. (Yichao Zhou)

Solution: Special case using **CTRL-]**. (Christian Brabandt)

Files: `src/getchar.c`, `src/edit.c`

Patch 7.3.490

Problem: Member confusion in Lua interface.

Solution: Fix it. Add `luaeval()`. (Taro Muraoka, Luis Carvalho)

Files: `runtime/doc/if_lua.txt`, `src/eval.c`, `src/if_lua.c`,
`src/proto/if_lua.pro`

Patch 7.3.491

Problem: No tests for Lua.

Solution: Add some simple tests for Lua. (Luis Carvalho)

Files: `src/testdir/test1.in`, `src/testdir/test85.in`, `src/testdir/test85.ok`,
`src/testdir/Make_amiga.mak`, `src/testdir/Make_dos.mak`,
`src/testdir/Make_ming.mak`, `src/testdir/Make_os2.mak`,
`src/testdir/Make_vms.mms`, `src/testdir/Makefile`

Patch 7.3.492

Problem: Can't indent conditions separately from function arguments.

Solution: Add the `'k'` flag in **'cino'**. (Lech Lorens)

Files: `runtime/doc/indent.txt`, `src/misc1.c`, `src/testdir/test3.in`,
`src/testdir/test3.ok`

Patch 7.3.493 (after 7.3.492)

Problem: Two unused variables.

Solution: Remove them. (Hong Xu)

Files: `src/misc1.c`

Patch 7.3.494 (after 7.3.491)

Problem: Can't compile with Lua 5.1 or dynamic Lua.

Solution: Fix `dll_methods`. Fix `luado()`. (Muraoka Taro, Luis Carvalho)

Files: `src/if_lua.c`

Patch 7.3.495 (after 7.3.492)

Problem: Compiler warnings.

Solution: Add function declaration. Remove "offset" argument.
Files: src/misc1.c

Patch 7.3.496

Problem: MS-DOS: When "diff" trips over difference in line separators some tests fail.

Solution: Make some .ok files use unix line separators. (David Pope)
Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak

Patch 7.3.497

Problem: Crash when doing ":python print" and compiled with gcc and the optimizer enabled.

Solution: Avoid the crash, doesn't really fix the problem. (Christian Brabandt)
Files: src/if_py_both.h

Patch 7.3.498

Problem: The behavior of the "- register changes depending on value of the '**clipboard**' option. (Szamotulski)

Solution: Also set the "- register when the register is "*" or "+". (Christian Brabandt)

Files: src/ops.c

Patch 7.3.499

Problem: When using any interface language when Vim is waiting for a child process it gets confused by a child process started through the interface.

Solution: Always used waitpid() instead of wait(). (Yasuhiro Matsumoto)
Files: src/os_unix.c

Patch 7.3.500

Problem: Ming makefile unconditionally sets WINVER.

Solution: Only defined when not already defined. (Yasuhiro Matsumoto)
Files: src/Make_ming.mak

Patch 7.3.501

Problem: Error for "flush" not being defined when using Ruby command.

Solution: Defined "flush" as a no-op method. (Kent Sibilev)
Files: src/if_ruby.c

Patch 7.3.502

Problem: Netbeans insert halfway a line actually appends to the line.

Solution: Insert halfway the line. (Brian Victor)
Files: src/netbeans.c

Patch 7.3.503 (after 7.3.501)

Problem: Warning for unused argument.

Solution: Add UNUSED.
Files: src/if_ruby.c

Patch 7.3.504

Problem: Commands in help files are not highlighted.

Solution: Allow for commands in backticks. Adjust **CTRL-]** to remove the backticks.

Files: src/ex_cmds.c

Patch 7.3.505

Problem: Test 11 fails on MS-Windows in some versions.

Solution: Fix #ifdefs for whether filtering through a pipe is possible. Move setting b_no_eol_lnum back to where it was before patch 7.3.124. (David Pope)

Files: src/feature.h, src/eval.c, src/ex_cmds.c, src/fileio.c

Patch 7.3.506

Problem: GTK gives an error when selecting a non-existent file.

Solution: Add a handler to avoid the error. (Christian Brabandt)

Files: src/gui_gtk.c

Patch 7.3.507

Problem: When exiting with unsaved changes, selecting an existing file in the file dialog, there is no dialog to ask whether the existing file should be overwritten. (Felipe G. Nievinski)

Solution: Call check_overwrite() before writing. (Christian Brabandt)

Files: src/ex_cmds.c, src/ex_cmds2.c, src/proto/ex_cmds.pro

Patch 7.3.508

Problem: Default for v:register is not set.

Solution: Init v:register in eval_init(). Correct for 'clipboard' before the main loop. (Ingo Karkat)

Files: src/eval.c, src/main.c

Patch 7.3.509

Problem: ":vimgrep" fails when 'autochdir' is set.

Solution: A more generic solution for changing directory. (Ben Fritz)

Files: src/quickfix.c

Patch 7.3.510

Problem: Test 77 fails on Solaris 7. (Michael Soyka)

Solution: Replace any tabs with spaces.

Files: src/testdir/test77.in

Patch 7.3.511

Problem: Using a FileReadCmd autocommand that does ":e! {file}" may cause a crash. (Christian Brabandt)

Solution: Properly restore curwin->w_s.

Files: src/fileio.c

Patch 7.3.512

Problem: undofile() returns a useless name when passed an empty string.

Solution: Return an empty string. (Christian Brabandt)

Files: src/eval.c

Patch 7.3.513

Problem: Cannot use **CTRL-E** and **CTRL-Y** with "r".

Solution: Make **CTRL-E** and **CTRL-Y** work like in Insert mode. (Christian Brabandt)

Files: src/edit.c, src/normal.c, src/proto/edit.pro

Patch 7.3.514

Problem: No completion for :history command.
Solution: Add the completion and update the docs. Also fix ":behave" completion. (Dominique Pelle)
Files: runtime/doc/cmdline.txt, runtime/doc/map.txt, src/ex_docmd.c, src/ex_getln.c, src/vim.h

Patch 7.3.515

Problem: **'wildignorecase'** only applies to the last part of the path.
Solution: Also ignore case for letters earlier in the path.
Files: src/misc1.c

Patch 7.3.516

Problem: extend(o, o) may crash Vim.
Solution: Fix crash and add test. (Thinca and Hirohito Higashi)
Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.3.517

Problem: Crash when using "vipvv". (Alexandre Provencio)
Solution: Don't let the text length become negative.
Files: src/ops.c

Patch 7.3.518

Problem: When **'encoding'** is a double-byte encoding ":helptags" may not find tags correctly.
Solution: Use vim_strbyte() instead of vim_strchr(). (Yasuhiro Matsumoto)
Files: src/ex_cmds.c

Patch 7.3.519

Problem: When completefunction returns it cannot indicate end of completion mode.
Solution: Recognize completefunction returning -3. (Matsushita Shougo)
Files: src/edit.c

Patch 7.3.520

Problem: gvim starts up slow on Ubuntu 12.04.
Solution: Move the call to gui_mch_init_check() to after fork(). (Yasuhiro Matsumoto) Do check \$DISPLAY being set.
Files: src/gui.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro

Patch 7.3.521

Problem: Using "z=" on a multibyte character may cause a crash.
Solution: Don't use strlen() on an int pointer.
Files: src/spell.c

Patch 7.3.522

Problem: Crash in vim_realloc() when using MEM_PROFILE.
Solution: Avoid using a NULL argument. (Dominique Pelle)
Files: src/eval.c

Patch 7.3.523

Problem: ":diffupdate" doesn't check for files changed elsewhere.
Solution: Add the ! flag. (Christian Brabandt)
Files: runtime/doc/diff.txt, src/diff.c, src/ex_cmds.h

Patch 7.3.524 (after 7.3.523)

Problem: Missing comma.
Solution: Add the comma.
Files: src/version.c

Patch 7.3.525

Problem: Compiler warning on 64 bit MS-Windows.
Solution: Add type cast. (Mike Williams)
Files: src/ex_getln.c

Patch 7.3.526

Problem: Confusing indenting for #ifdef.
Solution: Remove and add indent. (Elias Diem)
Files: src/normal.c

Patch 7.3.527

Problem: Clang complains about non-ASCII characters in a string.
Solution: Change to \x88 form. (Dominique Pelle)
Files: src/charset.c

Patch 7.3.528

Problem: Crash when closing last window in a tab. (Alex Efros)
Solution: Use common code in close_last_window_tabpage(). (Christian Brabandt)
Files: src/window.c

Patch 7.3.529

Problem: Using a count before "v" and "V" does not work (Kikyous)
Solution: Make the count select that many characters or lines. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.530 (after 7.3.520)

Problem: gvim does not work when **'guioptions'** includes "f". (Davido)
Solution: Call gui_mch_init_check() when running GUI in the foreground. (Yasuhiro Matsumoto)
Files: src/gui.c

Patch 7.3.531 (after 7.3.530)

Problem: GUI does not work on MS-Windows.
Solution: Add the missing #ifdef. (Patrick Avery)
Files: src/gui.c

Patch 7.3.532

Problem: Compiler warning from Clang.
Solution: Use a different way to point inside a string. (Dominique Pelle)
Files: src/syntax.c

Patch 7.3.533

Problem: Memory leak when writing undo file.
Solution: Free the ACL. (Dominique Pelle)
Files: src/undo.c

Patch 7.3.534 (after 7.3.461)

Problem: When using an InsertCharPre autocommand autoindent fails.

Solution: Proper handling of v:char. (Alexey Radkov)

Files: src/edit.c

Patch 7.3.535

Problem: Many #ifdefs for MB_MAXBYTES.

Solution: Also define MB_MAXBYTES without the +multi_byte feature. Fix places where the buffer didn't include space for a NUL byte.

Files: src/arabic.c, src/edit.c, src/eval.c, src/getchar.c, src/mbyte.c, src/misc1.c, src/screen.c, src/spell.c, src/vim.h

Patch 7.3.536

Problem: When spell checking the German sharp s is not seen as a word character. (Aexl Bender)

Solution: In utf_islower() return true for the sharp s. **Note:** also need updated spell file for this to take effect.

Files: src/mbyte.c

Patch 7.3.537

Problem: Unnecessary call to init_spell_chartab().

Solution: Delete the call.

Files: src/spell.c

Patch 7.3.538

Problem: **'efm'** does not handle Tabs in pointer lines.

Solution: Add Tab support. Improve tests. (Lech Lorens)

Files: src/quickfix.c, src/testdir/test10.in, src/testdir/test10.ok

Patch 7.3.539

Problem: Redrawing a character on the command line does not work properly for multibyte characters.

Solution: Count the number of bytes in a character. (Yukihiro Nakadaira)

Files: src/ex_getln.c

Patch 7.3.540

Problem: Cursor is left on the text instead of the command line.

Solution: Don't call setcursor() in command line mode.

Files: src/getchar.c

Patch 7.3.541

Problem: When joining lines comment leaders need to be removed manually.

Solution: Add the 'j' flag to **'formatoptions'**. (Lech Lorens)

Files: runtime/doc/change.txt, src/edit.c, src/ex_docmd.c, src/misc1.c, src/normal.c, src/ops.c, src/option.h, src/proto/misc1.pro, src/proto/ops.pro, src/search.c, src/testdir/test29.in, src/testdir/test29.ok

Patch 7.3.542 (after 7.3.506)

Problem: Function is sometimes unused.

Solution: Add #ifdef.

Files: src/gui_gtk.c

Patch 7.3.543

Problem: The cursor is in the wrong line after using ":copen". (John Beckett)
Solution: Invoke more drastic redraw method.
Files: src/eval.c

Patch 7.3.544

Problem: There is no good way to close a quickfix window when closing the last ordinary window.
Solution: Add the QuitPre autocommand.
Files: src/ex_docmd.c, src/fileio.c, src/vim.h

Patch 7.3.545

Problem: When closing a window or buffer autocommands may close it too, causing problems for where the autocommand was invoked from.
Solution: Add the w_closing and b_closing flags. When set disallow ":q" and ":close" to prevent recursive closing.
Files: src/structs.h, src/buffer.c, src/ex_docmd.c, src/window.c

Patch 7.3.546

Problem: Bogus line break.
Solution: Remove the line break.
Files: src/screen.c

Patch 7.3.547 (after 7.3.541)

Problem: Compiler warning for uninitialized variable.
Solution: Initialize it.
Files: src/ops.c

Patch 7.3.548

Problem: Compiler warning on 64 bit Windows.
Solution: Add type cast. (Mike Williams)
Files: src/ops.c

Patch 7.3.549

Problem: In '**cinoptions**' "0s" is interpreted as one shiftwidth. (David Pineau)
Solution: Use the zero as zero. (Lech Lorens)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.550 (after 7.3.541)

Problem: With "j" in '**formatoptions**' a list leader is not removed. (Gary Johnson)
Solution: Don't ignore the start of a three part comment. (Lech Lorens)
Files: src/ops.c, src/testdir/test29.in, src/testdir/test29.ok

Patch 7.3.551

Problem: When using :tablose a TabEnter autocommand is triggered too early. (Karthick)
Solution: Don't trigger *Enter autocommands before closing the tab. (Christian Brabandt)
Files: src/buffer.c, src/eval.c, src/ex_cmds2.c, src/fileio.c, src/proto/window.pro, src/window.c

Patch 7.3.552

Problem: Formatting inside comments does not use the "2" flag in **'formatoptions'**.

Solution: Support the "2" flag. (Tor Perkins)

Files: src/vim.h, src/ops.c, src/edit.c, src/misc1.c,
src/testdir/test68.in, src/testdir/test68.ok

Patch 7.3.553

Problem: With double-width characters and **'listchars'** containing "precedes" the text is displayed one cell off.

Solution: Check for double-width character being overwritten by the "precedes" character. (Yasuhiro Matsumoto)

Files: src/screen.c

Patch 7.3.554 (after 7.3.551)

Problem: Compiler warning for unused argument.

Solution: Add UNUSED.

Files: src/window.c

Patch 7.3.555

Problem: Building on IBM z/OS fails.

Solution: Adjust configure. Use the QUOTESED value from config.mk instead of the hard coded one in Makefile. (Stephen Bovy)

Files: src/configure.in, src/auto/configure, src/Makefile

Patch 7.3.556

Problem: Compiler warnings on 64 bit Windows.

Solution: Add type casts. (Mike Williams)

Files: src/misc1.c

Patch 7.3.557

Problem: Crash when an autocommand wipes out a buffer when it is hidden.

Solution: Restore the current window when needed. (Christian Brabandt)

Files: src/buffer.c

Patch 7.3.558

Problem: Memory access error. (Gary Johnson)

Solution: Allocate one more byte. (Dominique Pelle)

Files: src/misc1.c

Patch 7.3.559

Problem: home_replace() does not work with 8.3 filename.

Solution: Make ":p" expand 8.3 name to full path. (Yasuhiro Matsumoto)

Files: src/eval.c, src/misc1.c

Patch 7.3.560

Problem: Get an error for a locked argument in extend().

Solution: Initialize the lock flag for a dictionary. (Yukihiro Nakadaira)

Files: src/eval.c

Patch 7.3.561

Problem: Using refresh: always in a complete function breaks the "." command. (Val Markovic)

Solution: Add match leader to the redo buffer. (Yasuhiro Matsumoto)

Files: src/edit.c

Patch 7.3.562

Problem: ":profdel" should not work when the +profile feature is disabled.
Solution: Call ex_ni(). (Yasuhiro Matsumoto)
Files: src/ex_cmds2.c

Patch 7.3.563 (after 7.3.557)

Problem: Can't build with tiny features.
Solution: Add #ifdef.
Files: src/buffer.c

Patch 7.3.564 (after 7.3.559)

Problem: Warning for pointer conversion.
Solution: Add type cast.
Files: src/misc1.c

Patch 7.3.565

Problem: Can't generate proto file for Python 3.
Solution: Add PYTHON3_CFLAGS to LINT_CFLAGS.
Files: src/Makefile

Patch 7.3.566 (after 7.3.561)

Problem: Redo after completion does not work correctly when refresh: always is not used. (Raymond Ko)
Solution: Check the compl_opt_refresh_always flag. (Christian Brabandt)
Files: src/edit.c

Patch 7.3.567

Problem: Missing copyright notice.
Solution: Add Vim copyright notice. (Taro Muraoka)
Files: src/dehqx.py

Patch 7.3.568

Problem: Bad indents for #ifdefs.
Solution: Add and remove spaces. (Elias Diem)
Files: src/globals.h

Patch 7.3.569

Problem: Evaluating Vim expression in Python is insufficient.
Solution: Add vim.bindeval(). Also add pyeval() and py3eval(). (ZyX)
Files: runtime/doc/eval.txt, runtime/doc/if_pyth.txt, src/eval.c, src/if_lua.c, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/proto/eval.pro, src/proto/if_python.pro, src/proto/if_python3.pro, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Makefile, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.570

Problem: ":vimgrep" does not obey 'wildignore'.
Solution: Apply 'wildignore' and 'suffixes' to ":vimgrep". (Ingo Karkat)
Files: src/ex_cmds2.c, src/proto/ex_cmds2.pro, src/quickfix.c, src/spell.c

Patch 7.3.571

Problem: Duplicated condition.
Solution: Remove one. (Dominique Pelle)
Files: src/os_win32.c

Patch 7.3.572

Problem: Duplicate statement in if and else. (Dominique Pelle)
Solution: Remove the condition and add a TODO.
Files: src/gui_xmebw.c

Patch 7.3.573

Problem: Using array index before bounds checking.
Solution: Swap the parts of the condition. (Dominique Pelle)
Files: src/ops.c

Patch 7.3.574

Problem: When pasting a register in the search command line a **CTRL-L** character is not pasted. (Dominique Pelle)
Solution: Escape the **CTRL-L**. (Christian Brabandt)
Files: src/ex_getln.c

Patch 7.3.575

Problem: "ygt" tries to yank instead of giving an error. (Daniel Mueller)
Solution: Check for a pending operator.
Files: src/normal.c

Patch 7.3.576

Problem: Formatting of lists inside comments is not right yet.
Solution: Use another solution and add a test. (Tor Perkins)
Files: src/edit.c, src/misc1.c, src/testdir/test68.in,
src/testdir/test69.ok

Patch 7.3.577

Problem: Size of memory does not fit in 32 bit unsigned.
Solution: Use Kbyte instead of byte. Call GlobalMemoryStatusEx() instead of GlobalMemoryStatus() when available.
Files: src/misc2.c, src/option.c, src/os_amiga.c, src/os_msdos.c,
src/os_win16.c, src/os_win32.c

Patch 7.3.578

Problem: Misplaced declaration.
Solution: Move declaration to start of block.
Files: src/if_py_both.h

Patch 7.3.579 (after 7.3.569)

Problem: Can't compile with Python 2.5.
Solution: Use PyCObject when Capsules are not available.
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.580

Problem: Warning on 64 bit MS-Windows.
Solution: Add type cast. (Mike Williams)
Files: src/if_py_both.h

Patch 7.3.581

Problem: Problems compiling with Python.
Solution: Pick UCS2 or UCS4 function at runtime. (lilydjwg)
Files: src/if_python.c

Patch 7.3.582 (after 7.3.576)

Problem: Missing parts of the test OK file.
Solution: Add the missing parts.
Files: src/testdir/test68.ok

Patch 7.3.583

Problem: PyObject_NextNotImplemented is not defined before Python 2.7.
(Danek Duvall)
Solution: Add #ifdefs.
Files: src/if_python.c

Patch 7.3.584

Problem: PyCObject is not always defined.
Solution: Use PyObject instead.
Files: src/if_py_both.h, src/if_python.c

Patch 7.3.585

Problem: Calling changed_bytes() too often.
Solution: Move changed_bytes() out of a loop. (Tor Perkins)
Files: src/edit.c

Patch 7.3.586

Problem: When compiling with Cygwin or MingW MEMORYSTATUSEX is not defined.
Solution: Set the default for WINVER to 0x0500.
Files: src/Make_ming.mak, src/Make_cyg.mak

Patch 7.3.587

Problem: Compiler warning for local var shadowing global var.
Solution: Rename the var and move it to an inner block. (Christian Brabandt)
Files: src/buffer.c

Patch 7.3.588

Problem: Crash on NULL pointer.
Solution: Fix the immediate problem by checking for NULL. (Lech Lorens)
Files: src/window.c

Patch 7.3.589

Problem: Crash when \$HOME is not set.
Solution: Check for a NULL pointer. (Chris Webb)
Files: src/misc1.c

Patch 7.3.590

Problem: The '<' and '>' marks cannot be set directly.
Solution: Allow setting '<' and '>'. (Christian Brabandt)
Files: src/mark.c

Patch 7.3.591

Problem: Can only move to a tab by absolute number.
Solution: Move a number of tabs to the left or the right. (Lech Lorens)

Files: runtime/doc/tabpage.txt, src/ex_cmds.h, src/ex_docmd.c,
src/testdir/test62.in, src/testdir/test62.ok, src/window.c

Patch 7.3.592

Problem: Vim on GTK does not support g:browsefilter.
Solution: Add a GtkFileFilter to the file chooser. (Christian Brabandt)
Files: src/gui_gtk.c

Patch 7.3.593

Problem: No easy way to decide if b:browsefilter will work.
Solution: Add the browsefilter feature.
Files: src/gui_gtk.c, src/eval.c, src/vim.h

Patch 7.3.594

Problem: The X command server doesn't work perfectly. It sends an empty
reply for as-keys requests.
Solution: Remove duplicate ga_init2(). Do not send a reply for as-keys
requests. (Brian Burns)
Files: src/if_xcmdsrv.c

Patch 7.3.595

Problem: The X command server responds slowly
Solution: Change the loop that waits for replies. (Brian Burns)
Files: src/if_xcmdsrv.c

Patch 7.3.596

Problem: Can't remove all signs for a file or buffer.
Solution: Support "*" for the sign id. (Christian Brabandt)
Files: runtime/doc/sign.txt, src/buffer.c, src/ex_cmds.c,
src/proto/buffer.pro

Patch 7.3.597

Problem: **'clipboard'** "autoselect" only applies to the * register. (Sergey
Vakulenko)
Solution: Make **'autoselect'** work for the + register. (Christian Brabandt)
Add the "autoselectplus" option in **'clipboard'** and the "P" flag in
'guioptions'.
Files: runtime/doc/options.txt, src/normal.c, src/ops.c, src/screen.c,
src/ui.c, src/globals.h, src/proto/ui.pro, src/option.h, src/gui.c

Patch 7.3.598

Problem: Cannot act upon end of completion. (Taro Muraoka)
Solution: Add an autocommand event that is triggered when completion has
finished. (Idea by Florian Klein)
Files: src/edit.c, src/fileio.c, src/vim.h

Patch 7.3.599 (after 7.3.597)

Problem: Missing change in one file.
Solution: Patch for changed clip_autoselect().
Files: src/option.c

Patch 7.3.600

Problem: **<f-args>** is not expanded properly with DBCS encoding.
Solution: Skip over character instead of byte. (Yukihiro Nakadaira)

Files: src/ex_docmd.c

Patch 7.3.601

Problem: Bad code style.

Solution: Insert space, remove parens.

Files: src/farsi.c

Patch 7.3.602

Problem: Missing files in distribution.

Solution: Update the list of files.

Files: Filelist

Patch 7.3.603

Problem: It is possible to add replace builtin functions by calling extend() on g:.

Solution: Add a flag to a dict to indicate it is a scope. Check for existing functions. (ZyX)

Files: src/buffer.c, src/eval.c, src/proto/eval.pro, src/structs.h, src/testdir/test34.in, src/testdir/test34.ok, src/window.c

Patch 7.3.604

Problem: inputdialog() doesn't use the cancel argument in the console. (David Fishburn)

Solution: Use the third argument. (Christian Brabandt)

Files: src/eval.c

Patch 7.3.605 (after 7.3.577)

Problem: MS-Windows: Can't compile with older compilers. (Titov Anatoly)

Solution: Add #ifdef for MEMORYSTATUSEX.

Files: src/os_win32.c

Patch 7.3.606

Problem: **CTRL-P** completion has a problem with multibyte characters.

Solution: Check for next character being NUL properly. (Yasuhiro Matsumoto)

Files: src/search.c, src/macros.h

Patch 7.3.607

Problem: With an 8 color terminal the selected menu item is black on black, because darkGrey as bg is the same as black.

Solution: Swap fg and bg colors. (James McCoy)

Files: src/syntax.c

Patch 7.3.608

Problem: winrestview() does not always restore the view correctly.

Solution: Call win_new_height() and win_new_width(). (Lech Lorens)

Files: src/eval.c, src/proto/window.pro, src/window.c

Patch 7.3.609

Problem: File names in :checkpath! output are garbled.

Solution: Check for \zs in the pattern. (Lech Lorens)

Files: src/search.c, src/testdir/test17.in, src/testdir/test17.ok

Patch 7.3.610

Problem: Cannot operate on the text that a search pattern matches.

Solution: Add the "gn" and "gN" commands. (Christian Brabandt)
Files: runtime/doc/index.txt, runtime/doc/visual.txt, src/normal.c,
src/proto/search.pro, src/search.c, src/testdir/test53.in,
src/testdir/test53.ok

Patch 7.3.611

Problem: Can't use Vim dictionary as self argument in Python.
Solution: Fix the check for the "self" argument. (ZyX)
Files: src/if_py_both.h

Patch 7.3.612

Problem: Auto formatting messes up text when 'fo' contains "2". (ZyX)
Solution: Decrement "less_cols". (Tor Perkins)
Files: src/misc1.c, src/testdir/test68.in, src/testdir/test68.ok

Patch 7.3.613

Problem: Including Python's config.c in the build causes trouble. It is not clear why it was there.
Solution: Omit the config file. (James McCoy)
Files: src/Makefile, src/auto/configure, src/configure.in

Patch 7.3.614

Problem: Number argument gets turned into a number while it should be a string.
Solution: Add flag to the call_vim_function() call. (Yasuhiro Matsumoto)
Files: src/edit.c, src/eval.c, src/proto/eval.pro

Patch 7.3.615

Problem: Completion for a user command does not recognize backslash before a space.
Solution: Recognize escaped characters. (Yasuhiro Matsumoto)
Files: src/ex_docmd.c

Patch 7.3.616 (after 7.3.610)

Problem: Can't compile without +visual.
Solution: Add #ifdef.
Files: src/normal.c

Patch 7.3.617 (after 7.3.615)

Problem: Hang on completion.
Solution: Skip over the space. (Yasuhiro Matsumoto)
Files: src/ex_docmd.c

Patch 7.3.618 (after 7.3.616)

Problem: Still doesn't compile with small features.
Solution: Move current_search() out of #ifdef. (Dominique Pelle)
Files: src/normal.c, src/search.c

Patch 7.3.619

Problem: When executing a shell command Vim may become slow to respond.
Solution: Don't wait after every processed message. (idea by Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.3.620

Problem: Building with recent Ruby on Win32 doesn't work.
Solution: Add a separate argument for the API version. (Yasuhiro Matsumoto)
Files: src/Make_ming.mak, src/Make_mvc.mak

Patch 7.3.621

Problem: Compiler warnings on 64 bit windows.
Solution: Add type casts. (Mike Williams)
Files: src/ex_docmd.c, src/search.c

Patch 7.3.622

Problem: XPM library for Win32 can't be found.
Solution: Suggest using the one from the Vim ftp site.
Files: src/Make_mvc.mak

Patch 7.3.623

Problem: Perl 5.14 commands crash Vim on MS-Windows.
Solution: Use perl_get_sv() instead of GvSV(). (Raymond Ko)
Files: src/if_perl.xs

Patch 7.3.624

Problem: When cancelling input() it returns the third argument. That should only happen for inputdialog().
Solution: Check if inputdialog() was used. (Hirohito Higashi)
Files: src/eval.c

Patch 7.3.625

Problem: "gn" does not handle zero-width matches correctly.
Solution: Handle zero-width patterns specially. (Christian Brabandt)
Files: src/search.c

Patch 7.3.626

Problem: Python interface doesn't build with Python 2.4 or older.
Solution: Define Py_ssize_t. (Benjamin Bannier)
Files: src/if_py_both.h

Patch 7.3.627

Problem: When using the "n" flag with the ":s" command a \= substitution will not be evaluated.
Solution: Do perform the evaluation, so that a function can be invoked at every matching position without changing the text. (Christian Brabandt)
Files: src/ex_cmds.c

Patch 7.3.628

Problem: ":open" does not allow for a !, which results in a confusing error message. (Shawn Wilson)
Solution: Allow ! on ":open". (Christian Brabandt)
Files: src/ex_cmds.h

Patch 7.3.629

Problem: There is no way to make 'shiftwidth' follow 'tabstop'.
Solution: When 'shiftwidth' is zero use the value of 'tabstop'. (Christian Brabandt)

Files: src/edit.c, src/ex_getln.c, src/fold.c, src/misc1.c, src/ops.c,
src/option.c, src/proto/option.pro

Patch 7.3.630

Problem: "|" does not behave correctly when '**virtualedit**' is set.
Solution: Call validate_virtcol(). (David Bürgin)
Files: src/normal.c

Patch 7.3.631

Problem: Cannot complete user names.
Solution: Add user name completion. (Dominique Pelle)
Files: runtime/doc/map.txt, src/auto/configure, src/config.h.in,
src/configure.in, src/ex_docmd.c, src/ex_getln.c, src/misc1.c,
src/misc2.c, src/proto/misc1.pro, src/vim.h

Patch 7.3.632

Problem: Cannot select beyond 222 columns with the mouse in xterm.
Solution: Add support for SGR mouse tracking. (Hayaki Saito)
Files: runtime/doc/options.txt, src/feature.h, src/keymap.h, src/misc2.c,
src/option.h, src/os_unix.c, src/term.c, src/version.c

Patch 7.3.633

Problem: Selection remains displayed as selected after selecting another
text.
Solution: Call xterm_update() before select(). (Andrew Pimlott)
Files: src/os_unix.c

Patch 7.3.634

Problem: Month/Day format for undo is confusing. (Marcin Szamotulski)
Solution: Always use Year/Month/Day, should work for everybody.
Files: src/undo.c

Patch 7.3.635

Problem: Issue 21: System call during startup sets '**lines**' to a wrong
value. (Karl Yngve)
Solution: Don't set the shell size while the GUI is still starting up.
(Christian Brabandt)
Files: src/ui.c

Patch 7.3.636 (after 7.3.625)

Problem: Not all zero-width matches handled correctly for "gn".
Solution: Move zero-width detection to a separate function. (Christian
Brabandt)
Files: src/search.c

Patch 7.3.637

Problem: Cannot catch the error caused by a foldopen when there is no fold.
(ZyX, Issue 48)
Solution: Do not break out of the loop early when inside try/catch.
(Christian Brabandt) Except when there is a syntax error.
Files: src/ex_docmd.c, src/globals.h

Patch 7.3.638

Problem: Unnecessary redraw of the previous character.

Solution: Check if the character is double-width. (Jon Long)
Files: src/screen.c

Patch 7.3.639

Problem: It's not easy to build Vim on Windows with XPM support.
Solution: Include the required files, they are quite small. Update the MSVC makefile to use them. Binary files are in the next patch. (Sergey Khorev)
Files: src/xpm/COPYRIGHT, src/xpm/README.txt, src/xpm/include/simx.h, src/xpm/include/xpm.h, src/Make_mvc.mak, src/bigvim.bat, src/bigvim64.bat, Filelist

Patch 7.3.640

Problem: It's not easy to build Vim on Windows with XPM support.
Solution: Binary files for 7.3.639. (Sergey Khorev)
Files: src/xpm/x64/lib/libXpm.lib, src/xpm/x86/lib/libXpm.a, src/xpm/x86/lib/libXpm.lib

Patch 7.3.641

Problem: ":mkview" uses ":normal" instead of ":normal!" for folds. (Dan)
Solution: Add the bang. (Christian Brabandt)
Files: src/fold.c

Patch 7.3.642

Problem: Segfault with specific autocommands. Was OK after 7.3.449 and before 7.3.545. (Richard Brown)
Solution: Pass TRUE for abort_if_last in the call to close_buffer(). (Christian Brabandt)
Files: src/window.c

Patch 7.3.643 (after 7.3.635)

Problem: MS-Windows: When starting gvim maximized '**lines**' and '**columns**' are wrong. (Christian Robinson)
Solution: Move the check for gui.starting from ui_get_shellsize() to check_shellsize().
Files: src/ui.c, src/term.c

Patch 7.3.644

Problem: Dead code for BeOS GUI.
Solution: Remove unused __BEOS__ stuff.
Files: src/gui.c

Patch 7.3.645

Problem: No tests for patch 7.3.625 and 7.3.637.
Solution: Add more tests for the "gn" command and try/catch. (Christian Brabandt)
Files: src/testdir/test53.in, src/testdir/test53.ok, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.3.646

Problem: When reloading a buffer the undo file becomes unusable unless ":w" is executed. (Dmitri Frank)
Solution: After reloading the buffer write the undo file. (Christian Brabandt)

Files: src/fileio.c

Patch 7.3.647

Problem: "gnd" doesn't work correctly in Visual mode.

Solution: Handle Visual mode differently in "gn". (Christian Brabandt)

Files: src/search.c, src/testdir/test53.in, src/testdir/test53.ok

Patch 7.3.648

Problem: Crash when using a very long file name. (ZyX)

Solution: Properly check length of buffer space.

Files: src/buffer.c

Patch 7.3.649

Problem: When '**clipboard**' is set to "unnamed" small deletes end up in the numbered registers. (Ingo Karkat)

Solution: Use the original register name to decide whether to put a delete in a numbered register. (Christian Brabandt)

Files: src/ops.c

Patch 7.3.650

Problem: Completion after ":help \{-" gives an error message and messes up the command line.

Solution: Cancel the tag search if the pattern can't be compiled. (Yasuhiro Matsumoto)

Files: src/tag.c

Patch 7.3.651

Problem: Completion after ":help \{-" gives an error message.

Solution: Prepend a backslash.

Files: src/ex_cmds.c

Patch 7.3.652

Problem: Workaround for Python crash isn't perfect.

Solution: Change the type of the length argument. (Sean Estabrooks)

Files: src/if_py_both.h

Patch 7.3.653

Problem: MingW needs build rule for included XPM files. Object directory for 32 and 64 builds is the same, also for MSVC.

Solution: Add MingW build rule to use included XPM files. Add the CPU or architecture to the object directory name. (Sergey Khorev)

Files: src/Make_ming.mak, src/Make_mvc.mak, src/xpm/README.txt

Patch 7.3.654

Problem: When creating a Vim dictionary from Python objects an empty key might be used.

Solution: Do not use empty keys, throw an IndexError. (ZyX)

Files: src/if_py_both.h

Patch 7.3.655

Problem: 64 bit MingW xpm .a file is missing.

Solution: Add the file. (Sergey Khorev)

Files: src/xpm/x64/lib/libXpm.a

Patch 7.3.656

Problem: Internal error in :pyeval.
Solution: Handle failed object conversion. (ZyX)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.657

Problem: Python bindings silently truncate string values containing NUL.
Solution: Fail when a string contains NUL. (ZyX)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.658

Problem: NUL bytes truncate strings when converted from Python.
Solution: Handle truncation as an error. (ZyX)
Files: src/if_py_both.h, src/if_python3.c

Patch 7.3.659

Problem: Recent Python changes are not tested.
Solution: Add tests for Python bindings. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.660

Problem: ":help !" jumps to help for "!!".
Solution: Adjust check for tag header line. (Andy Wokula)
Files: src/tag.c

Patch 7.3.661 (after 7.3.652)

Problem: SEGV in Python code.
Solution: Initialize len to zero. Use the right function depending on version. (Maxim Philippov)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.662

Problem: Can't build Ruby interface with Ruby 1.9.3.
Solution: Add missing functions. (V. Ondruch)
Files: src/if_ruby.c

Patch 7.3.663

Problem: End of color scheme name not clear in E185. (Aaron Lewis)
Solution: Put the name in single quotes.
Files: src/ex_docmd.c

Patch 7.3.664

Problem: Buffer overflow in unescaping text. (Raymond Ko)
Solution: Limit check for multibyte character to 4 bytes.
Files: src/mbyte.c

Patch 7.3.665

Problem: MSVC 11 is not supported. (Raymond Ko)
Solution: Recognize MSVC 11. (Gary Willoughby)
Files: src/Make_mvc.mak

Patch 7.3.666

Problem: With MSVC 11 Win32.mak is not found.

Solution: Add the SDK_INCLUDE_DIR variable. (Raymond Ko)
Files: src/Make_mvc.mak

Patch 7.3.667

Problem: Unused variables in Perl interface.
Solution: Adjust #ifdefs.
Files: src/if_perl.xs

Patch 7.3.668

Problem: Building with Perl loaded dynamically still uses static library.
Solution: Adjust use of PL_thr_key. (Ken Takata)
Files: src/if_perl.xs

Patch 7.3.669

Problem: When building with Cygwin loading Python dynamically fails.
Solution: Use DLLLIBRARY instead of INSTSONAME. (Ken Takata)
Files: src/configure.in, src/auto/configure

Patch 7.3.670

Problem: Python: memory leaks when there are exceptions.
Solution: Add DICTKEY_UNREF in the right places. (ZyX)
Files: src/if_py_both.h

Patch 7.3.671

Problem: More Python code can be shared between Python 2 and 3.
Solution: Move code to if_py_both.h. (ZyX)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.672

Problem: Not possible to lock/unlock lists in Python interface.
Solution: Add .locked and .scope attributes. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/if_python.c,
src/if_python3.c, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.673

Problem: Using "gN" while '**selection**' is "exclusive" misses one character.
(Ben Fritz)
Solution: Check the direction when compensating for exclusive selection.
(Christian Brabandt)
Files: src/search.c

Patch 7.3.674

Problem: Can't compile with Lua/dyn on Cygwin.
Solution: Adjust configure to use the right library name. (Ken Takata)
Files: src/configure.in, src/auto/configure

Patch 7.3.675

Problem: Using uninitialized memory with very long file name.
Solution: Put NUL after text when it is truncated. (ZyX)
Files: src/buffer.c

Patch 7.3.676

Problem: Ruby compilation on Windows 32 bit doesn't work.

Solution: Only use some functions for 64 bit. (Ken Takata)
Files: src/if_ruby.c

Patch 7.3.677

Problem: buf_spname() is used inconsistently.
Solution: Make the return type a char_u pointer. Check the size of the returned string.
Files: src/buffer.c, src/proto/buffer.pro, src/ex_cmds2.c, src/ex_docmd.c, src/memline.c, src/screen.c

Patch 7.3.678

Problem: Ruby .so name may not be correct.
Solution: Use the LIBRUBY_S0 entry from the config. (Vit Ondruch)
Files: src/configure.in, src/auto/configure

Patch 7.3.679

Problem: Ruby detection uses Config, newer Ruby versions use RbConfig.
Solution: Detect the need to use RbConfig. (Vit Ondruch)
Files: src/configure.in, src/auto/configure

Patch 7.3.680

Problem: Some files missing in the list of distributed files.
Solution: Add lines for new files.
Files: Filelist

Patch 7.3.681 (after 7.3.680)

Problem: List of distributed files picks up backup files.
Solution: Make tutor patterns more specific.
Files: Filelist

Patch 7.3.682 (after 7.3.677)

Problem: Compiler complains about incompatible types.
Solution: Remove type casts. (hint by Danek Duvall)
Files: src/edit.c

Patch 7.3.683

Problem: ":python" may crash when vimbindeval() returns None.
Solution: Check for v_string to be NULL. (Yukihiro Nakadaira)
Files: src/if_py_both.h

Patch 7.3.684

Problem: "make test" does not delete lua.vim.
Solution: Add lua.vim to the clean target. (Simon Ruderich)
Files: src/testdir/Makefile, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms

Patch 7.3.685

Problem: No test for what patch 7.3.673 fixes.
Solution: Add a test. (Christian Brabandt)
Files: src/testdir/test53.in, src/testdir/test53.ok

Patch 7.3.686

Problem: Using CTRL-\ e mappings is useful also when entering an expression, but it doesn't work. (Marcin Szamotulski)

Solution: Allow using CTRL-\ e when entering an expression if it was not typed.
Files: src/ex_getln.c

Patch 7.3.687

Problem: Test 16 fails when \$DISPLAY is not set.
Solution: Skip the test when \$DISPLAY is not set.
Files: src/testdir/test16.in

Patch 7.3.688

Problem: Python 3.3 is not supported.
Solution: Add Python 3.3 support (Ken Takata)
Files: src/if_python3.c

Patch 7.3.689

Problem: MzScheme and Lua may use a NULL string.
Solution: Use an empty string instead of NULL. (Yukihiro Nakadaira)
Files: src/if_lua.c, src/if_mzsch.c

Patch 7.3.690

Problem: When the current directory name is exactly the maximum path length Vim may crash.
Solution: Only add "/" when there is room. (Danek Duvall)
Files: src/os_unix.c

Patch 7.3.691

Problem: State specific to the Python thread is discarded.
Solution: Keep state between threads. (Paul)
Files: src/if_python.c

Patch 7.3.692

Problem: Can't build GTK version with GTK 2.0.
Solution: Put GtkFileFilter declaration in the right place. (Yegappan Lakshmanan)
Files: src/gui_gtk.c

Patch 7.3.693

Problem: Can't make 'softtabstop' follow 'shiftwidth'.
Solution: When 'softtabstop' is negative use the value of 'shiftwidth'. (so8res)
Files: src/edit.c, src/option.c, src/proto/option.pro

Patch 7.3.694

Problem: Now that 'shiftwidth' may use the value of 'tabstop' it is not so easy to use in indent files.
Solution: Add the shiftwidth() function. (so8res)
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.3.695

Problem: Balloon cannot show multibyte text.
Solution: Properly deal with multibyte characters. (Dominique Pelle)
Files: src/gui_beval.c, src/ui.c

Patch 7.3.696

Problem: Message about added spell language can be wrong.
Solution: Give correct message. Add g:menutrans_set_lang_to to allow for translation. (Jiri Sedlak)
Files: runtime/menu.vim

Patch 7.3.697

Problem: Leaking resources when setting GUI font.
Solution: Free the font. (Ken Takata)
Files: src/syntax.c

Patch 7.3.698

Problem: Python 3 does not preserve state between commands.
Solution: Preserve the state. (Paul Ollis)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.699

Problem: When **'tymouse'** is set to "sgr" manually, it is overruled by automatic detection.
Solution: Do not use automatic detection when **'tymouse'** was set manually. (Hayaki Saito)
Files: src/term.c

Patch 7.3.700

Problem: Cannot detect URXVT and SGR mouse support.
Solution: add +mouse_urxvt and +mouse_sgr. (Hayaki Saito)
Files: src/feature.h, src/eval.c

Patch 7.3.701

Problem: MS-Windows: Crash with stack overflow when setting **'encoding'**.
Solution: Handle that loading the iconv library may be called recursively. (Jiri Sedlak)
Files: src/os_win32.c

Patch 7.3.702

Problem: Nmake from VS6 service pack 6 is not recognized.
Solution: Detect the version number. (Jiri Sedlak)
Files: src/Make_mvc.mak

Patch 7.3.703

Problem: When **'undofile'** is reset the hash is computed unnecessarily.
Solution: Only compute the hash when the option was set. (Christian Brabandt)
Files: src/option.c

Patch 7.3.704

Problem: Repeating "cgn" does not always work correctly.
Solution: Also fetch the operator character. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.705

Problem: Mouse features are not sorted properly. (Tony Mechelynck)
Solution: Put the mouse features in alphabetical order.
Files: src/version.c

Patch 7.3.706 (after 7.3.697)

Problem: Can't build Motif version.
Solution: Fix wrongly named variable. (Ike Devolder)
Files: src/syntax.c

Patch 7.3.707 (after 7.3.701)

Problem: Problems loading a library for a file name with non-latin characters.
Solution: Use wide system functions when possible. (Ken Takata)
Files: src/os_win32.c, src/os_win32.h

Patch 7.3.708

Problem: Filler lines above the first line may be hidden when opening Vim.
Solution: Change how topfill is computed. (Christian Brabandt)
Files: src/diff.c, src/testdir/test47.in, src/testdir/test47.ok

Patch 7.3.709

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/eval.c

Patch 7.3.710 (after 7.3.704)

Problem: Patch 7.3.704 breaks "fn".
Solution: Add check for ca.cmdchar. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.711 (after 7.3.688)

Problem: vim.current.buffer is not available. (lilydjwg)
Solution: Use py3_PyUnicode_AsUTF8 instead of py3_PyUnicode_AsUTF8String.
(Ken Takata)
Files: src/if_python3.c

Patch 7.3.712

Problem: Nmake from VS2010 SP1 is not recognized.
Solution: Add the version number. (Ken Takata)
Files: src/Make_mvc.mak

Patch 7.3.713

Problem: printf() can only align to bytes, not characters.
Solution: Add the "S" item. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/message.c

Patch 7.3.714

Problem: Inconsistency: :set can be used in the sandbox, but :setlocal and :setglobal cannot. (Michael Henry)
Solution: Fix the flags for :setlocal and :setglobal. (Christian Brabandt)
Files: src/ex_cmds.h

Patch 7.3.715

Problem: Crash when calling setloclist() in BufUnload autocmd. (Marcin Szamotulski)
Solution: Set w_llist to NULL when it was freed. Also add a test.
(Christian Brabandt)
Files: src/quickfix.c, src/testdir/test49.ok, src/testdir/test49.vim

Patch 7.3.716

Problem: Error on exit when using Python 3.
Solution: Remove PythonIO_Fini(). (Roland Puntaier)
Files: src/if_python3.c

Patch 7.3.717

Problem: When changing the font size, only MS-Windows limits the window size.
Solution: Also limit the window size on other systems. (Roland Puntaier)
Files: src/gui.c

Patch 7.3.718

Problem: When re-using the current buffer the buffer-local options stay.
Solution: Re-initialize the buffer-local options. (Christian Brabandt)
Files: src/buffer.c

Patch 7.3.719

Problem: Cannot run new version of cproto, it fails on missing include files.
Solution: Add lots of #ifndef PROTO
Files: src/os_amiga.c, src/os_amiga.h, src/gui_w16.c, src/gui_w48.c, src/gui_w32.c, src/vimio.h, src/os_msdos.c, src/os_msdos.h, src/os_win16.h, src/os_win16.c, src/os_win32.h, src/os_win32.c, src/os_mswin.c, src/gui_photon.c, src/os_unix.h, src/os_beos.c, src/os_beos.h

Patch 7.3.720

Problem: Proto files are outdated.
Solution: Update the newly generated proto files.
Files: src/proto/digraph.pro, src/proto/fold.pro, src/proto/misc1.pro, src/proto/move.pro, src/proto/screen.pro, src/proto/search.pro, src/proto/os_win32.pro, src/proto/os_mswin.pro, src/proto/os_beos.pro

Patch 7.3.721

Problem: Ruby interface defines local functions globally.
Solution: Make the functions static.
Files: src/if_ruby.c

Patch 7.3.722

Problem: Perl flags may contain "-g", which breaks "make proto".
Solution: Filter out the "-g" flag for cproto. (Ken Takata)
Files: src/Makefile

Patch 7.3.723

Problem: Various tiny problems.
Solution: Various tiny fixes.
Files: src/gui_mac.c, src/xpm_w32.c, src/netbeans.c, src/sha256.c, src/if_sniff.c, README.txt

Patch 7.3.724

Problem: Building with Ruby and Tcl on MS-Windows 64 bit does not work.
Solution: Remove Ruby and Tcl from the big MS-Windows build.
Files: src/bigvim64.bat

Patch 7.3.725

Problem: :aboveleft and :belowright have no effect on :copen.
Solution: Check for cmdmod.split. (Christian Brabandt)
Files: src/quickfix.c

Patch 7.3.726

Problem: Typos and duplicate info in README.
Solution: Fix the text.
Files: README.txt

Patch 7.3.727

Problem: Can't always find Win32.mak when building GvimExt.
Solution: Use same mechanism as in Make_mvc.mak. (Cade Foster)
Files: src/GvimExt/Makefile

Patch 7.3.728

Problem: Cannot compile with MzScheme interface on Ubuntu 12.10.
Solution: Find the collects directory under /usr/share.
Files: src/configure.in, src/auto/configure

Patch 7.3.729

Problem: Building with Ruby fails on some systems.
Solution: Remove "static" and add #ifndef PROTO. (Ken Takata)
Files: src/if_ruby.c

Patch 7.3.730

Problem: Crash in PHP file when using syntastic. (Ike Devolder)
Solution: Avoid using NULL pointer. (Christian Brabandt)
Files: src/quickfix.c

Patch 7.3.731

Problem: Py3Init_vim() is exported unnecessarily.
Solution: Make it static. (Ken Takata)
Files: src/if_python3.c

Patch 7.3.732

Problem: Compiler warnings for function arguments.
Solution: Use inteptr_t instead of long.
Files: src/if_mzsch.c, src/main.c

Patch 7.3.733

Problem: Tests fail when including MzScheme.
Solution: Change #ifdefs for vim_main2().
Files: src/main.c

Patch 7.3.734

Problem: Cannot put help files in a sub-directory.
Solution: Make :helptags work for sub-directories. (Charles Campbell)
Files: src/ex_cmds.c

Patch 7.3.735

Problem: Cannot build Ruby 1.9 with MingW or Cygwin.
Solution: Add another include directory. (Ken Takata)

Files: src/Make_cyg.mak, src/Make_ming.mak

Patch 7.3.736

Problem: File name completion in input() escapes white space. (Frederic Hardy)

Solution: Do not escape white space. (Christian Brabandt)

Files: src/ex_getln.c

Patch 7.3.737

Problem: When using do_cmdline() recursively did_endif is not reset, causing messages to be overwritten.

Solution: Reset did_endif. (Christian Brabandt)

Files: src/ex_docmd.c

Patch 7.3.738 (after 7.3.730)

Problem: Unused function argument.

Solution: Remove it. (Christian Brabandt)

Files: src/quickfix.c

Patch 7.3.739

Problem: Computing number of lines may have an integer overflow.

Solution: Check for MAXCOL explicitly. (Dominique Pelle)

Files: src/move.c

Patch 7.3.740

Problem: IOC tool complains about undefined behavior for int.

Solution: Change to unsigned int. (Dominique Pelle)

Files: src/hashtab.c, src/misc2.c

Patch 7.3.741 (after 7.3.737)

Problem: Tiny build fails.

Solution: Move #ifdef. (Ike Devolder)

Files: src/ex_docmd.c

Patch 7.3.742

Problem: Leaking memory when :vimgrep restores the directory.

Solution: Free the allocated memory. (Christian Brabandt)

Files: src/quickfix.c

Patch 7.3.743 (after 7.3.741)

Problem: Tiny build still fails.

Solution: Add #else in the right place.

Files: src/ex_docmd.c

Patch 7.3.744

Problem: 64 bit compiler warning.

Solution: Add type cast. (Mike Williams)

Files: src/ex_cmds.c

Patch 7.3.745

Problem: Automatically setting 'ttymouse' doesn't work.

Solution: Reset the "option was set" flag when using the default.

Files: src/option.c, src/proto/option.pro, src/term.c

Patch 7.3.746

Problem: Memory leaks when using location lists.
Solution: Set qf_title to something. (Christian Brabandt)
Files: src/eval.c, src/quickfix.c

Patch 7.3.747

Problem: When characters are concealed text aligned with tabs are no longer aligned, e.g. at ":help :index".
Solution: Compensate space for tabs for concealed characters. (Dominique Pelle)
Files: src/screen.c

Patch 7.3.748

Problem: Cannot properly test conceal mode.
Solution: Add the screencol() and screenrow() functions. Use them in test88. (Simon Ruderich)
Files: runtime/doc/eval.txt, src/eval.c, src/proto/screen.pro, src/screen.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test88.in, src/testdir/test88.ok,

Patch 7.3.749

Problem: Python interface doesn't build without the multibyte feature.
Solution: Add #ifdef. (Ken Takata)
Files: src/if_py_both.h

Patch 7.3.750

Problem: The justify macro does not always work correctly.
Solution: Fix off-by-one error (James McCoy)
Files: runtime/macros/justify.vim

Patch 7.3.751

Problem: Test 61 is flaky, it fails once in a while.
Solution: When it fails retry once.
Files: src/testdir/Makefile

Patch 7.3.752

Problem: Test 49 script file doesn't fold properly.
Solution: Add a colon.
Files: src/testdir/test49.vim

Patch 7.3.753

Problem: When there is a QuitPre autocommand using ":q" twice does not work for exiting when there are more files to edit.
Solution: Do not decrement quitmore in an autocommand. (Techlive Zheng)
Files: src/ex_docmd.c, src/fileio.c, src/proto/fileio.pro

Patch 7.3.754

Problem: Latest nmake is not recognized.
Solution: Add nmake version 11.00.51106.1. (Raymond Ko)
Files: src/Make_mvc.mak

Patch 7.3.755

Problem: Autoconf doesn't find Python 3 if it's called "python".
Solution: Search for "python2" and "python3" first, then "python".
Files: src/configure.in, src/auto/configure

Patch 7.3.756

Problem: A location list can get a wrong count in :lvimgrep.
Solution: Check if the list was changed by autocommands. (mostly by Christian Brabandt)
Files: src/quickfix.c

Patch 7.3.757

Problem: Issue 96: May access freed memory when a put command triggers autocommands. (Dominique Pelle)
Solution: Call u_save() before getting y_array.
Files: src/ops.c

Patch 7.3.758

Problem: Matchit plugin does not handle space in #ifdef.
Solution: Change matching pattern to allow spaces. (Mike Morearty)
Files: runtime/macros/matchit.vim

Patch 7.3.759

Problem: MS-Windows: Updating the tabline is slow when there are many tabs.
Solution: Disable redrawing while performing the update. (Arseny Kapoulkine)
Files: src/gui_w48.c

Patch 7.3.760

Problem: dv_ deletes the white space before the line.
Solution: Move the cursor to the first non-white. (Christian Brabandt)
Files: src/normal.c, src/testdir/test19.in, src/testdir/test19.ok

Patch 7.3.761

Problem: In Visual mode a "-p does not work. (Marcin Szamotulski)
Solution: Avoid writing to "-" before putting it. (Christian Brabandt)
Files: src/normal.c, src/testdir/test48.in, src/testdir/test48.ok

Patch 7.3.762 (after 7.3.759)

Problem: On some systems the tabline is not redrawn.
Solution: Call RedrawWindow(). (Charles Peacech)
Files: src/gui_w48.c

Patch 7.3.763

Problem: Jumping to a mark does not open a fold if it is in the same line. (Wiktor Ruben)
Solution: Also compare the column after the jump. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.764

Problem: Not all message translation files are installed.
Solution: Also install the converted files.
Files: src/po/Makefile

Patch 7.3.765

Problem: Segfault when doing "cclose" on BufUnload in a python function.
(Sean Reifschneider)
Solution: Skip window with NULL buffer. (Christian Brabandt)
Files: src/main.c, src/window.c

Patch 7.3.766

Problem: ":help cpo-*" jumps to the wrong place.
Solution: Make it equivalent to ":help cpo-star".
Files: src/ex_cmds.c

Patch 7.3.767

Problem: (Win32) The _errno used for iconv may be the wrong one.
Solution: Use the _errno from iconv.dll. (Ken Takata)
Files: src/mbyte.c

Patch 7.3.768

Problem: settabvar() and setwinvar() may move the cursor.
Solution: Save and restore the cursor position when appropriate. (idea by
Yasuhiro Matsumoto)
Files: src/edit.c

Patch 7.3.769

Problem: **'matchpairs'** does not work with multibyte characters.
Solution: Make it work. (Christian Brabandt)
Files: src/misc1.c, src/option.c, src/proto/option.pro, src/search.c,
src/testdir/test69.in, src/testdir/test69.ok

Patch 7.3.770

Problem: Vim.h indentation is inconsistent.
Solution: Adjust the indentation. (Elias Diem)
Files: src/vim.h

Patch 7.3.771 (after 7.3.769)

Problem: Uninitialized variable. (Yasuhiro Matsumoto)
Solution: Set x2 to -1.
Files: src/option.c

Patch 7.3.772

Problem: Cursor is at the wrong location and below the end of the file
after doing substitutions with confirm flag: %s/x/y/c
(Dominique Pelle)
Solution: Update the cursor position. (Christian Brabandt & Dominique)
Files: src/ex_cmds.c

Patch 7.3.773 (after 7.3.767)

Problem: Crash when OriginalFirstThunk is zero.
Solution: Skip items with OriginalFirstThunk not set. (Ken Takata)
Files: src/mbyte.c

Patch 7.3.774

Problem: Tiny GUI version misses console dialog feature.
Solution: Define FEAT_CON_DIALOG when appropriate. (Christian Brabandt)
Files: src/feature.h, src/gui.h

Patch 7.3.775

Problem: Cygwin and Mingw builds miss dependency on gui_w48.c.
Solution: Add a build rule. (Ken Takata)
Files: src/Make_cyg.mak, src/Make_ming.mak

Patch 7.3.776

Problem: ml_get error when searching, caused by curwin not matching curbuf.
Solution: Avoid changing curbuf. (Lech Lorens)
Files: src/charset.c, src/eval.c, src/mark.c, src/proto/charset.pro,
src/proto/mark.pro, src/regexp.c, src/syntax.c,

Patch 7.3.777

Problem: When building with Gnome locale gets reset.
Solution: Set locale after gnome_program_init(). (Christian Brabandt)
Files: src/gui_gtk_x11.c

Patch 7.3.778

Problem: Compiler error for adding up two pointers. (Titov Anatoly)
Solution: Add a type cast. (Ken Takata)
Files: src/mbyte.c

Patch 7.3.779

Problem: Backwards search lands in wrong place when started on a multibyte character.
Solution: Do not set extra_col for a backwards search. (Sung Pae)
Files: src/search.c, src/testdir/test44.in, src/testdir/test44.ok

Patch 7.3.780

Problem: char2nr() and nr2char() always use **'encoding'**.
Solution: Add argument to use utf-8 characters. (Yasuhiro Matsumoto)
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.3.781

Problem: Drawing with **'guifontwide'** can be slow.
Solution: Draw multiple characters at a time. (Taro Muraoka)
Files: src/gui.c

Patch 7.3.782

Problem: Windows: IME composition may use a wrong font.
Solution: Use **'guifontwide'** for IME when it is set. (Taro Muraoka)
Files: runtime/doc/options.txt, src/gui.c, src/gui_w48.c,
src/proto/gui_w16.pro, src/proto/gui_w32.pro

Patch 7.3.783

Problem: Crash when mark is not set. (Dominique Pelle)
Solution: Check for NULL.
Files: src/normal.c

Patch 7.3.784 (after 7.3.781)

Problem: Error when **'guifontwide'** has a comma.
Solution: Use gui.wide_font. (Taro Muraoka)
Files: src/gui_w48.c

Patch 7.3.785 (after 7.3.776)

Problem: Crash with specific use of search pattern.
Solution: Initialize reg_buf to curbuf.
Files: src/regexp.c

Patch 7.3.786

Problem: Python threads don't run in the background (issue 103).
Solution: Move the statements to manipulate thread state.
Files: src/if_python.c

Patch 7.3.787

Problem: With '**relativenumber**' set it is not possible to see the absolute line number.
Solution: For the cursor line show the absolute line number instead of a zero. (Nazri Ramliy)
Files: src/screen.c

Patch 7.3.788

Problem: When only using patches build fails on missing nl.po.
Solution: Create an empty nl.po file.
Files: src/po/Makefile

Patch 7.3.789 (after 7.3.776)

Problem: "\k" in regexp does not work in other window.
Solution: Use the right buffer. (Yukihiro Nakadaira)
Files: src/mbyte.c, src/proto/mbyte.pro, src/regexp.c

Patch 7.3.790

Problem: After reloading a buffer the modelines are not processed.
Solution: call do_modelines(). (Ken Takata)
Files: src/fileio.c

Patch 7.3.791

Problem: MzScheme interface doesn't work properly.
Solution: Make it work better. (Sergey Khorev)
Files: runtime/doc/if_mzsch.txt, src/configure.in, src/auto/configure, src/eval.c, src/if_mzsch.c, src/if_mzsch.h, src/Make_ming.mak, src/Make_mvc.mak, src/os_unix.c, src/proto/eval.pro, src/testdir/test70.in, src/testdir/test70.ok

Patch 7.3.792

Problem: ":substitute" works differently without confirmation.
Solution: Do not change the text when asking for confirmation, only display it.
Files: src/ex_cmds.c

Patch 7.3.793 (after 7.3.792)

Problem: New interactive :substitute behavior is not tested.
Solution: Add tests. (Christian Brabandt)
Files: src/testdir/test80.in, src/testdir/test80.ok

Patch 7.3.794

Problem: Tiny build fails. (Tony Mechelynck)
Solution: Adjust #ifdefs.
Files: src/charset.c

Patch 7.3.795

Problem: MzScheme does not build with tiny features.
Solution: Add #ifdefs. Also add UNUSED to avoid warnings. And change library ordering.
Files: src/if_mzsch.c, src/Makefile

Patch 7.3.796

Problem: "[^\n]" does match at a line break.
Solution: Make it do the same as "/.". (Christian Brabandt)
Files: src/regexp.c, src/testdir/test79.in, src/testdir/test79.ok

Patch 7.3.797 (after 7.3.792)

Problem: Compiler warning for size_t to int conversion. (Skept)
Solution: Add type casts.
Files: src/ex_cmds.c

Patch 7.3.798 (after 7.3.791)

Problem: MzScheme: circular list does not work correctly.
Solution: Separate Mac-specific code from generic code. (Sergey Khorev)
Files: src/if_mzsch.c, src/testdir/test70.in

Patch 7.3.799

Problem: The color column is not correct when entering a buffer. (Ben Fritz)
Solution: Call check_colorcolumn() if 'textwidth' changed. (Christian Brabandt)
Files: src/buffer.c

Patch 7.3.800

Problem: The " mark is not adjusted when inserting lines. (Roland Eggner)
Solution: Adjust the line number. (Christian Brabandt)
Files: src/mark.c

Patch 7.3.801

Problem: ":window set nu?" displays the cursor line. (Nazri Ramliy)
Solution: Do not update the cursor line when conceallevel is zero or the screen has scrolled. (partly by Christian Brabandt)
Files: src/window.c

Patch 7.3.802

Problem: After setting 'isk' to a value ending in a comma appending to the option fails.
Solution: Disallow a trailing comma for 'isk' and similar options.
Files: src/charset.c

Patch 7.3.803 (after 7.3.792)

Problem: Substitute with confirmation and then "q" does not replace anything. (John McGowan)
Solution: Do not break the loop, skip to the end.
Files: src/ex_cmds.c, src/testdir/test80.in, src/testdir/test80.ok

Patch 7.3.804 (after 7.3.799)

Problem: Compiler warning for tiny build. (Tony Mechelynck)

Solution: Add #ifdefs around variable.
Files: src/buffer.c

Patch 7.3.805

Problem: Lua version 5.2 is not detected properly on Arch Linux.
Solution: Adjust autoconf. (lilydjwg)
Files: src/configure.in, src/auto/configure

Patch 7.3.806

Problem: Compiler warnings in Perl code when building with Visual Studio 2012. (skept)
Solution: Add type casts. (Christian Brabandt, 2013 Jan 30)
Files: src/if_perl.xs

Patch 7.3.807

Problem: Popup menu does not work properly with the preview window, folds and '**cursorcolumn**'.
Solution: Redraw the popup menu after redrawing windows. (Christian Brabandt)
Files: src/screen.c

Patch 7.3.808

Problem: Python threads still do not work properly.
Solution: Fix both Python 2 and 3. Add tests. (Ken Takata)
Files: src/if_python.c, src/if_python3.c, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.809

Problem: The dosinst.c program has a buffer overflow. (Thomas Gwae)
Solution: Ignore \$VIMRUNTIME if it is too long.
Files: src/dosinst.c

Patch 7.3.810

Problem: '**relativenumber**' is reset unexpectedly. (François Ingelrest)
Solution: After an option was reset also reset the global value. Add a test. (Christian Brabandt)
Files: src/option.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test89.in, src/testdir/test89.ok

Patch 7.3.811

Problem: Useless termresponse parsing for SGR mouse.
Solution: Skip the parsing. (Hayaki Saito)
Files: src/term.c

Patch 7.3.812

Problem: When '**indentexpr**' moves the cursor "curswant" not restored.
Solution: Restore "curswant". (Sung Pae)
Files: src/misc1.c

Patch 7.3.813

Problem: The CompleteDone event is not triggered when there are no pattern matches. (Jianjun Mao)
Solution: Trigger the event. (Christian Brabandt)
Files: src/edit.c

Patch 7.3.814

Problem: Can't input multibyte characters on Win32 console if 'encoding' is different from current codepage.
Solution: Use convert_input_safe() instead of convert_input(). Make string_convert_ext() return an error for incomplete input. (Ken Takata)
Files: src/mbyte.c, src/os_win32.c

Patch 7.3.815

Problem: Building with Cygwin and Ruby doesn't work.
Solution: Copy some things from the MingW build file. (Ken Takata)
Files: src/Make_cyg.mak

Patch 7.3.816

Problem: Can't compute a hash.
Solution: Add the sha256() function. (Tyru, Hirohito Higashi)
Files: runtime/doc/eval.txt, src/eval.c, src/proto/sha256.pro, src/sha256.c, src/testdir/test90.in, src/testdir/test90.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.3.817

Problem: Test 89 fails with tiny and small features.
Solution: Add sourcing small.vim.
Files: src/testdir/test89.in

Patch 7.3.818

Problem: When test 40 fails because of a bad build it may leave files behind that cause it to fail later.
Solution: Let the file names start with "X".
Files: src/testdir/test40.in

Patch 7.3.819

Problem: Compiling without +eval and with Python isn't working.
Solution: Add the eval feature when building with Python.
Files: src/if_py_both.h, src/feature.h, src/eval.c, src/ex_docmd.c, src/normal.c, src/ex_docmd.c, src/gui_gtk_x11.c

Patch 7.3.820

Problem: Build errors and warnings when building with small features and Lua, Perl or Ruby.
Solution: Add #ifdefs and UNUSED.
Files: src/if_perl.xs, src/if_lua.c, src/if_ruby.c

Patch 7.3.821

Problem: Build with OLE and Cygwin is broken. (Steve Hall)
Solution: Select static or shared stdc library. (Ken Takata)
Files: src/Make_cyg.mak

Patch 7.3.822 (after 7.3.799)

Problem: Crash when accessing freed buffer.

Solution: Get **'textwidth'** in caller of enter_buffer(). (Christian Brabandt)

Files: src/buffer.c

Patch 7.3.823 (after 7.3.821)

Problem: Building with Cygwin: '-lsupc++' is not needed.

Solution: Remove it. (Ken Takata)

Files: src/Make_cyg.mak

Patch 7.3.824

Problem: Can redefine builtin functions. (ZyX)

Solution: Disallow adding a function to g:.

Files: src/eval.c

Patch 7.3.825

Problem: With Python errors are not always clear.

Solution: Print the stack trace, unless :silent is used. (ZyX)

Files: src/if_python3.c, src/if_python.c

Patch 7.3.826

Problem: List of features in :version output is hard to read.

Solution: Make columns. (Nazri Ramliy)

Files: src/version.c

Patch 7.3.827 (after 7.3.825)

Problem: Python tests fail.

Solution: Adjust the output for the stack trace.

Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.ok

Patch 7.3.828

Problem: Mappings are not aware of wildmenu mode.

Solution: Add wildmenumode(). (Christian Brabandt)

Files: src/eval.c, runtime/doc/eval.txt

Patch 7.3.829

Problem: When compiled with the +rightleft feature **'showmatch'** also shows a match for the opening paren. When **'revins'** is set the screen may scroll.

Solution: Only check the opening paren when the +rightleft feature was enabled. Do not show a match that is not visible. (partly by Christian Brabandt)

Files: src/search.c

Patch 7.3.830

Problem: :mksession confuses bytes, columns and characters when positioning the cursor.

Solution: Use w_virtcol with "|" instead of w_cursor.col with "l".

Files: src/ex_docmd.c

Patch 7.3.831

Problem: Clumsy to handle the situation that a variable does not exist.

Solution: Add default value to getbufvar() et al. (Shougo Matsushita, Hirohito Higashi)

Files: runtime/doc/eval.txt, src/eval.c src/testdir/test91.in, src/testdir/test91.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.3.832

Problem: Compiler warning.

Solution: Add type cast. (Mike Williams)

Files: src/version.c

Patch 7.3.833

Problem: In the terminal the scroll wheel always scrolls the active window.

Solution: Scroll the window under the mouse pointer, like in the GUI. (Bradie Rao)

Files: src/edit.c, src/normal.c

Patch 7.3.834

Problem: Ruby 2.0 has a few API changes.

Solution: Add handling of Ruby 2.0. (Yasuhiro Matsumoto)

Files: src/if_ruby.c

Patch 7.3.835

Problem: "xxd -i" fails on an empty file.

Solution: Do output the closing } for an empty file. (partly by Lawrence Woodman)

Files: src/xxd/xxd.c

Patch 7.3.836

Problem: Clipboard does not work on Win32 when compiled with Cygwin.

Solution: Move the Win32 clipboard code to a separate file and use it when building with os_unix.c. (Frodak Baksik, Ken Takata)

Files: src/Make_bc5.mak, src/Make_cyg.mak, src/Make_ivc.mak, src/Make_ming.mak, src/Make_mvc.mak, src/Make_w16.mak, src/Makefile, src/config.h.in, src/configure.in, src/auto/configure, src/feature.h, src/globals.h, src/mbyte.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/proto.h, src/proto/os_mswin.pro, src/proto/winclip.pro, src/term.c, src/vim.h, src/winclip.c

Patch 7.3.837 (after 7.3.826)

Problem: Empty lines in :version output when 'columns' is 320.

Solution: Simplify the logic of making columns. (Nazri Ramliy, Roland Eggner)

Files: src/version.c

Patch 7.3.838 (after 7.3.830)

Problem: Insufficient testing for mksession.

Solution: Add tests. (mostly by Roland Eggner)

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile,

src/testdir/test92.in, src/testdir/test92.ok,
src/testdir/test93.in, src/testdir/test93.ok,
src/ex_docmd.c

Patch 7.3.839

Problem: Some files missing in the list of distributed files.
Solution: Add lines for new files.
Files: Filelist

Patch 7.3.840

Problem: "\\@<!" in regexp does not work correctly with multibyte characters, especially cp932.
Solution: Move column to start of multibyte character. (Yasuhiro Matsumoto)
Files: src/regexp.c

Patch 7.3.841

Problem: When a "cond ? one : two" expression has a subscript it is not parsed correctly. (Andy Wokula)
Solution: Handle a subscript also when the type is unknown. (Christian Brabandt)
Files: src/eval.c

Patch 7.3.842

Problem: Compiler warning for signed/unsigned pointer.
Solution: Add type cast. (Christian Brabandt)
Files: src/eval.c

Patch 7.3.843 (after 7.3.841)

Problem: Missing test file changes.
Solution: Change the tests.
Files: src/testdir/test49.vim, src/testdir/test49.ok

Patch 7.3.844

Problem: Enum is not indented correctly with "public" etc.
Solution: Skip "public", "private" and "protected". (Hong Xu)
Files: src/misc1.c

Patch 7.3.845 (after 7.3.844)

Problem: Enum indenting is not tested.
Solution: Add tests. (Hong Xu)
Files: src/testdir/test3.in, src/testdir/test3.ok

Patch 7.3.846

Problem: Missing proto files.
Solution: Add the files.
Files: Filelist, src/proto/os_beos.pro

Patch 7.3.847

Problem: Test 55 fails when messages are translated.
Solution: Set language to C. (Ken Takata)
Files: src/testdir/test55.in

Patch 7.3.848

Problem: Can't build with Ruby 2.0 when using MinGW x64 or MSVC10.

Solution: Fix it. Also detect RUBY_PLATFORM and RUBY_INSTALL_NAME for x64.
(Ken Takata)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/if_ruby.c

Patch 7.3.849

Problem: ":g//" gives "Pattern not found error" with E486. Should not use the error number, it's not a regular error message.
Solution: Use a normal message. (David Bürgin)
Files: src/ex_cmds.c

Patch 7.3.850

Problem: ":vimgrep //" matches everywhere.
Solution: Make it use the previous search pattern. (David Bürgin)
Files: runtime/doc/quickfix.txt, src/quickfix.c

Patch 7.3.851

Problem: Using an empty pattern with :sort silently continues when there is no previous search pattern.
Solution: Give an error message. (David Bürgin)
Files: src/ex_cmds.c

Patch 7.3.852

Problem: system() breaks clipboard text. (Yukihiro Nakadaira)
Solution: Use Xutf8TextPropertyToTextList(). (Christian Brabandt)
Also do not put the text in the clip buffer if conversion fails.
Files: src/ui.c, src/ops.c

Patch 7.3.853

Problem: Using "ra" in multiple lines on multibyte characters leaves a few characters not replaced.
Solution: Adjust the end column only in the last line. (Yasuhiro Matsumoto)
Files: src/testdir/test69.in, src/testdir/test69.ok, src/ops.c

Patch 7.3.854

Problem: After using backspace in insert mode completion, **CTRL-N** and **CTRL-P** do not highlight the right entry. (Olivier Teuliere)
Solution: Set the current item to the shown item after using backspace.
Files: src/edit.c

Patch 7.3.855

Problem: Compiler warnings.
Solution: Add type casts. (Mike Williams)
Files: src/misc1.c

Patch 7.3.856

Problem: When calling system() multibyte clipboard contents is garbled.
Solution: Save and restore the clipboard contents. (Yukihiro Nakadaira)
Files: src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/ops.c, src/proto/ops.pro, src/os_unix.c, src/proto/ui.pro, src/ui.c

Patch 7.3.857

Problem: The QuitPre autocommand event does not trigger for :qa and :wq.
Solution: Trigger the event. (Tatsuro Fujii)
Files: src/ex_docmd.c

Patch 7.3.858

Problem: "gv" selects the wrong area after some operators.
Solution: Save and restore the type of selection. (Christian Brabandt)
Files: src/testdir/test66.in, src/testdir/test66.ok, src/normal.c

Patch 7.3.859

Problem: **'ambiwidth'** must be set by the user.
Solution: Detects East Asian ambiguous width (UAX #11) state of the terminal at the start-up time and **'ambiwidth'** accordingly. (Hayaki Saito)
Files: src/main.c, src/option.c, src/term.c, src/term.h, src/proto/term.pro

Patch 7.3.860

Problem: When using --remote-expr try/catch does not work. (Andrey Radev)
Solution: Set emsg_silent instead of emsg_skip.
Files: src/main.c

Patch 7.3.861

Problem: ":setlocal number" clears global value of **'relativenumber'**.
Solution: Do it properly. (Markus Heidelberg)
Files: src/testdir/test89.in, src/testdir/test89.ok, src/option.c

Patch 7.3.862

Problem: Dragging the status line can be slow.
Solution: Look ahead and drop the drag event if there is a next one.
Files: src/eval.c, src/misc1.c, src/proto/misc1.pro, src/normal.c

Patch 7.3.863 (after 7.3.859)

Problem: Problem with **'ambiwidth'** detection for ANSI terminal.
Solution: Work around not recognizing a term response. (Hayaki Saito)
Files: src/term.c

Patch 7.3.864 (after 7.3.862)

Problem: Can't build without the mouse feature.
Solution: Add an #ifdef. (Ike Devolder)
Files: src/misc1.c

Patch 7.3.865 (after 7.3.862)

Problem: Mouse position may be wrong.
Solution: Let vungetc() restore the mouse position.
Files: src/getchar.c

Patch 7.3.866

Problem: Not serving the X selection during system() isn't nice.
Solution: When using fork() do not lose the selection, keep serving it. Add a loop similar to handling I/O. (Yukihiro Nakadaira)
Files: src/os_unix.c

Patch 7.3.867

Problem: Matchparen does not update match when using auto-indenting. (Marc Aldorasi)
Solution: Add the TextChanged and TextChangedI autocommand events.
Files: runtime/plugin/matchparen.vim, src/main.c, src/edit.c,

src/globals.h, src/vim.h, src/fileio.c, src/proto/fileio.pro,
runtime/doc/autocmd.txt

Patch 7.3.868

Problem: When at the hit-return prompt and using "k" while no text has
scrolled off screen, then using "j", an empty line is displayed.
Solution: Only act on "k" when text scrolled off screen. Also accept
page-up and page-down. (cptstuding)
Files: src/message.c

Patch 7.3.869

Problem: bufwinnr() matches buffers in other tabs.
Solution: For bufwinnr() and ? only match buffers in the current tab.
(Alexey Radkov)
Files: src/buffer.c, src/diff.c, src/eval.c, src/ex_docmd.c,
src/if_perl.xs, src/proto/buffer.pro

Patch 7.3.870

Problem: Compiler warnings when using MingW 4.5.3.
Solution: Do not use MAKEINTRESOURCE. Adjust #if. (Ken Takata)
Files: src/gui_w32.c, src/gui_w48.c, src/os_mswin.c, src/os_win32.c,
src/os_win32.h

Patch 7.3.871

Problem: search('^\$', 'c') does not use the empty match under the cursor.
Solution: Special handling of the 'c' flag. (Christian Brabandt)
Add tests.
Files: src/search.c, src/testdir/test14.in, src/testdir/test14.ok

Patch 7.3.872

Problem: On some systems case of file names is always ignored, on others
never.
Solution: Add the **'fileignorecase'** option to control this at runtime.
Implies **'wildignorecase'**.
Files: src/buffer.c, src/edit.c, src/ex_cmds2.c, src/ex_getln.c,
src/fileio.c, src/misc1.c, src/misc2.c, src/option.c,
src/option.h, src/vim.h, runtime/doc/options.txt

Patch 7.3.873

Problem: Cannot easily use :s to make title case.
Solution: Have "\L\u" result in title case. (James McCoy)
Files: src/regexp.c, src/testdir/test79.in, src/testdir/test79.ok,
src/testdir/test80.in, src/testdir/test80.ok

Patch 7.3.874

Problem: Comparing file names does not handle multibyte characters
properly.
Solution: Implement multibyte handling.
Files: src/misc1.c, src/misc2.c

Patch 7.3.875 (after 7.3.866)

Problem: Build problem with some combination of features.
Solution: Use FEAT_XCLIPBOARD instead of FEAT_CLIPBOARD.
Files: src/os_unix.c

Patch 7.3.876

Problem: #if indents are off.
Solution: Insert a space where appropriate. (Taro Muraoka)
Files: src/gui.c

Patch 7.3.877 (after 7.3.871)

Problem: Forward searching with search() is broken.
Solution: Fix it and add tests. (Sung Pae)
Files: src/search.c, src/testdir/test14.in, src/testdir/test14.ok

Patch 7.3.878

Problem: **'fileignorecase'** is missing in options window and quickref.
Solution: Add the option.
Files: runtime/optwin.vim, runtime/doc/quickref.txt

Patch 7.3.879

Problem: When using an ex command in operator pending mode, using Esc to abort the command still executes the operator. (David Bürgin)
Solution: Clear the operator when the ex command fails. (Christian Brabandt)
Files: src/normal.c

Patch 7.3.880

Problem: When writing viminfo, old history lines may replace lines written more recently by another Vim instance.
Solution: Mark history entries that were read from viminfo and overwrite them when merging with the current viminfo.
Files: src/ex_getln.c

Patch 7.3.881

Problem: Python list does not work correctly.
Solution: Fix it and add a test. (Yukihiro Nakadaira)
Files: src/testdir/test86.in, src/testdir/test86.ok, src/if_py_both.h

Patch 7.3.882

Problem: CursorHold may trigger after receiving the termresponse.
Solution: Set the did_cursorhold flag. (Hayaki Saito)
Files: src/term.c

Patch 7.3.883 (after 7.3.880)

Problem: Can't build with some combination of features.
Solution: Adjust #ifdefs.
Files: src/ex_getln.c

Patch 7.3.884

Problem: Compiler warning for variable shadowing another. (John Little)
Solution: Rename the variable. (Christian Brabandt)
Files: src/term.c

Patch 7.3.885

Problem: Double free for list and dict in Lua. (Shougo Matsu)
Solution: Do not unref list and dict. (Yasuhiro Matsumoto)
Files: src/if_lua.c

Patch 7.3.886

Problem: Can't build with multibyte on Solaris 10.
Solution: Add #ifdef X_HAVE_UTF8_STRING. (Laurent Blume)
Files: src/ui.c

Patch 7.3.887

Problem: No tests for Visual mode operators, what 7.3.879 fixes.
Solution: Add a new test file. (David Bürgin)
Files: src/testdir/test94.in, src/testdir/test94.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.3.888

Problem: Filename completion with '**fileignorecase**' does not work for multibyte characters.
Solution: Make '**fileignorecase**' work properly. (Hirohito Higashi)
Files: src/misc2.c

Patch 7.3.889

Problem: Can't build with Ruby 2.0 on a 64 bit system.
Solution: Define rb_fix2int and rb_num2int. (Kohei Suzuki)
Files: src/if_ruby.c

Patch 7.3.890

Problem: Test 79 fails on Windows. (Michael Soyka)
Solution: Add comment below line causing an error.
Files: src/testdir/test79.in

Patch 7.3.891

Problem: Merging viminfo history doesn't work well.
Solution: Don't stop when one type of history is empty. Don't merge history when writing viminfo.
Files: src/ex_getln.c

Patch 7.3.892 (after 7.3.891)

Problem: Still merging problems for viminfo history.
Solution: Do not merge lines when writing, don't write old viminfo lines.
Files: src/ex_getln.c, src/ex_cmds.c, src/proto/ex_getln.pro

Patch 7.3.893

Problem: Crash when using b:, w: or t: after closing the buffer, window or tabpage.
Solution: Allocate the dictionary instead of having it part of the buffer/window/tabpage struct. (Yukihiro Nakadaira)
Files: src/buffer.c, src/eval.c, src/fileio.c, src/structs.h,
src/window.c, src/proto/eval.pro

Patch 7.3.894

Problem: Using wrong RUBY_VER causing Ruby build to break.
Solution: Correct the RUBY_VER value. (Yongwei Wu)
Files: src/bigvim.bat

Patch 7.3.895

Problem: Valgrind error in test 91. (Issue 128)
Solution: Pass scope name to find_var_in_ht().
Files: src/eval.c

Patch 7.3.896

Problem: Memory leaks in Lua interface.
Solution: Fix the leaks, add tests. (Yukihiro Nakadaira)
Files: src/testdir/test85.in, src/testdir/test85.ok, src/if_lua.c

Patch 7.3.897

Problem: Configure doesn't always find the shared library.
Solution: Change the configure script. (Ken Takata)
Files: src/configure.in, src/auto/configure

Patch 7.3.898

Problem: Memory leak reported by valgrind in test 91.
Solution: Only use default argument when needed.
Files: src/eval.c, src/testdir/test91.in, src/testdir/test91.ok

Patch 7.3.899

Problem: #if indents are off.
Solution: Fix the indents.
Files: src/os_unix.c

Patch 7.3.900

Problem: Not obvious that some mouse features are mutual-exclusive.
Solution: Add a comment.
Files: src/feature.h

Patch 7.3.901

Problem: Outdated comment, ugly condition.
Solution: Update a few comments, break line.
Files: src/getchar.c, src/misc1.c, src/undo.c

Patch 7.3.902

Problem: When deleting last buffer in other tab the tabline is not updated.
Solution: Set the redraw_tabline flag. (Yukihiro Nakadaira)
Files: src/window.c

Patch 7.3.903 (after 7.3.892)

Problem: Crash on exit writing viminfo. (Ron Aaron)
Solution: Check for the history to be empty.
Files: src/ex_getln.c

Patch 7.3.904 (after 7.3.893)

Problem: Using memory freed by the garbage collector.
Solution: Mark items in aucmd_win as used.
Files: src/eval.c

Patch 7.3.905 (after 7.3.903)

Problem: Crash when writing viminfo. (Ron Aaron)
Solution: Prevent freed history info to be used.
Files: src/ex_getln.c

Patch 7.3.906

Problem: The "sleep .2" for running tests does not work on Solaris.
Solution: Fall back to using "sleep 1". (Laurent Blume)
Files: src/testdir/Makefile

Patch 7.3.907

Problem: Python uses IndexError when a dict key is not found.
Solution: Use KeyError instead. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.908

Problem: Possible crash when using a list in Python.
Solution: Return early if the list is NULL. (ZyX)
Files: src/if_py_both.h

Patch 7.3.909

Problem: Duplicate Python code.
Solution: Move more items to if_py_both.h. (ZyX) Also avoid compiler warnings for missing initializers.
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.910

Problem: Python code in #ifdef branches with only minor differences.
Solution: Merge the #ifdef branches. (ZyX)
Files: src/if_py_both.h, src/if_python.c

Patch 7.3.911

Problem: Python: Access to Vim variables is not so easy.
Solution: Define vim.vars and vim.vvars. (ZyX)
Files: runtime/doc/if_pyth.txt, src/eval.c, src/globals.h,
src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.912

Problem: Typing a ":" command at the hit-enter dialog does not work if the "file changed" dialog happens next.
Solution: Check for changed files before giving the hit-enter dialog.
Files: src/message.c

Patch 7.3.913 (after 7.3.905)

Problem: Still a crash when writing viminfo.
Solution: Add checks for NULL pointers. (Ron Aaron)
Files: src/ex_getln.c

Patch 7.3.914

Problem: ~/.viminfo is messed up when running tests.
Solution: Set the viminfo filename.
Files: src/testdir/test89.in, src/testdir/test94.in

Patch 7.3.915

Problem: When reading a file with encoding conversion fails at the end the next encoding in '**fencs**' is not used.

Solution: Retry with another encoding when possible. (Taro Muraoka)
Files: src/fileio.c

Patch 7.3.916

Problem: Using freed memory when pasting with the mouse (Issue 130).
Solution: Get the byte value early. (hint by Dominique Pelle)
Files: src/buffer.c

Patch 7.3.917

Problem: When a path ends in a backslash appending a comma has the wrong effect.
Solution: Replace a trailing backslash with a slash. (Nazri Ramliy)
Files: src/misc1.c, src/testdir/test73.in, src/testdir/test73.ok

Patch 7.3.918

Problem: Repeating an Ex command after using a Visual motion does not work.
Solution: Check for an Ex command being used. (David Bürgin)
Files: src/normal.c

Patch 7.3.919 (after 7.3.788)

Problem: An empty nl.po file does not work with an old msgfmt.
Solution: Put a single # in the file. (Laurent Blume)
Files: src/po/Makefile

Patch 7.3.920

Problem: Compiler warning for size_t to int.
Solution: Add a type cast. (Mike Williams)
Files: src/misc1.c

Patch 7.3.921 (after 7.3.697)

Problem: Trying to create a fontset handle when '**guifontset**' is not set.
Solution: Add curly braces around the code block. (Max Kirillov)
Files: src/syntax.c

Patch 7.3.922

Problem: No test for what 7.3.918 fixes.
Solution: Add a test. (David Bürgin)
Files: src/testdir/test94.in, src/testdir/test94.ok

Patch 7.3.923

Problem: Check for X11 header files fails on Solaris.
Solution: Only use -Werror for gcc. (Laurent Blume)
Files: src/configure.in, src/auto/configure

Patch 7.3.924

Problem: Python interface can't easily access options.
Solution: Add vim.options, vim.window.options and vim.buffer.options. (ZyX)
Files: runtime/doc/if_pyth.txt, src/eval.c, src/if_py_both.h,
src/if_python.c, src/if_python3.c, src/option.c,
src/proto/eval.pro, src/proto/option.pro, src/testdir/test86.in,
src/testdir/test86.ok, src/testdir/test87.in,
src/testdir/test87.ok, src/vim.h

Patch 7.3.925

Problem: Typos in source files.
Solution: Fix the typos. (Ken Takata)
Files: runtime/plugin/matchparen.vim, runtime/tools/vim_vs_net.cmd,
src/GvimExt/gvimext.cpp, src/INSTALLvms.txt, src/Make_cyg.mak,
src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms,
src/Make_w16.mak, src/Makefile, src/VisVim/OleAut.cpp,
src/VisVim/README_VisVim.txt, src/auto/configure, src/buffer.c,
src/configure.in, src/diff.c, src/dosinst.c, src/edit.c,
src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/farsi.c, src/feature.h, src/fileio.c, src/glbl_ime.cpp,
src/gui.c, src/gui_athena.c, src/gui_beval.c, src/gui_gtk_x11.c,
src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_w16.c,
src/gui_w32.c, src/gui_w48.c, src/gui_xmew.c, src/gui_xmewp.h,
src/hardcopy.c, src/if_cscope.c, src/if_mzsch.c, src/if_ole.cpp,
src/if_perl.xs, src/if_py_both.h, src/if_python.c,
src/if_python3.c, src/if_ruby.c, src/main.aap, src/mbyte.c,
src/memfile.c, src/memline.c, src/misc1.c, src/misc2.c,
src/nbdebug.c, src/normal.c, src/ops.c, src/os_amiga.c,
src/os_mac.h, src/os_msdos.c, src/os_mswin.c, src/os_win16.h,
src/os_win32.c, src/os_win32.h, src/quickfix.c, src/screen.c,
src/search.c, src/spell.c, src/structs.h, src/syntax.c,
src/window.c, vimtutor.com

Patch 7.3.926

Problem: Autocommands are triggered by setwinvar() et al. Missing BufEnter on :tabclose. Duplicate WinEnter on :tabclose. Wrong order of events for :tablose and :tabnew.
Solution: Fix these autocommand events. (ZyX)
Files: runtime/doc/eval.txt, src/buffer.c, src/eval.c, src/ex_cmds2.c,
src/fileio.c, src/proto/window.pro, src/testdir/test62.in,
src/testdir/test62.ok, src/window.c

Patch 7.3.927

Problem: Missing combining characters when putting text in a register.
Solution: Include combining characters. (David Bürgin)
Files: src/getchar.c, src/testdir/test44.in, src/testdir/test44.ok

Patch 7.3.928 (after 7.3.924)

Problem: Can't build with strict C compiler.
Solution: Move declaration to start of block. (Taro Muraoka)
Files: src/if_py_both.h

Patch 7.3.929 (after 7.3.924)

Problem: Compiler warning for unused variable. Not freeing unused string.
Solution: Remove the variable. Clear the options.
Files: src/option.c

Patch 7.3.930

Problem: MSVC 2012 update is not recognized.
Solution: Update the version in the makefile. (Raymond Ko)
Files: src/Make_mvc.mak

Patch 7.3.931

Problem: No completion for :xmap and :smap. (Yukihiro Nakadaira)
Solution: Add the case statements. (Christian Brabandt)
Files: src/ex_docmd.c

Patch 7.3.932

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize the variable.
Files: src/option.c

Patch 7.3.933

Problem: Ruby on Mac crashes due to GC failure.
Solution: Init the stack from main(). (Hiroshi Shirosaki)
Files: src/main.c, src/if_ruby.c, src/proto/if_ruby.pro

Patch 7.3.934

Problem: E381 and E380 make the user think nothing happened.
Solution: Display the message indicating what error list is now active.
(Christian Brabandt)
Files: src/quickfix.c

Patch 7.3.935 (after 7.3.933)

Problem: Ruby: Init stack works differently on 64 bit systems.
Solution: Handle 64 bit systems and also static library. (Yukihiro Nakadaira)
Files: src/if_ruby.c

Patch 7.3.936 (after 7.3.935)

Problem: Ruby 1.8: Missing piece for static linking on 64 bit systems.
Solution: Define ruby_init_stack() (Hiroshi Shirosaki)
Also fix preprocessor indents.
Files: src/if_ruby.c

Patch 7.3.937

Problem: More can be shared between Python 2 and 3.
Solution: Move code to if_py_both.h. (ZyX)
Files: src/if_python.c, src/if_python3.c, src/if_py_both.h

Patch 7.3.938

Problem: Python: not easy to get to window number.
Solution: Add vim.window.number. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/proto/window.pro, src/window.c

Patch 7.3.939

Problem: Using Py_BuildValue is inefficient sometimes.
Solution: Use PyLong_FromLong(). (ZyX)
Files: src/if_py_both.h

Patch 7.3.940

Problem: Python: Can't get position of window.
Solution: Add window.row and window.col. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h

Patch 7.3.941

Problem: Stuff in if_py_both.h is ordered badly.
Solution: Reorder by type. (ZyX)
Files: src/if_py_both.h, src/if_python.c

Patch 7.3.942

Problem: Python: SEGV in Buffer functions.
Solution: Call CheckBuffer() at the right time. (ZyX)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.943

Problem: Python: Negative indices were failing.
Solution: Fix negative indices. Add tests. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/testdir/test86.in,
src/testdir/test86.ok, src/testdir/test87.in,
src/testdir/test87.ok

Patch 7.3.944

Problem: External program receives the termresponse.
Solution: Insert a delay and discard input. (Hayaki Saito)
Files: src/term.c

Patch 7.3.945

Problem: Python: List of buffers is not very useful.
Solution: Make vim.buffers a map. No iterator yet. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/if_python3.c,
src/if_python.c, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.946

Problem: Sometimes get stuck in waiting for cursor position report,
resulting in keys starting with <Esc>[not working.
Solution: Only wait for more characters after <Esc>[if followed by '?', '>'
or a digit.
Files: src/term.c

Patch 7.3.947

Problem: Python: No iterator for vim.list and vim.bufferlist.
Solution: Add the iterators. Also fix name of FunctionType. Add tests for
vim.buffers. (ZyX)
Files: runtime/doc/if_pyth.txt, src/eval.c, src/if_py_both.h,
src/if_python3.c, src/if_python.c, src/proto/eval.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.948

Problem: Cannot build with Python 2.2
Solution: Make Python interface work with Python 2.2
Make 2.2 the first supported version. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.ok, src/configure.in, src/auto/configure

Patch 7.3.949

Problem: Python: no easy access to tabpages.
Solution: Add vim.tabpages and vim.current.tabpage. (ZyX)

Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/if_python3.c,
src/if_python.c, src/proto/if_python3.pro,
src/proto/if_python.pro, src/proto/window.pro, src/structs.h,
src/window.c

Patch 7.3.950

Problem: Python: Stack trace printer can't handle messages.
Solution: Make KeyError use PyErr_SetObject. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.951

Problem: Python exceptions have problems.
Solution: Change some IndexError to TypeError. Make "line number out of
range" an IndexError. Make "unable to get option value" a
RuntimeError. Make all PyErr_SetString messages start with
lowercase letter and use _(). (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.952

Problem: Python: It's not easy to change window/buffer/tabpage.
Solution: Add ability to assign to vim.current.{tabpage,buffer,window}.
(ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h

Patch 7.3.953

Problem: Python: string exceptions are deprecated.
Solution: Make vim.error an Exception subclass. (ZyX)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.954

Problem: No check if PyObject_IsTrue fails.
Solution: Add a check for -1 value. (ZyX)
Files: src/if_py_both.h

Patch 7.3.955

Problem: Python: Not enough tests.
Solution: Add tests for vim.{current>window*,tabpage*}. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.956

Problem: Python vim.bindeval() causes SIGABRT.
Solution: Make pygilstate a local variable. (Yukihiro Nakadaira)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.957

Problem: Python does not have a "do" command like Perl or Lua.
Solution: Add the ":py3do" command. (Lilydjwg)
Files: runtime/doc/if_pyth.txt, src/ex_cmds.h, src/ex_docmd.c,
src/if_python3.c, src/proto/if_python3.pro

Patch 7.3.958

Problem: Python: Iteration destructor not set.

Solution: Put IterDestructor to use. (ZyX)
Files: src/if_py_both.c

Patch 7.3.959 (after 7.3.957)
Problem: Missing error number.
Solution: Assign an error number.
Files: src/if_python3.c

Patch 7.3.960
Problem: Compiler warning for unused variable.
Solution: Put declaration in #ifdef.
Files: src/window.c

Patch 7.3.961
Problem: Tests 86 and 87 fail when using another language than English.
Solution: Set the language to C in the test. (Dominique Pelle)
Files: src/testdir/test86.in, src/testdir/test87.in,
src/testdir/test87.ok

Patch 7.3.962
Problem: Python tests are not portable.
Solution: Use shiftwidth instead of iminsert. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.963
Problem: Setting curbuf without curwin causes trouble.
Solution: Add switch_buffer() and restore_buffer(). Block autocommands to avoid trouble.
Files: src/eval.c, src/proto/eval.pro, src/proto/window.pro,
src/if_py_both.h, src/window.c, src/testdir/test86.ok

Patch 7.3.964
Problem: Python: not so easy to access tab pages.
Solution: Add window.tabpage, make window.number work with non-current tab pages. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/if_python3.c,
src/if_python.c, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.965
Problem: Python garbage collection not working properly.
Solution: Add support for garbage collection. (ZyX)
Files: src/if_py_both.h

Patch 7.3.966
Problem: There is ":py3do" but no ":pydo".
Solution: Add the ":pydo" command. (Lilydjpg)
Files: runtime/doc/if_pyth.txt, src/ex_cmds.h, src/ex_docmd.c,
src/if_py_both.h, src/if_python.c, src/if_python3.c,
src/proto/if_python.pro

Patch 7.3.967 (after 7.3.965)
Problem: Build fails on Mac OSX. (Greg Novack)
Solution: Undefine clear().

Files: src/if_py_both.h

Patch 7.3.968

Problem: Multi-byte support is only available when compiled with "big" features.

Solution: Include multibyte by default, with "normal" features.

Files: src/feature.h

Patch 7.3.969

Problem: Can't build with Python 3 and without Python 2.

Solution: Adjust #ifdef. (Xavier de Gaye)

Files: src/window.c

Patch 7.3.970

Problem: Syntax highlighting can be slow.

Solution: Include the NFA regexp engine. Add the '**regexpengine**' option to select which one is used. (various authors, including Ken Takata, Andrei Aiordachioaie, Russ Cox, Xiaozhou Liua, Ian Young)

Files: src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak, src/Makefile, src/regexp.c, src/regexp.h, src/regexp_nfa.c, src/structs.h, src/testdir/Makefile, src/testdir/test64.in, src/testdir/test64.ok, Filelist, runtime/doc/pattern.txt, runtime/doc/option.txt, src/option.c, src/option.h, src/testdir/test95.in, src/testdir/test95.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.3.971

Problem: No support for VS2012 static code analysis.

Solution: Add the ANALYZE option. (Mike Williams)

Files: src/Make_mvc.mak

Patch 7.3.972

Problem: Cursor not restored after InsertEnter autocommand if it moved to another line.

Solution: Also restore if the saved line number is still valid. Allow setting v:char to skip restoring.

Files: src/edit.c, runtime/doc/autocmd.txt

Patch 7.3.973

Problem: Compiler warnings. Crash on startup. (Tony Mechelynck)

Solution: Change EMSG2 to EMSGN. Make array one character longer.

Files: src/regexp_nfa.c

Patch 7.3.974

Problem: Can't build with ruby 1.8.5.

Solution: Only use ruby_init_stack() when RUBY_INIT_STACK is defined. (Yukihiro Nakadaira)

Files: src/if_ruby.c

Patch 7.3.975

Problem: Crash in regexp parsing.

Solution: Correctly compute the end of allocated memory.

Files: src/regexp_nfa.c

Patch 7.3.976

Problem: Can't build on HP-UX.

Solution: Remove modern initialization. (John Marriott)

Files: src/regexp_nfa.c

Patch 7.3.977

Problem: Compiler warnings on 64 bit Windows.

Solution: Add type casts. (Mike Williams) Also fix some white space and uncomment what was commented-out for testing.

Files: src/regexp_nfa.c

Patch 7.3.978

Problem: Regexp debug logs don't have a good name.

Solution: Use clear names and make it possible to write logs for the old and new engines separately. (Taro Muraoka)

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.979

Problem: Complex NFA regexp doesn't work.

Solution: Set actual state stack end instead of using an arbitrary number. (Yasuhiro Matsumoto)

Files: src/regexp_nfa.c

Patch 7.3.980

Problem: Regexp logs may contain garbage. Character classes don't work correctly for multibyte characters.

Solution: Check for end of post list. Only use "is" functions for characters up to 255. (Ken Takata)

Files: src/regexp_nfa.c

Patch 7.3.981

Problem: In the old regexp engine \i, \I, \f and \F don't work on multibyte characters.

Solution: Dereference pointer properly.

Files: src/regexp.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.982

Problem: In the new regexp engine \p does not work on multibyte characters.

Solution: Don't point to an integer but the characters.

Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.983

Problem: Unnecessary temp variable.

Solution: Remove the variable.

Files: src/regexp_nfa.c

Patch 7.3.984

Problem: A Visual mapping that uses **CTRL-G** works differently when started from Insert mode. (Ein Brown)

Solution: Reset old_mapped_len when handling typed text in Select mode.

Files: src/normal.c

Patch 7.3.985

Problem: GTK vim not started as gvim doesn't set WM_CLASS property to a useful value.

Solution: Call g_set_prpname() on startup. (James McCoy)

Files: src/gui_gtk_x11.c

Patch 7.3.986

Problem: Test 95 doesn't pass when **'encoding'** isn't utf-8. (Yasuhiro Matsumoto)

Solution: Force **'encoding'** to be utf-8.

Files: src/testdir/test95.in

Patch 7.3.987

Problem: No easy to run an individual test. Tests 64 fails when **'encoding'** is not utf-8.

Solution: Add individual test targets to the Makefile. Move some lines from test 64 to 95.

Files: src/Makefile, src/testdir/test64.in, src/testdir/test64.ok, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.988

Problem: New regexp engine is slow.

Solution: Break out of the loop when the state list is empty.

Files: src/regexp_nfa.c

Patch 7.3.989

Problem: New regexp engine compares negative numbers to character.

Solution: Add missing case statements.

Files: src/regexp_nfa.c

Patch 7.3.990

Problem: Memory leak in new regexp engine.

Solution: Jump to end of function to free memory. (Dominique Pelle)

Files: src/regexp_nfa.c

Patch 7.3.991

Problem: More can be shared by Python 2 and 3.

Solution: Move more stuff to if_py_both. (ZyX)

Files: src/if_py_both.h, src/if_python3.c, src/if_python.c, src/testdir/test87.ok

Patch 7.3.992

Problem: Python: Too many type casts.

Solution: Change argument types. (ZyX)

Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.993

Problem: Python: Later patch does things slightly differently.

Solution: Adjusted argument type changes. (ZyX)

Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.994

Problem: Python: using magic constants.

Solution: Use descriptive values for ml_flags. (ZyX)
Files: src/if_py_both.h, src/if_python3.c

Patch 7.3.995

Problem: Python: Module initialization is duplicated.
Solution: Move to shared file. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.996

Problem: Python: Can't check types of what is returned by bindeval().
Solution: Add vim.List, vim.Dictionary and vim.Function types. (ZyX)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/testdir/test86.in,
src/testdir/test86.ok, src/testdir/test87.in,
src/testdir/test87.ok

Patch 7.3.997

Problem: Vim and Python exceptions are different.
Solution: Make Vim exceptions be Python exceptions. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.998

Problem: Python: garbage collection issues.
Solution: Fix the GC issues: Use proper DESTRUCTOR_FINISH: avoids negative
refcounts, use PyObject_GC_* for objects with tp_traverse and
tp_clear, add RangeTraverse and RangeClear, use Py_XDECREF in some
places. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.999

Problem: New regexp engine sets curbuf temporarily.
Solution: Use reg_buf instead, like the old engine.
Files: src/regexp_nfa.c

Patch 7.3.1000 (whoa!)

Problem: Typo in char value causes out of bounds access.
Solution: Fix character value. (Klemens Baum)
Files: src/regexp.c

Patch 7.3.1001

Problem: Duplicate condition in if.
Solution: Remove one condition.
Files: src/regexp_nfa.c

Patch 7.3.1002

Problem: Valgrind errors for Python interface.
Solution: Fix memory leaks when running tests. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1003

Problem: Python interface does not compile with Python 2.2
Solution: Fix thread issues and True/False. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.in, src/testdir/test86.ok,

src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1004

Problem: No error when option could not be set.
Solution: Report an error. (ZyX)
Files: src/if_py_both.h, src/option.c, src/proto/option.pro,
src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1005

Problem: Get stuck on regexp "\n*" and on "%s/^\n\+/\r".
Solution: Fix handling of matching a line break. (idea by Hirohito Higashi)
Files: src/regexp_nfa.c

Patch 7.3.1006

Problem: NFA engine not used for "_[0-9]".
Solution: Enable this, fixed in patch 1005.
Files: src/regexp_nfa.c

Patch 7.3.1007

Problem: Can't build on Minix 3.2.1.
Solution: Add a condition to an #ifdef. (Gautam Tirumala)
Files: src/memfile.c

Patch 7.3.1008

Problem: Test 95 fails on MS-Windows.
Solution: Set 'nomore'. Change \i to \f. Change multibyte character to something that is not matching \i. (Ken Takata)
Files: src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1009

Problem: Compiler warning for ambiguous else.
Solution: Add curly braces.
Files: src/if_py_both.h

Patch 7.3.1010

Problem: New regexp: adding \Z makes every character match.
Solution: Only apply ireg_icombine for composing characters.
Also add missing change from patch 1008. (Ken Takata)
Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1011

Problem: New regexp engine is inefficient with multibyte characters.
Solution: Handle a character at a time instead of a byte at a time. Also make \Z partly work.
Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1012

Problem: \Z does not work properly with the new regexp engine.
Solution: Make \Z work. Add tests.
Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1013

Problem: New regexp logging is a bit messy.
Solution: Consistently use #defines, add explanatory comment. (Taro Muraoka)

Files: src/regexp_nfa.c

Patch 7.3.1014

Problem: New regexp state dump is hard to read.
Solution: Make the state dump more pretty. (Taro Muraoka)
Files: src/regexp_nfa.c

Patch 7.3.1015

Problem: New regexp engine: Matching composing characters is wrong.
Solution: Fix matching composing characters.
Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1016

Problem: Unused field in nfa_state.
Solution: Remove lastthread.
Files: src/regexp.h, src/regexp_nfa.c

Patch 7.3.1017

Problem: Zero width match changes length of match.
Solution: For a zero width match put new states in the current position in the state list.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok, src/regexp.h

Patch 7.3.1018

Problem: New regexp engine wastes memory.
Solution: Allocate prog with actual number of states, not estimated maximum number of states.
Files: src/regexp_nfa.c

Patch 7.3.1019

Problem: These do not work with the new regexp engine: `\%o123`, `\%x123`, `\%d123`, `\%u123` and `\%U123`.
Solution: Implement these items.
Files: src/regexp_nfa.c

Patch 7.3.1020

Problem: Not all patterns are tested with auto / old / new engine.
Solution: Test patterns with three values of **'regengine'**.
Files: src/testdir/test64.in, src/testdir/test64.ok, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1021

Problem: New regexp engine does not ignore order of composing chars.
Solution: Ignore composing chars order.
Files: src/regexp_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1022

Problem: Compiler warning for shadowed variable. (John Little)
Solution: Move declaration, rename variables.
Files: src/regexp_nfa.c

Patch 7.3.1023

Problem: Searching for composing char only and using `\Z` has different

results.
Solution: Make it match the composing char, matching everything is not useful.
Files: src/regex_nfa.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.3.1024

Problem: New regexp: End of matching pattern not set correctly. (Cesar Romani)
Solution: Quit the loop after finding the match. Store nfa_has_zend in the program.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok, src/regex.h

Patch 7.3.1025

Problem: New regexp: not matching newline in string. (Marc Weber)
Solution: Check for "\n" character.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1026

Problem: New regexp: pattern that includes a new-line matches too early. (John McGowan)
Solution: Do not start searching in the second line.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1027

Problem: New regexp performance: Calling no_Magic() very often.
Solution: Remove magicness inline.
Files: src/regex_nfa.c

Patch 7.3.1028

Problem: New regexp performance: Copying a lot of position state.
Solution: Only copy the sub-expressions that are being used.
Files: src/regex_nfa.c, src/regex.h

Patch 7.3.1029

Problem: New regexp performance: Unused position state being copied.
Solution: Keep track of which positions are actually valid.
Files: src/regex_nfa.c

Patch 7.3.1030 (after 7.3.1028)

Problem: Can't build for debugging.
Solution: Fix struct member names.
Files: src/regex_nfa.c

Patch 7.3.1031

Problem: Compiler warnings for shadowed variable. (John Little)
Solution: Move the variable declarations to the scope where they are used.
Files: src/regex_nfa.c

Patch 7.3.1032

Problem: "\ze" is not supported by the new regexp engine.
Solution: Make "\ze" work.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1033

Problem: "\1" .. "\9" are not supported in the new regexp engine.
Solution: Implement them. Add a few more tests.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok,
src/regexp.h

Patch 7.3.1034

Problem: New regexp code using strange multibyte code.
Solution: Use the normal code to advance and backup pointers.
Files: src/regexp_nfa.c

Patch 7.3.1035

Problem: Compiler warning on 64 bit windows.
Solution: Add type cast. (Mike Williams)
Files: src/if_py_both.h

Patch 7.3.1036

Problem: Can't build on HP-UX.
Solution: Give the union a name. (John Marriott)
Files: src/regexp_nfa.c

Patch 7.3.1037

Problem: Look-behind matching is very slow on long lines.
Solution: Add a byte limit to how far back an attempt is made.
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1038

Problem: Crash when using Cscope.
Solution: Avoid negative argument to vim_strncpy(). (Narendran
Gopalakrishnan)
Files: src/if_cscope.c

Patch 7.3.1039

Problem: New regexp engine does not support \%23c, \%<23c and the like.
Solution: Implement them. (partly by Yasuhiro Matsumoto)
Files: src/regexp.h, src/regexp_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1040

Problem: Python: Problems with debugging dynamic build.
Solution: Python patch 1. (ZyX)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.1041

Problem: Python: Invalid read valgrind errors.
Solution: Python patch 2: defer DICTKEY_UNREF until key is no longer needed.
(ZyX)
Files: src/if_py_both.h

Patch 7.3.1042

Problem: Python: can't assign to vim.Buffer.name.
Solution: Python patch 3. (ZyX)
Files: runtime/doc/if_pyth.txt, src/ex_cmds.c, src/if_py_both.h,

src/if_python3.c, src/if_python.c, src/proto/ex_cmds.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1043

Problem: Python: Dynamic compilation with 2.3 fails.
Solution: Python patch 4. (ZyX)
Files: src/if_python.c

Patch 7.3.1044

Problem: Python: No {Buffer,TabPage,Window}.valid attributes.
Solution: Python patch 5: add .valid (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1045

Problem: Python: No error handling for VimToPython function.
Solution: Python patch 6. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1046

Problem: Python: Using Py_BuildValue for building strings.
Solution: Python patch 7 and 7.5: Replace Py_BuildValue with
PyString_FromString. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1047

Problem: Python: dir() does not work properly.
Solution: Python patch 8. Add __dir__ method to all objects with custom
tp_getattr supplemented by __members__ attribute for at least
python-2* versions. __members__ is not mentioned in python-3*
dir() output even if it is accessible. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1048

Problem: Python: no consistent naming.
Solution: Python patch 9: Rename d to dict and lookupDict to lookup_dict.
(ZyX)
Files: src/if_py_both.h

Patch 7.3.1049

Problem: Python: no consistent naming
Solution: Python patch 10: Rename DICTKEY_GET_NOTEMPTY to DICTKEY_GET. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1050

Problem: Python: Typo in pyiter_to_tv.
Solution: Python patch 11. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1051

Problem: Python: possible memory leaks.
Solution: Python patch 12: fix the leaks (ZyX)
Files: src/if_py_both.h

Patch 7.3.1052

Problem: Python: possible SEGV and negative refcount.
Solution: Python patch 13: Fix IterIter function. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1053

Problem: Python: no flag for types with tp_traverse+tp_clear.
Solution: Python patch 14: Add Py_TPFLAGS_HAVE_GC. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1054 (after 7.3.1042)

Problem: Can't build without the +autocmd feature. (Elimar Riesebieter)
Solution: Fix use of buf and curbuf.
Files: src/ex_cmds.c, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1055

Problem: Negated collection does not match newline.
Solution: Handle newline differently. (Hiroshi Shiroasaki)
Files: src/regexp_nfa.c, src/testdir/test64.ok, src/testdir/test64.in

Patch 7.3.1056

Problem: Python: possible memory leaks.
Solution: Python patch 15. (ZyX) Fix will follow later.
Files: src/eval.c, src/if_py_both.h, src/proto/eval.pro

Patch 7.3.1057

Problem: Python: not enough compatibility.
Solution: Python patch 16: Make OutputWritelines support any sequence object (ZyX) **Note:** tests fail
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1058

Problem: Call of funcref does not succeed in other script.
Solution: Python patch 17: add get_expanded_name(). (ZyX)
Files: src/eval.c, src/proto/eval.pro

Patch 7.3.1059

Problem: Python: Using fixed size buffers.
Solution: Python patch 18: Use python's own formatter. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.1060

Problem: Python: can't repr() a function.
Solution: Python patch 19: add FunctionRepr(). (ZyX)
Files: src/if_py_both.h

Patch 7.3.1061

Problem: Python: Dictionary is not standard.
Solution: Python patch 20: Add standard methods and fields. (ZyX)

Files: runtime/doc/if_pyth.txt, src/eval.c, src/if_py_both.h,
src/if_python3.c, src/if_python.c, src/proto/eval.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1062

Problem: Python: List is not standard.
Solution: Python patch 21: Add standard methods and fields. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1063

Problem: Python: Function is not standard.
Solution: Python patch 22: make Function subclassable. (ZyX)
Files: src/eval.c, src/if_py_both.h, src/proto/eval.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1064

Problem: Python: insufficient error checking.
Solution: Python patch 23. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1065

Problem: Python: key mapping is not standard.
Solution: Python patch 24: use PyMapping_Keys. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.1066

Problem: Python: Insufficient exception and error testing.
Solution: Python patch 25. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1067

Problem: Python: documentation lags behind.
Solution: Python patch 26. (ZyX)
Files: runtime/doc/if_pyth.txt

Patch 7.3.1068

Problem: Python: Script is auto-loaded on function creation.
Solution: Python patch 27. (ZyX)
Files: src/eval.c, src/if_py_both.h, src/proto/eval.pro,
src/testdir/test86.ok, src/testdir/test87.ok, src/vim.h

Patch 7.3.1069

Problem: Python: memory leaks.
Solution: Python patch 28: Purge out DICTKEY_CHECK_EMPTY macros. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1070

Problem: Vim crashes in Python tests. Compiler warning for unused function.
Solution: Disable the tests for now. Move the function.
Files: src/if_py_both.h, src/if_python.c, src/testdir/test86.in,

src/testdir/test87.in

Patch 7.3.1071

Problem: New regexp engine: backreferences don't work correctly.
Solution: Add every possible start/end position on the state stack.
Files: src/regexp_nfa.c, src/regexp.h, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1072

Problem: Compiler warning for uninitialized variable.
Solution: Initialize it.
Files: src/regexp_nfa.c

Patch 7.3.1073

Problem: New regexp engine may run out of states.
Solution: Allocate states dynamically. Also make the test report errors.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok,
src/testdir/test95.in

Patch 7.3.1074

Problem: Compiler warning for printf format. (Manuel Ortega)
Solution: Add type casts.
Files: src/if_py_both.h

Patch 7.3.1075

Problem: Compiler warning for storing a long_u in an int.
Solution: Declare the number as an int. (Mike Williams)
Files: src/regexp_nfa.c

Patch 7.3.1076

Problem: New regexp engine: \@= and \& don't work.
Solution: Make these items work. Add column info to logging.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1077

Problem: Python: Allocating dict the wrong way, causing a crash.
Solution: Use py_dict_alloc(). Fix some exception problems. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1078

Problem: New regexp engine: \@! doesn't work.
Solution: Implement the negated version of \@=.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1079

Problem: Test 87 fails.
Solution: Fix the test for Python 3.3. (ZyX) Make it pass on 32 bit systems.
Files: src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1080

Problem: Test 86 fails.
Solution: Comment out the parts that don't work. Make it pass on 32 bit systems.
Files: src/testdir/test86.in, src/testdir/test86.ok

Patch 7.3.1081

Problem: Compiler warnings on 64-bit Windows.
Solution: Change variable types. (Mike Williams)
Files: src/if_py_both.h, src/regex_nfa.c

Patch 7.3.1082

Problem: New regex engine: Problem with \@= matching.
Solution: Save and restore nfa_match.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1083

Problem: New regex engine: Does not support \%^ and \%\$.
Solution: Support matching start and end of file.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1084

Problem: New regex engine: only accepts up to \{,10}.
Solution: Remove upper limit. Remove dead code with NFA_PLUS.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1085

Problem: New regex engine: Non-greedy multi doesn't work.
Solution: Implement \{-}.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1086

Problem: Old regex engine accepts illegal range, new one doesn't.
Solution: Also accept the illegal range with the new engine.
Files: src/regex_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1087

Problem: A leading star is not seen as a normal char when \{} follows.
Solution: Save and restore the parse state properly.
Files: src/regex.c, src/regex_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1088

Problem: New regex engine: \@<= and \@<! are not implemented.
Solution: Implement look-behind matching. Fix off-by-one error in old
regex engine.
Files: src/regex.c, src/regex_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1089

Problem: Tests 86 and 87 fail on MS-Windows. (Ken Takata)
Solution: Fix platform-specific stuff. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1090

Problem: New regex engine does not support \z1 .. \z9 and \z(.
Solution: Implement the syntax submatches.
Files: src/regex.h, src/regex_nfa.c

Patch 7.3.1091

Problem: New regexp engine: no error when using \z1 or \z(where it does not work.

Solution: Give an error message.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1092

Problem: Can't build with regexp debugging. NFA debug output shows wrong pattern.

Solution: Fix debugging code for recent changes. Add the pattern to the program.

Files: src/regexp_nfa.c, src/regexp.h

Patch 7.3.1093

Problem: New regexp engine: When a sub expression is empty \1 skips a character.

Solution: Make \1 try the current position when the match is empty.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1094

Problem: New regexp engine: Attempts to match "^" at every character.

Solution: Only try "^" at the start of a line.

Files: src/regexp_nfa.c

Patch 7.3.1095

Problem: Compiler warnings for shadowed variables. (Christian Brabandt)

Solution: Rename new_state() to alloc_state(). Remove unnecessary declaration.

Files: src/regexp_nfa.c

Patch 7.3.1096

Problem: Python: popitem() was not defined in a standard way.

Solution: Remove the argument from popitem(). (ZyX)

Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1097

Problem: Python: a few recently added items are not documented.

Solution: Update the documentation. (ZyX)

Files: runtime/doc/if_pyth.txt

Patch 7.3.1098

Problem: Python: Possible memory leaks

Solution: Add Py_XDECREF() calls. (ZyX)

Files: src/if_py_both.h

Patch 7.3.1099

Problem: Python: Changing directory with os.chdir() causes problems for Vim's notion of directories.

Solution: Add vim.chdir() and vim.fchdir(). (ZyX)

Files: runtime/doc/if_pyth.txt, src/ex_docmd.c, src/if_py_both.h, src/if_python3.c, src/if_python.c, src/proto/ex_docmd.pro,

src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1100

Problem: Python: a few more memory problems.
Solution: Add and remove Py_XDECREF(). (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1101

Problem: Configure doesn't find Python 3 on Ubuntu 13.04.
Solution: First try distutils.sysconfig. Also fix some indents. (Ken Takata)
Files: src/configure.in, src/auto/configure

Patch 7.3.1102

Problem: Completion of ":py3do" and ":py3file" does not work after ":py3".
Solution: Make completion work. (Taro Muraoka)
Files: src/ex_docmd.c

Patch 7.3.1103

Problem: New regexp engine: overhead in saving and restoring.
Solution: Make saving and restoring list IDs faster. Don't copy or check \z subexpressions when they are not used.
Files: src/regexp_nfa.c

Patch 7.3.1104

Problem: New regexp engine does not handle "~".
Solution: Add support for "~".
Files: src/regexp_nfa.c, src/testdir/test24.in, src/testdir/test24.ok

Patch 7.3.1105

Problem: New regexp engine: too much code in one function. Dead code.
Solution: Move the recursive nfa_regmatch call to a separate function.
Remove the dead code.
Files: src/regexp_nfa.c

Patch 7.3.1106

Problem: New regexp engine: saving and restoring lastlist in the states takes a lot of time.
Solution: Use a second lastlist value for the first recursive call.
Files: src/regexp.h, src/regexp_nfa.c

Patch 7.3.1107

Problem: Compiler warnings for unused variables.
Solution: Put the variables inside #ifdef.
Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1108

Problem: Error message for os.fchdir() (Charles Peacech)
Solution: Clear the error. (ZyX)
Files: src/if_py_both.h

Patch 7.3.1109

Problem: Building on MS-Windows doesn't see changes in if_py_both.h.
Solution: Add a dependency. (Ken Takata)
Files: src/Make_bc5.mak, src/Make_cyg.mak, src/Make_ming.mak,
src/Make_mvc.mak

Patch 7.3.1110

Problem: New regexp matching: Using \@= and the like can be slow.
Solution: Decide whether to first try matching the zero-width part or what follows, whatever is more likely to fail.
Files: src/regexp_nfa.c

Patch 7.3.1111

Problem: nfa_recognize_char_class() implementation is inefficient.
Solution: Use bits in an int instead of chars in a string. (Dominique Pelle)
Files: src/regexp_nfa.c, src/testdir/test36.in, src/testdir/test36.ok

Patch 7.3.1112

Problem: New regexp engine: \%V not supported.
Solution: Implement \%V. Add tests.
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1113

Problem: New regexp engine: \% 'm not supported.
Solution: Implement \% 'm. Add tests.
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1114 (after 7.3.1110)

Problem: Can't build without the syntax feature.
Solution: Add #ifdefs. (Erik Falor)
Files: src/regexp_nfa.c

Patch 7.3.1115

Problem: Many users don't like the cursor line number when **'relativenumber'** is set.
Solution: Have four combinations with **'number'** and **'relativenumber'**. (Christian Brabandt)
Files: runtime/doc/options.txt, src/option.c, src/screen.c,
src/testdir/test89.in, src/testdir/test89.ok

Patch 7.3.1116

Problem: Can't build without Visual mode.
Solution: Add #ifdefs.
Files: src/regexp_nfa.c

Patch 7.3.1117

Problem: New regexp engine: \%[abc] not supported.
Solution: Implement \%[abc]. Add tests.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1118

Problem: Match failure rate is not very specific.
Solution: Tune the failure rate for match items.

Files: src/regexp_nfa.c

Patch 7.3.1119

Problem: Flags in 'cpo' are search for several times.

Solution: Store the result and re-use the flags.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1120

Problem: Crash when regexp logging is enabled.

Solution: Avoid using NULL pointers. Advance over count argument.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1121

Problem: New regexp engine: adding states that are not used.

Solution: Don't add the states.

Files: src/regexp_nfa.c

Patch 7.3.1122

Problem: New regexp engine: \@> not supported.

Solution: Implement \%>.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1123

Problem: Can't build tiny Vim on MS-Windows.

Solution: Adjust #ifdef around using modif_fname(). (Mike Williams)

Files: src/misc1.c

Patch 7.3.1124

Problem: Python: Crash on MS-Windows when os.fchdir() is not available.

Solution: Check for _chdir to be NULL. (Ken Takata)

Files: src/if_py_both.h

Patch 7.3.1125

Problem: Error for using \%V in a pattern in tiny Vim.

Solution: Allow using \%V but never match. (Dominique Pelle)

Files: src/regexp_nfa.c

Patch 7.3.1126

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)

Solution: Assign something to the variable.

Files: src/regexp_nfa.c

Patch 7.3.1127

Problem: No error for using empty \%[].

Solution: Give error message.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1128

Problem: Now that the NFA engine handles everything every failure is a syntax error.

Solution: Remove the syntax_error flag.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1129

Problem: Can't see what pattern in syntax highlighting is slow.
Solution: Add the ":syntime" command.
Files: src/structs.h, src/syntax.c, src/ex_cmds.h, src/ex_docmd.c,
src/proto/syntax.pro, src/ex_cmds2.c, src/proto/ex_cmds2.pro,
runtime/doc/syntax.txt

Patch 7.3.1130 (after 7.3.1129)

Problem: Can't build with anything but huge features.
Solution: Check for FEAT_PROFILE. (Yasuhiro Matsumoto)
Files: src/ex_docmd.c, src/structs.h, src/syntax.c

Patch 7.3.1131

Problem: New regexp engine is a bit slow.
Solution: Do not clear the state list. Don't copy syntax submatches when
not used.
Files: src/regexp_nfa.c

Patch 7.3.1132

Problem: Crash when debugging regexp.
Solution: Do not try to dump subexpr that were not set. Skip over count of
\% items.
Files: src/regexp.c, src/regexp_nfa.c

Patch 7.3.1133

Problem: New regexp engine is a bit slow.
Solution: Skip ahead to a character that must match. Don't try matching a
"^" patten past the start of line.
Files: src/regexp_nfa.c, src/regexp.h

Patch 7.3.1134

Problem: Running test 49 takes a long time.
Solution: Don't have it grep all files.
Files: src/testdir/test49.vim

Patch 7.3.1135

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/syntax.c

Patch 7.3.1136

Problem: ":func Foo" does not show attributes.
Solution: Add "abort", "dict" and "range". (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.1137

Problem: New regexp engine: collections are slow.
Solution: Handle all characters in one go.
Files: src/regexp_nfa.c

Patch 7.3.1138

Problem: New regexp engine: neglist no longer used.
Solution: Remove the now unused neglist.
Files: src/regexp_nfa.c

Patch 7.3.1139

Problem: New regexp engine: negated flag is hardly used.
Solution: Add separate _NEG states, remove negated flag.
Files: src/regexp_nfa.c, src/regexp.h

Patch 7.3.1140

Problem: New regexp engine: trying expensive match while the result is not going to be used.
Solution: Check for output state already being in the state list.
Files: src/regexp_nfa.c

Patch 7.3.1141

Problem: Win32: Check for available memory is not reliable and adds overhead.
Solution: Remove mch_avail_mem(). (Mike Williams)
Files: src/os_win32.c, src/os_win32.h

Patch 7.3.1142

Problem: Memory leak in ":syntime report".
Solution: Clear the grow array. (Dominique Pelle)
Files: src/syntax.c

Patch 7.3.1143

Problem: When mapping NUL it is displayed as an X.
Solution: Check for KS_ZERO instead of K_ZERO. (Yasuhiro Matsumoto)
Files: src/message.c

Patch 7.3.1144

Problem: "R0" is not translated everywhere.
Solution: Put inside _(). (Sergey Alyoshin)
Files: src/buffer.c, src/screen.c

Patch 7.3.1145

Problem: New regexp engine: addstate() is called very often.
Solution: Optimize adding the start state.
Files: src/regexp_nfa.c

Patch 7.3.1146

Problem: New regexp engine: look-behind match not checked when followed by zero-width match.
Solution: Do the look-behind match before adding the zero-width state.
Files: src/regexp_nfa.c

Patch 7.3.1147

Problem: New regexp engine: regstart is only used to find the first match.
Solution: Use regstart whenever adding the start state.
Files: src/regexp_nfa.c

Patch 7.3.1148

Problem: No command line completion for ":syntime".
Solution: Implement the completion. (Dominique Pelle)
Files: runtime/doc/map.txt, src/ex_cmds.h, src/ex_docmd.c,
src/ex_getln.c, src/proto/syntax.pro, src/syntax.c, src/vim.h

Patch 7.3.1149

Problem: New regexp engine: Matching plain text could be faster.

Solution: Detect a plain text match and handle it specifically. Add vim_regfree().

Files: src/regexp.c, src/regexp.h, src/regexp_nfa.c, src/proto/regexp.pro, src/buffer.c, src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/fileio.c, src/gui.c, src/misc1.c, src/misc2.c, src/option.c, src/syntax.c, src/quickfix.c, src/search.c, src/spell.c, src/tag.c, src/window.c, src/screen.c, src/macros.h, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1150

Problem: New regexp engine: Slow when a look-behind match does not have a width specified.

Solution: Try to compute the maximum width.

Files: src/regexp_nfa.c

Patch 7.3.1151

Problem: New regexp engine: Slow when a look-behind match is followed by a zero-width match.

Solution: Postpone the look-behind match more often.

Files: src/regexp_nfa.c

Patch 7.3.1152

Problem: In tiny build ireg_icombine is undefined. (Tony Mechelynck)

Solution: Add #ifdef.

Files: src/regexp_nfa.c

Patch 7.3.1153

Problem: New regexp engine: Some look-behind matches are very expensive.

Solution: Postpone invisible matches further, until a match is almost found.

Files: src/regexp_nfa.c

Patch 7.3.1154

Problem: New regexp_nfa engine: Unnecessary code.

Solution: Remove unnecessary code.

Files: src/regexp_nfa.c

Patch 7.3.1155

Problem: MS-DOS: "make test" uses external rmdir command.

Solution: Rename "rmdir" to "rd". (Taro Muraoka)

Files: src/testdir/Make_dos.mak

Patch 7.3.1156

Problem: Compiler warnings. (dv1445)

Solution: Initialize variables, even when the value isn't really used.

Files: src/regexp_nfa.c, src/eval.c

Patch 7.3.1157

Problem: New regexp engine fails on "\\(\\<command\\)\\@<=.*"

Solution: Fix rule for postponing match. Further tune estimating whether postponing works better. Add test.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1158

Problem: Crash when running test 86. (Jun Takimoto)
Solution: Define PY_SSIZE_T_CLEAN early. (Elimar Riesebieter)
Files: src/if_python.c, src/if_python3.c

Patch 7.3.1159

Problem: The round() function is not always available. (Christ van Willegen)
Solution: Use the solution from f_round().
Files: src/ex_cmds2.c, src/eval.c, src/proto/eval.pro

Patch 7.3.1160

Problem: Mixing long and pointer doesn't always work.
Solution: Avoid cast to pointer.
Files: src/undo.c

Patch 7.3.1161

Problem: Python: PyList_SetItem() is inefficient.
Solution: Use PyList_SET_ITEM() (ZyX)
Files: src/if_py_both.h

Patch 7.3.1162

Problem: Python: Memory leaks
Solution: Add more Py_DECREF(). (ZyX)
Files: src/if_py_both.h, src/if_python.c

Patch 7.3.1163

Problem: Not easy to load Python modules.
Solution: Search "python2", "python3" and "pythonx" directories in **'runtimepath'** for Python modules. (ZyX)
Files: runtime/doc/if_pyth.txt, src/configure.in, src/ex_cmds2.c, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/testdir/test86.in, src/testdir/test87.in, src/auto/configure

Patch 7.3.1164

Problem: Can't test what is actually displayed on screen.
Solution: Add the screenchar() and screenattr() functions.
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.3.1165

Problem: HP-UX compiler can't handle zero size array. (Charles Cooper)
Solution: Make the array one item big.
Files: src/regexp.h, src/regexp_nfa.c

Patch 7.3.1166

Problem: Loading Python modules is not tested.
Solution: Enable commented-out tests, add missing files. (ZyX)
Files: src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok, src/testdir/python2/module.py, src/testdir/python3/module.py, src/testdir/pythonx/module.py, src/testdir/pythonx/modulex.py, Filelist

Patch 7.3.1167

Problem: Python configure check doesn't reject Python 2 when requesting Python 3. Some systems need `-pthreads` instead of `-pthread`.
Solution: Adjust configure accordingly. (Andrei Olsen)
Files: `src/configure.in`, `src/auto/configure`

Patch 7.3.1168

Problem: Python "sane" configure checks give a warning message.
Solution: Use single quotes instead of escaped double quotes. (Ben Fritz)
Files: `src/configure.in`, `src/auto/configure`

Patch 7.3.1169

Problem: New regexp engine: some work is done while executing a pattern, even though the result is predictable.
Solution: Do the work while compiling the pattern.
Files: `src/regexp_nfa.c`

Patch 7.3.1170

Problem: Patch 7.3.1058 breaks backwards compatibility, not possible to use a function reference as a string. (lilydjwg)
Solution: Instead of translating the function name only translate "s:".
Files: `src/eval.c`

Patch 7.3.1171

Problem: Check for digits and ascii letters can be faster.
Solution: Use a trick with one comparison. (Dominique Pelle)
Files: `src/macros.h`

Patch 7.3.1172

Problem: Python 2: loading modules doesn't work well.
Solution: Fix the code. Add more tests. (ZyX)
Files: `runtime/doc/if_pyth.txt`, `src/if_py_both.h`, `src/if_python.c`, `src/testdir/python2/module.py`, `src/testdir/python3/module.py`, `src/testdir/python_after/after.py`, `src/testdir/python_before/before.py`, `src/testdir/test86.in`, `src/testdir/test86.ok`, `src/testdir/test87.in`, `src/testdir/test87.ok`, `Filelist`

Patch 7.3.1173

Problem: Python 2 tests don't have the same output everywhere.
Solution: Make the Python 2 tests more portable. (ZyX)
Files: `src/testdir/test86.in`, `src/testdir/test86.ok`

Patch 7.3.1174

Problem: Python 2 and 3 use different ways to load modules.
Solution: Use the same method. (ZyX)
Files: `runtime/doc/if_pyth.txt`, `src/if_py_both.h`, `src/if_python3.c`, `src/if_python.c`

Patch 7.3.1175

Problem: Using `isalpha()` and `isalnum()` can be slow.
Solution: Use range checks. (Mike Williams)
Files: `src/ex_docmd.c`, `src/macros.h`

Patch 7.3.1176

Problem: Compiler warnings on 64 bit system.
Solution: Add type casts. (Mike Williams)
Files: src/eval.c, src/if_py_both.h

Patch 7.3.1177

Problem: Wasting memory on padding.
Solution: Reorder struct fields. (Dominique Pelle)
Files: src/structs.h, src/fileio.c

Patch 7.3.1178

Problem: Can't put all Vim config files together in one directory.
Solution: Load ~/.vim/vimrc if ~/.vimrc does not exist. (Lech Lorens)
Files: runtime/doc/gui.txt, runtime/doc/starting.txt, src/gui.c, src/main.c, src/os_amiga.h, src/os_dos.h, src/os_unix.h

Patch 7.3.1179

Problem: When a global mapping starts with the same characters as a buffer-local mapping Vim waits for a character to be typed to find out whether the global mapping is to be used. (Andy Wokula)
Solution: Use the local mapping without waiting. (Michael Henry)
Files: runtime/doc/map.txt, src/getchar.c

Patch 7.3.1180

Problem: When current directory changes, path from cscope may no longer be valid. (AS Budden)
Solution: Always store the absolute path. (Christian Brabandt)
Files: src/if_cscope.c

Patch 7.3.1181

Problem: Wrong error message for 1.0[0].
Solution: Check for funcref and float separately. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.3.1182

Problem: **'backupcopy'** default on MS-Windows does not work for hard and soft links.
Solution: Check for links. (David Pope, Ken Takata)
Files: src/fileio.c, src/os_win32.c, src/proto/os_win32.pro

Patch 7.3.1183

Problem: Python tests 86 and 87 fail.
Solution: Add "empty" files. (ZyX)
Files: src/testdir/python_before/before_1.py, src/testdir/python_before/before_2.py

Patch 7.3.1184

Problem: Highlighting is sometimes wrong. (Axel Bender)
Solution: Fetch regline again when returning from recursive regmatch.
Files: src/regexp_nfa.c

Patch 7.3.1185

Problem: New regexp engine: no match with ^ after \n. (SungHyun Nam)
Solution: Fix it, add a test.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1186

Problem: Python 3: test 87 may crash.

Solution: Use `_PyArg_Parse_SizeT` instead of `PyArg_Parse`. (Jun Takimoto)

Files: src/if_python3.c

Patch 7.3.1187 (after 7.3.1170)

Problem: "s:" is recognized but "<SID>" is not. (ZyX)

Solution: Translate "<SID>" like "s:".

Files: src/eval.c

Patch 7.3.1188

Problem: Newline characters messing up error message.

Solution: Remove the newlines. (Kazunobu Kuriyama)

Files: src/gui_x11.c

Patch 7.3.1189 (after 7.3.1185)

Problem: Highlighting is still wrong sometimes. (Dominique Pelle)

Solution: Also restore `reginput` properly.

Files: src/regexp_nfa.c

Patch 7.3.1190

Problem: Compiler warning for parentheses. (Christian Wellenbrock)

Solution: Change `#ifdef`.

Files: src/ex_docmd.c

Patch 7.3.1191

Problem: Backreference to previous line doesn't work. (Lech Lorens)

Solution: Implement looking in another line.

Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test64.in,
src/testdir/test64.ok

Patch 7.3.1192

Problem: Valgrind reports errors when using backreferences. (Dominique Pelle)

Solution: Do not check the end of submatches.

Files: src/regexp_nfa.c

Patch 7.3.1193

Problem: `fail_if_missing` not used for Python 3.

Solution: Give an error when Python 3 can't be configured. (Andrei Olsen)

Files: src/configure.in, src/auto/configure

Patch 7.3.1194

Problem: Yaml highlighting is slow.

Solution: Tune the estimation of pattern failure chance.

Files: src/regexp_nfa.c

Patch 7.3.1195

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)

Solution: Set the length to the matching backref.

Files: src/regexp.c

Patch 7.3.1196

Problem: Old regexp engine does not match pattern with backref correctly.
(Dominique Pelle)
Solution: Fix setting status. Test multi-line patterns better.
Files: src/regexp.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1197

Problem: ":wviminfo!" does not write history previously read from a viminfo
file. (Roland Eggner)
Solution: When not merging history write all entries.
Files: src/ex_cmds.c, src/ex_getln.c, src/proto/ex_getln.pro

Patch 7.3.1198

Problem: Build error when using Perl 5.18.0 and dynamic loading.
Solution: Change #ifdefs for Perl_croak_xs_usage. (Ike Devolder)
Files: src/if_perl.xs

Patch 7.3.1199

Problem: When evaluating '**foldexpr**' causes an error this is silently
ignored and evaluation is retried every time.
Solution: Set emsg_silent instead of emsg_off. Stop evaluating '**foldexpr**' if
it is causing errors. (Christian Brabandt)
Files: src/fold.c

Patch 7.3.1200

Problem: When calling setline() from Insert mode, using **CTRL-R** =, undo does
not work properly. (Israel Chauca)
Solution: Sync undo after evaluating the expression. (Christian Brabandt)
Files: src/edit.c, src/testdir/test61.in, src/testdir/test61.ok

Patch 7.3.1201

Problem: When a startup script creates a preview window, it probably
becomes the current window.
Solution: Make another window the current one. (Christian Brabandt)
Files: src/main.c

Patch 7.3.1202 (after 7.3.660)

Problem: Tags are not found in case-folded tags file. (Darren cole, Issue
90)
Solution: Take into account that when case folding was used for the tags
file "!rm" sorts before the "!_TAG" header lines.
Files: src/tag.c

Patch 7.3.1203

Problem: Matches from matchadd() might be highlighted incorrectly when they
are at a fixed position and inserting lines. (John Szakmeister)
Solution: Redraw all lines below a change if there are highlighted matches.
(idea by Christian Brabandt)
Files: src/screen.c

Patch 7.3.1204

Problem: Calling gettabwinvar() in '**tabline**' cancels Visual mode. (Hirohito
Higashi)
Solution: Don't always use goto_tabpage_tp().

Files: src/window.c, src/proto/window.pro, src/eval.c, src/if_py_both.h

Patch 7.3.1205

Problem: logtalk.dict is not removed on uninstall.

Solution: Remove the file. (Kazunobu Kuriyama)

Files: src/Makefile

Patch 7.3.1206

Problem: Inconsistent function argument declarations.

Solution: Use ANSI style.

Files: src/if_py_both.h

Patch 7.3.1207

Problem: New regexp engine: no match found on "#if F00". (Lech Lorens)

Solution: When adding a state gets skipped don't adjust the index.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1208

Problem: Compiler warnings on MS-Windows.

Solution: Add type cast. Move variable declaration. (Mike Williams)

Files: src/option.c, src/os_mswin.c

Patch 7.3.1209

Problem: No completion for ":tabdo".

Solution: Add tabdo to the list of modifiers. (Dominique Pelle)

Files: src/ex_docmd.c

Patch 7.3.1210 (after 7.3.1182)

Problem: **'backupcopy'** default on MS-Windows is wrong when **'encoding'** equals the current codepage.

Solution: Change the #else block. (Ken Takata)

Files: src/os_win32.c

Patch 7.3.1211

Problem: MS-Windows: When **'encoding'** differs from the current codepage **":hardcopy"** does not work properly.

Solution: Use TextOutW() and SetDlgItemTextW(). (Ken Takata)

Files: src/os_mswin.c, src/vim.rc

Patch 7.3.1212

Problem: "make test" on MS-Windows does not report failure like Unix does.

Solution: Make it work like on Unix. (Taro Muraoka)

Files: src/testdir/Make_dos.mak

Patch 7.3.1213

Problem: Can't build with small features and Python.

Solution: Adjust #ifdefs.

Files: src/eval.c, src/buffer.c, src/eval.c, src/window.c

Patch 7.3.1214

Problem: Missing declaration for init_users() and realloc_post_list(). (Salman Halim)

Solution: Add the declarations.

Files: src/misc1.c, src/regexp_nfa.c

Patch 7.3.1215

Problem: Compiler warning for function not defined.
Solution: Add #ifdef.
Files: src/misc1.c

Patch 7.3.1216

Problem: Configure can't find Motif on Ubuntu.
Solution: Search for libXm in /usr/lib/*-linux-gnu.
Files: src/configure.in, src/auto/configure

Patch 7.3.1217

Problem: New regexp engine: Can't handle %[ao]]. (Yukihiro Nakadaira)
Solution: Support nested atoms inside %[].
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1218

Problem: "make test" on MS-Windows does not clean all temporary files and gives some unnecessary message.
Solution: Clean the right files. Create .failed files. (Ken Takata)
Files: src/testdir/Make_dos.mak

Patch 7.3.1219

Problem: No test for using [] inside %[].
Solution: Add a test.
Files: src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1220

Problem: MS-Windows: When using wide font italic and bold are not included.
Solution: Support wide-bold, wide-italic and wide-bold-italic. (Ken Takata, Taro Muraoka)
Files: src/gui.c, src/gui.h, src/gui_w48.c

Patch 7.3.1221

Problem: When build flags change "make distclean" run into a configure error.
Solution: When CFLAGS changes delete auto/config.cache. Also avoid adding duplicate text to flags. (Ken Takata)
Files: src/Makefile, src/configure.in, src/auto/configure

Patch 7.3.1222

Problem: Cannot execute some tests from the src directly.
Solution: Add missing targets.
Files: src/Makefile

Patch 7.3.1223

Problem: Tests fail on MS-Windows.
Solution: Avoid depending on OS version. Use DOS commands instead of Unix commands. (Taro Muraoka, Ken Takata)
Files: src/testdir/test17.in, src/testdir/test50.in, src/testdir/test71.in, src/testdir/test77.in

Patch 7.3.1224

Problem: Clang gives warnings on xxd.

Solution: Change how to use part of a string. (Dominique Pelle) Also avoid warning for return not reached.
Files: src/xxd/xxd.c, src/regex_nfa.c

Patch 7.3.1225

Problem: Compiler warnings when building with Motif.
Solution: Change set_label() argument. (Kazunobu Kuriyama)
Files: src/gui_motif.c

Patch 7.3.1226

Problem: Python: duplicate code.
Solution: Share code between OutputWrite() and OutputWritelines(). (ZyX)
Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1227

Problem: Inconsistent string conversion.
Solution: Use **'encoding'** instead of utf-8. Use METH_0 in place of METH_VARARGS where appropriate. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1228

Problem: Python: various inconsistencies and problems.
Solution: StringToLine now supports both bytes() and unicode() objects. Make function names consistent. Fix memory leak fixed in StringToLine. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c

Patch 7.3.1229

Problem: Python: not so easy to delete/restore translating.
Solution: Make macros do translation of exception messages. (ZyX)
Note: this breaks translations!
Files: src/if_py_both.h, src/if_python3.c

Patch 7.3.1230

Problem: Python: Exception messages are not clear.
Solution: Make exception messages more verbose. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1231

Problem: Python: use of numbers not consistent.
Solution: Add support for Number protocol. (ZyX)
Files: src/if_py_both.h, src/if_python3.c, src/if_python.c, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1232

Problem: Python: inconsistencies in variable names.
Solution: Rename variables. (ZyX)
Files: src/eval.c, src/if_py_both.h

Patch 7.3.1233

Problem: Various Python problems.
Solution: Fix VimTryEnd. Crash with debug build and PYTHONDUMPREFS=1. Memory leaks in StringToLine(), BufferMark() and convert_dl. (ZyX)

Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.3.1234 (after 7.3.1229)

Problem: Python: Strings are not marked for translation.

Solution: Add N_() where appropriate. (ZyX)

Files: src/if_py_both.h

Patch 7.3.1235

Problem: In insert mode **CTRL-]** is not inserted, on the command-line it is.

Solution: Don't insert **CTRL-]** on the command line. (Yukihiro Nakadaira)

Files: src/ex_getln.c

Patch 7.3.1236

Problem: Python: WindowSetattr() missing support for NUMBER_UNSIGNED.

Solution: Add NUMBER_UNSIGNED, add more tests. Various fixes. (ZyX)

Files: src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/testdir/pythonx/failing.py,
src/testdir/pythonx/failing_import.py, src/testdir/test86.in,
src/testdir/test86.ok, src/testdir/test87.in,
src/testdir/test87.ok, src/testdir/pythonx/topmodule/__init__.py,
src/testdir/pythonx/topmodule/submodule/__init__.py,
src/testdir/pythonx/topmodule/submodule/subsubmodule/__init__.py,
src/testdir/pythonx/topmodule/submodule/subsubmodule/subsubsubmodule.py

Patch 7.3.1237

Problem: Python: non-import errors not handled correctly.

Solution: Let non-ImportError exceptions pass the finder. (ZyX)

Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok

Patch 7.3.1238

Problem: Crash in Python interface on 64 bit machines.

Solution: Change argument type of PyString_AsStringAndSize. (Taro Muraoka,
Jun Takimoto)

Files: src/if_python.c

Patch 7.3.1239

Problem: Can't build with Python and MSVC10.

Solution: Move #if outside of macro. (Taro Muraoka)

Files: src/if_py_both.h

Patch 7.3.1240

Problem: Memory leak in findfile().

Solution: Free the memory. (Christian Brabandt)

Files: src/eval.c

Patch 7.3.1241 (after 7.3.1236)

Problem: Some test files missing from the distribution.

Solution: Update the list of files.

Files: Filelist

Patch 7.3.1242

Problem: No failure when trying to use a number as a string.

Solution: Give an error when StringToLine() is called with an instance of

Files: the wrong type. (Jun Takimoto)
src/if_py_both.h

Patch 7.3.1243

Problem: New regexp engine: back references in look-behind match don't work. (Lech Lorens)
Solution: Copy the submatches before a recursive match. Also fix function prototypes.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1244

Problem: MS-Windows: confirm() dialog text may not fit.
Solution: Use GetTextWidthEnc() instead of GetTextWidth(). (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 7.3.1245

Problem: MS-Windows: confirm() dialog text may still not fit.
Solution: Use GetTextWidthEnc() instead of GetTextWidth() in two more places. (Yasuhiro Matsumoto)
Files: src/gui_w32.c

Patch 7.3.1246

Problem: When setting **'winfixheight'** and resizing the window causes the window layout to be wrong.
Solution: Add frame_check_height() and frame_check_width() (Yukihiro Nakadaira)
Files: src/window.c

Patch 7.3.1247

Problem: New regexp engine: '[]@!\p%([]@!\p\)*:' does not always match.
Solution: When there is a PIM add a duplicate state that starts at another position.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1248

Problem: Still have old hacking code for Input Method.
Solution: Add **'imactivatefunc'** and **'imstatusfunc'** as a generic solution to Input Method activation. (Yukihiro Nakadaira)
Files: runtime/doc/options.txt, src/fileio.c, src/mbyte.c, src/option.c, src/option.h, src/proto/fileio.pro

Patch 7.3.1249

Problem: Modeline not recognized when using "Vim" instead of "vim".
Solution: Also accept "Vim".
Files: src/buffer.c

Patch 7.3.1250

Problem: Python tests fail on MS-Windows.
Solution: Change backslashes to slashes. (Taro Muraoka)
Files: src/testdir/test86.in, src/testdir/test87.in

Patch 7.3.1251

Problem: Test 61 messes up viminfo.

Solution: Specify a separate viminfo file.
Files: src/testdir/test61.in

Patch 7.3.1252

Problem: gvim does not find the toolbar bitmap files in ~/vimfiles/bitmaps if the corresponding menu command contains additional characters like the shortcut marker '&' or if you use a non-english locale.
Solution: Use menu->en_dname or menu->dname. (Martin Giesecking)
Files: src/gui_w32.c

Patch 7.3.1253 (after 7.3.1200)

Problem: Still undo problem after using **CTRL-R** = setline(). (Hirohito Higashi)
Solution: Set the ins_need_undo flag.
Files: src/edit.c

Patch 7.3.1254 (after 7.3.1252)

Problem: Can't build without the multi-lang feature. (John Marriott)
Solution: Add #ifdef.
Files: src/gui_w32.c

Patch 7.3.1255

Problem: Clang warnings when building with Athena.
Solution: Add type casts. (Dominique Pelle)
Files: src/gui_at_fs.c

Patch 7.3.1256

Problem: Can't build without eval or autocmd feature.
Solution: Add #ifdefs.
Files: src/mbyte.c, src/window.c

Patch 7.3.1257

Problem: With GNU gettext() ":lang de_DE.utf8" does not always result in German messages.
Solution: Clear the \$LANGUAGE environment variable.
Files: src/ex_cmds2.c

Patch 7.3.1258

Problem: Using submatch() may crash Vim. (Ingo Karkat)
Solution: Restore the number of subexpressions used.
Files: src/regexp_nfa.c

Patch 7.3.1259

Problem: No test for patch 7.3.1258
Solution: Add a test entry.
Files: src/testdir/test64.in, src/testdir/test64.ok

Patch 7.3.1260

Problem: User completion does not get the whole command line in the command line window.
Solution: Pass on the whole command line. (Daniel Thau)
Files: src/ex_getln.c, src/structs.h

Patch 7.3.1261 (after patch 7.3.1179)

Problem: A buffer-local language mapping from a keymap stops a global insert mode mapping from working. (Ron Aaron)
Solution: Do not wait for more characters to be typed only when the mapping was defined with `<nowait>`.
Files: runtime/doc/map.txt, src/eval.c, src/getchar.c, src/testdir/test75.in, src/testdir/test75.ok

Patch 7.3.1262

Problem: Crash and compilation warnings with Cygwin.
Solution: Check return value of `XmbTextListToTextProperty()`. Add type casts. Adjust `#ifdefs`. (Lech Lorens)
Files: src/main.c, src/os_unix.c, src/ui.c

Patch 7.3.1263

Problem: Typo in short option name.
Solution: Change "imse" to "imsf".
Files: src/option.c

Patch 7.3.1264 (after 7.3.1261)

Problem: Missing `m_nowait`.
Solution: Include missing part of the patch.
Files: src/structs.h

Patch 7.3.1265 (after 7.3.1249)

Problem: Accepting "Vim:" for a modeline causes errors too often.
Solution: Require "Vim:" to be followed by "set".
Files: src/buffer.c

Patch 7.3.1266

Problem: QNX: GUI fails to start.
Solution: Remove the QNX-specific `#ifdef`. (Sean Boudreau)
Files: src/gui.c

Patch 7.3.1267

Problem: MS-Windows ACL support doesn't work well.
Solution: Implement more ACL support. (Ken Takata)
Files: src/os_win32.c

Patch 7.3.1268

Problem: ACL support doesn't work when compiled with MingW.
Solution: Support ACL on MingW. (Ken Takata)
Files: src/os_win32.c, src/os_win32.h

Patch 7.3.1269

Problem: Insert completion keeps entry selected even though the list has changed. (Olivier Teuliere)
Solution: Reset `compl_shown_match` and `compl_curr_match`. (Christian Brabandt)
Files: src/edit.c

Patch 7.3.1270

Problem: Using "Vp" in an empty buffer can't be undone. (Hauke Petersen)
Solution: Save one line in an empty buffer. (Christian Brabandt)
Files: src/ops.c

Patch 7.3.1271 (after 7.3.1260)

Problem: Command line completion does not work.
Solution: Move setting xp_line down. (Daniel Thau)
Files: src/ex_getln.c

Patch 7.3.1272

Problem: Crash when editing Ruby file. (Aliaksandr Rahalevich)
Solution: Reallocate the state list when necessary.
Files: src/regexp_nfa.c

Patch 7.3.1273

Problem: When copying a location list the index might be wrong.
Solution: Set the index to one when using the first entry. (Lech Lorens)
Files: src/quickfix.c

Patch 7.3.1274

Problem: When selecting an entry from a location list it may pick an arbitrary window or open a new one.
Solution: Prefer using a window related to the location list. (Lech Lorens)
Files: src/quickfix.c

Patch 7.3.1275

Problem: "gn" does not work when the match is a single character.
Solution: Fix it, add a test. (Christian Brabandt)
Files: src/search.c, src/testdir/test53.in, src/testdir/test53.ok

Patch 7.3.1276

Problem: When using a cscope connection resizing the window may send SIGWINCH to cscope and it quits.
Solution: Call setpgid(0, 0) in the child process. (Narendran Gopalakrishnan)
Files: src/if_cscope.c

Patch 7.3.1277

Problem: In diff mode '**cursorline**' also draws in the non-active window. When '**nu**' and '**sbr**' are set the '**sbr**' string is not underlined.
Solution: Only draw the cursor line in the current window. Combine the '**cursorline**' and other highlighting attributes. (Christian Brabandt)
Files: src/screen.c

Patch 7.3.1278

Problem: When someone sets the screen size to a huge value with "stty" Vim runs out of memory before reducing the size.
Solution: Limit Rows and Columns in more places.
Files: src/gui.c, src/gui_gtk_x11.c, src/option.c, src/os_unix.c, src/proto/term.pro, src/term.c

Patch 7.3.1279

Problem: Compiler warning for variable uninitialized. (Tony Mechelynck)
Solution: Add an init.
Files: src/ex_getln.c

Patch 7.3.1280

Problem: Reading memory already freed since patch 7.3.1247. (Simon

Ruderich, Dominique Pelle)
Solution: Copy submatches before reallocating the state list.
Files: src/regexp_nfa.c

Patch 7.3.1281

Problem: When **'ttymouse'** is set to "xterm2" clicking in column 123 moves the cursor to column 96. (Kevin Goodsell)
Solution: Decode KE_CSI.
Files: src/term.c

Patch 7.3.1282 (after 7.3.1277)

Problem: **'cursorline'** not drawn in any other window. (Charles Campbell)
Solution: Do draw the cursor line in other windows.
Files: src/screen.c

Patch 7.3.1283

Problem: Test 71 fails on MS-Windows.
Solution: Put the binary data in a separate file. (Ken Takata)
Files: src/testdir/test71.in, src/testdir/test71a.in

Patch 7.3.1284

Problem: Compiler warnings in MS-Windows clipboard handling.
Solution: Add type casts. (Ken Takata)
Files: src/winclip.c

Patch 7.3.1285

Problem: No tests for picking a window when selecting an entry in a location list. Not picking the right window sometimes.
Solution: Add test 96. Set usable_win appropriately. (Lech Lorens)
Files: src/quickfix.c, src/testdir/Makefile, src/testdir/test96.in, src/testdir/test96.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.3.1286

Problem: Check for screen size missing for Athena and Motif.
Solution: Add call to limit_screen_size().
Files: src/gui_x11.c

Patch 7.3.1287

Problem: Python SystemExit exception is not handled properly.
Solution: Catch the exception and give an error. (Yasuhiro Matsumoto, Ken Takata)
Files: runtime/doc/if_pyth.txt, src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.3.1288

Problem: The first `":echo 'hello'"` command output doesn't show. Mapping for `<S-F3>` gets triggered during startup.
Solution: Add debugging code for the termresponse. When receiving the "Co" entry and when setting **'ambiwidht'** redraw right away if possible. Add redraw_asap(). Don't set **'ambiwidht'** if it already had the right value. Do the **'ambiwidht'** check in the second row to avoid confusion with `<S-F3>`.

Files: src/term.c, src/screen.c, src/proto/screen.pro

Patch 7.3.1289

Problem: Get GLIB warning when removing a menu item.

Solution: Reference menu-id and also call gtk_container_remove(). (Ivan Krasilnikov)

Files: src/gui_gtk.c

Patch 7.3.1290 (after 7.3.1253)

Problem: **CTRL-R** = in Insert mode changes the start of the insert position. (Ingo Karkat)

Solution: Only break undo, don't start a new insert.

Files: src/edit.c

Patch 7.3.1291 (after 7.3.1288)

Problem: Compiler warnings for uninitialized variables. (Tony Mechelynck)

Solution: Initialize the variables.

Files: src/screen.c

Patch 7.3.1292

Problem: Possibly using invalid pointer when searching for window. (Raichoo)

Solution: Use "firstwin" instead of "tp_firstwin" for current tab.

Files: src/window.c

Patch 7.3.1293

Problem: Put in empty buffer cannot be undone.

Solution: Save one more line for undo. (Ozaki)

Files: src/ops.c

Patch 7.3.1294

Problem: ":diffoff" resets options.

Solution: Save and restore option values. (Christian Brabandt)

Files: src/diff.c, src/structs.h, src/option.c

Patch 7.3.1295

Problem: glob() and globpath() do not handle escaped special characters properly.

Solution: Handle escaped characters differently. (Adnan Zafar)

Files: src/testdir/Makefile, src/testdir/test97.in, src/testdir/test97.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/fileio.c, src/misc1.c

Patch 7.3.1296

Problem: Only MS-Windows limits the GUI window size to what fits on the monitor.

Solution: Limit the size for all systems. (Daniel Harding)

Files: src/ui.c

Patch 7.3.1297

Problem: findfile() directory matching does not work when a star follows text. (Markus Braun)

Solution: Make a wildcard work properly. (Christian Brabandt)

Files: src/misc2.c, src/testdir/test89.in, src/testdir/test89.ok

Patch 7.3.1298 (after 7.3.1297)

Problem: Crash.

Solution: Use STRCPY() instead of STRCAT() and allocate one more byte.

Files: src/misc2.c

Patch 7.3.1299

Problem: Errors when doing "make proto". Didn't do "make depend" for a while.

Solution: Add #ifdefs. Update dependencies. Update proto files.

Files: src/if_python3.c, src/os_win32.c, src/Makefile,
src/proto/ex_docmd.pro, src/proto/if_python.pro,
src/proto/if_python3.pro, src/proto/gui_w16.pro,
src/proto/gui_w32.pro, src/proto/os_win32.pro

Patch 7.3.1300

Problem: Mac: tiny and small build fails.

Solution: Don't include os_macosx.m in tiny build. Include mouse support in small build. (Kazunobu Kuriyama)

Files: src/configure.in, src/auto/configure, src/vim.h

Patch 7.3.1301

Problem: Some tests fail on MS-Windows.

Solution: Fix path separators in test 89 and 96. Omit test 97, escaping works differently. Make findfile() work on MS-Windows.

Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/test89.in,
src/testdir/test96.in, src/misc2.c

Patch 7.3.1302

Problem: Test 17 fails on MS-Windows. Includes line break in file name everywhere.

Solution: Fix '**fileformat**'. Omit CR-LF from a line read from an included file.

Files: src/search.c, src/testdir/test17.in, src/testdir/test17.ok

Patch 7.3.1303 (after 7.3.1290)

Problem: Undo is synced whenever **CTRL-R** = is called, breaking some plugins.

Solution: Only break undo when calling setline() or append().

Files: src/globals.h, src/eval.c, src/edit.c, src/testdir/test61.in,
src/testdir/test61.ok

Patch 7.3.1304

Problem: Test 89 still fails on MS-Windows.

Solution: Set '**shellslash**'. (Taro Muraoka)

Files: src/testdir/test89.in

Patch 7.3.1305

Problem: Warnings from 64 bit compiler.

Solution: Add type casts.

Files: src/misc2.c

Patch 7.3.1306

Problem: When redrawing the screen during startup the intro message may be cleared.
Solution: Redisplay the intro message when appropriate.
Files: src/screen.c, src/version.c, src/proto/version.pro

Patch 7.3.1307

Problem: MS-Windows build instructions are outdated.
Solution: Adjust for building on Windows 7. Drop Windows 95/98/ME support.
Files: Makefile, nsis/gvim.nsi

Patch 7.3.1308

Problem: Typos in MS-Windows build settings and README.
Solution: Minor changes to MS-Windows files.
Files: src/msvc2008.bat, src/msvc2010.bat, src/VisVim/README_VisVim.txt

Patch 7.3.1309

Problem: When a script defines a function the flag to wait for the user to hit enter is reset.
Solution: Restore the flag. (Yasuhiro Matsumoto) Except when the user was typing the function.
Files: src/eval.c

Patch 7.3.1310

Problem: Typos in nsis script. Can use better compression.
Solution: Fix typos. Use lzma compression. (Ken Takata)
Files: nsis/gvim.nsi

Patch 7.3.1311

Problem: Compiler warnings on Cygwin.
Solution: Add type casts. Add windows include files. (Ken Takata)
Files: src/mbyte.c, src/ui.c

Patch 7.3.1312 (after 7.3.1287)

Problem: Not giving correct error messages for SystemExit().
Solution: Move E858 into an else. (Ken Takata)
Files: src/if_py_both.h

Patch 7.3.1313

Problem: :py and :py3 don't work when compiled with Cygwin or MingW with 64 bit.
Solution: Add -DMS_WIN64 to the build command. (Ken Takata)
Files: src/Make_cyg.mak, src/Make_ming.mak

Patch 7.3.1314

Problem: Test 87 fails with Python 3.3.
Solution: Filter the error messages. (Taro Muraoka)
Files: src/testdir/test87.in

Patch 7.4a.001

Problem: Script to update syntax menu is outdated.
Solution: Add the missing items.
Files: runtime/makemenu.vim

Patch 7.4a.002

Problem: Valgrind errors in test 89. (Simon Ruderich)
Solution: Allocate one more byte. (Dominique Pelle)
Files: src/misc2.c

Patch 7.4a.003

Problem: Copyright year is outdated.
Solution: Only use the first year.
Files: src/vim.rc, src/vim16.rc

Patch 7.4a.004

Problem: MSVC 2012 Update 3 is not recognized.
Solution: Add the version number. (Raymond Ko)
Files: src/Make_mvc.mak

Patch 7.4a.005

Problem: Scroll binding causes unexpected scroll.
Solution: Store the topline after updating scroll binding. Add a test.
(Lech Lorens)
Files: src/testdir/test98.in, src/testdir/test98a.in,
src/testdir/test98.ok, src/option.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile

Patch 7.4a.006

Problem: Failure in po file check goes unnoticed.
Solution: Fail "make test" if the po file check fails.
Files: src/Makefile

Patch 7.4a.007

Problem: After "g\$" with '**virtualedit**' set, "k" moves to a different
column. (Dimitar Dimitrov)
Solution: Set w_curswant. (Christian Brabandt)
Files: src/normal.c

Patch 7.4a.008

Problem: Python 3 doesn't handle multibyte characters properly when
'**encoding**' is not utf-8.
Solution: Use PyUnicode_Decode() instead of PyUnicode_FromString(). (Ken
Takata)
Files: src/if_python3.c

Patch 7.4a.009

Problem: Compiler warnings for function prototypes.
Solution: Add "void". Move list_features() prototype. (Ken Takata)
Files: src/gui_w48.c, src/if_py_both.h, src/version.c

Patch 7.4a.010

Problem: Test 86 and 87 fail when building with Python or Python 3 and
using a static library.
Solution: Add configure check to add -fPIE compiler flag.
Files: src/configure.in, src/auto/configure

Patch 7.4a.011

Problem: Configure check for Python 3 config name isn't right.
Solution: Always include `vi_cv_var_python3_version`. (Tim Harder)
Files: `src/configure.in`, `src/auto/configure`

Patch 7.4a.012

Problem: "make test" fails when using a shadow directory.
Solution: Create links for files in `src/po`. (James McCoy)
Files: `src/Makefile`

Patch 7.4a.013

Problem: Setting/resetting `'\lbr'` in the main help file changes alignment after a Tab. (Dimitar Dimitrov)
Solution: Also use the code for conceal mode where `n_extra` is computed for `'\lbr'`.
Files: `src/screen.c`, `src/testdir/test88.in`, `src/testdir/test88.ok`

Patch 7.4a.014

Problem: Test 86 and 89 have a problem with using a shadow dir.
Solution: Adjust for the different directory structure. (James McCoy)
Files: `src/testdir/test89.in`, `src/testdir/test86.in`, `src/Makefile`

Patch 7.4a.015

Problem: No Japanese man pages.
Solution: Add Japanese translations of man pages. (Ken Takata, Yukihiro Nakadaira, et al.)
Files: `Filelist`, `src/Makefile`, `runtime/doc/evim-ja.UTF-8.1`,
`runtime/doc/vim-ja.UTF-8.1`, `runtime/doc/vimdiff-ja.UTF-8.1`,
`runtime/doc/vimtutor-ja.UTF-8.1`, `runtime/doc/xxd-ja.UTF-8.1`

Patch 7.4a.016 (after 7.4a.014)

Problem: Features enabled in `Makefile`.
Solution: Undo accidental changes.
Files: `src/Makefile`

Patch 7.4a.017

Problem: When `'foldmethod'` is "indent", using ">>" on a line just above a fold makes the cursor line folded. (Evan Laforge)
Solution: Call `foldOpenCursor()`. (Christian Brabandt)
Files: `src/ops.c`

Patch 7.4a.018

Problem: Compiler warning for code unreachable. (Charles Campbell)
Solution: Use "while" instead of endless loop. Change break to continue.
Files: `src/regexp_nfa.c`, `src/ui.c`

Patch 7.4a.019

Problem: Invalid closing parenthesis in test 62. Command truncated at double quote.
Solution: Remove the parenthesis. Change double quote to `'`. (ZyX)
Files: `src/testdir/test62.in`, `src/testdir/test62.ok`

Patch 7.4a.020

Problem: Superfluous `mb_ptr_adv()`.
Solution: Remove the call. (Dominique Pelle)

Files: src/regexp_nfa.c

Patch 7.4a.021

Problem: Using feedkeys() doesn't always work.

Solution: Omit feedkeys(). (Ken Takata)

Files: src/testdir/test98a.in

Patch 7.4a.022

Problem: Using "d2g\$" does not delete the last character. (ZyX)

Solution: Set the "inclusive" flag properly.

Files: src/normal.c

Patch 7.4a.023 (after 7.4a.019)

Problem: Still another superfluous parenthesis. (ZyX)

Solution: Remove it.

Files: src/testdir/test62.in

Patch 7.4a.024

Problem: X11 GUI: Checking icon height twice.

Solution: Check height and width. (Dominique Pelle)

Files: src/gui_x11.c

Patch 7.4a.025

Problem: Get the press-Enter prompt even after using :redraw.

Solution: Clear need_wait_return when executing :redraw.

Files: src/ex_docmd.c

Patch 7.4a.026

Problem: ":diffoff" does not remove folds. (Ramel)

Solution: Do not restore '**foldenable**' when '**foldmethod**' is "manual".

Files: src/diff.c

Patch 7.4a.027

Problem: When Python adds lines to another buffer the cursor position is wrong, it might be below the last line causing ml_get errors. (Vlad Irnov)

Solution: Temporarily change the current window, so that marks are corrected properly.

Files: src/if_py_both.h, src/window.c, src/proto/buffer.pro

Patch 7.4a.028

Problem: Crash when spell checking in new buffer.

Solution: Set the b_p_key field. (Mike Williams)

Files: src/spell.c, src/testdir/test58.in

Patch 7.4a.029

Problem: Can't build with MzScheme on Ubuntu 13.04.

Solution: Add configure check for the "ffi" library.

Files: src/configure.in, src/auto/configure

Patch 7.4a.030 (after 7.4.027)

Problem: Missing find_win_for_buf(). (toothpik)

Solution: Add missing changes.

Files: src/buffer.c

Patch 7.4a.031

Problem: Compiler warnings. (Charles Campbell)
Solution: Initialize variables even when not needed.
Files: src/regexp_nfa.c, src/search.c

Patch 7.4a.032

Problem: New regexp engine: Does not match shorter alternative. (Ingo Karkat)
Solution: Do not drop a new state when the PIM info is different.
Files: src/regexp_nfa.c

Patch 7.4a.033

Problem: Test 98 always passes.
Solution: Include test98a.in in test98.in, execute the crucial command in one line. (Yukihiro Nakadaira)
Files: src/testdir/test98.in, src/testdir/test98a.in

Patch 7.4a.034

Problem: The tabline may flicker when opening a new tab after 7.3.759 on Win32.
Solution: Move call to TabCtrl_SetCurSel(). (Ken Takata)
Files: src/gui_w48.c

Patch 7.4a.035

Problem: Fix in patch 7.4a.032 is not tested.
Solution: Add test.
Files: src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4a.036

Problem: "\p" in a regexp does not match double-width characters. (Yukihiro Nakadaira)
Solution: Don't count display cells, use vim_isprintc().
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.4a.037

Problem: Win32: When mouse is hidden and in the toolbar, moving it won't make it appear. (Sami Salonen)
Solution: Add tabline_wndproc() and toolbar_wndproc(). (Ken Takata)
Files: src/gui_w32.c, src/gui_w48.c

Patch 7.4a.038

Problem: When using MSVC 2012 there are various issues, including GUI size computations.
Solution: Use SM_CXPADDED BORDER. (Mike Williams)
Files: src/gui_w32.c, src/gui_w48.c, src/os_win32.h

Patch 7.4a.039

Problem: New regexp engine doesn't match pattern. (Ingo Karkat)
Solution: When adding a state also check for different PIM if the list of states has any state with a PIM.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4a.040

Problem: Win32: using uninitialized variable.
Solution: (Yukihiro Nakadaira)
Files: src/os_win32.c

Patch 7.4a.041

Problem: When using ":new ++ff=unix" and "dos" is first in **'fileformats'** then **'ff'** is set to "dos" instead of "unix". (Ingo Karkat)
Solution: Create set_file_options() and invoke it from do_ecmd().
Files: src/fileio.c, src/proto/fileio.pro, src/ex_cmds.c, src/testdir/test91.in, src/testdir/test91.ok

Patch 7.4a.042

Problem: Crash when BufUnload autocommands close all buffers. (Andrew Pimlott)
Solution: Set curwin->w_buffer to curbuf to avoid NULL.
Files: src/window.c, src/testdir/test8.in, src/testdir/test8.ok

Patch 7.4a.043

Problem: More ml_get errors when adding or deleting lines from Python. (Vlad Irnov)
Solution: Switch to a window with the buffer when possible.
Files: src/if_py_both.h

Patch 7.4a.044

Problem: Test 96 sometimes fails.
Solution: Clear window from b_wininfo in win_free(). (Suggestion by Yukihiro Nakadaira)
Files: src/window.c

Patch 7.4a.045

Problem: Configure does not always find the right library for Lua. Missing support for LuaJit.
Solution: Improve the configure detection of Lua. (Hiroshi Shirosaki)
Files: src/Makefile, src/configure.in, src/auto/configure

Patch 7.4a.046

Problem: Can't build without mbyte feature.
Solution: Add #ifdefs.
Files: src/ex_cmds.c

Patch 7.4a.047

Problem: Some comments are not so nice.
Solution: Change the comments.
Files: src/ex_docmd.c, src/message.c, src/ops.c, src/option.c

Patch 7.4b.001

Problem: Win32: dialog may extend off-screen.
Solution: Reduce the size, use correct borders. (Andrei Olsen)
Files: src/gui_w32.c

Patch 7.4b.002

Problem: Crash searching for `\%(\\%(\|\\d\|-\\|\.)*\|*\)`. (Marcin

Szamotołski) Also for `\(\)*`.
Solution: Do add a state for opening parenthesis, so that we can check if it was added before at the same position.
Files: `src/regexp_nfa.c`, `src/testdir/test64.in`, `src/testdir/test64.ok`

Patch 7.4b.003

Problem: Regexp code is not nicely aligned.
Solution: Adjust white space. (Ken Takata)
Files: `src/regexp_nfa.c`

Patch 7.4b.004

Problem: Regexp crash on pattern `"@%[\w\-\]*"`. (Axel Kielhorn)
Solution: Add `\(\)` around `\%[]` internally.
Files: `src/regexp_nfa.c`, `src/testdir/test64.in`, `src/testdir/test64.ok`

Patch 7.4b.005

Problem: Finding `%s` in `shellpipe` and `shellredir` does not ignore `%%s`.
Solution: Skip over `%%`. (lcd 47)
Files: `src/ex_cmds.c`

Patch 7.4b.006 (after 7.3.1295)

Problem: Using `\{n,m}` in an autocommand pattern no longer works. Specifically, `mutt` temp files are not recognized. (Gary Johnson)
Solution: Make `\\{n,m}` work.
Files: `runtime/doc/autocmd.txt`, `src/fileio.c`

Patch 7.4b.007

Problem: On 32 bit MS-Windows `:perl` does not work.
Solution: Make sure `time_t` uses 32 bits. (Ken Takata)
Files: `src/if_perl.xs`, `src/vim.h`

Patch 7.4b.008

Problem: **'autochdir'** causes `setbufvar()` to change the current directory. (Ben Fritz)
Solution: When disabling autocommands also reset **'acd'** temporarily. (Christian Brabandt)
Files: `src/fileio.c`

Patch 7.4b.009

Problem: When setting the Visual area manually and **'selection'** is exclusive, a yank includes one character too much. (Ingo Karkat)
Solution: Default the Visual operation to `"v"`. (Christian Brabandt)
Files: `src/mark.c`

Patch 7.4b.010

Problem: Win32: Tcl library load does not use standard mechanism.
Solution: Call `vimLoadLib()` instead of `LoadLibraryEx()`. (Ken Takata)
Files: `src/if_perl.xs`, `src/if_tcl.c`

Patch 7.4b.011

Problem: `":he \%(\\)"` does not work. (ZyX)
Solution: Add an exception to the list.
Files: `src/ex_cmds.c`

Patch 7.4b.012

Problem: Output from a shell command is truncated at a NUL. (lcd 47)
Solution: Change every NUL into an SOH.
Files: src/misc1.c

Patch 7.4b.013

Problem: Install dir for JP man pages is wrong.
Solution: Remove ".UTF-8" from the directory name. (Ken Takata)
Files: src/Makefile

Patch 7.4b.014 (after 7.4b.012)

Problem: Stupid mistake.
Solution: Changle "len" to "i".
Files: src/misc1.c

Patch 7.4b.015 (after 7.4b.008)

Problem: Can't compile without the 'acd' feature.
Solution: Add #ifdefs. (Kazunobu Kuriyama)
Files: src/fileio.c

Patch 7.4b.016

Problem: Ruby detection fails on Fedora 19.
Solution: Use one way to get the Ruby version. (Michael Henry)
Files: src/configure.in, src/auto/configure

Patch 7.4b.017

Problem: ":he ^x" gives a strange error message. (glts)
Solution: Do not translate ^x to _CTRL-x.
Files: src/ex_cmds.c

Patch 7.4b.018 (after 7.4b.001)

Problem: Win32: Dialog can still be too big.
Solution: Move the check for height further down. (Andrei Olsen)
Files: src/gui_w32.c

Patch 7.4b.019 (after 7.4a.034)

Problem: Tabline is not updated properly when closing a tab on Win32.
Solution: Only reduce flickering when adding a tab. (Ken Takata)
Files: src/gui_w48.c

Patch 7.4b.020

Problem: "g~ap" changes first character of next paragraph. (Manuel Ortega)
Solution: Avoid subtracting (0 - 1) from todo. (Mike Williams)
Files: src/ops.c, src/testdir/test82.in, src/testdir/test82.ok

Patch 7.4b.021

Problem: Pressing "u" after an external command results in multiple press-enter messages. (glts)
Solution: Don't call hit_return_msg() when we have K_IGNORE. (Christian Brabandt)
Files: src/message.c

Patch 7.4b.022

Problem: Not waiting for a character when the tick count overflows.

Solution: Subtract the unsigned numbers and cast to int. (Ken Takata)
Files: src/os_win32.c

VIM REFERENCE MANUAL by Bram Moolenaar

vim8 vim-8 version-8.0 version8.0

Welcome to Vim 8! A large number of bugs have been fixed and several nice features have been added. This file mentions all the new items and changes to existing features since Vim 7.4. The patches up to Vim 7.4 can be found here: vim-7.4 .

Use this command to see the full version and features information of the Vim program you are using:

:version

NEW FEATURES	new-8
Vim script enhancements	new-vim-script-8
Various new items	new-items-8
INCOMPATIBLE CHANGES	incompatible-8
IMPROVEMENTS	improvements-8
COMPILE TIME CHANGES	compile-changes-8
PATCHES	patches-8
VERSION 8.1	version-8.1
Changed	changed-8.1
Added	added-8.1
Patches	patches-8.1
VERSION 8.2	version-8.2
Changed	changed-8.2
Added	added-8.2
Patches	patches-8.2

See vi_diff.txt for an overview of differences between Vi and Vim 8.0. See version4.txt , version5.txt , version6.txt and version7.txt for differences between other versions.

vim-changelog

You can find an overview of the most important changes (according to Martin Tournoij) on this site: https://www.arp242.net/vimlog/

===== NEW FEATURES new-8

First an overview of the more interesting new features. A comprehensive list is below.

Asynchronous I/O support, channels

Vim can now exchange messages with other processes in the background. This makes it possible to have servers do work and send back the results to Vim. See `channel-demo` for an example, this shows communicating with a Python server.

Closely related to channels is JSON support. JSON is widely supported and can easily be used for inter-process communication, allowing for writing a server in any language. The functions to use are `json_encode()` and `json_decode()` .

This makes it possible to build very complex plugins, written in any language and running in a separate process.

Jobs

Vim can now start a job, communicate with it and stop it. This is very useful to run a process for completion, syntax checking, etc. Channels are used to communicate with the job. Jobs can also read from or write to a buffer or a file. See `job_start()` .

Timers

Also asynchronous are timers. They can fire once or repeatedly and invoke a function to do any work. For example:

```
let tempTimer = timer_start(4000, 'CheckTemp')
```

This will call the `CheckTemp()` function four seconds (4000 milliseconds) later. See `timer_start()` .

Partials

Vim already had a `Funcref`, a reference to a function. A partial also refers to a function, and additionally binds arguments and/or a dictionary. This is especially useful for callbacks on channels and timers. E.g., for the timer example above, to pass an argument to the function:

```
let tempTimer = timer_start(4000, function('CheckTemp', ['out']))
```

This will call `CheckTemp('out')` four seconds later.

Lambda and Closure

A short way to create a function has been added: `{args -> expr}`. See `lambda` . This is useful for functions such as `filter()` and `map()`, which now also accept a function argument. Example:

```
:call filter(mylist, {idx, val -> val > 20})
```

A lambda can use variables defined in the scope where the lambda is defined. This is usually called a `closure` .

User defined functions can also be a closure by adding the "closure" argument `:func-closure` .

Packages

Plugins keep growing and more of them are available than ever before. To keep the collection of plugins manageable package support has been added. This is a convenient way to get one or more plugins, drop them in a directory and possibly keep them updated. Vim will load them automatically, or only when desired. See [packages](#) .

New style tests

This is for Vim developers. So far writing tests for Vim has not been easy. Vim 8 adds assert functions and a framework to run tests. This makes it a lot simpler to write tests and keep them updated. Also new are several functions that are added specifically for testing. See [test-functions](#) .

Window IDs

Previously windows could only be accessed by their number. And every time a window would open, close or move that number changes. Each window now has a unique ID, so that they are easy to find. See [win_getid\(\)](#) and [win_id2win\(\)](#) .

Viminfo uses timestamps

Previously the information stored in viminfo was whatever the last Vim wrote there. Now timestamps are used to always keep the most recent items. See [viminfo-timestamp](#) .

Wrapping lines with indent

The **'breakindent'** option has been added to be able to wrap lines without changing the amount of indent.

Windows: DirectX support

This adds the **'renderoptions'** option to allow for switching on DirectX (DirectWrite) support on MS-Windows.

GTK+ 3 support

The GTK+ 3 GUI works just like GTK+ 2 except for hardly noticeable technical differences between them. Configure still chooses GTK+ 2 if both 2 and 3 are available. See `src/Makefile` for how to use GTK+ 3 instead. See [gui-x11-compiling](#) for other details.

Vim script enhancements

[new-vim-script-8](#)

In Vim script the following types have been added:

Special	<code>v:false</code> , <code>v:true</code> , <code>v:none</code> and <code>v:null</code>
Channel	connection to another process for asynchronous I/O
Job	process control

Many functions and commands have been added to support the new types.

On some systems the numbers used in Vim script are now 64 bit. This can be checked with the `+num64` feature.

Many items were added to support `new-style-testing` .

`printf()` now accepts any type of argument for `%s`. It is converted to a string like with `string()`.

Various new items

[new-items-8](#)

Visual mode commands:

<code>v_CTRL-A</code>	CTRL-A	add N to number in highlighted text
<code>v_CTRL-X</code>	CTRL-X	subtract N from number in highlighted text
<code>v_g_CTRL-A</code>	g CTRL-A	add N to number in highlighted text
<code>v_g_CTRL-X</code>	g CTRL-X	subtract N from number in highlighted text

Insert mode commands:

<code>i_CTRL-G_U</code>	CTRL-G U	don't break undo with next cursor movement
-------------------------	-----------------	--

Cmdline mode commands:

<code>/_CTRL-G</code>	CTRL-G	move to the next match in 'incsearch' mode
<code>/_CTRL-T</code>	CTRL-T	move to the previous match in 'incsearch' mode

Options:

'belloff'	do not ring the bell for these reasons
'breakindent'	wrapped line repeats indent
'breakindentopt'	settings for 'breakindent' .
'emoji'	emoji characters are considered full width
'fixendofline'	make sure last line in file has <code><EOL></code>
'langremap'	do apply 'langmap' to mapped characters
'luadll'	name of the Lua dynamic library
'packpath'	list of directories used for packages
'perldll'	name of the Perl dynamic library
'pythondll'	name of the Python 2 dynamic library
'pythonthreedll'	name of the Python 3 dynamic library

'renderoptions'	options for text rendering on Windows
'rubydll'	name of the Ruby dynamic library
'signcolumn'	when to display the sign column
'tagcase'	how to handle case when searching in tags files
'tcldll'	name of the Tcl dynamic library
'termguicolors'	use GUI colors for the terminal

Ex commands:

:cbottom	scroll to the bottom of the quickfix window
:cdo	execute command in each valid error list entry
:cfdo	execute command in each file in error list
:chistory	display quickfix list stack
:clearjumps	clear the jump list
:filter	only output lines that (do not) match a pattern
:helpclose	close one help window
:lbottom	scroll to the bottom of the location window
:ldo	execute command in valid location list entries
:lfdo	execute command in each file in location list
:lhistory	display location list stack
:noswapfile	following commands don't create a swap file
:packadd	add a plugin from 'packpath'
:packloadall	load all packages under 'packpath'
:smile	make the user happy

Ex command modifiers:

:keeppatterns	following command keeps search pattern history
<mods>	supply command modifiers to user defined commands

New and extended functions:

arglistid()	get id of the argument list
assert_equal()	assert that two expressions values are equal
assert_exception()	assert that a command throws an exception
assert_fails()	assert that a function call fails
assert_false()	assert that an expression is false
assert_inrange()	assert that an expression is inside a range
assert_match()	assert that a pattern matches the value
assert_notequal()	assert that two expressions values are not equal
assert_notmatch()	assert that a pattern does not match the value
assert_true()	assert that an expression is true
bufwinid()	get the window ID of a specific buffer
byteidxcomp()	like byteidx() but count composing characters
ch_close()	close a channel
ch_close_in()	close the in part of a channel
ch_evalexp()	evaluates an expression over channel
ch_evalraw()	evaluates a raw string over channel
ch_getbufnr()	get the buffer number of a channel
ch_getjob()	get the job associated with a channel
ch_info()	get channel information

<code>ch_log()</code>	write a message in the channel log file
<code>ch_logfile()</code>	set the channel log file
<code>ch_open()</code>	open a channel
<code>ch_read()</code>	read a message from a channel
<code>ch_readraw()</code>	read a raw message from a channel
<code>ch_sendexpr()</code>	send a JSON message over a channel
<code>ch_sendraw()</code>	send a raw message over a channel
<code>ch_setopts()</code>	set the options for a channel
<code>ch_status()</code>	get status of a channel
<code>execute()</code>	execute an Ex command and get the output
<code>exepath()</code>	full path of an executable program
<code>funcref()</code>	return a reference to function {name}
<code>getbufinfo()</code>	get a list with buffer information
<code>getcharsearch()</code>	return character search information
<code>getcmdwintype()</code>	return the current command-line window type
<code>getcompletion()</code>	return a list of command-line completion matches
<code>getcurpos()</code>	get position of the cursor
<code>gettabinfo()</code>	get a list with tab page information
<code>getwininfo()</code>	get a list with window information
<code>glob2regpat()</code>	convert a glob pattern into a search pattern
<code>isnan()</code>	check for not a number
<code>job_getchannel()</code>	get the channel used by a job
<code>job_info()</code>	get information about a job
<code>job_setopts()</code>	set options for a job
<code>job_start()</code>	start a job
<code>job_status()</code>	get the status of a job
<code>job_stop()</code>	stop a job
<code>js_decode()</code>	decode a JSON string to Vim types
<code>js_encode()</code>	encode an expression to a JSON string
<code>json_decode()</code>	decode a JSON string to Vim types
<code>json_encode()</code>	encode an expression to a JSON string
<code>matchaddpos()</code>	define a list of positions to highlight
<code>matchstrpos()</code>	match and positions of a pattern in a string
<code>perlevel()</code>	evaluate Perl expression
<code>reltimefloat()</code>	convert reltime() result to a Float
<code>setcharsearch()</code>	set character search information
<code>setfperm()</code>	set the permissions of a file
<code>strcharpart()</code>	get part of a string using char index
<code>strgetchar()</code>	get character from a string using char index
<code>systemlist()</code>	get the result of a shell command as a list
<code>test_alloc_fail()</code>	make memory allocation fail
<code>test_autochdir()</code>	test 'autochdir' functionality
<code>test_garbagecollect_now()</code>	free memory right now
<code>test_null_channel()</code>	return a null Channel
<code>test_null_dict()</code>	return a null Dict
<code>test_null_job()</code>	return a null Job
<code>test_null_list()</code>	return a null List
<code>test_null_partial()</code>	return a null Partial function
<code>test_null_string()</code>	return a null String
<code>test_settime()</code>	set the time Vim uses internally
<code>timer_info()</code>	get information about timers
<code>timer_pause()</code>	pause or unpause a timer
<code>timer_start()</code>	create a timer
<code>timer_stop()</code>	stop a timer

<code>timer_stopall()</code>	stop all timers
<code>uniq()</code>	remove copies of repeated adjacent items
<code>win_findbuf()</code>	find windows containing a buffer
<code>win_getid()</code>	get window ID of a window
<code>win_gotoid()</code>	go to window with ID
<code>win_id2tabwin()</code>	get tab and window nr from window ID
<code>win_id2win()</code>	get window nr from window ID
<code>wordcount()</code>	get byte/word/char count of buffer

New Vim variables:

<code>v:beval_winid</code>	Window ID of the window where the mouse pointer is
<code>v:completed_item</code>	complete items for the most recently completed word
<code>v:errors</code>	errors found by assert functions
<code>v:false</code>	a Number with value zero
<code>v:hlsearch</code>	indicates whether search highlighting is on
<code>v:mouse_winid</code>	Window ID for a mouse click obtained with <code>getchar()</code>
<code>v:none</code>	an empty String, used for JSON
<code>v:null</code>	an empty String, used for JSON
<code>v:option_new</code>	new value of the option, used by <code>OptionSet</code>
<code>v:option_old</code>	old value of the option, used by <code>OptionSet</code>
<code>v:option_oldlocal</code>	old local value of the option, used by <code>OptionSet</code>
<code>v:option_oldglobal</code>	old global value of the option, used by <code>OptionSet</code>
<code>v:option_type</code>	scope of the set command, used by <code>OptionSet</code>
<code>v:option_command</code>	command used to set the option, used by <code>OptionSet</code>
<code>v:progbath</code>	the command with which Vim was invoked
<code>v:t_bool</code>	value of Boolean type
<code>v:t_channel</code>	value of Channel type
<code>v:t_dict</code>	value of Dictionary type
<code>v:t_float</code>	value of Float type
<code>v:t_func</code>	value of Funcref type
<code>v:t_job</code>	value of Job type
<code>v:t_list</code>	value of List type
<code>v:t_none</code>	value of None type
<code>v:t_number</code>	value of Number type
<code>v:t_string</code>	value of String type
<code>v:testing</code>	must be set before using <code>test_garbagecollect_now()</code>
<code>v:true</code>	a Number with value one
<code>v:vim_did_enter</code>	set just before VimEnter autocommands are triggered

New autocommand events:

<code>CmdUndefined</code>	a user command is used but it isn't defined
<code>OptionSet</code>	after setting any option
<code>TabClosed</code>	after closing a tab page
<code>TabNew</code>	after creating a new tab page
<code>TextChanged</code>	after a change was made to the text in Normal mode
<code>TextChangedI</code>	after a change was made to the text in Insert mode
<code>WinNew</code>	after creating a new window

New highlight groups:

EndOfBuffer filler lines (~) after the last line in the buffer.
hl-EndOfBuffer

New items in search patterns:

/\%C \%C match any composing characters

New Syntax/Indent/FTplugin files:

AVR Assembler (Avra) syntax
Arduino syntax
Bazel syntax and indent and ftplugin
Dockerfile syntax and ftplugin
Eiffel ftplugin
Euphoria 3 and 4 syntax
Go syntax and indent and ftplugin
Godoc syntax
Groovy ftplugin
HGcommit ftplugin
Hog indent and ftplugin
Innovation Data Processing upstream.pt syntax
J syntax and indent and ftplugin
Jproperties ftplugin
Json syntax and indent and ftplugin
Kivy syntax
Less syntax and indent
Mix syntax
Motorola S-Record syntax
R ftplugin
ReStructuredText syntax and indent and ftplugin
Registry ftplugin
Rhelp indent and ftplugin
Rmd (markdown with R code chunks) syntax and indent
Rmd ftplugin
Rnoweb ftplugin
Rnoweb indent
Scala syntax and indent and ftplugin
SystemVerilog syntax and indent and ftplugin
Systemd syntax and indent and ftplugin
Teraterm (TTL) syntax and indent
Text ftplugin
Vroom syntax and indent and ftplugin

New Keymaps:

Armenian eastern and western
Russian jcukenwintype
Vietnamese telex and vni

=====

INCOMPATIBLE CHANGES

incompatible-8

These changes are incompatible with previous releases. Check this list if you run into a problem when upgrading from Vim 7.4 to 8.0.

Better defaults without a vimrc

When no vimrc file is found, the `defaults.vim` script is loaded to set more useful default values for new users. That includes setting `'nocompatible'`. Thus Vim no longer starts up in Vi compatible mode. If you do want that, either create a `.vimrc` file that does "set compatible" or start Vim with "vim -C".

Support removed

The support for MS-DOS has been removed. It hasn't been working for a while (Vim doesn't fit in memory) and removing it cleans up the code quite a bit.

The support for Windows 16 bit (Windows 95 and older) has been removed.

The support for OS/2 has been removed. It probably hasn't been working for a while since nobody uses it.

The SNIFF+ support has been removed.

Minor incompatibilities:

Probably...

IMPROVEMENTS

improvements-8

The existing blowfish encryption turned out to be much weaker than it was supposed to be. The blowfish2 method has been added to fix that. **Note** that this still isn't a state-of-the-art encryption, but good enough for most usage. See `'cryptmethod'`.

COMPILE TIME CHANGES

compile-changes-8

The Vim repository was moved from Google code to github, since Google code was shut down. It can now be found at <https://github.com/vim/vim>.

Functions now use ANSI-C declarations. At least a C-89 compatible compiler is required.

The +visual feature is now always included.

PATCHES

patches-8 bug-fixes-8

The list of patches that got included since 7.4.0. This includes all the new features, but does not include runtime file changes (syntax, indent, help, etc.)

Patch 7.4.001

Problem: Character classes such as `[a-z]` do not react to `'ignorecase'`. Breaks man page highlighting. (Mario Grgic)
Solution: Add separate items for classes that react to `'ignorecase'`. Clean up logic handling character classes. Add more tests.
Files: `src/regexp_nfa.c`, `src/testdir/test64.in`, `src/testdir/test64.ok`

Patch 7.4.002

Problem: Pattern with two alternative look-behind matches does not match. (Amadeus Demarzi)
Solution: When comparing PIMs also compare their state ID to see if they are different.
Files: `src/regexp_nfa.c`, `src/testdir/test64.in`, `src/testdir/test64.ok`

Patch 7.4.003

Problem: Memory access error in Ruby syntax highlighting. (Christopher Chow)
Solution: Refresh stale pointer. (James McCoy)
Files: `src/regexp_nfa.c`

Patch 7.4.004

Problem: When closing a window fails `":bwipe"` may hang.
Solution: Let `win_close()` return FAIL and break out of the loop.
Files: `src/window.c`, `src/proto/window.pro`, `src/buffer.c`

Patch 7.4.005

Problem: Using `"vaB"` while `'virtualedit'` is set selects the wrong area. (Dimitar Dimitrov)
Solution: Reset `coladd` when finding a match.
Files: `src/search.c`

Patch 7.4.006

Problem: `mkdir("foo/bar/", "p")` gives an error message. (David Barnett)
Solution: Remove the trailing slash. (lcd)
Files: `src/eval.c`

Patch 7.4.007

Problem: Creating a preview window on startup leaves the screen layout in a messed up state. (Marius Gedminas)
Solution: Don't change `firstwin`. (Christian Brabandt)
Files: `src/main.c`

Patch 7.4.008

Problem: New regexp engine can't be interrupted.
Solution: Check for **CTRL-C** pressed. (Yasuhiro Matsumoto)
Files: `src/regexp_nfa.c`, `src/regexp.c`

Patch 7.4.009

Problem: When a file was not decrypted (yet), writing it may destroy the contents.

Solution: Mark the file as readonly until decryption was done. (Christian Brabandt)
Files: src/fileio.c

Patch 7.4.010 (after 7.4.006)
Problem: Crash with invalid argument to mkdir().
Solution: Check for empty string. (lcd47)
Files: src/eval.c

Patch 7.4.011
Problem: Cannot find out if "acl" and "xpm" features are supported.
Solution: Add "acl" and "xpm" to the list of features. (Ken Takata)
Files: src/eval.c, src/version.c

Patch 7.4.012
Problem: MS-Windows: resolving shortcut does not work properly with multibyte characters.
Solution: Use wide system functions. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.013
Problem: MS-Windows: File name buffer too small for utf-8.
Solution: Use character count instead of byte count. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.014
Problem: MS-Windows: check for writing to device does not work.
Solution: Fix #ifdefs. (Ken Takata)
Files: src/fileio.c

Patch 7.4.015
Problem: MS-Windows: Detecting node type does not work for multibyte characters.
Solution: Use wide character function when needed. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.016
Problem: MS-Windows: File name case can be wrong.
Solution: Add fname_casew(). (Ken Takata)
Files: src/os_win32.c

Patch 7.4.017
Problem: ":help !!" does not find the "!!" tag in the help file. (Ben Fritz)
Solution: When reading the start of the tags file do parse lines that are not header lines.
Files: src/tag.c

Patch 7.4.018
Problem: When completing item becomes unselected. (Shougo Matsu)
Solution: Revert patch 7.3.1269.
Files: src/edit.c

Patch 7.4.019

Problem: MS-Windows: File name completion doesn't work properly with Chinese characters. (Yue Wu)
Solution: Take care of multibyte characters when looking for the start of the file name. (Ken Takata)
Files: src/edit.c

Patch 7.4.020

Problem: NFA engine matches too much with \@>. (John McGowan)
Solution: When a whole pattern match is found stop searching.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.021

Problem: NFA regexp: Using \ze in one branch which doesn't match may cause end of another branch to be wrong. (William Fugh)
Solution: Set end position if it wasn't set yet.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.022

Problem: Deadlock while exiting, because of allocating memory.
Solution: Do not use gettext() in deathtrap(). (James McCoy)
Files: src/os_unix.c, src/misc1.c

Patch 7.4.023

Problem: Compiler warning on 64 bit windows.
Solution: Add type cast. (Mike Williams)
Files: src/edit.c

Patch 7.4.024

Problem: When root edits a file the undo file is owned by root while the edited file may be owned by another user, which is not allowed. (cac2s)
Solution: Accept an undo file owned by the current user.
Files: src/undo.c

Patch 7.4.025 (after 7.4.019)

Problem: Reading before start of a string.
Solution: Do not call mb_ptr_back() at start of a string. (Dominique Pelle)
Files: src/edit.c

Patch 7.4.026

Problem: Clang warning for int shift overflow.
Solution: Use unsigned and cast back to int. (Dominique Pelle)
Files: src/misc2.c

Patch 7.4.027 (after 7.4.025)

Problem: Another valgrind error when using **CTRL-X CTRL-F** at the start of the line. (Dominique Pelle)
Solution: Don't call mb_ptr_back() at the start of the line. Add a test.
Files: src/edit.c, src/testdir/test32.in

Patch 7.4.028

Problem: Equivalence classes are not working for multibyte characters.
Solution: Copy the rules from the old to the new regexp engine. Add a test to check both engines.

Files: src/regexp_nfa.c, src/testdir/test44.in, src/testdir/test99.in,
src/testdir/test99.ok, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile

Patch 7.4.029

Problem: An error in a pattern is reported twice.
Solution: Remove the retry with the backtracking engine, it won't work.
Files: src/regexp.c

Patch 7.4.030

Problem: The -mno-cygwin argument is no longer supported by Cygwin.
Solution: Remove the arguments. (Steve Hall)
Files: src/GvimExt/Make_cyg.mak, src/Make_cyg.mak, src/xxd/Make_cyg.mak

Patch 7.4.031

Problem: ":diffoff!" resets options even when 'diff' is not set. (Charles Cooper)
Solution: Only resets related options in a window where 'diff' is set.
Files: src/diff.c

Patch 7.4.032

Problem: NFA engine does not match the NUL character. (Jonathon Merz)
Solution: Use 0x0a instead of NUL. (Christian Brabandt)
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.033

Problem: When the terminal has only 20 lines test 92 and 93 overwrite the input file.
Solution: Explicitly write test.out. Check that the terminal is large enough to run the tests. (Hirohito Higashi)
Files: src/testdir/test92.in, src/testdir/test93.in,
src/testdir/test1.in, src/testdir/Makefile

Patch 7.4.034

Problem: Using "p" in Visual block mode only changes the first line.
Solution: Repeat the put in all text in the block. (Christian Brabandt)
Files: runtime/doc/change.txt, src/ops.c, src/normal.c,
src/testdir/test20.in, src/testdir/test20.ok

Patch 7.4.035

Problem: MS-Windows: The mouse pointer flickers when going from command line mode to Normal mode.
Solution: Check for WM_NCMOUSEMOVE. (Ken Takata)
Files: src/gui_w48.c

Patch 7.4.036

Problem: NFA engine does not capture group correctly when using \@>. (ZyX)
Solution: Copy submatches before doing the recursive match.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.037

Problem: Using "\ze" in a sub-pattern does not result in the end of the

match to be set. (Axel Bender)
Solution: Copy the end of match position when a recursive match was successful.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.038

Problem: Using "zw" and "zg" when '**spell**' is off give a confusing error message. (Gary Johnson)
Solution: Ignore the error when locating the word. Explicitly mention what word was added. (Christian Brabandt)
Files: src/normal.c, src/spell.c

Patch 7.4.039

Problem: MS-Windows: MSVC10 and earlier can't handle symlinks to a directory properly.
Solution: Add stat_symlink_aware() and wstat_symlink_aware(). (Ken Takata)
Files: src/os_mswin.c, src/os_win32.c, src/os_win32.h

Patch 7.4.040

Problem: Valgrind error on exit when a script-local variable holds a reference to the scope of another script.
Solution: First clear all variables, then free the scopes. (ZyX)
Files: src/eval.c

Patch 7.4.041 (after 7.4.034)

Problem: Visual selection does not remain after being copied over. (Axel Bender)
Solution: Move when VIsual_active is reset. (Christian Brabandt)
Files: src/ops.c

Patch 7.4.042

Problem: When using ":setlocal" for '**spell**' and '**spelllang**' then :spelldump doesn't work. (Dimitar Dimitrov)
Solution: Copy the option variables to the new window used to show the dump. (Christian Brabandt)
Files: src/spell.c

Patch 7.4.043

Problem: VMS can't handle long function names.
Solution: Shorten may_req_ambiguous_character_width. (Samuel Ferencik)
Files: src/main.c, src/term.c, src/proto/term.pro

Patch 7.4.044 (after 7.4.039)

Problem: Can't build with old MSVC. (Wang Shoulin)
Solution: Define OPEN_OH_ARGTYPE instead of using intptr_t directly.
Files: src/os_mswin.c

Patch 7.4.045

Problem: substitute() does not work properly when the pattern starts with "\\ze".
Solution: Detect an empty match. (Christian Brabandt)
Files: src/eval.c, src/testdir/test80.in, src/testdir/test80.ok

Patch 7.4.046

Problem: Can't use Tcl 8.6.
Solution: Change how Tcl_FindExecutable is called. (Jan Nijtmans)
Files: src/if_tcl.c

Patch 7.4.047

Problem: When using input() in a function invoked by a mapping it doesn't work.
Solution: Temporarily reset ex_normal_busy. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.048

Problem: Recent clang version complains about -fno-strength-reduce.
Solution: Add a configure check for the clang version. (Kazunobu Kuriyama)
Files: src/configure.in, src/auto/configure

Patch 7.4.049

Problem: In Ex mode, when line numbers are enabled the substitute prompt is wrong.
Solution: Adjust for the line number size. (Benoit Pierre)
Files: src/ex_cmds.c

Patch 7.4.050

Problem: "gn" selects too much for the pattern "\d" when there are two lines with a single digit. (Ryan Carney)
Solution: Adjust the logic of is_one_char(). (Christian Brabandt)
Files: src/search.c, src/testdir/test53.in, src/testdir/test53.ok

Patch 7.4.051

Problem: Syntax highlighting a Yaml file causes a crash. (Blake Preston)
Solution: Copy the pim structure before calling addstate() to avoid it becoming invalid when the state list is reallocated.
Files: src/regex_nfa.c

Patch 7.4.052

Problem: With 'fo' set to "a2" inserting a space in the first column may cause the cursor to jump to the previous line.
Solution: Handle the case when there is no comment leader properly. (Tor Perkins) Also fix that cursor is in the wrong place when spaces get replaced with a Tab.
Files: src/misc1.c, src/ops.c, src/testdir/test68.in, src/testdir/test68.ok

Patch 7.4.053

Problem: Test75 has a wrong header. (ZyX)
Solution: Fix the text and remove leading ".
Files: src/testdir/test75.in

Patch 7.4.054

Problem: Reading past end of the 'stl' string.
Solution: Don't increment pointer when already at the NUL. (Christian Brabandt)
Files: src/buffer.c

Patch 7.4.055

Problem: Mac: Where availability macros are defined depends on the system.
Solution: Add a configure check. (Felix Bünemann)
Files: src/config.h.in, src/configure.in, src/auto/configure,
src/os_mac.h

Patch 7.4.056

Problem: Mac: Compilation problem with OS X 10.9 Mavericks.
Solution: Include AvailabilityMacros.h when available. (Kazunobu Kuriyama)
Files: src/os_unix.c

Patch 7.4.057

Problem: byteidx() does not work for composing characters.
Solution: Add byteidxcomp().
Files: src/eval.c, src/testdir/test69.in, src/testdir/test69.ok,
runtime/doc/eval.txt

Patch 7.4.058

Problem: Warnings on 64 bit Windows.
Solution: Add type casts. (Mike Williams)
Files: src/ops.c

Patch 7.4.059

Problem: set_last_cursor() may encounter w_buffer being NULL. (Matt Mkaniaris)
Solution: Check for NULL.
Files: src/mark.c

Patch 7.4.060

Problem: Declaration has wrong return type for PyObject_SetAttrString().
Solution: Use int instead of PyObject. (Andreas Schwab)
Files: src/if_python.c, src/if_python3.c

Patch 7.4.061 (after 7.4.055 and 7.4.056)

Problem: Availability macros configure check in wrong place.
Solution: Also check when not using Darwin. Remove version check.
Files: src/configure.in, src/auto/configure, src/os_unix.c

Patch 7.4.062 (after 7.4.061)

Problem: Configure check for AvailabilityMacros.h is wrong.
Solution: Use AC_CHECK_HEADERS().
Files: src/configure.in, src/auto/configure

Patch 7.4.063

Problem: Crash when using invalid key in Python dictionary.
Solution: Check for object to be NULL. Add tests. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.064

Problem: When replacing a character in Visual block mode, entering a CR does not cause a repeated line break.
Solution: Recognize the situation and repeat the line break. (Christian Brabandt)

Files: src/normal.c, src/ops.c, src/testdir/test39.in,
src/testdir/test39.ok

Patch 7.4.065

Problem: When recording, the character typed at the hit-enter prompt is recorded twice. (Urtica Dioica)

Solution: Avoid recording the character twice. (Christian Brabandt)

Files: src/message.c

Patch 7.4.066

Problem: MS-Windows: When there is a colon in the file name (sub-stream feature) the swap file name is wrong.

Solution: Change the colon to "%". (Yasuhiro Matsumoto)

Files: src/fileio.c, src/memline.c, src/misc1.c, src/proto/misc1.pro

Patch 7.4.067

Problem: After inserting comment leader, CTRL-\ CTRL-O does move the cursor. (Wiktor Ruben)

Solution: Avoid moving the cursor. (Christian Brabandt)

Files: src/edit.c

Patch 7.4.068

Problem: Cannot build Vim on Mac with non-Apple compilers.

Solution: Remove the -no-cpp-precomp flag. (Misty De Meo)

Files: src/configure.in, src/auto/configure, src/osdef.sh

Patch 7.4.069

Problem: Cannot right shift lines starting with #.

Solution: Allow the right shift when '**cin**' contains #N with N > 0. (Christian Brabandt)

Refactor parsing '**cin**', store the values in the buffer.

Files: runtime/doc/indent.txt, src/buffer.c, src/edit.c, src/eval.c,
src/ex_getln.c, src/fold.c, src/misc1.c, src/ops.c,
src/proto/misc1.pro, src/proto/option.pro, src/structs.h,
src/option.c

Patch 7.4.070 (after 7.4.069)

Problem: Can't compile with tiny features. (Tony Mechelynck)

Solution: Add #ifdef.

Files: src/buffer.c

Patch 7.4.071 (after 7.4.069)

Problem: Passing limits around too often.

Solution: Use limits from buffer.

Files: src/edit.c, src/misc1.c, src/proto/misc1.pro

Patch 7.4.072

Problem: Crash when using Insert mode completion.

Solution: Avoid going past the end of pum_array. (idea by Francisco Lopes)

Files: src/popupmnu.c

Patch 7.4.073

Problem: Setting undolevels for one buffer changes undo in another.

Solution: Make '**undolevels**' a global-local option. (Christian Brabandt)

Files: runtime/doc/options.txt, src/buffer.c, src/option.c, src/option.h
src/structs.h, src/undo.c

Patch 7.4.074

Problem: When undo'ing all changes and creating a new change the undo structure is incorrect. (Christian Brabandt)
Solution: When deleting the branch starting at the old header, delete the whole branch, not just the first entry.
Files: src/undo.c

Patch 7.4.075

Problem: Locally setting **'undolevels'** is not tested.
Solution: Add a test. (Christian Brabandt)
Files: src/testdir/test100.in, src/testdir/test100.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile, src/Makefile

Patch 7.4.076

Problem: "cgn" does not wrap around the end of the file. (Dimitar Dimitrov)
Solution: Restore **'wrapscan'** earlier. (Christian Brabandt)
Files: src/search.c

Patch 7.4.077

Problem: DOS installer creates shortcut without a path, resulting in the current directory to be C:\Windows\system32.
Solution: Use environment variables.
Files: src/dosinst.c

Patch 7.4.078

Problem: MSVC 2013 is not supported.
Solution: Recognize and support MSVC 2013. (Ed Brown)
Files: src/Make_mvc.mak

Patch 7.4.079

Problem: A script cannot detect whether **'hlsearch'** highlighting is actually displayed.
Solution: Add the "v:hlsearch" variable. (ZyX)
Files: src/eval.c, src/ex_docmd.c,
src/option.c, src/screen.c, src/search.c, src/tag.c, src/vim.h,
src/testdir/test101.in, src/testdir/test101.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.080 (after 7.4.079)

Problem: Missing documentation for v:hlsearch.
Solution: Include the right file in the patch.
Files: runtime/doc/eval.txt

Patch 7.4.081 (after 7.4.078)

Problem: Wrong logic when ANALYZE is "yes".
Solution: Use or instead of and. (KF Leong)
Files: src/Make_mvc.mak

Patch 7.4.082

Problem: Using "gf" in a changed buffer suggests adding "!", which is not possible. (Tim Chase)
Solution: Pass a flag to check_changed() whether adding ! make sense.
Files: src/vim.h, src/ex_cmds2.c, src/proto/ex_cmds2.pro, src/globals.h, src/ex_cmds.c, src/ex_docmd.c

Patch 7.4.083

Problem: It's hard to avoid adding a used pattern to the search history.
Solution: Add the ":keeppatterns" modifier. (Christian Brabandt)
Files: runtime/doc/cmdline.txt, src/ex_cmds.h, src/ex_docmd.c, src/ex_getln.c, src/structs.h

Patch 7.4.084

Problem: Python: interrupt not being properly discarded. (Yggdroot Chen)
Solution: Discard interrupt in VimTryEnd. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.085

Problem: When inserting text in Visual block mode and moving the cursor the wrong text gets repeated in other lines.
Solution: Use the '[' mark to find the start of the actually inserted text. (Christian Brabandt)
Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.086

Problem: Skipping over an expression when not evaluating it does not work properly for dict members.
Solution: Skip over unrecognized expression. (ZyX)
Files: src/eval.c, src/testdir/test34.in, src/testdir/test34.ok

Patch 7.4.087

Problem: Compiler warning on 64 bit Windows systems.
Solution: Fix type cast. (Mike Williams)
Files: src/ops.c

Patch 7.4.088

Problem: When spell checking is enabled Asian characters are always marked as error.
Solution: When '**spelllang**' contains "cjk" do not mark Asian characters as error. (Ken Takata)
Files: runtime/doc/options.txt, runtime/doc/spell.txt, src/mbyte.c, src/option.c, src/spell.c, src/structs.h

Patch 7.4.089

Problem: When editing a file in a directory mounted through sshfs Vim doesn't set the security context on a renamed file.
Solution: Add mch_copy_sec() to vim_rename(). (Peter Backes)
Files: src/fileio.c

Patch 7.4.090

Problem: Win32: When a directory name contains an exclamation mark,

completion doesn't complete the contents of the directory.
Solution: Escape the exclamation mark. (Jan Stocker)
Files: src/ex_getln.c, src/testdir/test102.in, src/testdir/test102.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.091 (after 7.4.089)
Problem: Missing semicolon.
Solution: Add the semicolon.
Files: src/fileio.c

Patch 7.4.092 (after 7.4.088)
Problem: Can't build small version.
Solution: Add #ifdef where the b_cjk flag is used. (Ken Takata)
Files: src/spell.c

Patch 7.4.093
Problem: Configure can't use LuaJIT on ubuntu 12.04.
Solution: Adjust the configure regexp that locates the version number.
(Charles Strahan)
Files: src/configure.in, src/auto/configure

Patch 7.4.094
Problem: Configure may not find that -lint is needed for gettext().
Solution: Check for gettext() with empty \$LIBS. (Thomas De Schampheleire)
Files: src/configure.in, src/auto/configure

Patch 7.4.095 (after 7.4.093)
Problem: Regexp for LuaJIT version doesn't work on BSD.
Solution: Use "*" instead of "\+" and "\?". (Ozaki Kiichi)
Files: src/configure.in, src/auto/configure

Patch 7.4.096
Problem: Can't change directory to an UNC path.
Solution: Use win32_getattns() in mch_getperm(). (Christian Brabandt)
Files: src/os_win32.c

Patch 7.4.097 (after 7.4.034)
Problem: Unexpected behavior change related to **'virtualedit'**. (Ingo Karkat)
Solution: Update the valid cursor position. (Christian Brabandt)
Files: src/ops.c

Patch 7.4.098
Problem: When using ":'<,'>del" errors may be given for the visual line
numbers being out of range.
Solution: Reset Visual mode in ":del". (Lech Lorens)
Files: src/ex_docmd.c, src/testdir/test103.in, src/testdir/test103.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.099
Problem: Append in blockwise Visual mode with "\$" is wrong.

Solution: After "\$" don't use the code that checks if the cursor was moved.
(Hirohito Higashi, Ken Takata)
Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.100

Problem: NFA regexp doesn't handle backreference correctly. (Ryuichi Hayashida, Urtica Dioica)
Solution: Always add NFA_SKIP, also when it already exists at the start position.
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.101

Problem: Using \1 in pattern goes one line too far. (Bohr Shaw, John Little)
Solution: Only advance the match end for the matched characters in the last line.
Files: src/regexp.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.102

Problem: Crash when interrupting "z=".
Solution: Add safety check for word length. (Christian Brabandt, Dominique Pelle)
Files: src/spell.c

Patch 7.4.103

Problem: Dos installer uses an old way to escape spaces in the diff command.
Solution: Adjust the quoting to the new default shellxquote. (Ben Fritz)
Files: src/dosinst.c

Patch 7.4.104

Problem: ":help s/_" reports an internal error. (John Beckett)
Solution: Check for NUL and invalid character classes.
Files: src/regexp_nfa.c

Patch 7.4.105

Problem: Completing a tag pattern may give an error for invalid pattern.
Solution: Suppress the error, just return no matches.
Files: src/tag.c

Patch 7.4.106

Problem: Can't build with Ruby using Cygwin.
Solution: Fix library name in makefile. (Steve Hall)
Files: src/Make_cyg.mak

Patch 7.4.107

Problem: Python: When vim.eval() encounters a Vim error, a try/catch in the Python code doesn't catch it. (Yggdroot Chen)
Solution: Throw exceptions on errors in vim.eval(). (ZyX)
Files: src/ex_eval.c, src/if_py_both.h, src/proto/ex_eval.pro, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.108

Problem: "zG" and "zW" leave temp files around on MS-Windows.

Solution: Delete the temp files when exiting. (Ken Takata)
Files: src/memline.c, src/proto/spell.pro, src/spell.c

Patch 7.4.109

Problem: ColorScheme autocommand matches with the current buffer name.
Solution: Match with the colorscheme name. (Christian Brabandt)
Files: runtime/doc/autocmd.txt, src/fileio.c, src/syntax.c

Patch 7.4.110

Problem: "gUgn" cannot be repeated. (Dimitar Dimitrov)
Solution: Don't put "gn" in a different order in the redo buffer. Restore **'wrapscan'** when the pattern isn't found. (Christian Wellenbrock)
Files: src/normal.c, src/search.c, src/test53.in, src/test53.ok

Patch 7.4.111

Problem: Memory leak in Python OptionsAssItem. (Ken Takata)
Solution: Call Py_XDECREF() where needed. (ZyX)
Files: src/if_py_both.h

Patch 7.4.112

Problem: The defaults for **'directory'** and **'backupdir'** on MS-Windows do not include a directory that exists.
Solution: Use \$TEMP.
Files: src/os_dos.h

Patch 7.4.113

Problem: MSVC static analysis gives warnings.
Solution: Avoid the warnings and avoid possible bugs. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.114

Problem: New GNU make outputs messages about changing directory in another format.
Solution: Recognize the new format.
Files: src/option.h

Patch 7.4.115

Problem: When using Zsh expanding ~abc doesn't work when the result contains a space.
Solution: Off-by-one error in detecting the NUL. (Pavol Juhas)
Files: src/os_unix.c

Patch 7.4.116

Problem: When a mapping starts with a space, the typed space does not show up for **'showcmd'**.
Solution: Show "<20>". (Brook Hong)
Files: src/normal.c

Patch 7.4.117

Problem: Can't build with Cygwin/MingW and Perl 5.18.
Solution: Add a linker argument for the Perl library. (Cesar Romani)
Adjust CFLAGS and LIB. (Cesar Romani)
Move including inline.h further down. (Ken Takata)
Files: src/Make_cyg.mak, src/Make_ming.mak, src/if_perl.xs

Patch 7.4.118

Problem: It's possible that redrawing the status lines causes win_redr_custom() to be called recursively.
Solution: Protect against recursiveness. (Yasuhiro Matsumoto)
Files: src/screen.c

Patch 7.4.119

Problem: Vim doesn't work well on OpenVMS.
Solution: Fix various problems. (Samuel Ferencik)
Files: src/os_unix.c, src/os_unix.h, src/os_vms.c

Patch 7.4.120 (after 7.4.117)

Problem: Can't build with Perl 5.18 on Linux. (Lcd 47)
Solution: Add #ifdef. (Ken Takata)
Files: src/if_perl.xs

Patch 7.4.121

Problem: Completion doesn't work for ":py3d" and ":py3f". (Bohr Shaw)
Solution: Skip over letters after ":py3".
Files: src/ex_docmd.c

Patch 7.4.122

Problem: Win32: When **'encoding'** is set to "utf-8" and the active codepage is cp932 then ":grep" and other commands don't work for multibyte characters.
Solution: (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.123

Problem: Win32: Getting user name does not use wide function.
Solution: Use GetUserNameW() if possible. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.124

Problem: Win32: Getting host name does not use wide function.
Solution: Use GetComputerNameW() if possible. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.125

Problem: Win32: Dealing with messages may not work for multibyte chars.
Solution: Use pDispatchMessage(). (Ken Takata)
Files: src/os_win32.c

Patch 7.4.126

Problem: Compiler warnings for "const" and incompatible types.
Solution: Remove "const", add type cast. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.127

Problem: Perl 5.18 on Unix doesn't work.
Solution: Move workaround to after including vim.h. (Ken Takata)
Files: src/if_perl.xs

Patch 7.4.128

Problem: Perl 5.18 for MSVC doesn't work.
Solution: Add check in makefile and define `__inline`. (Ken Takata)
Files: `src/Make_mvc.mak`, `src/if_perl.xs`

Patch 7.4.129

Problem: `getline(-1)` returns zero. (mvxxc)
Solution: Return an empty string.
Files: `src/eval.c`

Patch 7.4.130

Problem: Relative line numbers mix up windows when using folds.
Solution: Use `hasFoldingWin()` instead of `hasFolding()`. (Lech Lorens)
Files: `src/misc2.c`

Patch 7.4.131

Problem: Syncbind causes E315 errors in some situations. (Liang Li)
Solution: Set and restore `curbuf` in `ex_syncbind()`. (Christian Brabandt)
Files: `src/ex_docmd.c`, `src/testdir/test37.ok`

Patch 7.4.132 (after 7.4.122)

Problem: Win32: `flags` and `inherit_handles` arguments mixed up.
Solution: Swap the argument. (cs86661)
Files: `src/os_win32.c`

Patch 7.4.133

Problem: Clang warns for using `NUL`.
Solution: Change `NUL` to `NULL`. (Dominique Pelle)
Files: `src/eval.c`, `src/misc2.c`

Patch 7.4.134

Problem: Spurious space in MingW Makefile.
Solution: Remove the space. (Michael Soyka)
Files: `src/Make_ming.mak`

Patch 7.4.135

Problem: Missing dot in MingW test Makefile.
Solution: Add the dot. (Michael Soyka)
Files: `src/testdir/Make_ming.mak`

Patch 7.4.136 (after 7.4.096)

Problem: MS-Windows: When saving a file with a UNC path the file becomes read-only.
Solution: Don't mix up Win32 attributes and Unix attributes. (Ken Takata)
Files: `src/os_mswin.c`, `src/os_win32.c`

Patch 7.4.137

Problem: Cannot use IME with Windows 8 console.
Solution: Change the user of `ReadConsoleInput()` and `PeekConsoleInput()`. (Nobuhiro Takasaki)
Files: `src/os_win32.c`

Patch 7.4.138 (after 7.4.114)

Problem: Directory change messages are not recognized.

Solution: Fix using a character range literally. (Lech Lorens)
Files: src/option.h

Patch 7.4.139

Problem: Crash when using :cd in autocommand. (François Ingelrest)
Solution: Set w_localdir to NULL after freeing it. (Dominique Pelle)
Files: src/ex_docmd.c, src/window.c

Patch 7.4.140

Problem: Crash when wiping out buffer triggers autocommand that wipes out only other buffer.
Solution: Do not delete the last buffer, make it empty. (Hirohito Higashi)
Files: src/buffer.c

Patch 7.4.141

Problem: Problems when building with Borland: st_mode is signed short; can't build with Python; temp files not ignored by Mercurial; building with DEBUG doesn't define _DEBUG.
Solution: Fix the problems. (Ken Takata)
Files: src/Make_bc5.mak, src/if_py_both.h, src/os_win32.c

Patch 7.4.142 (after 7.4.137)

Problem: On MS-Windows 8 IME input doesn't work correctly.
Solution: Work around the problem. (Nobuhiro Takasaki)
Files: src/os_win32.c

Patch 7.4.143

Problem: TextChangedI is not triggered.
Solution: Reverse check for "ready". (lilydjpg)
Files: src/edit.c

Patch 7.4.144

Problem: MingW also supports intptr_t for OPEN_OH_ARGTYPE.
Solution: Adjust #ifdef. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.145

Problem: getregtype() does not return zero for unknown register.
Solution: Adjust documentation: return empty string for unknown register. Check the register name to be valid. (Yukihiro Nakadaira)
Files: runtime/doc/eval.txt, src/ops.c

Patch 7.4.146

Problem: When starting Vim with "-u NONE" v:oldfiles is NULL.
Solution: Set v:oldfiles to an empty list. (Yasuhiro Matsumoto)
Files: src/main.c

Patch 7.4.147

Problem: Cursor moves to wrong position when using "gj" after "\$" and virtual editing is active.
Solution: Make "gj" behave differently when virtual editing is active. (Hirohito Higashi)
Files: src/normal.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.148

Problem: Cannot build with Cygwin and X11.
Solution: Include Xwindows.h instead of windows.h. (Lech Lorens)
Files: src/mbyte.c

Patch 7.4.149

Problem: Get E685 error when assigning a function to an autoload variable.
(Yukihiro Nakadaira)
Solution: Instead of having a global no_autoload variable, pass an autoload
flag down to where it is used. (ZyX)
Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok,
src/testdir/test60.in, src/testdir/test60.ok,
src/testdir/sautest/autoload/footest.vim

Patch 7.4.150

Problem: :keeppatterns is not respected for :s.
Solution: Check the keeppatterns flag. (Yasuhiro Matsumoto)
Files: src/search.c, src/testdir/test14.in, src/testdir/test14.ok

Patch 7.4.151

Problem: Python: slices with steps are not supported.
Solution: Support slices in Python vim.List. (ZyX)
Files: src/eval.c, src/if_py_both.h, src/if_python3.c, src/if_python.c,
src/proto/eval.pro, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.152

Problem: Python: Cannot iterate over options.
Solution: Add options iterator. (ZyX)
Files: src/if_py_both.h, src/option.c, src/proto/option.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok, src/vim.h

Patch 7.4.153

Problem: Compiler warning for pointer type.
Solution: Add type cast.
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.4.154 (after 7.4.149)

Problem: Still a problem with auto-loading.
Solution: Pass no_autoload to deref_func_name(). (Yukihiro Nakadaira)
Files: src/eval.c

Patch 7.4.155

Problem: ":keeppatterns /pat" does not keep search pattern offset.
Solution: Restore the offset after doing the search.
Files: src/search.c, src/testdir/test14.in, src/testdir/test14.ok

Patch 7.4.156

Problem: Test file missing from distribution.
Solution: Add new directory to file list.
Files: Filelist

Patch 7.4.157

Problem: Error number used twice. (Yukihiro Nakadaira)
Solution: Change the one not referred in the docs.
Files: src/undo.c

Patch 7.4.158 (after 7.4.045)

Problem: Pattern containing \zs is not handled correctly by substitute().
Solution: Change how an empty match is skipped. (Yukihiro Nakadaira)
Files: src/eval.c, src/testdir/test80.in, src/testdir/test80.ok

Patch 7.4.159

Problem: Completion hangs when scanning the current buffer after doing keywords. (Christian Brabandt)
Solution: Set the first match position when starting to scan the current buffer.
Files: src/edit.c

Patch 7.4.160

Problem: Win32: Crash when executing external command.
Solution: Only close the handle when it was created. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.161

Problem: Crash in Python exception handling.
Solution: Only use exception variables if did_throw is set. (ZyX)
Files: src/if_py_both.h

Patch 7.4.162

Problem: Running tests in shadow dir doesn't work.
Solution: Add testdir/sautest to the shadow target. (James McCoy)
Files: src/Makefile

Patch 7.4.163 (after 7.4.142)

Problem: MS-Windows input doesn't work properly on Windows 7 and earlier.
Solution: Add a check for Windows 8. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.164 (after 7.4.163)

Problem: Problem with event handling on Windows 8.
Solution: Ignore duplicate WINDOW_BUFFER_SIZE_EVENTS. (Nobuhiro Takasaki)
Files: src/os_win32.c

Patch 7.4.165

Problem: By default, after closing a buffer changes can't be undone.
Solution: In the example vimrc file set **'undofile'**.
Files: runtime/vimrc_example.vim

Patch 7.4.166

Problem: Auto-loading a function for code that won't be executed.
Solution: Do not auto-load when evaluation is off. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.167 (after 7.4.149)

Problem: Fixes are not tested.
Solution: Add a test for not autoloading on assignment. (Yukihiro Nakadaira)

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/sautest/autoload/Test104.vim, src/testdir/test104.in,
src/testdir/test104.ok

Patch 7.4.168

Problem: Can't compile with Ruby 2.1.0.
Solution: Add support for new GC. (Kohei Suzuki)
Files: src/if_ruby.c

Patch 7.4.169

Problem: ":sleep" puts cursor in the wrong column. (Liang Li)
Solution: Add the window offset. (Christian Brabandt)
Files: src/ex_docmd.c

Patch 7.4.170

Problem: Some help tags don't work with ":help". (Tim Chase)
Solution: Add exceptions.
Files: src/ex_cmds.c

Patch 7.4.171

Problem: Redo does not set v:count and v:count1.
Solution: Use a separate buffer for redo, so that we can set the counts when performing redo.
Files: src/getchar.c, src/globals.h, src/normal.c, src/proto/getchar.pro, src/structs.h

Patch 7.4.172

Problem: The blowfish code mentions output feedback, but the code is actually doing cipher feedback.
Solution: Adjust names and comments.
Files: src/blowfish.c, src/fileio.c, src/proto/blowfish.pro, src/memline.c

Patch 7.4.173

Problem: When using scrollbind the cursor can end up below the last line. (mvxxc)
Solution: Reset w_botfill when scrolling up. (Christian Brabandt)
Files: src/move.c

Patch 7.4.174

Problem: Compiler warnings for Python interface. (Tony Mechelynck)
Solution: Add type casts, initialize variable.
Files: src/if_py_both.h

Patch 7.4.175

Problem: When a wide library function fails, falling back to the non-wide function may do the wrong thing.
Solution: Check the platform, when the wide function is supported don't fall back to the non-wide function. (Ken Takata)
Files: src/os_mswin.c, src/os_win32.c

Patch 7.4.176

Problem: Dictionary.update() throws an error when used without arguments. Python programmers don't expect that.
Solution: Make Dictionary.update() without arguments do nothing. (ZyX)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test87.in

Patch 7.4.177

Problem: Compiler warning for unused variable. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/move.c

Patch 7.4.178

Problem: The J command does not update '[' and ']' marks. (William Gardner)
Solution: Set the marks. (Christian Brabandt)
Files: src/ops.c

Patch 7.4.179

Problem: Warning for type-punned pointer. (Tony Mechelynck)
Solution: Use intermediate variable.
Files: src/if_py_both.h

Patch 7.4.180 (after 7.4.174)

Problem: Older Python versions don't support %ld.
Solution: Use %d instead. (ZyX)
Files: src/if_py_both.h

Patch 7.4.181

Problem: When using '**pastetoggle**' the status lines are not updated. (Samuel Ferencik, Jan Christoph Ebersbach)
Solution: Update the status lines. (Nobuhiro Takasaki)
Files: src/getchar.c

Patch 7.4.182

Problem: Building with mzscheme and racket does not work. (David Chimay)
Solution: Adjust autoconf. (Sergey Khorev)
Files: src/configure.in, src/auto/configure

Patch 7.4.183

Problem: MSVC Visual Studio update not supported.
Solution: Add version number. (Mike Williams)
Files: src/Make_mvc.mak

Patch 7.4.184

Problem: match() does not work properly with a {count} argument.
Solution: Compute the length once and update it. Quit the loop when at the end. (Hirohito Higashi)
Files: src/eval.c, src/testdir/test53.in, src/testdir/test53.ok

Patch 7.4.185

Problem: Clang gives warnings.
Solution: Adjust how bigness is set. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.4.186 (after 7.4.085)

Problem: Insert in Visual mode sometimes gives incorrect results.

(Dominique Pelle)
Solution: Remember the original insert start position. (Christian Brabandt, Dominique Pelle)
Files: src/edit.c, src/globals.h, src/ops.c, src/structs.h

Patch 7.4.187

Problem: Delete that crosses line break splits multibyte character.
Solution: Advance a character instead of a byte. (Cade Foster)
Files: src/normal.c, src/testdir/test69.in, src/testdir/test69.ok

Patch 7.4.188

Problem: SIZEOF_LONG clashes with similar defines in header files.
Solution: Rename to a name starting with VIM_. Also for SIZEOF_INT.
Files: src/if_ruby.c, src/vim.h, src/configure.in, src/auto/configure, src/config.h.in, src/fileio.c, src/if_python.c, src/message.c, src/spell.c, src/feature.h, src/os_os2_cfg.h, src/os_vms_conf.h, src/os_win16.h, src/structs.h

Patch 7.4.189

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/eval.c

Patch 7.4.190

Problem: Compiler warning for using %lld for off_t.
Solution: Add type cast.
Files: src/fileio.c

Patch 7.4.191

Problem: Escaping a file name for shell commands can't be done without a function.
Solution: Add the :S file name modifier.
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test105.in, src/testdir/test105.ok, runtime/doc/cmdline.txt, runtime/doc/eval.txt, runtime/doc/map.txt, runtime/doc/options.txt, runtime/doc/quickfix.txt, runtime/doc/usr_30.txt, runtime/doc/usr_40.txt, runtime/doc/usr_42.txt, runtime/doc/vi_diff.txt, src/eval.c, src/misc2.c, src/normal.c, src/proto/misc2.pro

Patch 7.4.192

Problem: Memory leak when giving E853.
Solution: Free the argument. (Dominique Pelle)
Files: src/eval.c

Patch 7.4.193

Problem: Typos in messages.
Solution: "then" -> "than". (Dominique Pelle)
Files: src/if_py_both.h, src/spell.c

Patch 7.4.194

Problem: Can't build for Android.
Solution: Add #if condition. (Fredrik Fornwall)
Files: src/mbyte.c

Patch 7.4.195 (after 7.4.193)

Problem: Python tests fail.
Solution: Change "then" to "than" in more places. (Dominique Pelle, Taro Muraoka)
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.196

Problem: Tests fail on Solaris 9 and 10.
Solution: Use "test -f" instead of "test -e". (Laurent Blume)
Files: src/testdir/Makefile

Patch 7.4.197

Problem: Various problems on VMS.
Solution: Fix several VMS problems. (Zoltan Arpadffy)
Files: runtime/doc/os_vms.txt, src/Make_vms.mms, src/fileio.c,
src/os_unix.c, src/os_unix.h, src/os_vms.c, src/os_vms_conf.h,
src/proto/os_vms.pro, src/testdir/Make_vms.mms,
src/testdir/test72.in, src/testdir/test77a.com,
src/testdir/test77a.in, src/testdir/test77a.ok src/undo.c

Patch 7.4.198

Problem: Can't build Vim with Perl when -Dusetthreads is not specified for building Perl, and building Vim with --enable-perlinterp=dynamic.
Solution: Adjust #ifdefs. (Yasuhiro Matsumoto)
Files: src/if_perl.xs

Patch 7.4.199

Problem: (issue 197)]P doesn't paste over Visual selection.
Solution: Handle Visual mode specifically. (Christian Brabandt)
Files: src/normal.c

Patch 7.4.200

Problem: Too many #ifdefs in the code.
Solution: Enable FEAT_VISUAL always, await any complaints
Files: src/feature.h

Patch 7.4.201

Problem: **'lispwords'** is a global option.
Solution: Make **'lispwords'** global-local. (Sung Pae)
Files: runtime/doc/options.txt, runtime/optwin.vim, src/buffer.c,
src/misc1.c, src/option.c, src/option.h, src/structs.h,
src/testdir/test100.in, src/testdir/test100.ok

Patch 7.4.202

Problem: MS-Windows: non-ASCII font names don't work.
Solution: Convert between the current code page and **'encoding'**. (Ken Takata)
Files: src/gui_w48.c, src/os_mswin.c, src/proto/winclip.pro,
src/winclip.c

Patch 7.4.203

Problem: Parsing **'errorformat'** is not correct.
Solution: Reset "multiignore" at the start of a multi-line message. (Lcd)
Files: src/quickfix.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test106.in,
src/testdir/test106.ok

Patch 7.4.204

Problem: A mapping where the second byte is 0x80 doesn't work.
Solution: Unescape before checking for incomplete multibyte char. (Nobuhiro Takasaki)
Files: src/getchar.c, src/testdir/test75.in, src/testdir/test75.ok

Patch 7.4.205

Problem: ":mksession" writes command to move to second argument while it does not exist. When it does exist the order might be wrong.
Solution: Use ":argadd" for each argument instead of using ":args" with a list of names. (Nobuhiro Takasaki)
Files: src/ex_docmd.c

Patch 7.4.206

Problem: Compiler warnings on 64 bit Windows.
Solution: Add type casts. (Mike Williams)
Files: src/gui_w48.c, src/os_mswin.c

Patch 7.4.207

Problem: The cursor report sequence is sometimes not recognized and results in entering replace mode.
Solution: Also check for the cursor report when not asked for.
Files: src/term.c

Patch 7.4.208

Problem: Mercurial picks up some files that are not distributed.
Solution: Add patterns to the ignore list. (Cade Forester)
Files: .hgignore

Patch 7.4.209

Problem: When repeating a filter command "%" and "#" are expanded.
Solution: Escape the command when storing for redo. (Christian Brabandt)
Files: src/ex_cmds.c

Patch 7.4.210

Problem: Visual block mode plus virtual edit doesn't work well with tabs. (Liang Li)
Solution: Take coladd into account. (Christian Brabandt)
Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.211

Problem: ":lu" is an abbreviation for ":lua", but it should be ":lunmap". (ZyX)
Solution: Move "lunmap" to above "lua".
Files: src/ex_cmds.h

Patch 7.4.212 (after 7.4.200)

Problem: Now that the +visual feature is always enabled the #ifdefs for it are not useful.

Solution: Remove the checks for FEAT_VISUAL.

Files: src/buffer.c, src/charset.c, src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_docmd.c, src/fold.c, src/getchar.c, src/gui.c, src/gui_mac.c, src/gui_w48.c, src/main.c, src/mark.c, src/menu.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_msdos.c, src/os_qnx.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/screen.c, src/search.c, src/spell.c, src/syntax.c, src/term.c, src/ui.c, src/undo.c, src/version.c, src/window.c, src/feature.h, src/globals.h, src/option.h, src/os_win32.h, src/structs.h

Patch 7.4.213

Problem: It's not possible to open a new buffer without creating a swap file.

Solution: Add the ":noswapfile" modifier. (Christian Brabandt)

Files: runtime/doc/recover.txt, src/ex_cmds.h, src/ex_docmd.c, src/memline.c, src/structs.h

Patch 7.4.214

Problem: Compilation problems on HP_nonStop (Tandem).

Solution: Add #defines. (Joachim Schmitz)

Files: src/vim.h

Patch 7.4.215

Problem: Inconsistency: ":sp foo" does not reload "foo", unless "foo" is the current buffer. (Liang Li)

Solution: Do not reload the current buffer on a split command.

Files: runtime/doc/windows.txt, src/ex_docmd.c

Patch 7.4.216

Problem: Compiler warnings. (Tony Mechelynck)

Solution: Initialize variables, add #ifdef.

Files: src/term.c, src/os_unix.h

Patch 7.4.217

Problem: When src/auto/configure was updated, "make clean" would run configure pointlessly.

Solution: Do not run configure for "make clean" and "make distclean" when the make program supports \$MAKECMDGOALS. (Ken Takata)

Files: src/Makefile

Patch 7.4.218

Problem: It's not easy to remove duplicates from a list.

Solution: Add the uniq() function. (Lcd)

Files: runtime/doc/change.txt, runtime/doc/eval.txt, runtime/doc/usr_41.txt, runtime/doc/version7.txt, src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.219

Problem: When '**relativenumber**' or '**cursorline**' are set the window is

redrawn much too often. (Patrick Hemmer, Dominique Pelle)
Solution: Check the VALID_CROW flag instead of VALID_WROW.
Files: src/move.c

Patch 7.4.220

Problem: Test 105 does not work in a shadow dir. (James McCoy)
Solution: Omit "src/" from the checked path.
Files: src/testdir/test105.in, src/testdir/test105.ok

Patch 7.4.221

Problem: Quickfix doesn't resize on ":copen 20". (issue 199)
Solution: Resize the window when requested. (Christian Brabandt)
Files: src/quickfix.c

Patch 7.4.222

Problem: The Ruby directory is constructed from parts.
Solution: Use '**rubyarchhdrdir**' if it exists. (James McCoy)
Files: src/configure.in, src/auto/configure

Patch 7.4.223

Problem: Still using an older autoconf version.
Solution: Switch to autoconf 2.69.
Files: src/Makefile, src/configure.in, src/auto/configure

Patch 7.4.224

Problem: /usr/bin/grep on Solaris does not support -F.
Solution: Add configure check to find a good grep. (Danek Duvall)
Files: src/configure.in, src/auto/configure

Patch 7.4.225

Problem: Dynamic Ruby doesn't work on Solaris.
Solution: Always use the stubs. (Danek Duvall, Yukihiro Nakadaira)
Files: src/if_ruby.c

Patch 7.4.226 (after 7.4.219)

Problem: Cursorline highlighting not redrawn when scrolling. (John Marriott)
Solution: Check for required redraw in two places.
Files: src/move.c

Patch 7.4.227 (after 7.4.225)

Problem: Can't build with Ruby 1.8.
Solution: Do include a check for the Ruby version. (Ken Takata)
Files: src/if_ruby.c

Patch 7.4.228

Problem: Compiler warnings when building with Python 3.2.
Solution: Make type cast depend on Python version. (Ken Takata)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.4.229

Problem: Using ":let" for listing variables and the second one is a curly braces expression may fail.
Solution: Check for an "=" in a better way. (ZyX)

Files: src/eval.c, src/testdir/test104.in, src/testdir/test104.ok

Patch 7.4.230

Problem: Error when using ":options".
Solution: Fix the entry for '**lispwords**'. (Kenichi Ito)
Files: runtime/optwin.vim

Patch 7.4.231

Problem: An error in ":options" is not caught by the tests.
Solution: Add a test for ":options". Set \$VIMRUNTIME for the tests so that it uses the current runtime files instead of the installed ones.
Files: src/Makefile, src/testdir/Makefile, src/testdir/test_options.in, src/testdir/test_options.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.4.232

Problem: ":%s/\n/" uses a lot of memory. (Aidan Marlin)
Solution: Turn this into a join command. (Christian Brabandt)
Files: src/ex_cmds.c, src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 7.4.233

Problem: Escaping special characters for using "%" with a shell command is inconsistent, parentheses are escaped but spaces are not.
Solution: Only escape "!". (Gary Johnson)
Files: src/ex_docmd.c

Patch 7.4.234

Problem: Can't get the command that was used to start Vim.
Solution: Add v:progpath. (Viktor Kojouharov)
Files: runtime/doc/eval.txt, src/eval.c, src/main.c, src/vim.h

Patch 7.4.235

Problem: It is not easy to get the full path of a command.
Solution: Add the exepath() function.
Files: src/eval.c, src/misc1.c, src/os_amiga.c, src/os_msdos.c, src/os_unix.c, src/os_vms.c, src/os_win32.c, src/proto/os_amiga.pro, src/proto/os_msdos.pro, src/proto/os_unix.pro, src/proto/os_win32.pro, runtime/doc/eval.txt

Patch 7.4.236

Problem: It's not that easy to check the Vim patch version.
Solution: Make has("patch-7.4.123") work. (partly by Marc Weber)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test60.in, src/testdir/test60.ok

Patch 7.4.237 (after 7.4.236)

Problem: When some patches were not included has("patch-7.4.123") may return true falsely.
Solution: Check for the specific patch number.
Files: runtime/doc/eval.txt, src/eval.c

Patch 7.4.238

Problem: Vim does not support the smack library.
Solution: Add smack support (Jose Bollo)
Files: src/config.h.in, src/configure.in, src/fileio.c, src/memfile.c,
src/os_unix.c, src/undo.c, src/auto/configure

Patch 7.4.239

Problem: ":e +" does not position cursor at end of the file.
Solution: Check for "+" being the last character (ZyX)
Files: src/ex_docmd.c

Patch 7.4.240

Problem: ":tjump" shows "\n" as "\\n".
Solution: Skip over "\" that escapes a backslash. (Gary Johnson)
Files: src/tag.c

Patch 7.4.241

Problem: The string returned by submatch() does not distinguish between a NL from a line break and a NL that stands for a NUL character.
Solution: Add a second argument to return a list. (ZyX)
Files: runtime/doc/eval.txt, src/eval.c, src/proto/regexp.pro,
src/regexp.c, src/testdir/test79.in, src/testdir/test79.ok,
src/testdir/test80.in, src/testdir/test80.ok

Patch 7.4.242

Problem: getreg() does not distinguish between a NL used for a line break and a NL used for a NUL character.
Solution: Add another argument to return a list. (ZyX)
Files: runtime/doc/eval.txt, src/eval.c, src/ops.c, src/proto/ops.pro,
src/vim.h, src/Makefile, src/testdir/test_eval.in,
src/testdir/test_eval.ok, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms

Patch 7.4.243

Problem: Cannot use setreg() to add text that includes a NUL.
Solution: Make setreg() accept a list.
Files: runtime/doc/eval.txt, src/eval.c, src/ops.c, src/proto/ops.pro,
src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.244 (after 7.4.238)

Problem: The smack feature causes stray error messages.
Solution: Remove the error messages.
Files: src/os_unix.c

Patch 7.4.245

Problem: Crash for "vim -u NONE -N -c '&&'".
Solution: Check for the pattern to be NULL. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.4.246

Problem: Configure message for detecting smack are out of sequence.
Solution: Put the messages in the right place. (Kazunobu Kuriyama)
Files: src/configure.in, src/auto/configure

Patch 7.4.247

Problem: When passing input to system() there is no way to keep NUL and NL characters separate.

Solution: Optionally use a list for the system() input. (ZyX)

Files: runtime/doc/eval.txt, src/eval.c

Patch 7.4.248

Problem: Cannot distinguish between NL and NUL in output of system().

Solution: Add systemlist(). (ZyX)

Files: runtime/doc/eval.txt, src/eval.c, src/ex_cmds2.c, src/misc1.c, src/proto/misc1.pro

Patch 7.4.249

Problem: Using setreg() with a list of numbers does not work.

Solution: Use a separate buffer for numbers. (ZyX)

Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.250

Problem: Some test files missing from distribution.

Solution: Add pattern for newly added tests.

Files: Filelist

Patch 7.4.251

Problem: Crash when BufAdd autocommand wipes out the buffer.

Solution: Check for buffer to still be valid. Postpone freeing the buffer structure. (Hirohito Higashi)

Files: src/buffer.c, src/ex_cmds.c, src/fileio.c, src/globals.h

Patch 7.4.252

Problem: Critical error in GTK, removing timer twice.

Solution: Clear the timer after removing it. (James McCoy)

Files: src/gui_gtk_x11.c

Patch 7.4.253

Problem: Crash when using cpp syntax file with pattern using external match. (Havard Garnes)

Solution: Discard match when end column is before start column.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.4.254

Problem: Smack support detection is incomplete.

Solution: Check for attr/xattr.h and specific macro.

Files: src/configure.in, src/auto/configure

Patch 7.4.255

Problem: Configure check for smack doesn't work with all shells. (David Larson)

Solution: Remove spaces in set command.

Files: src/configure.in, src/auto/configure

Patch 7.4.256 (after 7.4.248)

Problem: Using systemlist() may cause a crash and does not handle NUL characters properly.

Solution: Increase the reference count, allocate memory by length. (Yasuhiro)

Matsumoto)
Files: src/eval.c

Patch 7.4.257

Problem: Compiler warning, possibly for mismatch in parameter name.
Solution: Rename the parameter in the declaration.
Files: src/ops.c

Patch 7.4.258

Problem: Configure fails if \$CC contains options.
Solution: Remove quotes around \$CC. (Paul Barker)
Files: src/configure.in, src/auto/configure

Patch 7.4.259

Problem: Warning for misplaced "const".
Solution: Move the "const". (Yukihiro Nakadaira)
Files: src/os_unix.c

Patch 7.4.260

Problem: It is possible to define a function with a colon in the name. It is possible to define a function with a lower case character if a "#" appears after the name.
Solution: Disallow using a colon other than with "s:". Ignore "#" after the name.
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.261

Problem: When updating the window involves a regexp pattern, an interactive substitute to replace a "\n" with a line break fails. (Ingo Karkat)
Solution: Set reg_line_lbr in vim_regsub() and vim_regsub_multi().
Files: src/regexp.c, src/testdir/test79.in, src/testdir/test79.ok

Patch 7.4.262

Problem: Duplicate code in regexec().
Solution: Add line_lbr flag to regexec_nl().
Files: src/regexp.c, src/regexp_nfa.c, src/regexp.h

Patch 7.4.263

Problem: GCC 4.8 compiler warning for hiding a declaration (François Gannaz)
Solution: Remove the second declaration.
Files: src/eval.c

Patch 7.4.264 (after 7.4.260)

Problem: Can't define a function starting with "g:". Can't assign a funcref to a buffer-local variable.
Solution: Skip "g:" at the start of a function name. Don't check for colons when assigning to a variable.
Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.265 (after 7.4.260)

Problem: Can't call a global function with "g:" in an expression.
Solution: Skip the "g:" when looking up the function.

Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.266

Problem: Test 62 fails.

Solution: Set the language to C. (Christian Brabandt)

Files: src/testdir/test62.in

Patch 7.4.267 (after 7.4.178)

Problem: The '[' mark is in the wrong position after "gq". (Ingo Karkat)

Solution: Add the setmark argument to do_join(). (Christian Brabandt)

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_autoformat_join.in,
src/testdir/test_autoformat_join.ok, src/Makefile, src/edit.c,
src/ex_cmds.c, src/ex_docmd.c, src/normal.c, src/ops.c,
src/proto/ops.pro

Patch 7.4.268

Problem: Using exists() on a funcref for a script-local function does not work.

Solution: Translate <SNR> to the special byte sequence. Add a test.

Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok,
src/testdir/test_eval_func.vim, Filelist

Patch 7.4.269

Problem: **CTRL-U** in Insert mode does not work after using a cursor key. (Pine Wu)

Solution: Use the original insert start position. (Christian Brabandt)

Files: src/edit.c, src/testdir/test29.in, src/testdir/test29.ok

Patch 7.4.270

Problem: Comparing pointers instead of the string they point to.

Solution: Use strcmp(). (Ken Takata)

Files: src/gui_gtk_x11.c

Patch 7.4.271

Problem: Compiler warning on 64 bit windows.

Solution: Add type cast. (Mike Williams)

Files: src/ops.c

Patch 7.4.272

Problem: Using just "\$" does not cause an error message.

Solution: Check for empty environment variable name. (Christian Brabandt)

Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.273

Problem: "make autoconf" and "make reconfig" may first run configure and then remove the output.

Solution: Add these targets to the exceptions. (Ken Takata)

Files: src/Makefile

Patch 7.4.274

Problem: When doing ":update" just before running an external command that

changes the file, the timestamp may be unchanged and the file is not reloaded.

Solution: Also check the file size.

Files: src/fileio.c

Patch 7.4.275

Problem: When changing the type of a sign that hasn't been placed there is no error message.

Solution: Add an error message. (Christian Brabandt)

Files: src/ex_cmds.c

Patch 7.4.276

Problem: The fish shell is not supported.

Solution: Use begin/end instead of () for fish. (Andy Russell)

Files: src/ex_cmds.c, src/misc1.c, src/option.c, src/proto/misc1.pro

Patch 7.4.277

Problem: Using ":sign unplace *" may leave the cursor in the wrong position (Christian Brabandt)

Solution: Update the cursor position when removing all signs.

Files: src/buffer.c

Patch 7.4.278

Problem: list_remove() conflicts with function defined in Sun header file.

Solution: Rename the function. (Richard Palo)

Files: src/eval.c, src/if_lua.c, src/if_py_both.h, src/proto/eval.pro

Patch 7.4.279

Problem: globpath() returns a string, making it difficult to get a list of matches. (Greg Novack)

Solution: Add an optional argument like with glob(). (Adnan Zafar)

Files: runtime/doc/eval.txt, src/eval.c, src/ex_getln.c, src/misc1.c, src/misc2.c, src/proto/ex_getln.pro, src/proto/misc2.pro, src/testdir/test97.in, src/testdir/test97.ok

Patch 7.4.280

Problem: When using a session file the relative position of the cursor is not restored if there is another tab. (Nobuhiro Takasaki)

Solution: Update w_wrow before calculating the fraction.

Files: src/window.c

Patch 7.4.281

Problem: When a session file has more than one tabpage and 'showtabline' is one the positions may be slightly off.

Solution: Set 'showtabline' to two while positioning windows.

Files: src/ex_docmd.c

Patch 7.4.282 (after 7.4.279)

Problem: Test 97 fails on Mac.

Solution: Do not ignore case in file names. (Jun Takimoto)

Files: src/testdir/test97.in

Patch 7.4.283 (after 7.4.276)

Problem: Compiler warning about unused variable. (Charles Cooper)

Solution: Move the variable inside the #if block.
Files: src/ex_cmds.c

Patch 7.4.284

Problem: Setting '**langmap**' in the modeline can cause trouble. E.g. mapping ":" breaks many commands. (Jens-Wolfhard Schicke-Uffmann)

Solution: Disallow setting '**langmap**' from the modeline.
Files: src/option.c

Patch 7.4.285

Problem: When '**relativenumber**' is set and deleting lines or undoing that, line numbers are not always updated. (Robert Arkwright)

Solution: (Christian Brabandt)
Files: src/misc1.c

Patch 7.4.286

Problem: Error messages are inconsistent. (ZyX)

Solution: Change "Lists" to "list".
Files: src/eval.c

Patch 7.4.287

Problem: Patches for .hgignore don't work, since the file is not in the distribution.

Solution: Add .hgignore to the distribution. Will be effective with the next version.

Files: Filelist

Patch 7.4.288

Problem: When '**spellfile**' is set the screen is not redrawn.

Solution: Redraw when updating the spelling info. (Christian Brabandt)
Files: src/spell.c

Patch 7.4.289

Problem: Pattern with repeated backreference does not match with new regexp engine. (Urtica Dioica)

Solution: Also check the end of a submatch when deciding to put a state in the state list.

Files: src/testdir/test64.in, src/testdir/test64.ok, src/regexp_nfa.c

Patch 7.4.290

Problem: A non-greedy match followed by a branch is too greedy. (Ingo Karkat)

Solution: Add NFA_MATCH when it is already in the state list if the position differs.

Files: src/testdir/test64.in, src/testdir/test64.ok, src/regexp_nfa.c

Patch 7.4.291

Problem: Compiler warning for int to pointer of different size when DEBUG is defined.

Solution: use msg() instead of MSG3().

Files: src/regexp.c

Patch 7.4.292

Problem: Searching for "a" does not match accented "a" with new regexp

engine, does match with old engine. (David Bürgin)
"ca" does not match "ca" with accented "a" with either engine.
Solution: Change the old engine, check for following composing character
also for single-byte patterns.
Files: src/regexp.c, src/testdir/test95.in, src/testdir/test95.ok

Patch 7.4.293

Problem: It is not possible to ignore composing characters at a specific
point in a pattern.
Solution: Add the %C item.
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test95.in,
src/testdir/test95.ok, runtime/doc/pattern.txt

Patch 7.4.294 (7.4.293)

Problem: Test files missing from patch.
Solution: Patch the test files.
Files: src/testdir/test95.in, src/testdir/test95.ok

Patch 7.4.295

Problem: Various typos, bad white space and unclear comments.
Solution: Fix typos. Improve white space. Update comments.
Files: src/testdir/test49.in, src/macros.h, src/screen.c, src/structs.h,
src/gui_gtk_x11.c, src/os_unix.c

Patch 7.4.296

Problem: Can't run tests on Solaris.
Solution: Change the way VIMRUNTIME is set. (Laurent Blume)
Files: src/testdir/Makefile

Patch 7.4.297

Problem: Memory leak from result of get_isolated_shell_name().
Solution: Free the memory. (Dominique Pelle)
Files: src/ex_cmds.c, src/misc1.c

Patch 7.4.298

Problem: Can't have a funcref start with "t:".
Solution: Add "t" to the list of accepted names. (Yukihiro Nakadaira)
Files: src/eval.c

Patch 7.4.299

Problem: When running configure twice DYNAMIC_PYTHON_DLL may become empty.
Solution: Use AC_CACHE_VAL. (Ken Takata)
Files: src/configure.in, src/auto/configure

Patch 7.4.300

Problem: The way config.cache is removed doesn't always work.
Solution: Always remove config.cache. (Ken Takata)
Files: src/Makefile

Patch 7.4.301 (after 7.4.280)

Problem: Still a scrolling problem when loading a session file.
Solution: Fix off-by-one mistake. (Nobuhiro Takasaki)
Files: src/window.c

Patch 7.4.302

Problem: Signs placed with **'foldcolumn'** set don't show up after filler lines.

Solution: Take filler lines into account. (Olaf Dabrunz)

Files: src/screen.c

Patch 7.4.303

Problem: When using double-width characters the text displayed on the command line is sometimes truncated.

Solution: Reset the string length. (Nobuhiro Takasaki)

Files: src/screen.c

Patch 7.4.304

Problem: Cannot always use Python with Vim.

Solution: Add the manifest to the executable. (Jacques Germishuys)

Files: src/Make_mvc.mak

Patch 7.4.305

Problem: Making **'ttymouse'** empty after the xterm version was requested causes problems. (Elijah Griffin)

Solution: Do not check for DEC mouse sequences when the xterm version was requested. Also don't request the xterm version when DEC mouse was enabled.

Files: src/term.c, src/os_unix.c, src/proto/term.pro, src/globals.h

Patch 7.4.306

Problem: getchar(0) does not return Esc.

Solution: Do not wait for an Esc sequence to be complete. (Yasuhiro Matsumoto)

Files: src/eval.c, src/getchar.c

Patch 7.4.307 (after 7.4.305)

Problem: Can't build without the +termresponse feature.

Solution: Add proper #ifdefs.

Files: src/os_unix.c, src/term.c

Patch 7.4.308

Problem: When using **":diffsplit"** on an empty file the cursor is displayed on the command line.

Solution: Limit the value of w_topfill.

Files: src/diff.c

Patch 7.4.309

Problem: When increasing the size of the lower window, the upper window jumps back to the top. (Ron Aaron)

Solution: Change setting the topline. (Nobuhiro Takasaki)

Files: src/window.c

Patch 7.4.310

Problem: getpos()/setpos() don't include curswant.

Solution: Add a fifth number when getting/setting the cursor.

Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok, runtime/doc/eval.txt

Patch 7.4.311

Problem: Can't use winrestview to only restore part of the view.
Solution: Handle missing items in the dict. (Christian Brabandt)
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.312

Problem: Cannot figure out what argument list is being used for a window.
Solution: Add the arglistid() function. (Marcin Szamotulski)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/eval.c, src/ex_docmd.c, src/globals.h, src/structs.h, src/main.c

Patch 7.4.313 (after 7.4.310)

Problem: Changing the return value of getpos() causes an error. (Jie Zhu)
Solution: Revert getpos() and add getcurpos().
Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok, runtime/doc/eval.txt

Patch 7.4.314

Problem: Completion messages can get in the way of a plugin.
Solution: Add 'c' flag to '**shortmess**' option. (Shougo Matsu)
Files: runtime/doc/options.txt, src/edit.c, src/option.h, src/screen.c

Patch 7.4.315 (after 7.4.309)

Problem: Fixes for computation of topline not tested.
Solution: Add test. (Hirohito Higashi)
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test107.in, src/testdir/test107.ok

Patch 7.4.316

Problem: Warning from 64-bit compiler.
Solution: Add type cast. (Mike Williams)
Files: src/ex_getln.c

Patch 7.4.317

Problem: Crash when starting gvim. Issue 230.
Solution: Check for a pointer to be NULL. (Christian Brabandt)
Files: src/window.c

Patch 7.4.318

Problem: Check for whether a highlight group has settings ignores fg and bg color settings.
Solution: Also check cterm and GUI color settings. (Christian Brabandt)
Files: src/syntax.c

Patch 7.4.319

Problem: Crash when putting zero bytes on the clipboard.
Solution: Do not support the utf8_atom target when not using a Unicode encoding. (Naofumi Honda)
Files: src/ui.c

Patch 7.4.320

Problem: Possible crash when an BufLeave autocommand deletes the buffer.

Solution: Check for the window pointer being valid. Postpone freeing the window until autocommands are done. (Yasuhiro Matsumoto)
Files: src/buffer.c, src/fileio.c, src/globals.h, src/window.c

Patch 7.4.321

Problem: Can't build with strawberry perl 5.20 + mingw-w64-4.9.0.
Solution: Define save_strlen. (Ken Takata)
Files: src/if_perl.xs

Patch 7.4.322

Problem: Using "msgfmt" is hard coded, cannot use "gmsgfmt".
Solution: Use the msgfmt command found by configure. (Danek Duvall)
Files: src/config.mk.in, src/po/Makefile

Patch 7.4.323

Problem: substitute() with zero width pattern breaks multibyte character.
Solution: Take multibyte character size into account. (Yukihiro Nakadaira)
Files: src/eval.c src/testdir/test69.in, src/testdir/test69.ok

Patch 7.4.324

Problem: In Ex mode, cyrillic characters are not handled. (Stas Malavin)
Solution: Support multibyte characters in Ex mode. (Yukihiro Nakadaira)
Files: src/ex_getln.c

Patch 7.4.325

Problem: When starting the gui and changing the window size the status line may not be drawn correctly.
Solution: Catch new_win_height() being called recursively. (Christian Brabandt)
Files: src/window.c

Patch 7.4.326

Problem: Can't build Tiny version. (Elimar Riesebieter)
Solution: Add #ifdef.
Files: src/window.c

Patch 7.4.327

Problem: When **'verbose'** is set to display the return value of a function, may get E724 repeatedly.
Solution: Do not give an error for verbose messages. Abort conversion to string after an error.
Files: src/eval.c

Patch 7.4.328

Problem: Selection of inner block is inconsistent.
Solution: Skip indent not only for '}' but all parens. (Tom McDonald)
Files: src/search.c

Patch 7.4.329

Problem: When moving the cursor and then switching to another window the previous window isn't scrolled. (Yukihiro Nakadaira)
Solution: Call update_topline() before leaving the window. (Christian Brabandt)
Files: src/window.c

Patch 7.4.330

Problem: Using a regexp pattern to highlight a specific position can be slow.

Solution: Add matchaddpos() to highlight specific positions efficiently. (Alexey Radkov)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, runtime/plugin/matchparen.vim, src/eval.c, src/ex_docmd.c, src/proto/window.pro, src/screen.c, src/structs.h, src/testdir/test63.in, src/testdir/test63.ok, src/window.c

Patch 7.4.331

Problem: Relative numbering not updated after a linewise yank. Issue 235.

Solution: Redraw after the yank. (Christian Brabandt)

Files: src/ops.c

Patch 7.4.332

Problem: GTK: When a sign icon doesn't fit exactly there can be ugly gaps.

Solution: Scale the sign to fit when the aspect ratio is not too far off. (Christian Brabandt)

Files: src/gui_gtk_x11.c

Patch 7.4.333

Problem: Compiler warning for unused function.

Solution: Put the function inside the #ifdef.

Files: src/screen.c

Patch 7.4.334 (after 7.4.330)

Problem: Uninitialized variables, causing some problems.

Solution: Initialize the variables. (Dominique Pelle)

Files: src/screen.c, src/window.c

Patch 7.4.335

Problem: No digraph for the new rouble sign.

Solution: Add the digraphs =R and =P.

Files: src/digraph.c, runtime/doc/digraph.txt

Patch 7.4.336

Problem: Setting **'history'** to a big value causes out-of-memory errors.

Solution: Limit the value to 10000. (Hirohito Higashi)

Files: runtime/doc/options.txt, src/option.c

Patch 7.4.337

Problem: When there is an error preparing to edit the command line, the command won't be executed. (Hirohito Higashi)

Solution: Reset did_emsg before editing.

Files: src/ex_getln.c

Patch 7.4.338

Problem: Cannot wrap lines taking indent into account.

Solution: Add the **'breakindent'** option. (many authors, final improvements by Christian Brabandt)

Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/optwin.vim, src/buffer.c, src/charset.c, src/edit.c, src/ex_getln.c,

src/getchar.c, src/misc1.c, src/misc2.c, src/ops.c, src/option.c,
src/option.h, src/proto/charset.pro, src/proto/misc1.pro,
src/proto/option.pro, src/screen.c, src/structs.h,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_breakindent.in, src/testdir/test_breakindent.ok,
src/ui.c, src/version.c

Patch 7.4.339

Problem: Local function is available globally.
Solution: Add "static".
Files: src/option.c, src/proto/option.pro

Patch 7.4.340

Problem: Error from sed about illegal bytes when installing Vim.
Solution: Prepend LC_ALL=C. (Itchyny)
Files: src/installman.sh

Patch 7.4.341

Problem: sort() doesn't handle numbers well.
Solution: Add an argument to specify sorting on numbers. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test55.in,
src/testdir/test55.ok

Patch 7.4.342

Problem: Clang gives warnings.
Solution: Add an else block. (Dominique Pelle)
Files: src/gui_beval.c

Patch 7.4.343

Problem: matchdelete() does not always update the right lines.
Solution: Fix off-by-one error. (Ozaki Kiichi)
Files: src/window.c

Patch 7.4.344

Problem: Unnecessary initializations and other things related to
matchaddpos().
Solution: Code cleanup. (Alexey Radkov)
Files: runtime/doc/eval.txt, src/screen.c, src/window.c

Patch 7.4.345 (after 7.4.338)

Problem: Indent is not updated when deleting indent.
Solution: Remember changedtick.
Files: src/misc1.c

Patch 7.4.346 (after 7.4.338)

Problem: Indent is not updated when changing **'breakindentopt'**. (itchyny)
Solution: Do not cache "brishift". (Christian Brabandt)
Files: src/misc1.c

Patch 7.4.347

Problem: test55 fails on some systems.
Solution: Remove the elements that all result in zero and can end up in an

Files: arbitrary position.
src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.348

Problem: When using "J1" in '**cinoptions**' a line below a continuation line gets too much indent.

Solution: Fix parentheses in condition.

Files: src/misc1.c

Patch 7.4.349

Problem: When there are matches to highlight the whole window is redrawn, which is slow.

Solution: Only redraw everything when lines were inserted or deleted.
Reset b_mod_xlines when needed. (Alexey Radkov)

Files: src/screen.c, src/window.c

Patch 7.4.350

Problem: Using C indenting for Javascript does not work well for a {} block inside parentheses.

Solution: When looking for a matching paren ignore one that is before the start of a {} block.

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.351

Problem: sort() is not stable.

Solution: When the items are identical, compare the pointers.

Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.352

Problem: With '**linebreak**' a tab causes a missing line break.

Solution: Count a tab for what it's worth also for shorter lines.
(Christian Brabandt)

Files: src/charset.c

Patch 7.4.353

Problem: '**linebreak**' doesn't work with the '**list**' option.

Solution: Make it work. (Christian Brabandt)

Files: runtime/doc/options.txt, src/charset.c, src/screen.c,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_listlbr.in, src/testdir/test_listlbr.ok

Patch 7.4.354

Problem: Compiler warning.

Solution: Change NUL to NULL. (Ken Takata)

Files: src/screen.c

Patch 7.4.355

Problem: Several problems with Javascript indenting.

Solution: Improve Javascript indenting.

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.356

Problem: Mercurial does not ignore memfile_test. (Daniel Hahler)
Solution: Add memfile_test to ignored files, remove trailing spaces.
Files: .hgignore

Patch 7.4.357

Problem: After completion some characters are not redrawn.
Solution: Clear the command line unconditionally. (Jacob Niehus)
Files: src/edit.c

Patch 7.4.358 (after 7.4.351)

Problem: Sort is not always stable.
Solution: Add an index instead of relying on the pointer to remain the same.
Idea by Jun Takimoto.
Files: src/eval.c

Patch 7.4.359

Problem: When **'ttymouse'** is set to **'uxterm'** the xterm version is not requested. (Tomas Janousek)
Solution: Do not mark uxterm as a conflict mouse and add resume_get_esc_sequence().
Files: src/term.c, src/os_unix.c, src/proto/term.pro

Patch 7.4.360

Problem: In a regexp pattern a "\$" followed by \v or \V is not seen as the end-of-line.
Solution: Handle the situation. (Ozaki Kiichi)
Files: src/regexp.c

Patch 7.4.361

Problem: Lots of flickering when filling the preview window for **'omnifunc'**.
Solution: Disable redrawing. (Hirohito Higashi)
Files: src/popupmnu.c

Patch 7.4.362

Problem: When matchaddpos() uses a length smaller than the number of bytes in the (last) character the highlight continues until the end of the line.
Solution: Change condition from equal to larger-or-equal.
Files: src/screen.c

Patch 7.4.363

Problem: In Windows console typing 0xCE does not work.
Solution: Convert 0xCE to K_NUL 3. (Nobuhiro Takasaki et al.)
Files: src/os_win32.c, src/term.c

Patch 7.4.364

Problem: When the viminfo file can't be renamed there is no error message. (Vladimir Berezhnoy)
Solution: Check for the rename to fail.
Files: src/ex_cmds.c

Patch 7.4.365

Problem: Crash when using ":botright split" when there isn't much space.
Solution: Add a check for the minimum width/height. (Yukihiro Nakadaira)

Files: src/window.c

Patch 7.4.366

Problem: Can't run the linebreak test on MS-Windows.

Solution: Fix the output file name. (Taro Muraoka)

Files: src/testdir/Make_dos.mak

Patch 7.4.367 (after 7.4.357)

Problem: Other solution for redrawing after completion.

Solution: Schedule a window redraw instead of just clearing the command line. (Jacob Niehus)

Files: src/edit.c

Patch 7.4.368

Problem: Restoring the window sizes after closing the command line window doesn't work properly if there are nested splits.

Solution: Restore the sizes twice. (Hirohito Higashi)

Files: src/window.c

Patch 7.4.369

Problem: Using freed memory when exiting while compiled with EXITFREE.

Solution: Set curwin to NULL and check for that. (Dominique Pelle)

Files: src/buffer.c, src/window.c

Patch 7.4.370

Problem: Linebreak test fails when encoding is not utf-8. (Danek Duvall)

Solution: Split the test in a single byte one and a utf-8 one. (Christian Brabandt)

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_listlbr.in, src/testdir/test_listlbr.ok, src/testdir/test_listlbr_utf8.in, src/testdir/test_listlbr_utf8.ok

Patch 7.4.371

Problem: When **'linebreak'** is set control characters are not correctly displayed. (Kimmy Lindvall)

Solution: Set n_extra. (Christian Brabandt)

Files: src/screen.c

Patch 7.4.372

Problem: When **'winminheight'** is zero there might not be one line for the current window.

Solution: Change the size computations. (Yukihiro Nakadaira)

Files: src/window.c

Patch 7.4.373

Problem: Compiler warning for unused argument and unused variable.

Solution: Add UNUSED. Move variable inside #ifdef.

Files: src/charset.c, src/window.c

Patch 7.4.374

Problem: Character after "fb" command not mapped if it might be a composing character.

Solution: Don't disable mapping when looking for a composing character.
(Jacob Niehus)
Files: src/normal.c

Patch 7.4.375

Problem: Test 63 fails when run with GUI-only Vim.
Solution: Add guibg attributes. (suggested by Mike Soyka)
Files: src/testdir/test63.in

Patch 7.4.376 (after 7.4.367)

Problem: Popup menu flickers too much.
Solution: Remove the forced redraw. (Hirohito Higashi)
Files: src/edit.c

Patch 7.4.377

Problem: When '**equalalways**' is set a split may report "no room" even though there is plenty of room.
Solution: Compute the available room properly. (Yukihiro Nakadaira)
Files: src/window.c

Patch 7.4.378

Problem: Title of quickfix list is not kept for setqflist(list, 'r').
Solution: Keep the title. Add a test. (Lcd)
Files: src/quickfix.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_qf_title.in,
src/testdir/test_qf_title.ok

Patch 7.4.379

Problem: Accessing freed memory after using setqflist(list, 'r'). (Lcd)
Solution: Reset qf_index.
Files: src/quickfix.c

Patch 7.4.380

Problem: Loading python may cause Vim to exit.
Solution: Avoid loading the "site" module. (Taro Muraoka)
Files: src/if_python.c

Patch 7.4.381

Problem: Get u_undo error when backspacing in Insert mode deletes more than one line break. (Ayberk Ozgur)
Solution: Also decrement Insstart.lnum.
Files: src/edit.c

Patch 7.4.382

Problem: Mapping characters may not work after typing Esc in Insert mode.
Solution: Fix the noremap flags for inserted characters. (Jacob Niehus)
Files: src/getchar.c

Patch 7.4.383

Problem: Bad interaction between preview window and omnifunc.
Solution: Avoid redrawing the status line. (Hirohito Higashi)
Files: src/popupmnu.c

Patch 7.4.384

Problem: Test 102 fails when compiled with small features.
Solution: Source small.vim. (Jacob Niehus)
Files: src/testdir/test102.in

Patch 7.4.385

Problem: When building with tiny or small features building the .mo files fails.
Solution: In autoconf do not setup for building the .mo files when it would fail.
Files: src/configure.in, src/auto/configure

Patch 7.4.386

Problem: When splitting a window the changelist position is wrong.
Solution: Copy the changelist position. (Jacob Niehus)
Files: src/window.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_changelist.in,
src/testdir/test_changelist.ok

Patch 7.4.387

Problem: "4gro" replaces one character then executes "ooo". (Urtica Dioica)
Solution: Write the ESC in the second stuff buffer.
Files: src/getchar.c, src/proto/getchar.pro, src/edit.c,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_insertcount.in, src/testdir/test_insertcount.ok

Patch 7.4.388

Problem: With '**linebreak**' set and '**list**' unset a Tab is not counted properly. (Kent Sibilev)
Solution: Check the '**list**' option. (Christian Brabandt)
Files: src/screen.c, src/testdir/test_listlbr_utf8.in,
src/testdir/test_listlbr_utf8.ok

Patch 7.4.389

Problem: Still sometimes Vim enters Replace mode when starting up.
Solution: Use a different solution in detecting the termresponse and location response. (Hayaki Saito)
Files: src/globals.h, src/os_unix.c, src/term.c, src/proto/term.pro

Patch 7.4.390

Problem: Advancing pointer over end of a string.
Solution: Init quote character to -1 instead of zero. (Dominique Pelle)
Files: src/misc1.c

Patch 7.4.391

Problem: No '**cursorline**' highlighting when the cursor is on a line with diff highlighting. (Benjamin Fritz)
Solution: Combine the highlight attributes. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.392

Problem: Not easy to detect type of command line window.
Solution: Add the getcmdwintype() function. (Jacob Niehus)
Files: src/eval.c

Patch 7.4.393

Problem: Text drawing on newer MS-Windows systems is suboptimal. Some multibyte characters are not displayed, even though the same font in Notepad can display them. (Srinath Avadhanula)
Solution: Add the **'renderoptions'** option to enable DirectX drawing. (Taro Muraoka)
Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/various.txt, src/Make_cyg.mak, src/Make_ming.mak, src/Make_mvc.mak, src/eval.c, src/gui_dwrite.cpp, src/gui_dwrite.h, src/gui_w32.c, src/gui_w48.c, src/option.c, src/option.h, src/version.c, src/vim.h, src/proto/gui_w32.pro

Patch 7.4.394 (after 7.4.393)

Problem: When using DirectX last italic character is incomplete.
Solution: Add one to the number of cells. (Ken Takata)
Files: src/gui_w32.c

Patch 7.4.395 (after 7.4.355)

Problem: C indent is wrong below an if with wrapped condition followed by curly braces. (Trevor Powell)
Solution: Make a copy of tryposBrace.
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.396

Problem: When **'clipboard'** is "unnamed", :g/pat/d is very slow. (Praful)
Solution: Only set the clipboard after the last delete. (Christian Brabandt)
Files: src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/globals.h, src/ops.c, src/proto/ui.pro, src/ui.c

Patch 7.4.397

Problem: Matchparen only uses the topmost syntax item.
Solution: Go through the syntax stack to find items. (James McCoy)
Also use getcurpos() when possible.
Files: runtime/plugin/matchparen.vim

Patch 7.4.398 (after 7.4.393)

Problem: Gcc error for the argument of InterlockedIncrement() and InterlockedDecrement(). (Axel Bender)
Solution: Remove "unsigned" from the cRefCount_ declaration.
Files: src/gui_dwrite.cpp

Patch 7.4.399

Problem: Encryption implementation is messy. Blowfish encryption has a weakness.
Solution: Refactor the encryption, store the state in an allocated struct instead of using a save/restore mechanism. Introduce the "blowfish2" method, which does not have the weakness and encrypts the whole undo file. (largely by David Leadbeater)

Files: runtime/doc/editing.txt, runtime/doc/options.txt, src/Makefile, src/blowfish.c, src/crypt.c, src/crypt_zip.c, src/ex_docmd.c, src/fileio.c, src/globals.h, src/main.c, src/memline.c, src/misc2.c, src/option.c, src/proto.h, src/proto/blowfish.pro, src/proto/crypt.pro, src/proto/crypt_zip.pro, src/proto/fileio.pro, src/proto/misc2.pro, src/structs.h, src/undo.c, src/testdir/test71.in, src/testdir/test71.ok, src/testdir/test71a.in, src/testdir/test72.in, src/testdir/test72.ok

Patch 7.4.400

Problem: List of distributed files is incomplete.

Solution: Add recently added files.

Files: Filelist

Patch 7.4.401 (after 7.4.399)

Problem: Can't build on MS-Windows.

Solution: Include the new files in all the Makefiles.

Files: src/Make_bc3.mak, src/Make_bc5.mak, src/Make_cyg.mak, src/Make_dice.mak, src/Make_djg.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_os2.mak, src/Make_sas.mak, Make_vms.mms

Patch 7.4.402

Problem: Test 72 crashes under certain conditions. (Kazunobu Kuriyama)

Solution: Clear the whole bufinfo_T early.

Files: src/undo.c

Patch 7.4.403

Problem: Valgrind reports errors when running test 72. (Dominique Pelle)

Solution: Reset the local '**cryptmethod**' option before storing the seed. Set the seed in the memfile even when there is no block0 yet.

Files: src/fileio.c, src/option.c, src/memline.c

Patch 7.4.404

Problem: Windows 64 bit compiler warnings.

Solution: Add type casts. (Mike Williams)

Files: src/crypt.c, src/undo.c

Patch 7.4.405

Problem: Screen updating is slow when using matches.

Solution: Do not use the ">=" as in patch 7.4.362, check the lnum.

Files: src/screen.c, src/testdir/test63.in, src/testdir/test63.ok

Patch 7.4.406

Problem: Test 72 and 100 fail on MS-Windows.

Solution: Set fileformat to unix in the tests. (Taro Muraoka)

Files: src/testdir/test72.in, src/testdir/test100.in

Patch 7.4.407

Problem: Inserting text for Visual block mode, with cursor movement, repeats the wrong text. (Aleksandar Ivanov)

Solution: Reset the update_Insstart_orig flag. (Christian Brabandt)

Files: src/edit.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.408

Problem: Visual block insert breaks a multibyte character.

Solution: Calculate the position properly. (Yasuhiro Matsumoto)

Files: src/ops.c, src/testdir/test_utf8.in, src/testdir/test_utf8.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.409

Problem: Can't build with Perl on Fedora 20.

Solution: Find xsubpp in another directory. (Michael Henry)

Files: src/Makefile, src/config.mk.in, src/configure.in,
src/auto/configure

Patch 7.4.410

Problem: Fold does not open after search when there is a CmdwinLeave autocommand.

Solution: Restore KeyTyped. (Jacob Niehus)

Files: src/ex_getln.c

Patch 7.4.411

Problem: "foo bar" sorts before "foo" with sort(). (John Little)

Solution: Avoid putting quotes around strings before comparing them.

Files: src/eval.c

Patch 7.4.412

Problem: Can't build on Windows XP with MSVC.

Solution: Add SUBSYSTEM_VER to the Makefile. (Yongwei Wu)

Files: src/Make_mvc.mak, src/INSTALLpc.txt

Patch 7.4.413

Problem: MS-Windows: Using US international keyboard layout, inserting dead key by pressing space does not always work. Issue 250.

Solution: Let MS-Windows translate the message. (John Wellesz)

Files: src/gui_w48.c

Patch 7.4.414

Problem: Cannot define a command only when it's used.

Solution: Add the CmdUndefined autocommand event. (partly by Yasuhiro Matsumoto)

Files: runtime/doc/autocmd.txt, src/ex_docmd.c, src/fileio.c,
src/proto/fileio.pro

Patch 7.4.415 (after 7.4.414)

Problem: Cannot build. Warning for shadowed variable. (John Little)

Solution: Add missing change. Remove declaration.

Files: src/vim.h, src/ex_docmd.c

Patch 7.4.416

Problem: Problem with breakindent/showbreak and tabs.

Solution: Handle tabs differently. (Christian Brabandt)

Files: src/testdir/test_breakindent.in, src/testdir/test_breakindent.ok,

src/charset.c

Patch 7.4.417

Problem: After splitting a window and setting **'breakindent'** the default minimum with is not respected.

Solution: Call briopt_check() when copying options to a new window.

Files: src/option.c, src/proto/option.pro,
src/testdir/test_breakindent.in

Patch 7.4.418

Problem: When leaving ":append" the cursor shape is like in Insert mode. (Jacob Niehus)

Solution: Do not have State set to INSERT when calling getline().

Files: src/ex_cmds.c

Patch 7.4.419

Problem: When part of a list is locked it's possible to make changes.

Solution: Check if any of the list items is locked before make a change. (ZyX)

Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.420

Problem: It's not obvious how to add a new test.

Solution: Add a README file. (Christian Brabandt)

Files: src/testdir/README.txt

Patch 7.4.421

Problem: Crash when searching for "\ze*". (Urtica Dioica)

Solution: Disallow a multi after \ze and \zs.

Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.422

Problem: When using conceal with linebreak some text is not displayed correctly. (Grüner Gimpel)

Solution: Check for conceal mode when using linebreak. (Christian Brabandt)

Files: src/screen.c, src/testdir/test_listlbr.in,
src/testdir/test_listlbr.ok

Patch 7.4.423

Problem: expand("\$shell") does not work as documented.

Solution: Do not escape the \$ when expanding environment variables.

Files: src/os_unix.c, src/misc1.c, src/vim.h

Patch 7.4.424

Problem: Get ml_get error when using Python to delete lines in a buffer that is not in a window. issue 248.

Solution: Do not try adjusting the cursor for a different buffer.

Files: src/if_py_both.h

Patch 7.4.425

Problem: When **'showbreak'** is used "gj" may move to the wrong position. (Nazri Ramliy)

Solution: Adjust virtcol when **'showbreak'** is set. (Christian Brabandt)

Files: src/normal.c

Patch 7.4.426

Problem: README File missing from list of files.
Solution: Update the list of files.
Files: Filelist

Patch 7.4.427

Problem: When an InsertCharPre autocommand executes system() typeahead may be echoed and messes up the display. (Jacob Niehus)
Solution: Do not set cooked mode when invoked from ":silent".
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.428

Problem: executable() may return a wrong result on MS-Windows.
Solution: Change the way SearchPath() is called. (Yasuhiro Matsumoto, Ken Takata)
Files: src/os_win32.c

Patch 7.4.429

Problem: Build fails with fewer features. (Elimar Riesebieter)
Solution: Add #ifdef.
Files: src/normal.c

Patch 7.4.430

Problem: test_listlbr fails when compiled with normal features.
Solution: Check for the +conceal feature.
Files: src/testdir/test_listlbr.in

Patch 7.4.431

Problem: Compiler warning.
Solution: Add type cast. (Mike Williams)
Files: src/ex_docmd.c

Patch 7.4.432

Problem: When the startup code expands command line arguments, setting **'encoding'** will not properly convert the arguments.
Solution: Call get_cmd_argsW() early in main(). (Yasuhiro Matsumoto)
Files: src/os_win32.c, src/main.c, src/os_mswin.c

Patch 7.4.433

Problem: Test 75 fails on MS-Windows.
Solution: Use ":normal" instead of feedkeys(). (Michael Soyka)
Files: src/testdir/test75.in

Patch 7.4.434

Problem: gettabvar() is not consistent with getwinvar() and getbufvar().
Solution: Return a dict with all variables when the varname is empty. (Yasuhiro Matsumoto)
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test91.in, src/testdir/test91.ok

Patch 7.4.435

Problem: Line formatting behaves differently when **'linebreak'** is set. (mvxxc)

Solution: Disable **'linebreak'** temporarily. (Christian Brabandt)
Files: src/edit.c

Patch 7.4.436

Problem: ml_get error for autocommand that moves the cursor of the current window.

Solution: Check the cursor position after switching back to the current buffer. (Christian Brabandt)

Files: src/fileio.c

Patch 7.4.437

Problem: New and old regexp engine are not consistent.

Solution: Also give an error for "\ze*" for the old regexp engine.

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.4.438

Problem: Cached values for **'cino'** not reset for ":set all&".

Solution: Call parse_cino(). (Yukihiro Nakadaira)

Files: src/option.c

Patch 7.4.439

Problem: Duplicate message in message history. Some quickfix messages appear twice. (Gary Johnson)

Solution: Do not reset keep_msg too early. (Hirohito Higashi)

Files: src/main.c

Patch 7.4.440

Problem: Omni complete popup drawn incorrectly.

Solution: Call validate_cursor() instead of check_cursor(). (Hirohito Higashi)

Files: src/edit.c

Patch 7.4.441

Problem: Endless loop and other problems when **'cedit'** is set to **CTRL-C**.

Solution: Do not call ex_window() when ex_normal_busy or got_int was set. (Yasuhiro Matsumoto)

Files: src/ex_getln.c

Patch 7.4.442 (after 7.4.434)

Problem: Using uninitialized variable.

Solution: Pass the first window of the tabpage.

Files: src/eval.c

Patch 7.4.443

Problem: Error reported by ubsan when running test 72.

Solution: Add type cast to unsigned. (Dominique Pelle)

Files: src/undo.c

Patch 7.4.444

Problem: Reversed question mark not recognized as punctuation. (Issue 258)

Solution: Add the Supplemental Punctuation range.

Files: src/mbyte.c

Patch 7.4.445

Problem: Clipboard may be cleared on startup.
Solution: Set clip_did_set_selection to -1 during startup. (Christian Brabandt)
Files: src/main.c, src/ui.c

Patch 7.4.446

Problem: In some situations, when setting up an environment to trigger an autocommand, the environment is not properly restored.
Solution: Check the return value of switch_win() and call restore_win() always. (Daniel Hahler)
Files: src/eval.c, src/misc2.c, src/window.c

Patch 7.4.447

Problem: Spell files from Hunspell may generate a lot of errors.
Solution: Add the IGNOREEXTRA flag.
Files: src/spell.c, runtime/doc/spell.txt

Patch 7.4.448

Problem: Using ETO_IGNORELANGUAGE causes problems.
Solution: Remove this flag. (Paul Moore)
Files: src/gui_w32.c

Patch 7.4.449

Problem: Can't easily close the help window. (Chris Gaal)
Solution: Add ":helpclose". (Christian Brabandt)
Files: runtime/doc/helphelp.txt, runtime/doc/index.txt, src/ex_cmds.c, src/ex_cmds.h, src/proto/ex_cmds.pro

Patch 7.4.450

Problem: Not all commands that edit another buffer support the +cmd argument.
Solution: Add the +cmd argument to relevant commands. (Marcin Szamotulski)
Files: runtime/doc/windows.txt, src/ex_cmds.h, src/ex_docmd.c

Patch 7.4.451

Problem: Calling system() with empty input gives an error for writing the temp file.
Solution: Do not try writing if the string length is zero. (Olaf Dabrunz)
Files: src/eval.c

Patch 7.4.452

Problem: Can't build with tiny features. (Tony Mechelynck)
Solution: Use "return" instead of "break".
Files: src/ex_cmds.c

Patch 7.4.453

Problem: Still can't build with tiny features.
Solution: Add #ifdef.
Files: src/ex_cmds.c

Patch 7.4.454

Problem: When using a Visual selection of multiple words and doing **CTRL-W_]** it jumps to the tag matching the word under the cursor, not the selected text. (Patrick hemmer)

Solution: Do not reset Visual mode. (idea by Christian Brabandt)
Files: src/window.c

Patch 7.4.455

Problem: Completion for :buf does not use **'wildignorecase'**. (Akshay H)
Solution: Pass the **'wildignorecase'** flag around.
Files: src/buffer.c

Patch 7.4.456

Problem: **'backupcopy'** is global, cannot write only some files in a different way.
Solution: Make **'backupcopy'** global-local. (Christian Brabandt)
Files: runtime/doc/options.txt, src/buffer.c, src/fileio.c, src/option.c, src/option.h, src/proto/option.pro, src/structs.h

Patch 7.4.457

Problem: Using getchar() in an expression mapping may result in K_CURSORHOLD, which can't be recognized.
Solution: Add the <CursorHold> key. (Hirohito Higashi)
Files: src/misc2.c

Patch 7.4.458

Problem: Issue 252: Cursor moves in a zero-height window.
Solution: Check for zero height. (idea by Christian Brabandt)
Files: src/move.c

Patch 7.4.459

Problem: Can't change the icon after building Vim.
Solution: Load the icon from a file on startup. (Yasuhiro Matsumoto)
Files: src/gui_w32.c, src/os_mswin.c, src/os_win32.c, src/proto/os_mswin.pro

Patch 7.4.460 (after 7.4.454)

Problem: Can't build without the quickfix feature. (Erik Falor)
Solution: Add a #ifdef.
Files: src/window.c

Patch 7.4.461

Problem: MS-Windows: When collate is on the number of copies is too high.
Solution: Only set the collated/uncollated count when collate is on. (Yasuhiro Matsumoto)
Files: src/os_mswin.c

Patch 7.4.462

Problem: Setting the local value of **'backupcopy'** empty gives an error. (Peter Mattern)
Solution: When using an empty value set the flags to zero. (Hirohito Higashi)
Files: src/option.c

Patch 7.4.463

Problem: Test 86 and 87 may hang on MS-Windows.
Solution: Call inputrestore() after inputsave(). (Ken Takata)
Files: src/testdir/test86.in, src/testdir/test87.in

Patch 7.4.464 (after 7.4.459)

Problem: Compiler warning.
Solution: Add type cast. (Ken Takata)
Files: src/gui_w32.c

Patch 7.4.465 (after 7.4.016)

Problem: Crash when expanding a very long string.
Solution: Use wcsncpy() instead of wcscpy(). (Ken Takata)
Files: src/os_win32.c

Patch 7.4.466 (after 7.4.460)

Problem: **CTRL-W** } does not open preview window. (Erik Falor)
Solution: Don't set g_do_tagpreview for **CTRL-W** }.
Files: src/window.c

Patch 7.4.467

Problem: **'linebreak'** does not work well together with Visual mode.
Solution: Disable **'linebreak'** while applying an operator. Fix the test.
(Christian Brabandt)
Files: src/normal.c, src/screen.c, src/testdir/test_listlbr.in,
src/testdir/test_listlbr.ok

Patch 7.4.468

Problem: Issue 26: **CTRL-C** does not interrupt after it was mapped and then unmapped.
Solution: Reset mapped_ctrl_c. (Christian Brabandt)
Files: src/getchar.c

Patch 7.4.469 (after 7.4.467)

Problem: Can't build with MSVC. (Ken Takata)
Solution: Move the assignment after the declarations.
Files: src/normal.c

Patch 7.4.470

Problem: Test 11 and 100 do not work properly on Windows.
Solution: Avoid using feedkeys(). (Ken Takata)
Files: src/testdir/Make_dos.mak, src/testdir/test11.in,
src/testdir/test100.in

Patch 7.4.471

Problem: MS-Windows: When printer name contains multibyte, the name is displayed as ???.
Solution: Convert the printer name from the active codepage to **'encoding'**.
(Yasuhiro Matsumoto)
Files: src/os_mswin.c

Patch 7.4.472

Problem: The "precedes" entry in **'listchars'** will be drawn when **'showbreak'** is set and **'list'** is not.
Solution: Only draw this character when **'list'** is on. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.473

Problem: Cursor movement is incorrect when there is a number/sign/fold column and **'sbr'** is displayed.
Solution: Adjust the column for **'sbr'**. (Christian Brabandt)
Files: src/charset.c

Patch 7.4.474

Problem: AIX compiler can't handle // comment. Issue 265.
Solution: Remove that line.
Files: src/regex_nfa.c

Patch 7.4.475

Problem: Can't compile on a system where Xutf8SetWMPProperties() is not in the X11 library. Issue 265.
Solution: Add a configure check.
Files: src/configure.in, src/auto/configure, src/config.h.in, src/os_unix.c

Patch 7.4.476

Problem: MingW: compiling with "XPM=no" doesn't work.
Solution: Check for the "no" value. (KF Leong) Also for Cygwin. (Ken Takata)
Files: src/Make_ming.mak, src/Make_cyg.mak

Patch 7.4.477

Problem: When using ":%diffput" and the other file is empty an extra empty line remains.
Solution: Set the buf_empty flag.
Files: src/diff.c

Patch 7.4.478

Problem: Using byte length instead of character length for **'showbreak'**.
Solution: Compute the character length. (Marco Hinz)
Files: src/charset.c

Patch 7.4.479

Problem: MS-Windows: The console title can be wrong.
Solution: Take the encoding into account. When restoring the title use the right function. (Yasuhiro Matsumoto)
Files: src/os_mswin.c, src/os_win32.c

Patch 7.4.480 (after 7.4.479)

Problem: MS-Windows: Can't build.
Solution: Remove goto, use a flag instead.
Files: src/os_win32.c

Patch 7.4.481 (after 7.4.471)

Problem: Compiler warning on MS-Windows.
Solution: Add type casts. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.482

Problem: When **'balloonexpr'** results in a list, the text has a trailing newline. (Lcd)
Solution: Remove one trailing newline.

Files: src/gui_beval.c

Patch 7.4.483

Problem: A 0x80 byte is not handled correctly in abbreviations.
Solution: Unescape special characters. Add a test. (Christian Brabandt)
Files: src/getchar.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_mapping.in,
src/testdir/test_mapping.ok

Patch 7.4.484 (after 7.4.483)

Problem: Compiler warning on MS-Windows. (Ken Takata)
Solution: Add type cast.
Files: src/getchar.c

Patch 7.4.485 (after 7.4.484)

Problem: Abbreviations don't work. (Toothpik)
Solution: Move the length computation inside the for loop. Compare against the unescaped key.
Files: src/getchar.c

Patch 7.4.486

Problem: Check for writing to a yank register is wrong.
Solution: Negate the check. (Zyx). Also clean up the #ifdefs.
Files: src/ex_docmd.c, src/ex_cmds.h

Patch 7.4.487

Problem: ":sign jump" may use another window even though the file is already edited in the current window.
Solution: First check if the file is in the current window. (James McCoy)
Files: src/window.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_signs.in,
src/testdir/test_signs.ok

Patch 7.4.488

Problem: test_mapping fails for some people.
Solution: Set the '**encoding**' option. (Ken Takata)
Files: src/testdir/test_mapping.in

Patch 7.4.489

Problem: Cursor movement still wrong when '**\br**' is set and there is a number column. (Hirohito Higashi)
Solution: Add correction for number column. (Hiroyuki Takagi)
Files: src/charset.c

Patch 7.4.490

Problem: Cannot specify the buffer to use for "do" and "dp", making them useless for three-way diff.
Solution: Use the count as the buffer number. (James McCoy)
Files: runtime/doc/diff.txt, src/diff.c, src/normal.c, src/proto/diff.pro

Patch 7.4.491

Problem: When winrestview() has a negative "topline" value there are display errors.
Solution: Correct a negative value to 1. (Hirohito Higashi)
Files: src/eval.c

Patch 7.4.492

Problem: In Insert mode, after inserting a newline that inserts a comment leader, **CTRL-O** moves to the right. (ZyX) Issue 57.
Solution: Correct the condition for moving the cursor back to the NUL. (Christian Brabandt)
Files: src/edit.c, src/testdir/test4.in, src/testdir/test4.ok

Patch 7.4.493

Problem: A TextChanged autocommand is triggered when saving a file. (William Gardner)
Solution: Update last_changedtick after calling unchanged(). (Christian Brabandt)
Files: src/fileio.c

Patch 7.4.494

Problem: Cursor shape is wrong after a CompleteDone autocommand.
Solution: Update the cursor and mouse shape after ":normal" restores the state. (Jacob Niehus)
Files: src/ex_docmd.c

Patch 7.4.495

Problem: XPM isn't used correctly in the Cygwin Makefile.
Solution: Include the rules like in Make_ming.mak. (Ken Takata)
Files: src/Make_cyg.mak

Patch 7.4.496

Problem: Many lines are both in Make_cyg.mak and Make_ming.mak
Solution: Move the common parts to one file. (Ken Takata)
Files: src/INSTALLpc.txt, src/Make_cyg.mak, src/Make_cyg_ming.mak, src/Make_ming.mak, src/Make_mvc.mak, Filelist

Patch 7.4.497

Problem: With some regexp patterns the NFA engine uses many states and becomes very slow. To the user it looks like Vim freezes.
Solution: When the number of states reaches a limit fall back to the old engine. (Christian Brabandt)
Files: runtime/doc/options.txt, src/Makefile, src/regexp.c, src/regexp.h, src/regexp_nfa.c, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Makefile, src/testdir/samples/re.freeze.txt, src/testdir/bench_re_freeze.in, src/testdir/bench_re_freeze.vim, Filelist

Patch 7.4.498 (after 7.4.497)

Problem: Typo in DOS makefile.
Solution: Change exists to exist. (Ken Takata)
Files: src/testdir/Make_dos.mak

Patch 7.4.499

Problem: substitute() can be slow with long strings.
Solution: Store a pointer to the end, instead of calling strlen() every time. (Ozaki Kiichi)
Files: src/eval.c

Patch 7.4.500

Problem: Test 72 still fails once in a while.
Solution: Don't set **'fileformat'** to unix, reset it. (Ken Takata)
Files: src/testdir/test72.in

Patch 7.4.501 (after 7.4.497)

Problem: Typo in file pattern.
Solution: Insert a slash and remove a dot.
Files: Filelist

Patch 7.4.502

Problem: Language mapping also applies to mapped characters.
Solution: Add the **'langnoremap'** option, when on **'langmap'** does not apply to mapped characters. (Christian Brabandt)
Files: runtime/doc/options.txt, runtime/vimrc_example.vim, src/macros.h, src/option.c, src/option.h

Patch 7.4.503

Problem: Cannot append a list of lines to a file.
Solution: Add the append option to writefile(). (Yasuhiro Matsumoto)
Files: runtime/doc/eval.txt, src/Makefile, src/eval.c, src/testdir/test_writefile.in, src/testdir/test_writefile.ok

Patch 7.4.504

Problem: Restriction of the MS-Windows installer that the path must end in "Vim" prevents installing more than one version.
Solution: Remove the restriction. (Tim Lebedkov)
Files: nsis/gvim.nsi

Patch 7.4.505

Problem: On MS-Windows when **'encoding'** is a double-byte encoding a file name longer than MAX_PATH bytes but shorter than that in characters causes problems.
Solution: Fail on file names longer than MAX_PATH bytes. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.506

Problem: MS-Windows: Cannot open a file with 259 characters.
Solution: Fix off-by-one error. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.507 (after 7.4.496)

Problem: Building with MingW and Perl.
Solution: Remove quotes. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.508

Problem: When generating ja.sjis.po the header is not correctly adjusted.

Solution: Check for the right header string. (Ken Takata)
Files: src/po/sjiscorr.c

Patch 7.4.509

Problem: Users are not aware their encryption is weak.
Solution: Give a warning when prompting for the key.
Files: src/crypt.c, src/ex_docmd.c, src/fileio.c, src/main.c,
src/proto/crypt.pro

Patch 7.4.510

Problem: "-fwrapv" argument breaks use of cproto.
Solution: Remove the alphabetic arguments in a drastic way.
Files: src/Makefile

Patch 7.4.511

Problem: Generating proto for if_ruby.c uses type not defined elsewhere.
Solution: Do not generate a prototype for
rb_gc_writebarrier_unprotect_promoted()
Files: src/if_ruby.c

Patch 7.4.512

Problem: Cannot generate prototypes for Win32 files and VMS.
Solution: Add typedefs and #ifdef
Files: src/os_win32.c, src/gui_w32.c, src/os_vms.c

Patch 7.4.513

Problem: Crash because reference count is wrong for list returned by
getreg().
Solution: Increment the reference count. (Kimmy Lindvall)
Files: src/eval.c

Patch 7.4.514 (after 7.4.492)

Problem: Memory access error. (Dominique Pelle)
Solution: Update tpos. (Christian Brabandt)
Files: src/edit.c

Patch 7.4.515

Problem: In a help buffer the global '**foldmethod**' is used. (Paul Marshall)
Solution: Reset '**foldmethod**' when starting to edit a help file. Move the
code to a separate function.
Files: src/ex_cmds.c

Patch 7.4.516

Problem: Completing a function name containing a # does not work. Issue
253.
Solution: Recognize the # character. (Christian Brabandt)
Files: src/eval.c

Patch 7.4.517

Problem: With a wrapping line the cursor may not end up in the right place.
(Nazri Ramliy)
Solution: Adjust n_extra for a Tab that wraps. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.518

Problem: Using status line height in width computations.
Solution: Use one instead. (Hirohito Higashi)
Files: src/window.c

Patch 7.4.519 (after 7.4.497)

Problem: Crash when using syntax highlighting.
Solution: When regprog is freed and replaced, store the result.
Files: src/buffer.c, src/regexp.c, src/syntax.c, src/spell.c,
src/ex_cmds2.c, src/fileio.c, src/proto/fileio.pro,
src/proto/regexp.pro, src/os_unix.c

Patch 7.4.520

Problem: Sun PCK locale is not recognized.
Solution: Add PCK in the table. (Keiichi Oono)
Files: src/mbyte.c

Patch 7.4.521

Problem: When using "vcp" a mark is moved to the next line. (Maxi Padulo,
Issue 283)
Solution: Decrement the line number. (Christian Brabandt)
Files: src/ops.c

Patch 7.4.522

Problem: Specifying wrong buffer size for GetLongPathName().
Solution: Use the actual size. (Ken Takata)
Files: src/eval.c

Patch 7.4.523

Problem: When the X11 server is stopped and restarted, while Vim is kept in
the background, copy/paste no longer works. (Issue 203)
Solution: Setup the clipboard again. (Christian Brabandt)
Files: src/os_unix.c

Patch 7.4.524

Problem: When using ":ownsyntax" spell checking is messed up. (Issue 78)
Solution: Use the window-local option values. (Christian Brabandt)
Files: src/option.c, src/syntax.c

Patch 7.4.525

Problem: map() leaks memory when there is an error in the expression.
Solution: Call clear_tv(). (Christian Brabandt)
Files: src/eval.c

Patch 7.4.526

Problem: matchstr() fails on long text. (Daniel Hahler)
Solution: Return NFA_T00_EXPENSIVE from regexec_nl(). (Christian Brabandt)
Files: src/regexp.c

Patch 7.4.527

Problem: Still confusing regexp failure and NFA_T00_EXPENSIVE.
Solution: NFA changes equivalent of 7.4.526.
Files: src/regexp_nfa.c

Patch 7.4.528

Problem: Crash when using matchadd() (Yasuhiro Matsumoto)
Solution: Copy the match regprog.
Files: src/screen.c

Patch 7.4.529

Problem: No test for what 7.4.517 fixes.
Solution: Adjust the tests for breakindent. (Christian Brabandt)
Files: src/testdir/test_breakindent.in, src/testdir/test_breakindent.ok

Patch 7.4.530

Problem: Many commands take a count or range that is not using line numbers.
Solution: For each command specify what kind of count it uses. For windows, buffers and arguments have "\$" and "." have a relevant meaning. (Marcin Szamotulski)
Files: runtime/doc/editing.txt, runtime/doc/tabpage.txt, runtime/doc/windows.txt, src/Makefile, src/ex_cmds.h, src/ex_docmd.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_argument_count.in, src/testdir/test_argument_count.ok, src/testdir/test_close_count.in, src/testdir/test_close_count.ok, src/window.c

Patch 7.4.531

Problem: Comments about parsing an Ex command are wrong.
Solution: Correct the step numbers.
Files: src/ex_docmd.c

Patch 7.4.532

Problem: When using **'incsearch'** "2/pattern/e" highlights the first match.
Solution: Move the code to set extra_col inside the loop for count. (Ozaki Kiichi)
Files: src/search.c

Patch 7.4.533

Problem: ":hardcopy" leaks memory in case of errors.
Solution: Free memory in all code paths. (Christian Brabandt)
Files: src/hardcopy.c

Patch 7.4.534

Problem: Warnings when compiling if_ruby.c.
Solution: Avoid the warnings. (Ken Takata)
Files: src/if_ruby.c

Patch 7.4.535 (after 7.4.530)

Problem: Can't build with tiny features.
Solution: Add #ifdefs and skip a test.
Files: src/ex_docmd.c, src/testdir/test_argument_count.in

Patch 7.4.536

Problem: Test 63 fails when using a black&white terminal.

Solution: Add attributes for a non-color terminal. (Christian Brabandt)
Files: src/testdir/test63.in

Patch 7.4.537

Problem: Value of v:hsearch reflects an internal variable.
Solution: Make the value reflect whether search highlighting is actually displayed. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/testdir/test101.in, src/testdir/test101.ok, src/vim.h

Patch 7.4.538

Problem: Tests fail with small features plus Python.
Solution: Disallow weird combination of options. Do not set "fdm" when folding is disabled.
Files: src/option.c, src/ex_cmds.c, src/configure.in, src/auto/configure, src/feature.h

Patch 7.4.539 (after 7.4.530)

Problem: Crash when computing buffer count. Problem with range for user commands. Line range wrong in Visual area.
Solution: Avoid segfault in compute_buffer_local_count(). Check for CMD_USER when checking type of range. (Marcin Szamotulski)
Files: runtime/doc/windows.txt, src/ex_docmd.c

Patch 7.4.540 (after 7.4.539)

Problem: Cannot build with tiny and small features. (Taro Muraoka)
Solution: Add #ifdef around CMD_USER.
Files: src/ex_docmd.c

Patch 7.4.541

Problem: Crash when doing a range assign.
Solution: Check for NULL pointer. (Yukihiro Nakadaira)
Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.542

Problem: Using a range for window and buffer commands has a few problems. Cannot specify the type of range for a user command.
Solution: Add the -addr argument for user commands. Fix problems. (Marcin Szamotulski)
Files: src/testdir/test_command_count.in, src/testdir/test_command_count.ok src/testdir/Make_amiga.mak src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, runtime/doc/map.txt, src/Makefile, src/ex_cmds.h, src/ex_docmd.c, src/ex_getln.c, src/proto/ex_docmd.pro, src/vim.h,

Patch 7.4.543

Problem: Since patch 7.4.232 "1,3s/\n/" joins two lines instead of three. (Eliseo Martínez) Issue 287
Solution: Correct the line count. (Christian Brabandt)
Also set the last used search pattern.
Files: src/ex_cmds.c, src/search.c, src/proto/search.pro

Patch 7.4.544

Problem: Warnings for unused arguments when compiling with a combination of features.
Solution: Add "UNUSED".
Files: src/if_cscope.c

Patch 7.4.545

Problem: Highlighting for multi-line matches is not correct.
Solution: Stop highlight at the end of the match. (Hirohito Higashi)
Files: src/screen.c

Patch 7.4.546

Problem: Repeated use of vim_snprintf() with a number.
Solution: Move these vim_snprintf() calls into a function.
Files: src/window.c

Patch 7.4.547

Problem: Using "vit" does not select a multibyte character at the end correctly.
Solution: Advance the cursor over the multibyte character. (Christian Brabandt)
Files: src/search.c

Patch 7.4.548

Problem: Compilation fails with native version of MinGW-w64, because it doesn't have x86_64-w64-mingw32-windres.exe.
Solution: Use windres instead. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.549

Problem: Function name not recognized correctly when inside a function.
Solution: Don't check for an alpha character. (Ozaki Kiichi)
Files: src/eval.c, src/testdir/test_nested_function.in, src/testdir/test_nested_function.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.550

Problem: curs_rows() function is always called with the second argument false.
Solution: Remove the argument. (Christian Brabandt)
validate_botline_win() can then also be removed.
Files: src/move.c

Patch 7.4.551

Problem: "ygn" may yank too much. (Fritzophrenic) Issue 295.
Solution: Check the width of the next match. (Christian Brabandt)
Files: src/search.c, src/testdir/test53.in, src/testdir/test53.ok

Patch 7.4.552

Problem: Langmap applies to Insert mode expression mappings.
Solution: Check for Insert mode. (Daniel Hahler)
Files: src/getchar.c, src/testdir/test_mapping.in,

src/testdir/test_mapping.ok

Patch 7.4.553

Problem: Various small issues.

Solution: Fix those issues.

Files: src/ex_cmds.h, src/gui.h, src/message.c, src/testdir/test39.in,
src/proto/eval.pro, src/proto/misc1.pro, src/proto/ops.pro,
src/proto/screen.pro, src/proto/window.pro. src/os_unix.c,
src/Make_vms.mms, src/proto/os_vms.pro, src/INSTALL

Patch 7.4.554

Problem: Missing part of patch 7.4.519.

Solution: Copy back regprog after calling vim_regexec.

Files: src/quickfix.c

Patch 7.4.555

Problem: test_close_count may fail for some combination of features.

Solution: Require normal features.

Files: src/testdir/test_close_count.in

Patch 7.4.556

Problem: Failed commands in Python interface not handled correctly.

Solution: Restore window and buffer on failure.

Files: src/if_py_both.h

Patch 7.4.557

Problem: One more small issue.

Solution: Update function proto.

Files: src/proto/window.pro

Patch 7.4.558

Problem: When the X server restarts Vim may get stuck.

Solution: Destroy the application context and create it again. (Issue 203)

Files: src/os_unix.c

Patch 7.4.559

Problem: Appending a block in the middle of a tab does not work correctly when virtualedit is set.

Solution: Decrement spaces and count, don't reset them. (James McCoy)

Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.560

Problem: Memory leak using :wviminfo. Issue 296.

Solution: Free memory when needed. (idea by Christian Brabandt)

Files: src/ops.c

Patch 7.4.561

Problem: Ex range handling is wrong for buffer-local user commands.

Solution: Check for CMD_USER_BUF. (Marcin Szamotulski)

Files: src/ex_docmd.c, src/testdir/test_command_count.in,
src/testdir/test_command_count.ok

Patch 7.4.562

Problem: Segfault with wide screen and error in **'rulerformat'**. (Ingo Karkat)

Solution: Check there is enough space. (Christian Brabandt)
Files: src/buffer.c, src/screen.c

Patch 7.4.563

Problem: No test for replacing on a tab in Virtual replace mode.
Solution: Add a test. (Elias Diem)
Files: src/testdir/test48.in, src/testdir/test48.ok

Patch 7.4.564

Problem: FEAT_OSFILETYPE is used even though it's never defined.
Solution: Remove the code. (Christian Brabandt)
Files: src/fileio.c

Patch 7.4.565

Problem: Ranges for arguments, buffers, tabs, etc. are not checked to be valid but limited to the maximum. This can cause the wrong thing to happen.
Solution: Give an error for an invalid value. (Marcin Szamotulski)
Use windows range for ":wincmd".
Files: src/ex_docmd.c, src/ex_cmds.h, src/testdir/test62.in,
src/testdir/test_argument_count.in,
src/testdir/test_argument_count.ok,
src/testdir/test_close_count.in,
src/testdir/test_command_count.in,
src/testdir/test_command_count.ok

Patch 7.4.566

Problem: :argdo, :bufdo, :windo and :tabdo don't take a range.
Solution: Support the range. (Marcin Szamotulski)
Files: runtime/doc/editing.txt, runtime/doc/tabpage.txt,
runtime/doc/windows.txt, src/ex_cmds.h, src/ex_cmds2.c,
src/testdir/test_command_count.in,
src/testdir/test_command_count.ok

Patch 7.4.567

Problem: Non-ascii vertical separator characters are always redrawn.
Solution: Compare only the one byte that's stored. (Thiago Padilha)
Files: src/screen.c

Patch 7.4.568

Problem: Giving an error for ":@wincmd w" is a problem for some plugins.
Solution: Allow the zero in the range. (Marcin Szamotulski)
Files: src/ex_docmd.c, src/testdir/test_command_count.ok

Patch 7.4.569 (after 7.4.468)

Problem: Having **CTRL-C** interrupt or not does not check the mode of the mapping. (Ingo Karkat)
Solution: Use a bitmask with the map mode. (Christian Brabandt)
Files: src/getchar.c, src/structs.h, src/testdir/test_mapping.in,
src/testdir/test_mapping.ok, src/ui.c, src/globals.h

Patch 7.4.570

Problem: Building with dynamic library does not work for Ruby 2.2.0
Solution: Change #ifdefs and #defines. (Ken Takata)

Files: src/if_ruby.c

Patch 7.4.571 (after 7.4.569)

Problem: Can't build with tiny features. (Ike Devolder)

Solution: Add #ifdef.

Files: src/getchar.c

Patch 7.4.572

Problem: Address type of :wincmd depends on the argument.

Solution: Check the argument.

Files: src/ex_docmd.c, src/window.c, src/proto/window.pro

Patch 7.4.573 (after 7.4.569)

Problem: Mapping **CTRL-C** in Visual mode doesn't work. (Ingo Karkat)

Solution: Call get_real_state() instead of using State directly.

Files: src/ui.c, src/testdir/test_mapping.in, src/testdir/test_mapping.ok

Patch 7.4.574

Problem: No error for eval('\$').

Solution: Check for empty name. (Yasuhiro Matsumoto)

Files: src/eval.c

Patch 7.4.575

Problem: Unicode character properties are outdated.

Solution: Update the tables with the latest version.

Files: src/mbyte.c

Patch 7.4.576

Problem: Redrawing problem with **'relativenumber'** and **'linebreak'**.

Solution: Temporarily reset **'linebreak'** and restore it in more places.
(Christian Brabandt)

Files: src/normal.c

Patch 7.4.577

Problem: Matching with a virtual column has a lot of overhead on very long lines. (Issue 310)

Solution: Bail out early if there can't be a match. (Christian Brabandt)
Also check for **CTRL-C** at every position.

Files: src/regexp_nfa.c

Patch 7.4.578

Problem: Using getcurpos() after "\$" in an empty line returns a negative number.

Solution: Don't add one when this would overflow. (Hirohito Higashi)

Files: src/eval.c

Patch 7.4.579

Problem: Wrong cursor positioning when **'linebreak'** is set and lines wrap.

Solution: Fix it. (Christian Brabandt)

Files: src/charset.c, src/screen.c

Patch 7.4.580

Problem: ":52wincmd v" still gives an invalid range error. (Charles Campbell)

Solution: Skip over white space.
Files: src/ex_docmd.c

Patch 7.4.581

Problem: Compiler warnings for uninitialized variables. (John Little)
Solution: Initialize the variables.
Files: src/ops.c

Patch 7.4.582 (after 7.4.577)

Problem: Can't match "%>80v" properly. (Axel Bender)
Solution: Correctly handle ">". (Christian Brabandt)
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.583

Problem: With tiny features test 16 may fail.
Solution: Source small.vim. (Christian Brabandt)
Files: src/testdir/test16.in

Patch 7.4.584

Problem: With tiny features test_command_count may fail.
Solution: Source small.vim. (Christian Brabandt)
Files: src/testdir/test_command_count.in

Patch 7.4.585

Problem: Range for :bdelete does not work. (Ronald Schild)
Solution: Also allow unloaded buffers.
Files: src/ex_cmds.h, src/testdir/test_command_count.in,
src/testdir/test_command_count.ok

Patch 7.4.586

Problem: Parallel building of the documentation html files is not reliable.
Solution: Remove a cyclic dependency. (Reiner Herrmann)
Files: runtime/doc/Makefile

Patch 7.4.587

Problem: Conceal does not work properly with '**linebreak**'. (cs86661)
Solution: Save and restore boguscols. (Christian Brabandt)
Files: src/screen.c, src/testdir/test_listlbr_utf8.in,
src/testdir/test_listlbr_utf8.ok

Patch 7.4.588

Problem: ":0argedit foo" puts the new argument in the second place instead of the first.
Solution: Adjust the range type. (Ingo Karkat)
Files: src/ex_cmds.h, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_argument_0count.in,
src/testdir/test_argument_0count.ok

Patch 7.4.589

Problem: In the MS-Windows console Vim can't handle greek characters when encoding is utf-8.
Solution: Escape K_NUL. (Yasuhiro Matsumoto)

Files: src/os_win32.c

Patch 7.4.590

Problem: Using ctrl_x_mode as if it contains flags.

Solution: Don't use AND with CTRL_X_OMNI. (Hirohito Higashi)

Files: src/edit.c

Patch 7.4.591 (after 7.4.587)

Problem: test_listlbr_utf8 fails when the conceal feature is not available.

Solution: Check for the conceal feature. (Kazunobu Kuriyama)

Files: src/testdir/test_listlbr_utf8.in

Patch 7.4.592

Problem: When doing ":e foobar" when already editing "foobar" and **'buftype'** is "nofile" the buffer is cleared. (Xavier de Gaye)

Solution: Do not clear the buffer.

Files: src/ex_cmds.c

Patch 7.4.593

Problem: Crash when searching for "x\{0,90000}". (Dominique Pelle)

Solution: Bail out from the NFA engine when the max limit is much higher than the min limit.

Files: src/regexp_nfa.c, src/regexp.c, src/vim.h

Patch 7.4.594

Problem: Using a block delete while **'breakindent'** is set does not work properly.

Solution: Use "line" instead of "prev_pend" as the first argument to lbr_chartabsz_adv(). (Hirohito Higashi)

Files: src/ops.c, src/testdir/test_breakindent.in, src/testdir/test_breakindent.ok

Patch 7.4.595

Problem: The test_command_count test fails when using Japanese.

Solution: Force the language to C. (Hirohito Higashi)

Files: src/testdir/test_command_count.in

Patch 7.4.596 (after 7.4.592)

Problem: Tiny build doesn't compile. (Ike Devolder)

Solution: Add #ifdef.

Files: src/ex_cmds.c

Patch 7.4.597

Problem: Cannot change the result of systemlist().

Solution: Initialize v_lock. (Yukihiro Nakadaira)

Files: src/eval.c

Patch 7.4.598

Problem: ":tabdo windo echo **'hi'**" causes "*" register not to be changed. (Salman Halim)

Solution: Change how clip_did_set_selection is used and add clipboard_needs_update and global_change_count. (Christian Brabandt)

Files: src/main.c, src/ui.c, src/testdir/test_eval.in,

src/testdir/test_eval.ok

Patch 7.4.599

Problem: Out-of-memory error.

Solution: Avoid trying to allocate a negative amount of memory, use size_t instead of int. (Dominique Pelle)

Files: src/regexp_nfa.c

Patch 7.4.600

Problem: Memory wasted in struct because of aligning.

Solution: Split pos in lnum and col. (Dominique Pelle)

Files: src/regexp_nfa.c

Patch 7.4.601

Problem: It is not possible to have feedkeys() insert characters.

Solution: Add the 'i' flag.

Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.602

Problem: ":set" does not accept hex numbers as documented.

Solution: Use vim_str2nr(). (ZyX)

Files: src/option.c, runtime/doc/options.txt

Patch 7.4.603

Problem: **'foldcolumn'** may be set such that it fills the whole window, not leaving space for text.

Solution: Reduce the foldcolumn width when there is not sufficient room. (idea by Christian Brabandt)

Files: src/screen.c

Patch 7.4.604

Problem: Running tests changes viminfo.

Solution: Disable viminfo.

Files: src/testdir/test_breakindent.in

Patch 7.4.605

Problem: The # register is not writable, it cannot be restored after jumping around.

Solution: Make the # register writable. (Marcin Szamotulski)

Files: runtime/doc/change.txt, src/ops.c, src/buffer.c, src/globals.h

Patch 7.4.606

Problem: May crash when using a small window.

Solution: Avoid dividing by zero. (Christian Brabandt)

Files: src/normal.c

Patch 7.4.607 (after 7.4.598)

Problem: Compiler warnings for unused variables.

Solution: Move them inside #ifdef. (Kazunobu Kuriyama)

Files: src/ui.c

Patch 7.4.608 (after 7.4.598)

Problem: test_eval fails when the clipboard feature is missing.

Solution: Skip part of the test. Reduce the text used.

Files: src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.609

Problem: For complicated list and dict use the garbage collector can run out of stack space.

Solution: Use a stack of dicts and lists to be marked, thus making it iterative instead of recursive. (Ben Fritz)

Files: src/eval.c, src/if_lua.c, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/proto/eval.pro, src/proto/if_lua.pro, src/proto/if_python.pro, src/proto/if_python3.pro, src/structs.h

Patch 7.4.610

Problem: Some function headers may be missing from generated .pro files.

Solution: Add PROTO to the #ifdef.

Files: src/option.c, src/syntax.c

Patch 7.4.611 (after 7.4.609)

Problem: Syntax error.

Solution: Change statement to return.

Files: src/if_python3.c

Patch 7.4.612

Problem: test_eval fails on Mac.

Solution: Use the * register instead of the + register. (Jun Takimoto)

Files: src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.613

Problem: The NFA engine does not implement the 'redrawtime' time limit.

Solution: Implement the time limit.

Files: src/regexp_nfa.c

Patch 7.4.614

Problem: There is no test for what patch 7.4.601 fixes.

Solution: Add a test. (Christian Brabandt)

Files: src/testdir/test_mapping.in, src/testdir/test_mapping.ok

Patch 7.4.615

Problem: Vim hangs when freeing a lot of objects.

Solution: Do not go back to the start of the list every time. (Yasuhiro Matsumoto and Ariya Mizutani)

Files: src/eval.c

Patch 7.4.616

Problem: Cannot insert a tab in front of a block.

Solution: Correctly compute aop->start. (Christian Brabandt)

Files: src/ops.c, src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.617

Problem: Wrong ":argdo" range does not cause an error.

Solution: Reset "cmd" to NULL. (Marcin Szamotulski, Ingo Karkat)

Files: src/ex_docmd.c

Patch 7.4.618 (after 7.4.609)

Problem: luaV_setref() is missing a return statement. (Ozaki Kiichi)

Solution: Put the return statement back.
Files: src/if_lua.c

Patch 7.4.619 (after 7.4.618)
Problem: luaV_setref() not returning the correct value.
Solution: Return one.
Files: src/if_lua.c

Patch 7.4.620
Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize "did_free". (Ben Fritz)
Files: src/eval.c

Patch 7.4.621 (after 7.4.619)
Problem: Returning 1 in the wrong function. (Raymond Ko)
Solution: Return 1 in the right function (hopefully).
Files: src/if_lua.c

Patch 7.4.622
Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/regexp_nfa.c

Patch 7.4.623
Problem: Crash with pattern: \(\)\{80000} (Dominique Pelle)
Solution: When the max limit is large fall back to the old engine.
Files: src/regexp_nfa.c

Patch 7.4.624
Problem: May leak memory or crash when vim_realloc() returns NULL.
Solution: Handle a NULL value properly. (Mike Williams)
Files: src/if_cscope.c, src/memline.c, src/misc1.c, src/netbeans.c

Patch 7.4.625
Problem: Possible NULL pointer dereference.
Solution: Check for NULL before using it. (Mike Williams)
Files: src/if_py_both.h

Patch 7.4.626
Problem: MSVC with W4 gives useless warnings.
Solution: Disable more warnings. (Mike Williams)
Files: src/vim.h

Patch 7.4.627
Problem: The last screen cell is not updated.
Solution: Respect the "tn" termcap feature. (Hayaki Saito)
Files: runtime/doc/term.txt, src/option.c, src/screen.c, src/term.c, src/term.h

Patch 7.4.628
Problem: Compiler warning for variable might be clobbered by longjmp.
Solution: Add volatile. (Michael Jarvis)
Files: src/main.c

Patch 7.4.629

Problem: Coverity warning for Out-of-bounds read.
Solution: Increase MAXWLEN to 254. (Eliseo Martínez)
Files: src/spell.c

Patch 7.4.630

Problem: When using Insert mode completion combined with autocommands the redo command may not work.
Solution: Do not save the redo buffer when executing autocommands. (Yasuhiro Matsumoto)
Files: src/fileio.c

Patch 7.4.631

Problem: The default conceal character is documented to be a space but it's initially a dash. (Christian Brabandt)
Solution: Make the initial value a space.
Files: src/globals.h

Patch 7.4.632 (after 7.4.592)

Problem: 7.4.592 breaks the netrw plugin, because the autocommands are skipped.
Solution: Roll back the change.
Files: src/ex_cmds.c

Patch 7.4.633

Problem: After 7.4.630 the problem persists.
Solution: Also skip redo when calling a user function.
Files: src/eval.c

Patch 7.4.634

Problem: Marks are not restored after redo + undo.
Solution: Fix the way marks are restored. (Olaf Dabrunz)
Files: src/undo.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_marks.in, src/testdir/test_marks.ok

Patch 7.4.635

Problem: If no NL or CR is found in the first block of a file then the **'fileformat'** may be set to "mac". (Issue 77)
Solution: Check if a CR was found. (eswald)
Files: src/fileio.c

Patch 7.4.636

Problem: A search with end offset gets stuck at end of file. (Gary Johnson)
Solution: When a search doesn't move the cursor repeat it with a higher count. (Christian Brabandt)
Files: src/normal.c, src/testdir/test44.in, src/testdir/test44.ok

Patch 7.4.637

Problem: Incorrectly read the number of buffer for which an autocommand should be registered.
Solution: Reverse check for "<buffer=abuf>". (Lech Lorens)
Files: src/fileio.c

Patch 7.4.638

Problem: Can't build with Lua 5.3 on Windows.
Solution: use luaL_optinteger() instead of LuaL_optlong(). (Ken Takata)
Files: src/if_lua.c

Patch 7.4.639

Problem: Combination of linebreak and conceal doesn't work well.
Solution: Fix the display problems. (Christian Brabandt)
Files: src/screen.c, src/testdir/test88.in, src/testdir/test88.ok,
src/testdir/test_listlbr_utf8.in, src/testdir/test_listlbr_utf8.ok

Patch 7.4.640

Problem: After deleting characters in Insert mode such that lines are
joined undo does not work properly. (issue 324)
Solution: Use Insstart instead of Insstart_orig. (Christian Brabandt)
Files: src/edit.c

Patch 7.4.641

Problem: The tabline menu was using ":999tabnew" which is now invalid.
Solution: Use ":%tabnew" instead. (Florian Degner)
Files: src/normal.c

Patch 7.4.642

Problem: When using "gf" escaped spaces are not handled.
Solution: Recognize escaped spaces.
Files: src/vim.h, src/window.c, src/misc2.c

Patch 7.4.643

Problem: Using the default file format for Mac files. (Issue 77)
Solution: Reset the try_mac counter in the right place. (Oswald)
Files: src/fileio.c, src/testdir/test30.in, src/testdir/test30.ok

Patch 7.4.644

Problem: Stratus VOS doesn't have sync().
Solution: Use fflush(). (Karli Aurelia)
Files: src/memfile.c

Patch 7.4.645

Problem: When splitting the window in a BufAdd autocommand while still in
the first, empty buffer the window count is wrong.
Solution: Do not reset b_nwindows to zero and don't increment it.
Files: src/buffer.c, src/ex_cmds.c

Patch 7.4.646

Problem: ":bufdo" may start at a deleted buffer.
Solution: Find the first not deleted buffer. (Shane Harper)
Files: src/ex_cmds2.c, src/testdir/test_command_count.in,
src/testdir/test_command_count.ok

Patch 7.4.647

Problem: After running the tests on MS-Windows many files differ from their
originals as they were checked out.
Solution: Use a temp directory for executing the tests. (Ken Takata, Taro

Muraoka)
Files: src/testdir/Make_dos.mak

Patch 7.4.648 (after 7.4.647)
Problem: Tests broken on MS-Windows.
Solution: Delete wrong copy line. (Ken Takata)
Files: src/testdir/Make_dos.mak

Patch 7.4.649
Problem: Compiler complains about ignoring return value of fwrite().
(Michael Jarvis)
Solution: Add (void).
Files: src/misc2.c

Patch 7.4.650
Problem: Configure check may fail because the dl library is not used.
Solution: Put "-ldl" in LIBS rather than LDFLAGS. (Ozaki Kiichi)
Files: src/configure.in, src/auto/configure

Patch 7.4.651 (after 7.4.582)
Problem: Can't match "%>80v" properly for multibyte characters.
Solution: Multiply the character number by the maximum number of bytes in a
character. (Yasuhiro Matsumoto)
Files: src/regexp_nfa.c

Patch 7.4.652
Problem: Xxd lacks a few features.
Solution: Use 8 characters for the file position. Add the -e and -o
arguments. (Vadim Vygonets)
Files: src/xxd/xxd.c, runtime/doc/xxd.1

Patch 7.4.653
Problem: Insert mode completion with complete() may have **CTRL-L** work like
CTRL-P.
Solution: Handle completion with complete() differently. (Yasuhiro
Matsumoto, Christian Brabandt, Hirohito Higashi)
Files: src/edit.c

Patch 7.4.654
Problem: glob() and globpath() cannot include links to non-existing files.
(Charles Campbell)
Solution: Add an argument to include all links with glob(). (James McCoy)
Also for globpath().
Files: src/vim.h, src/eval.c, src/ex_getln.c

Patch 7.4.655
Problem: Text deleted by "dit" depends on indent of closing tag.
(Jan Parthey)
Solution: Do not adjust oap->end in do_pending_operator(). (Christian
Brabandt)
Files: src/normal.c, src/search.c, src/testdir/test53.in,
src/testdir/test53.ok

Patch 7.4.656 (after 7.4.654)

Problem: Missing changes for glob() in one file.
Solution: Add the missing changes.
Files: src/misc1.c

Patch 7.4.657 (after 7.4.656)
Problem: Compiler warnings for pointer mismatch.
Solution: Add a typecast. (John Marriott)
Files: src/misc1.c

Patch 7.4.658
Problem: **'formatexpr'** is evaluated too often.
Solution: Only invoke it when beyond the **'textwidth'** column, as it is documented. (James McCoy)
Files: src/edit.c

Patch 7.4.659
Problem: When **'ruler'** is set the preferred column is reset. (Issue 339)
Solution: Don't set curswant when redrawing the status lines.
Files: src/option.c

Patch 7.4.660
Problem: Using freed memory when g:colors_name is changed in the colors script. (oni-link)
Solution: Make a copy of the variable value.
Files: src/syntax.c

Patch 7.4.661
Problem: Using "\0 **CTRL-D**" in Insert mode may have CursorHoldI interfere. (Gary Johnson)
Solution: Don't store K_CURSORHOLD as the last character. (Christian Brabandt)
Files: src/edit.c

Patch 7.4.662
Problem: When 'M' is in the **'cpo'** option then selecting a text object in parentheses does not work correctly.
Solution: Keep 'M' in **'cpo'** when finding a match. (Hirohito Higashi)
Files: src/search.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_textobjects.in, src/testdir/test_textobjects.ok

Patch 7.4.663
Problem: When using netbeans a buffer is not found in another tab.
Solution: When **'switchbuf'** is set to "usetab" then switch to another tab when possible. (Xavier de Gaye)
Files: src/netbeans.c

Patch 7.4.664
Problem: When **'compatible'** is reset **'numberwidth'** is set to 4, but the effect doesn't show until a change is made.
Solution: Check if **'numberwidth'** changed. (Christian Brabandt)
Files: src/screen.c, src/structs.h

Patch 7.4.665

Problem: **'linebreak'** does not work properly with multibyte characters.
Solution: Compute the pointer offset with mb_head_off(). (Yasuhiro Matsumoto)
Files: src/screen.c

Patch 7.4.666

Problem: There is a chance that Vim may lock up.
Solution: Handle timer events differently. (Aaron Burrow)
Files: src/os_unix.c

Patch 7.4.667

Problem: **'colorcolumn'** isn't drawn in a closed fold while **'cursorcolumn'** is. (Carlos Pita)
Solution: Make it consistent. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.668

Problem: Can't use a glob pattern as a regexp pattern.
Solution: Add glob2regpat(). (Christian Brabandt)
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.669

Problem: When netbeans is active the sign column always shows up.
Solution: Only show the sign column once a sign has been added. (Xavier de Gaye)
Files: src/buffer.c, src/edit.c, src/move.c, src/netbeans.c, src/screen.c, src/structs.h

Patch 7.4.670

Problem: Using **'cindent'** for Javascript is less than perfect.
Solution: Improve indenting of continuation lines. (Hirohito Higashi)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.671 (after 7.4.665)

Problem: Warning for shadowing a variable.
Solution: Rename off to mb_off. (Kazunobu Kuriyama)
Files: src/screen.c

Patch 7.4.672

Problem: When completing a shell command, directories in the current directory are not listed.
Solution: When "." is not in \$PATH also look in the current directory for directories.
Files: src/ex_getln.c, src/vim.h, src/misc1.c, src/eval.c, src/os_amiga.c, src/os_msdos.c, src/os_unix.c, src/os_vms.c, src/proto/os_amiga.pro, src/proto/os_msdos.pro, src/proto/os_unix.pro, src/proto/os_win32.pro

Patch 7.4.673

Problem: The first syntax entry gets sequence number zero, which doesn't work. (Clinton McKay)
Solution: Start at number one. (Bjorn Linse)

Files: src/syntax.c

Patch 7.4.674 (after 7.4.672)

Problem: Missing changes in one file.

Solution: Also change the win32 file.

Files: src/os_win32.c

Patch 7.4.675

Problem: When a FileReadPost autocommand moves the cursor inside a line it gets moved back.

Solution: When checking whether an autocommand moved the cursor store the column as well. (Christian Brabandt)

Files: src/ex_cmds.c

Patch 7.4.676

Problem: On Mac, when not using the default Python framework configure doesn't do the right thing.

Solution: Use a linker search path. (Kazunobu Kuriyama)

Files: src/configure.in, src/auto/configure

Patch 7.4.677 (after 7.4.676)

Problem: Configure fails when specifying a python-config-dir. (Lcd)

Solution: Check if PYTHONFRAMEWORKPREFIX is set.

Files: src/configure.in, src/auto/configure

Patch 7.4.678

Problem: When using --remote the directory may end up being wrong.

Solution: Use localdir() to find out what to do. (Xaizek)

Files: src/main.c

Patch 7.4.679

Problem: Color values greater than 255 cause problems on MS-Windows.

Solution: Truncate to 255 colors. (Yasuhiro Matsumoto)

Files: src/os_win32.c

Patch 7.4.680

Problem: **CTRL-W** in Insert mode does not work well for multibyte characters.

Solution: Use mb_get_class(). (Yasuhiro Matsumoto)

Files: src/edit.c, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_erasebackward.in, src/testdir/test_erasebackward.ok,

Patch 7.4.681

Problem: MS-Windows: When Vim is minimized the window height is computed incorrectly.

Solution: When minimized use the previously computed size. (Ingo Karkat)

Files: src/gui_w32.c

Patch 7.4.682

Problem: The search highlighting and match highlighting replaces the cursorline highlighting, this doesn't look good.

Solution: Combine the highlighting. (Yasuhiro Matsumoto)
Files: src/screen.c

Patch 7.4.683

Problem: Typo in the vimtutor command.
Solution: Fix the typo. (Corey Farwell, github pull 349)
Files: vimtutor.com

Patch 7.4.684

Problem: When starting several Vim instances in diff mode, the temp files used may not be unique. (Issue 353)
Solution: Add an argument to vim_tempname() to keep the file.
Files: src/diff.c, src/eval.c, src/ex_cmds.c, src/fileio.c, src/hardcopy.c, src/proto/fileio.pro, src/if_cscope.c, src/memline.c, src/misc1.c, src/os_unix.c, src/quickfix.c, src/spell.c

Patch 7.4.685

Problem: When there are illegal utf-8 characters the old regexp engine may go past the end of a string.
Solution: Only advance to the end of the string. (Dominique Pelle)
Files: src/regexp.c

Patch 7.4.686

Problem: "zr" and "zm" do not take a count.
Solution: Implement the count, restrict the fold level to the maximum nesting depth. (Marcin Szamotulski)
Files: runtime/doc/fold.txt, src/normal.c

Patch 7.4.687

Problem: There is no way to use a different in Replace mode for a terminal.
Solution: Add t_SR. (Omar Sandoval)
Files: runtime/doc/options.txt, runtime/doc/term.txt, runtime/syntax/vim.vim, src/option.c, src/term.c, src/term.h

Patch 7.4.688

Problem: When "\$" is in '**cpo**' the popup menu isn't undrawn correctly. (Issue 166)
Solution: When using the popup menu remove the "\$".
Files: src/edit.c

Patch 7.4.689

Problem: On MS-Windows, when '**autochdir**' is set, diff mode with files in different directories does not work. (Axel Bender)
Solution: Remember the current directory and use it where needed. (Christian Brabandt)
Files: src/main.c

Patch 7.4.690

Problem: Memory access errors when changing indent in Ex mode. Also missing redraw when using **CTRL-U**. (Knill Ino)
Solution: Update pointers after calling ga_grow().
Files: src/ex_getln.c

Patch 7.4.691 (after 7.4.689)

Problem: Can't build with MzScheme.
Solution: Change "cwd" into the global variable "start_dir".
Files: src/main.c

Patch 7.4.692

Problem: Defining SOLARIS for no good reason. (Danek Duvall)
Solution: Remove it.
Files: src/os_unix.h

Patch 7.4.693

Problem: Session file is not correct when there are multiple tab pages.
Solution: Reset the current window number for each tab page. (Jacob Niehus)
Files: src/ex_docmd.c

Patch 7.4.694

Problem: Running tests changes the .viminfo file.
Solution: Disable viminfo in the text objects test.
Files: src/testdir/test_textobjects.in

Patch 7.4.695

Problem: Out-of-bounds read, detected by Coverity.
Solution: Remember the value of cmap for the first matching encoding. Reset cmap to that value if first matching encoding is going to be used. (Eliseo Martínez)
Files: src/hardcopy.c

Patch 7.4.696

Problem: Not freeing memory when encountering an error.
Solution: Free the stack before returning. (Eliseo Martínez)
Files: src/regexp_nfa.c

Patch 7.4.697

Problem: The filename used for ":profile" must be given literally.
Solution: Expand "~" and environment variables. (Marco Hinz)
Files: src/ex_cmds2.c

Patch 7.4.698

Problem: Various problems with locked and fixed lists and dictionaries.
Solution: Disallow changing locked items, fix a crash, add tests. (Olaf Dabrunz)
Files: src/structs.h, src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.699

Problem: E315 when trying to delete a fold. (Yutao Yuan)
Solution: Make sure the fold doesn't go beyond the last buffer line. (Christian Brabandt)
Files: src/fold.c

Patch 7.4.700

Problem: Fold can't be opened after ":move". (Ein Brown)
Solution: Delete the folding information and update it afterwards. (Christian Brabandt)

Files: src/ex_cmds.c, src/fold.c, src/testdir/test45.in,
src/testdir/test45.ok

Patch 7.4.701

Problem: Compiler warning for using uninitialized variable. (Yasuhiro
Matsumoto)

Solution: Initialize it.

Files: src/hardcopy.c

Patch 7.4.702

Problem: Joining an empty list does unnecessary work.

Solution: Let join() return early. (Marco Hinz)

Files: src/eval.c

Patch 7.4.703

Problem: Compiler warning for start_dir unused when building unittests.

Solution: Move start_dir inside the #ifdef.

Files: src/main.c

Patch 7.4.704

Problem: Searching for a character matches an illegal byte and causes
invalid memory access. (Dominique Pelle)

Solution: Do not match an invalid byte when search for a character in a
string. Fix equivalence classes using negative numbers, which
result in illegal bytes.

Files: src/misc2.c, src/regexp.c, src/testdir/test44.in

Patch 7.4.705

Problem: Can't build with Ruby 2.2.

Solution: Add #ifdefs to handle the incompatible change. (Andrei Olsen)

Files: src/if_ruby.c

Patch 7.4.706

Problem: Window drawn wrong when '**laststatus**' is zero and there is a
command-line window. (Yclept Nemo)

Solution: Set the status height a bit later. (Christian Brabandt)

Files: src/window.c

Patch 7.4.707

Problem: Undo files can have their executable bit set.

Solution: Strip of the executable bit. (Mikael Berthe)

Files: src/undo.c

Patch 7.4.708

Problem: gettext() is called too often.

Solution: Do not call gettext() for messages until they are actually used.
(idea by Yasuhiro Matsumoto)

Files: src/eval.c

Patch 7.4.709

Problem: ":tabmove" does not work as documented.

Solution: Make it work consistently. Update documentation and add tests.
(Hirohito Higashi)

Files: src/window.c, runtime/doc/tabpage.txt, src/ex_docmd.c,

src/testdir/test62.in, src/testdir/test62.ok

Patch 7.4.710

Problem: It is not possible to make spaces visible in list mode.
Solution: Add the "space" item to '**listchars**'. (David Bürgin, issue 350)
Files: runtime/doc/options.txt, src/globals.h, src/message.h,
src/screen.c, src/testdir/test_listchars.in,
src/testdir/test_listchars.ok, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile

Patch 7.4.711 (after 7.4.710)

Problem: Missing change in one file.
Solution: Also change option.c
Files: src/option.c

Patch 7.4.712 (after 7.4.710)

Problem: Missing change in another file.
Solution: Also change message.c
Files: src/message.c

Patch 7.4.713

Problem: Wrong condition for #ifdef.
Solution: Change USR_EXRC_FILE2 to USR_VIMRC_FILE2. (Mikael Fourier)
Files: src/os_unix.h

Patch 7.4.714

Problem: Illegal memory access when there are illegal bytes.
Solution: Check the byte length of the character. (Dominique Pelle)
Files: src/regexp.c

Patch 7.4.715

Problem: Invalid memory access when there are illegal bytes.
Solution: Get the length from the text, not from the character. (Dominique Pelle)
Files: src/regexp_nfa.c

Patch 7.4.716

Problem: When using the 'c' flag of ":substitute" and selecting "a" or "l" at the prompt the flags are not remembered for ":%&". (Ingo Karkat)
Solution: Save the flag values and restore them. (Hirohito Higashi)
Files: src/ex_cmds.c

Patch 7.4.717

Problem: ":let list += list" can change a locked list.
Solution: Check for the lock earlier. (Olaf Dabrunz)
Files: src/eval.c, src/testdir/test55.in, src/testdir/test55.ok

Patch 7.4.718

Problem: Autocommands triggered by quickfix cannot get the current title value.
Solution: Set w:quickfix_title earlier. (Yannick)

Also move the check for a title into the function.
Files: src/quickfix.c

Patch 7.4.719

Problem: Overflow when adding MAXCOL to a pointer.
Solution: Subtract pointers instead. (James McCoy)
Files: src/screen.c

Patch 7.4.720

Problem: Can't build with Visual Studio 2015.
Solution: Recognize the "version 14" numbers and omit /nodefaultlib when appropriate. (Paul Moore)
Files: src/Make_mvc.mak

Patch 7.4.721

Problem: When '**list**' is set Visual mode does not highlight anything in empty lines. (mgaleski)
Solution: Check the value of lcs_eol in another place. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.722

Problem: 0x202f is not recognized as a non-breaking space character.
Solution: Add 0x202f to the list. (Christian Brabandt)
Files: runtime/doc/options.txt, src/message.c, src/screen.c

Patch 7.4.723

Problem: For indenting, finding the C++ baseclass can be slow.
Solution: Cache the result. (Hirohito Higashi)
Files: src/misc1.c

Patch 7.4.724

Problem: Vim icon does not show in Windows context menu. (issue 249)
Solution: Load the icon in GvimExt.
Files: src/GvimExt/gvimext.cpp, src/GvimExt/gvimext.h

Patch 7.4.725

Problem: ":call setreg('"', [])" reports an internal error.
Solution: Make the register empty. (Yasuhiro Matsumoto)
Files: src/ops.c

Patch 7.4.726 (after 7.4.724)

Problem: Cannot build GvimExt.
Solution: Set APPVER to 5.0. (KF Leong)
Files: src/GvimExt/Makefile

Patch 7.4.727 (after 7.4.724)

Problem: Cannot build GvimExt with MingW.
Solution: Add -lgdi32. (KF Leong)
Files: src/GvimExt/Make_ming.mak

Patch 7.4.728

Problem: Can't build with some version of Visual Studio 2015.
Solution: Recognize another version 14 number. (Sinan)
Files: src/Make_mvc.mak

Patch 7.4.729 (after 7.4.721)

Problem: Occasional crash with 'list' set.
Solution: Fix off-by-one error. (Christian Brabandt)
Files: src/screen.c

Patch 7.4.730

Problem: When setting the crypt key and using a swap file, text may be encrypted twice or unencrypted text remains in the swap file. (Issue 369)
Solution: Call ml_preserve() before re-encrypting. Set correct index for next pointer block.
Files: src/memfile.c, src/memline.c, src/proto/memline.pro, src/option.c

Patch 7.4.731

Problem: The tab menu shows "Close tab" even when it doesn't work.
Solution: Don't show "Close tab" for the last tab. (John Marriott)
Files: src/gui_w48.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c

Patch 7.4.732

Problem: The cursor line is not always updated for the "O" command.
Solution: Reset the VALID_CROW flag. (Christian Brabandt)
Files: src/normal.c

Patch 7.4.733

Problem: test_listchars breaks on MS-Windows. (Kenichi Ito)
Solution: Set fileformat to "unix". (Christian Brabandt)
Files: src/testdir/test_listchars.in

Patch 7.4.734

Problem: ml_get error when using "p" in a Visual selection in the last line.
Solution: Change the behavior at the last line. (Yukihiro Nakadaira)
Files: src/normal.c, src/ops.c, src/testdir/test94.in, src/testdir/test94.ok

Patch 7.4.735

Problem: Wrong argument for sizeof().
Solution: Use a pointer argument. (Chris Hall)
Files: src/eval.c

Patch 7.4.736

Problem: Invalid memory access.
Solution: Avoid going over the end of a NUL terminated string. (Dominique Pelle)
Files: src/regexp.c

Patch 7.4.737

Problem: On MS-Windows vimgrep over arglist doesn't work (Issue 361)
Solution: Only escape backslashes in ## expansion when it is not used as the path separator. (James McCoy)
Files: src/ex_docmd.c

Patch 7.4.738 (after 7.4.732)

Problem: Can't compile without the syntax highlighting feature.
Solution: Add #ifdef around use of w_p_cul. (Hirohito Higashi)
Files: src/normal.c, src/screen.c

Patch 7.4.739

Problem: In a string "\U" only takes 4 digits, while after **CTRL-V** U eight digits can be used.
Solution: Make "\U" also take eight digits. (Christian Brabandt)
Files: src/eval.c

Patch 7.4.740

Problem: ":!quit" works like ":.quit". (Bohr Shaw)
Solution: Don't exit Vim when a range is specified. (Christian Brabandt)
Files: src/ex_docmd.c, src/testdir/test13.in, src/testdir/test13.ok

Patch 7.4.741

Problem: When using += with ":set" a trailing comma is not recognized. (Issue 365)
Solution: Don't add a second comma. Add a test. (partly by Christian Brabandt)
Files: src/option.c, src/testdir/test_set.in, src/testdir/test_set.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.742

Problem: Cannot specify a vertical split when loading a buffer for a quickfix command.
Solution: Add the "vsplit" value to 'switchbuf'. (Brook Hong)
Files: runtime/doc/options.txt, src/buffer.c, src/option.h

Patch 7.4.743

Problem: "p" in Visual mode causes an unexpected line split.
Solution: Advance the cursor first. (Yukihiro Nakadaira)
Files: src/ops.c, src/testdir/test94.in, src/testdir/test94.ok

Patch 7.4.744

Problem: No tests for Ruby and Perl.
Solution: Add minimal tests. (Ken Takata)
Files: src/testdir/test_perl.in, src/testdir/test_perl.ok, src/testdir/test_ruby.in, src/testdir/test_ruby.ok, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.745

Problem: The entries added by matchaddpos() are returned by getmatches() but can't be set with setmatches(). (Lcd)
Solution: Fix setmatches(). (Christian Brabandt)
Files: src/eval.c, src/testdir/test63.in, src/testdir/test63.ok

Patch 7.4.746

Problem: ":[count]tag" is not always working. (cs86661)
Solution: Set cur_match a bit later. (Hirohito Higashi)

Files: src/tag.c,

Patch 7.4.747

Problem: ":cnext" may jump to the wrong column when setting
'virtualedit=all' (cs86661)

Solution: Reset the coladd field. (Hirohito Higashi)

Files: src/quickfix.c

Patch 7.4.748 (after 7.4.745)

Problem: Buffer overflow.

Solution: Make the buffer larger. (Kazunobu Kuriyama)

Files: src/eval.c

Patch 7.4.749 (after 7.4.741)

Problem: For some options two consecutive commas are OK. (Nikolai Pavlov)

Solution: Add the P_ONECOMMA flag.

Files: src/option.c

Patch 7.4.750

Problem: Cannot build with clang 3.5 on Cygwin with perl enabled.

Solution: Strip "-fdebug-prefix-map" in configure. (Ken Takata)

Files: src/configure.in, src/auto/configure

Patch 7.4.751

Problem: It is not obvious how to enable the address sanitizer.

Solution: Add commented-out flags in the Makefile. (Dominique Pelle)

Also add missing test targets.

Files: src/Makefile

Patch 7.4.752

Problem: Unicode 8.0 not supported.

Solution: Update tables for Unicode 8.0. Avoid E36 when running the script.
(James McCoy)

Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 7.4.753

Problem: Appending in Visual mode with '**linebreak**' set does not work properly. Also when '**selection**' is "exclusive". (Ingo Karkat)

Solution: Recalculate virtual columns. (Christian Brabandt)

Files: src/normal.c, src/testdir/test_listlbr.in,
src/testdir/test_listlbr.ok, src/testdir/test_listlbr_utf8.in,
src/testdir/test_listlbr_utf8.ok

Patch 7.4.754

Problem: Using **CTRL-A** in Visual mode does not work well. (Gary Johnson)

Solution: Make it increment all numbers in the Visual area. (Christian Brabandt)

Files: runtime/doc/change.txt, src/normal.c, src/ops.c,
src/proto/ops.pro, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_increment.in,
src/testdir/test_increment.ok

Patch 7.4.755

Problem: It is not easy to count the number of characters.
Solution: Add the skipcc argument to strchr(). (Hirohito Higashi, Ken Takata)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_utf8.in, src/testdir/test_utf8.ok

Patch 7.4.756

Problem: Can't use strawberry Perl 5.22 x64 on MS-Windows.
Solution: Add new defines and #if. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/if_perl.xs

Patch 7.4.757

Problem: Cannot detect the background color of a terminal.
Solution: Add T_RBG to request the background color if possible. (Lubomir Rintel)
Files: src/main.c, src/term.c, src/term.h, src/proto/term.pro

Patch 7.4.758

Problem: When '**conceallevel**' is 1 and quitting the command-line window with **CTRL-C** the first character ':' is erased.
Solution: Reset '**conceallevel**' in the command-line window. (Hirohito Higashi)
Files: src/ex_getln.c

Patch 7.4.759

Problem: Building with Lua 5.3 doesn't work, symbols have changed.
Solution: Use the new names for the new version. (Felix Schnizlein)
Files: src/if_lua.c

Patch 7.4.760

Problem: Spelling mistakes are not displayed after ":syn spell".
Solution: Force a redraw after ":syn spell" command. (Christian Brabandt)
Files: src/syntax.c

Patch 7.4.761 (after 7.4.757)

Problem: The request-background termcode implementation is incomplete.
Solution: Add the missing pieces.
Files: src/option.c, src/term.c

Patch 7.4.762 (after 7.4.757)

Problem: Comment for may_req_bg_color() is wrong. (Christ van Willegen)
Solution: Rewrite the comment.
Files: src/term.c

Patch 7.4.763 (after 7.4.759)

Problem: Building with Lua 5.1 doesn't work.
Solution: Define lua_replace and lua_remove. (KF Leong)
Files: src/if_lua.c

Patch 7.4.764 (after 7.4.754)

Problem: test_increment fails on MS-Windows. (Ken Takata)
Solution: Clear Visual mappings. (Taro Muraoka)
Files: src/testdir/test_increment.in

Patch 7.4.765 (after 7.4.754)

Problem: **CTRL-A** and **CTRL-X** in Visual mode do not always work well.
Solution: Improvements for increment and decrement. (Christian Brabandt)
Files: src/normal.c, src/ops.c, src/testdir/test_increment.in,
src/testdir/test_increment.ok

Patch 7.4.766 (after 7.4.757)

Problem: Background color check does not work on Tera Term.
Solution: Also recognize ST as a termination character. (Hirohito Higashi)
Files: src/term.c

Patch 7.4.767

Problem: --remote-tab-silent can fail on MS-Windows.
Solution: Use single quotes to avoid problems with backslashes. (Idea by
Weiyong Mao)
Files: src/main.c

Patch 7.4.768

Problem: :diffoff only works properly once.
Solution: Also make :diffoff work when used a second time. (Olaf Dabrunz)
Files: src/diff.c

Patch 7.4.769 (after 7.4.768)

Problem: Behavior of :diffoff is not tested.
Solution: Add a bit of testing. (Olaf Dabrunz)
Files: src/testdir/test47.in, src/testdir/test47.ok

Patch 7.4.770 (after 7.4.766)

Problem: Background color response with transparency is not ignored.
Solution: Change the way escape sequences are recognized. (partly by
Hirohito Higashi)
Files: src/ascii.h, src/term.c

Patch 7.4.771

Problem: Search does not handle multibyte character at the start position
correctly.
Solution: Take byte size of character into account. (Yukihiro Nakadaira)
Files: src/search.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_search_mbyte.in,
src/testdir/test_search_mbyte.ok

Patch 7.4.772

Problem: Racket 6.2 is not supported on MS-Windows.
Solution: Check for the "racket" subdirectory. (Weiyong Mao)
Files: src/Make_mvc.mak, src/if_mzsch.c

Patch 7.4.773

Problem: **'langmap'** is used in command-line mode when checking for mappings.
Issue 376.
Solution: Do not use **'langmap'** in command-line mode. (Larry Velazquez)
Files: src/getchar.c, src/testdir/test_mapping.in,

src/testdir/test_mapping.ok

Patch 7.4.774

Problem: When using the CompleteDone autocommand event it's difficult to get to the completed items.

Solution: Add the v:completed_items variable. (Shougo Matsu)

Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, src/edit.c, src/eval.c, src/macros.h, src/proto/eval.pro, src/vim.h

Patch 7.4.775

Problem: It is not possible to avoid using the first item of completion.

Solution: Add the "noinsert" and "noselect" values to '**completeopt**'. (Shougo Matsu)

Files: runtime/doc/options.txt, src/edit.c, src/option.c

Patch 7.4.776

Problem: Equivalence class for 'd' does not work correctly.

Solution: Fix 0x1e0f and 0x1d0b. (Dominique Pelle)

Files: src/regexp.c, src/regexp_nfa.c

Patch 7.4.777

Problem: The README file doesn't look nice on github.

Solution: Add a markdown version of the README file.

Files: Filelist, README.md

Patch 7.4.778

Problem: Coverity warns for uninitialized variable.

Solution: Change condition of assignment.

Files: src/ops.c

Patch 7.4.779

Problem: Using **CTRL-A** in a line without a number moves the cursor. May cause a crash when at the start of the line. (Urtica Dioica)

Solution: Do not move the cursor if no number was changed.

Files: src/ops.c

Patch 7.4.780

Problem: Compiler complains about uninitialized variable and clobbered variables.

Solution: Add Initialization. Make variables static.

Files: src/ops.c, src/main.c

Patch 7.4.781

Problem: line2byte() returns one less when '**bin**' and '**noeol**' are set.

Solution: Only adjust the size for the last line. (Rob Wu)

Files: src/memline.c

Patch 7.4.782

Problem: Still a few problems with **CTRL-A** and **CTRL-X** in Visual mode.

Solution: Fix the reported problems. (Christian Brabandt)

Files: src/charset.c, src/eval.c, src/ex_cmds.c, src/ex_getln.c, src/misc2.c, src/normal.c, src/ops.c, src/option.c, src/proto/charset.pro, src/testdir/test_increment.in, src/testdir/test_increment.ok

Patch 7.4.783

Problem: copy_chars() and copy_spaces() are inefficient.
Solution: Use memset() instead. (Dominique Pelle)
Files: src/ex_getln.c, src/misc2.c, src/ops.c, src/proto/misc2.pro,
src/screen.c

Patch 7.4.784

Problem: Using both "noinselect" and "noselect" in '**completeopt**' does not work properly.
Solution: Change the ins_complete() calls. (Ozaki Kiichi)
Files: src/edit.c

Patch 7.4.785

Problem: On some systems automatically adding the missing EOL causes problems. Setting '**binary**' has too many side effects.
Solution: Add the '**fixeol**' option, default on. (Pavel Samarkin)
Files: src/buffer.c, src/fileio.c, src/memline.c, src/netbeans.c,
src/ops.c, src/option.c, src/option.h, src/os_unix.c,
src/os_win32.c, src/structs.h, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_fixeol.in,
src/testdir/test_fixeol.ok, runtime/doc/options.txt,
runtime/optwin.vim

Patch 7.4.786

Problem: It is not possible for a plugin to adjust to a changed setting.
Solution: Add the OptionSet autocommand event. (Christian Brabandt)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, src/eval.c,
src/fileio.c, src/option.c, src/proto/eval.pro,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_autocmd_option.in,
src/testdir/test_autocmd_option.ok, src/vim.h

Patch 7.4.787 (after 7.4.786)

Problem: snprintf() isn't available everywhere.
Solution: Use vim_snprintf(). (Ken Takata)
Files: src/option.c

Patch 7.4.788 (after 7.4.787)

Problem: Can't build without the crypt feature. (John Marriott)
Solution: Add #ifdef's.
Files: src/option.c

Patch 7.4.789 (after 7.4.788)

Problem: Using freed memory and crash. (Dominique Pelle)
Solution: Correct use of pointers. (Hirohito Higashi)
Files: src/option.c

Patch 7.4.790 (after 7.4.786)

Problem: Test fails when the autochdir feature is not available. Test

output contains the test script.
Solution: Check for the autochdir feature. (Kazunobu Kuriyama) Only write the relevant test output.
Files: src/testdir/test_autocmd_option.in,
src/testdir/test_autocmd_option.ok

Patch 7.4.791

Problem: The buffer list can be very long.
Solution: Add an argument to ":ls" to specify the type of buffer to list. (Marcin Szamotulski)
Files: runtime/doc/windows.txt, src/buffer.c, src/ex_cmds.h

Patch 7.4.792

Problem: Can only conceal text by defining syntax items.
Solution: Use matchadd() to define concealing. (Christian Brabandt)
Files: runtime/doc/eval.txt, src/eval.c, src/ex_docmd.c,
src/proto/window.pro, src/screen.c, src/structs.h,
src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_match_conceal.in,
src/testdir/test_match_conceal.ok, src/window.c

Patch 7.4.793

Problem: Can't specify when not to ring the bell.
Solution: Add the '**belloff**' option. (Christian Brabandt)
Files: runtime/doc/options.txt, src/edit.c, src/ex_getln.c,
src/hangulin.c, src/if_lua.c, src/if_mzsch.c, src/if_tcl.c,
src/message.c, src/misc1.c, src/normal.c, src/option.c,
src/option.h, src/proto/misc1.pro, src/search.c, src/spell.c

Patch 7.4.794

Problem: Visual Studio 2015 is not recognized.
Solution: Add the version numbers to the makefile. (Taro Muraoka)
Files: src/Make_mvc.mak

Patch 7.4.795

Problem: The '**fixeol**' option is not copied to a new window.
Solution: Copy the option value. (Yasuhiro Matsumoto)
Files: src/option.c

Patch 7.4.796

Problem: Warning from 64 bit compiler.
Solution: Add type cast. (Mike Williams)
Files: src/ops.c

Patch 7.4.797

Problem: Crash when using more lines for the command line than '**maxcombine**'.
Solution: Use the correct array index. Also, do not try redrawing when exiting. And use screen_Columns instead of Columns.
Files: src/screen.c

Patch 7.4.798 (after 7.4.753)

Problem: Repeating a change in Visual mode does not work as expected.
(Urtica Dioica)
Solution: Make redo in Visual mode work better. (Christian Brabandt)
Files: src/normal.c, src/testdir/test_listlbr.in,
src/testdir/test_listlbr.ok

Patch 7.4.799

Problem: Accessing memory before an allocated block.
Solution: Check for not going before the start of a pattern. (Dominique Pelle)
Files: src/fileio.c

Patch 7.4.800

Problem: Using freed memory when triggering CmdUndefined autocommands.
Solution: Set pointer to NULL. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.4.801 (after 7.4.769)

Problem: Test for ":diffoff" doesn't catch all potential problems.
Solution: Add a :diffthis and a :diffoff command. (Olaf Dabrunz)
Files: src/testdir/test47.in

Patch 7.4.802

Problem: Using "A" in Visual mode while '**linebreak**' is set is not tested.
Solution: Add a test for this, verifies the problem is fixed. (Ingo Karkat)
Files: src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.803

Problem: C indent does not support C11 raw strings. (Mark Lodato)
Solution: Do not change indent inside the raw string.
Files: src/search.c, src/misc1.c, src/edit.c, src/ops.c,
src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.804

Problem: Xxd doesn't have a license notice.
Solution: Add license as indicated by Juergen.
Files: src/xxd/xxd.c

Patch 7.4.805

Problem: The ruler shows "Bot" even when there are only filler lines missing. (Gary Johnson)
Solution: Use "All" when the first line and one filler line are visible.
Files: src/buffer.c

Patch 7.4.806

Problem: **CTRL-A** in Visual mode doesn't work properly with "alpha" in '**nrformats**'.
Solution: Make it work. (Christian Brabandt)
Files: src/ops.c, src/testdir/test_increment.in,
src/testdir/test_increment.ok

Patch 7.4.807 (after 7.4.798)

Problem: After **CTRL-V CTRL-A** mode isn't updated. (Hirohito Higashi)
Solution: Clear the command line or update the displayed command.

Files: src/normal.c

Patch 7.4.808

Problem: On MS-Windows 8 IME input doesn't work correctly.

Solution: Read console input before calling MsgWaitForMultipleObjects().
(vim-jp, Nobuhiro Takasaki)

Files: src/os_win32.c

Patch 7.4.809 (after 7.4.802)

Problem: Test is duplicated.

Solution: Roll back 7.4.802.

Files: src/testdir/test39.in, src/testdir/test39.ok

Patch 7.4.810

Problem: With a sequence of commands using buffers in diff mode E749 is given. (itchyny)

Solution: Skip unloaded buffer. (Hirohito Higashi)

Files: src/diff.c

Patch 7.4.811

Problem: Invalid memory access when using "exe 'sc'".

Solution: Avoid going over the end of the string. (Dominique Pelle)

Files: src/ex_docmd.c

Patch 7.4.812

Problem: Gcc sanitizer complains about using a NULL pointer to memmove().

Solution: Only call memmove when there is something to move. (Vittorio Zecca)

Files: src/memline.c

Patch 7.4.813

Problem: It is not possible to save and restore character search state.

Solution: Add getcharsearch() and setcharsearch(). (James McCoy)

Files: runtime/doc/eval.txt, src/eval.c, src/proto/search.pro,
src/search.c, src/testdir/test_charsearch.in,
src/testdir/test_charsearch.ok, src/testdir/Makefile,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms

Patch 7.4.814

Problem: Illegal memory access with "sy match a fold".

Solution: Check for empty string. (Dominique Pelle)

Files: src/syntax.c

Patch 7.4.815

Problem: Invalid memory access when doing ":call g:".

Solution: Check for an empty name. (Dominique Pelle)

Files: src/eval.c

Patch 7.4.816

Problem: Invalid memory access when doing ":fun X(".

Solution: Check for missing ')'. (Dominique Pelle)

Files: src/eval.c

Patch 7.4.817

Problem: Invalid memory access in file_pat_to_reg_pat().
Solution: Use vim_isspace() instead of checking for a space only. (Dominique Pelle)
Files: src/fileio.c

Patch 7.4.818

Problem: **'linebreak'** breaks c% if the last Visual selection was block. (Chris Morganiser, Issue 389)
Solution: Handle Visual block mode differently. (Christian Brabandt)
Files: src/normal.c, src/testdir/test_listlbr.in, src/testdir/test_listlbr.ok

Patch 7.4.819

Problem: Beeping when running the tests.
Solution: Fix 41 beeps. (Roland Eggner)
Files: src/testdir/test17.in, src/testdir/test29.in, src/testdir/test4.in, src/testdir/test61.in, src/testdir/test82.in, src/testdir/test83.in, src/testdir/test90.in, src/testdir/test95.in, src/testdir/test_autoformat_join.in

Patch 7.4.820

Problem: Invalid memory access in file_pat_to_reg_pat.
Solution: Avoid looking before the start of a string. (Dominique Pelle)
Files: src/fileio.c

Patch 7.4.821

Problem: Coverity reports a few problems.
Solution: Avoid the warnings. (Christian Brabandt)
Files: src/ex_docmd.c, src/option.c, src/screen.c

Patch 7.4.822

Problem: More problems reported by coverity.
Solution: Avoid the warnings. (Christian Brabandt)
Files: src/os_unix.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_getln.c, src/fold.c, src/gui.c, src/gui_w16.c, src/gui_w32.c, src/if_ctype.c, src/if_xcmdsrv.c, src/move.c, src/normal.c, src/regexp.c, src/syntax.c, src/ui.c, src/window.c

Patch 7.4.823

Problem: Cursor moves after **CTRL-A** on alphabetic character.
Solution: (Hirohito Higashi, test by Christian Brabandt)
Files: src/testdir/test_increment.in, src/testdir/test_increment.ok, src/ops.c

Patch 7.4.824 (after 7.4.813)

Problem: Can't compile without the multibyte feature. (John Marriott)
Solution: Add #ifdef.
Files: src/eval.c

Patch 7.4.825

Problem: Invalid memory access for ":syn keyword x a[".

Solution: Do not skip over the NUL. (Dominique Pelle)
Files: src/syntax.c

Patch 7.4.826

Problem: Compiler warnings and errors.
Solution: Make it build properly without the multibyte feature.
Files: src/eval.c, src/search.c

Patch 7.4.827

Problem: Not all test targets are in the Makefile.
Solution: Add the missing targets.
Files: src/Makefile

Patch 7.4.828

Problem: Crash when using "syn keyword x c". (Dominique Pelle)
Solution: Initialize the keyword table. (Raymond Ko, PR 397)
Files: src/syntax.c

Patch 7.4.829

Problem: Crash when clicking in beval balloon. (Travis Lebsock)
Solution: Use PostMessage() instead of DestroyWindow(). (Raymond Ko, PR 298)
Files: src/gui_w32.c

Patch 7.4.830

Problem: Resetting **'encoding'** when doing ":set all&" causes problems.
(Bjorn Linse) Display is not updated.
Solution: Do not reset **'encoding'**. Do a full redraw.
Files: src/option.c

Patch 7.4.831

Problem: When expanding **'=expr'** on the command line and encountering an error, the command is executed anyway.
Solution: Bail out when an error is detected.
Files: src/misc1.c

Patch 7.4.832

Problem: \$HOME in **'=\$HOME . '/.vimrc'** is expanded too early.
Solution: Skip over **'=expr'** when expanding environment names.
Files: src/misc1.c

Patch 7.4.833

Problem: More side effects of ":set all&" are missing. (Björn Linse)
Solution: Call didset_options() and add didset_options2() to collect more side effects to take care of. Still not everything...
Files: src/option.c

Patch 7.4.834

Problem: gettabvar() doesn't work after Vim start. (Szymon Wrozynski)
Solution: Handle first window in tab still being NULL. (Christian Brabandt)
Files: src/eval.c, src/testdir/test91.in, src/testdir/test91.ok

Patch 7.4.835

Problem: Comparing utf-8 sequences does not handle different byte sizes correctly.

Solution: Get the byte size of each character. (Dominique Pelle)
Files: src/misc2.c

Patch 7.4.836

Problem: Accessing uninitialized memory.
Solution: Add missing calls to init_tv(). (Dominique Pelle)
Files: src/eval.c

Patch 7.4.837

Problem: Compiler warning with MSVC compiler when using +sniff.
Solution: Use Sleep() instead of _sleep(). (Tux)
Files: src/if_sniff.c

Patch 7.4.838 (after 7.4.833)

Problem: Can't compile without the crypt feature. (John Marriott)
Solution: Add #ifdef.
Files: src/option.c

Patch 7.4.839

Problem: Compiler warning on 64-bit system.
Solution: Add cast to int. (Mike Williams)
Files: src/search.c

Patch 7.4.840 (after 7.4.829)

Problem: Tooltip window stays open.
Solution: Send a WM_CLOSE message. (Jurgen Kramer)
Files: src/gui_w32.c

Patch 7.4.841

Problem: Can't compile without the multibyte feature. (John Marriott)
Solution: Add more #ifdef's.
Files: src/option.c

Patch 7.4.842 (after 7.4.840)

Problem: Sending too many messages to close the balloon.
Solution: Only send a WM_CLOSE message. (Jurgen Kramer)
Files: src/gui_w32.c

Patch 7.4.843 (after 7.4.835)

Problem: Still possible to go beyond the end of a string.
Solution: Check for NUL also in second string. (Dominique Pelle)
Files: src/misc2.c

Patch 7.4.844

Problem: When '#' is in **'isident'** the is# comparator doesn't work.
Solution: Don't use vim_isIDc(). (Yasuhiro Matsumoto)
Files: src/eval.c, src/testdir/test_comparators.in,
src/testdir/test_comparators.ok, src/testdir/Makefile,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms

Patch 7.4.845

Problem: Compiler warning for possible loss of data.

Solution: Add a type cast. (Erich Ritz)
Files: src/misc1.c

Patch 7.4.846

Problem: Some GitHub users don't know how to use issues.
Solution: Add a file that explains the basics of contributing.
Files: Filelist, CONTRIBUTING.md

Patch 7.4.847

Problem: "vi)d" may leave a character behind.
Solution: Skip over multibyte character. (Christian Brabandt)
Files: src/search.c

Patch 7.4.848

Problem: **CTRL-A** on hex number in Visual block mode is incorrect.
Solution: Account for the "0x". (Hirohito Higashi)
Files: src/charset.c, src/testdir/test_increment.in,
src/testdir/test_increment.ok

Patch 7.4.849

Problem: Moving the cursor in Insert mode starts new undo sequence.
Solution: Add **CTRL-G U** to keep the undo sequence for the following cursor movement command. (Christian Brabandt)
Files: runtime/doc/insert.txt, src/edit.c, src/testdir/test_mapping.in,
src/testdir/test_mapping.ok

Patch 7.4.850 (after 7.4.846)

Problem: <Esc> does not show up.
Solution: Use > and <. (Kazunobu Kuriyama)
Files: CONTRIBUTING.md

Patch 7.4.851

Problem: Saving and restoring the console buffer does not work properly.
Solution: Instead of ReadConsoleOutputA/WriteConsoleOutputA use CreateConsoleScreenBuffer and SetConsoleActiveScreenBuffer. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.852

Problem: On MS-Windows console Vim uses ANSI APIs for keyboard input and console output, it cannot input/output Unicode characters.
Solution: Use Unicode APIs for console I/O. (Ken Takata, Yasuhiro Matsumoto)
Files: src/os_win32.c, src/ui.c, runtime/doc/options.txt

Patch 7.4.853

Problem: "zt" in diff mode does not always work properly. (Gary Johnson)
Solution: Don't count filler lines twice. (Christian Brabandt)
Files: src/move.c

Patch 7.4.854 (after 7.4.850)

Problem: Missing information about runtime files.
Solution: Add section about runtime files. (Christian Brabandt)
Files: CONTRIBUTING.md

Patch 7.4.855

Problem: GTK: font glitches for combining characters
Solution: Use pango_shape_full() instead of pango_shape(). (luchr, PR #393)
Files: src/gui_gtk_x11.c

Patch 7.4.856

Problem: "zt" still doesn't work well with filler lines. (Gary Johnson)
Solution: Check for filler lines above the cursor. (Christian Brabandt)
Files: src/move.c

Patch 7.4.857

Problem: Dragging the current tab with the mouse doesn't work properly.
Solution: Take the current tabpage index into account. (Hirohito Higashi)
Files: src/normal.c

Patch 7.4.858

Problem: It's a bit clumsy to execute a command on a list of matches.
Solution: Add the ":ldo", ":lfdo", ":cdo" and ":cfdo" commands. (Yegappan Lakshmanan)
Files: runtime/doc/cmdline.txt, runtime/doc/editing.txt,
runtime/doc/index.txt, runtime/doc/quickfix.txt,
runtime/doc/tabpage.txt, runtime/doc/windows.txt, src/ex_cmds.h,
src/ex_cmds2.c, src/ex_docmd.c, src/proto/quickfix.pro,
src/quickfix.c, src/testdir/Make_amiga.mak,
src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Make_os2.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/test_cdo.in,
src/testdir/test_cdo.ok

Patch 7.4.859

Problem: Vim doesn't recognize all htmldjango files.
Solution: Recognize a comment. (Daniel Hahler, PR #410)
Files: runtime/filetype.vim

Patch 7.4.860

Problem: Filetype detection is outdated.
Solution: Include all recent and not-so-recent changes.
Files: runtime/filetype.vim

Patch 7.4.861 (after 7.4.855)

Problem: pango_shape_full() is not always available.
Solution: Add a configure check.
Files: src/configure.in, src/auto/configure, src/config.h.in,
src/gui_gtk_x11.c

Patch 7.4.862 (after 7.4.861)

Problem: Still problems with pango_shape_full() not available.
Solution: Change AC_TRY_COMPILE to AC_TRY_LINK.
Files: src/configure.in, src/auto/configure

Patch 7.4.863 (after 7.4.856)

Problem: plines_nofill() used without the diff feature.
Solution: Define PLINES_NOFILL().
Files: src/macros.h, src/move.c

Patch 7.4.864 (after 7.4.858)

Problem: Tiny build fails.
Solution: Put `qf_items` inside `#ifdef`.
Files: `src/ex_docmd.c`

Patch 7.4.865

Problem: Compiler warning for uninitialized variable.
Solution: Initialize.
Files: `src/ex_cmds2.c`

Patch 7.4.866

Problem: Crash when changing the **'tags'** option from a remote command.
(Benjamin Fritz)
Solution: Instead of executing messages immediately, use a queue, like for netbeans. (James Kolb)
Files: `src/ex_docmd.c`, `src/getchar.c`, `src/gui_gtk_x11.c`, `src/gui_w48.c`,
`src/gui_x11.c`, `src/if_xcmdsrv.c`, `src/misc2.c`, `src/os_unix.c`,
`src/proto/if_xcmdsrv.pro`, `src/proto/misc2.pro`, `src/macros.h`

Patch 7.4.867 (after 7.4.866)

Problem: Can't build on MS-Windows. (Taro Muraoka)
Solution: Adjust `#ifdef`.
Files: `src/misc2.c`

Patch 7.4.868

Problem: **'smarttab'** is also effective when **'paste'** is enabled. (Alexander Monakov)
Solution: Disable **'smarttab'** when **'paste'** is set. (Christian Brabandt)
Do the same for **'expandtab'**.
Files: `src/option.c`, `src/structs.h`

Patch 7.4.869

Problem: MS-Windows: scrolling may cause text to disappear when using an Intel GPU.
Solution: Call `GetPixel()`. (Yohei Endo)
Files: `src/gui_w48.c`

Patch 7.4.870

Problem: May get into an invalid state when using `getchar()` in an expression mapping.
Solution: Anticipate `mod_mask` to change. (idea by Yukihiro Nakadaira)
Files: `src/getchar.c`

Patch 7.4.871

Problem: Vim leaks memory, when **'wildignore'** filters out all matches.
Solution: Free the files array when it becomes empty.
Files: `src/misc1.c`

Patch 7.4.872

Problem: Not using CI services available.
Solution: Add configuration files for travis and appveyor. (Ken Takata, vim-jp, PR #401)
Files: `.travis.yml`, `appveyor.yml`, `Filelist`

Patch 7.4.873 (after 7.4.866)

Problem: Compiler warning for unused variable. (Tony Mechelynck)
Solution: Remove the variable. Also fix int vs long_u mixup.
Files: src/if_xcmdsrv.c

Patch 7.4.874

Problem: MS-Windows: When Vim runs inside another application, the size isn't right.
Solution: When in child mode compute the size differently. (Agorgianitis Loukas)
Files: src/gui_w48.c

Patch 7.4.875

Problem: Not obvious how to contribute.
Solution: Add a remark about CONTRIBUTING.md to README.md
Files: README.md

Patch 7.4.876

Problem: Windows7: when using vim.exe with msys or msys2, conhost.exe (console window provider on Windows7) will freeze or crash.
Solution: Make original screen buffer active, before executing external program. And when the program is finished, revert to vim's one. (Taro Muraoka)
Files: src/os_win32.c

Patch 7.4.877 (after 7.4.843)

Problem: ":find" sometimes fails. (Excanoe)
Solution: Compare current characters instead of previous ones.
Files: src/misc2.c

Patch 7.4.878

Problem: Coverity error for clearing only one byte of struct.
Solution: Clear the whole struct. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.4.879

Problem: Can't see line numbers in nested function calls.
Solution: Add line number to the file name. (Alberto Fanjul)
Files: src/eval.c

Patch 7.4.880

Problem: No build and coverage status.
Solution: Add links to the README file. (Christian Brabandt)
Files: README.md

Patch 7.4.881 (after 7.4.879)

Problem: Test 49 fails.
Solution: Add line number to check of call stack.
Files: src/testdir/test49.vim

Patch 7.4.882

Problem: When leaving the command line window with **CTRL-C** while a completion menu is displayed the menu isn't removed.

Solution: Force a screen update. (Hirohito Higashi)
Files: src/edit.c

Patch 7.4.883 (after 7.4.818)

Problem: Block-mode replace works characterwise instead of blockwise after column 147. (Issue #422)

Solution: Set Visual mode. (Christian Brabandt)
Files: src/normal.c, src/testdir/test_listlbr.in,
src/testdir/test_listlbr.ok

Patch 7.4.884

Problem: Travis also builds on a tag push.
Solution: Filter out tag pushes. (Kenichi Ito)
Files: .travis.yml

Patch 7.4.885

Problem: When doing an upwards search without wildcards the search fails if the initial directory doesn't exist.

Solution: Fix the non-wildcard case. (Stefan Kempf)
Files: src/misc2.c

Patch 7.4.886 (after 7.4.876)

Problem: Windows7: Switching screen buffer causes flicker when using system().
Solution: Instead of actually switching screen buffer, duplicate the handle. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.887

Problem: Using uninitialized memory for regexp with back reference. (Dominique Pelle)
Solution: Initialize end_lnum.
Files: src/regexp_nfa.c

Patch 7.4.888

Problem: The OptionSet autocommands are not triggered from setwinvar().
Solution: Do not use switch_win() when not needed. (Hirohito Higashi)
Files: src/eval.c

Patch 7.4.889

Problem: Triggering OptionSet from setwinvar() isn't tested.
Solution: Add a test. (Christian Brabandt)
Files: src/testdir/test_autocmd_option.in,
src/testdir/test_autocmd_option.ok

Patch 7.4.890

Problem: Build failure when using dynamic python but not python3.
Solution: Adjust the #if to also include DYNAMIC_PYTHON3 and UNIX.
Files: src/if_python3.c

Patch 7.4.891

Problem: Indentation of array initializer is wrong.
Solution: Avoid that calling find_start_rawstring() changes the position returned by find_start_comment(), add a test. (Hirohito Higashi)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.892

Problem: On MS-Windows the iconv DLL may have a different name.
Solution: Also try libiconv2.dll and libiconv-2.dll. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 7.4.893

Problem: C indenting is wrong below a "case (foo):" because it is recognized as a C++ base class construct. Issue #38.
Solution: Check for the case keyword.
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.894

Problem: vimrun.exe is picky about the number of spaces before -s.
Solution: Skip all spaces. (Cam Sinclair)
Files: src/vimrun.c

Patch 7.4.895

Problem: Custom command line completion does not work for a command containing digits.
Solution: Skip over the digits. (suggested by Yasuhiro Matsumoto)
Files: src/ex_docmd.c

Patch 7.4.896

Problem: Editing a URL, which netrw should handle, doesn't work.
Solution: Avoid changing slashes to backslashes. (Yasuhiro Matsumoto)
Files: src/fileio.c, src/os_mswin.c

Patch 7.4.897

Problem: Freeze and crash when there is a sleep in a remote command. (Karl Yngve Lervåg)
Solution: Remove a message from the queue before dealing with it. (James Kolb)
Files: src/if_xcmdsrv.c

Patch 7.4.898

Problem: The '**fixendofline**' option is set on with ":edit".
Solution: Don't set the option when clearing a buffer. (Yasuhiro Matsumoto)
Files: src/buffer.c

Patch 7.4.899

Problem: README file is not optimal.
Solution: Move buttons, update some text. (closes #460)
Files: README.txt, README.md

Patch 7.4.900 (after 7.4.899)

Problem: README file can still be improved
Solution: Add a couple of links. (Christian Brabandt)
Files: README.md

Patch 7.4.901

Problem: When a BufLeave autocommand changes folding in a way it syncs undo, undo can be corrupted.

Solution: Prevent undo sync. (Jacob Niehus)
Files: src/popupmnu.c

Patch 7.4.902

Problem: Problems with using the MS-Windows console.
Solution: Revert patches 7.4.851, 7.4.876 and 7.4.886 until we find a better solution. (suggested by Ken Takata)
Files: src/os_win32.c

Patch 7.4.903

Problem: MS-Windows: When **'encoding'** differs from the current code page, expanding wildcards may cause illegal memory access.
Solution: Allocate a longer buffer. (Ken Takata)
Files: src/misc1.c

Patch 7.4.904

Problem: Vim does not provide .desktop files.
Solution: Include and install .desktop files. (James McCoy, closes #455)
Files: Filelist, runtime/vim.desktop, runtime/gvim.desktop, src/Makefile

Patch 7.4.905

Problem: Python interface can produce error "vim.message' object has no attribute **'isatty'**".
Solution: Add dummy isatty(), readable(), etc. (closes #464)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok, src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.906

Problem: On MS-Windows the viminfo file is (always) given the hidden attribute. (raulnac)
Solution: Check the hidden attribute in a different way. (Ken Takata)
Files: src/ex_cmds.c, src/os_win32.c, src/os_win32.pro

Patch 7.4.907

Problem: Libraries for dynamically loading interfaces can only be defined at compile time.
Solution: Add options to specify the dll names. (Kazuki Sakamoto, closes #452)
Files: runtime/doc/if_lua.txt, runtime/doc/if_perl.txt, runtime/doc/if_pyth.txt, runtime/doc/if_ruby.txt, runtime/doc/options.txt, src/if_lua.c, src/if_perl.xs, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/option.c, src/option.h

Patch 7.4.908 (after 7.4.907)

Problem: Build error with MingW compiler. (Cesar Romani)
Solution: Change #if into #ifdef.
Files: src/if_perl.xs

Patch 7.4.909 (after 7.4.905)

Problem: "make install" fails.
Solution: Only try installing desktop files if the destination directory exists.
Files: src/Makefile

Patch 7.4.910 (after 7.4.905)

Problem: Compiler complains about type punned pointer.

Solution: Use another way to increment the ref count.

Files: src/if_py_both.h

Patch 7.4.911

Problem: t_Ce and t-Cs are documented but not supported. (Hirohito Higashi)

Solution: Define the options.

Files: src/option.c

Patch 7.4.912

Problem: Wrong indenting for C++ constructor.

Solution: Recognize ::. (Anhong)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 7.4.913

Problem: No utf-8 support for the hangul input feature.

Solution: Add utf-8 support. (Namsh)

Files: src/gui.c, src/hangulin.c, src/proto/hangulin.pro, src/screen.c, src/ui.c, runtime/doc/hangulin.txt, src/feature.h

Patch 7.4.914

Problem: New compiler warning: logical-not-parentheses

Solution: Silence the warning.

Files: src/term.c

Patch 7.4.915

Problem: When removing from **'path'** and then adding, a comma may go missing. (Malcolm Rowe)

Solution: Fix the check for P_ONECOMMA. (closes #471)

Files: src/option.c, src/testdir/test_options.in, src/testdir/test_options.ok

Patch 7.4.916

Problem: When running out of memory while copying a dict memory may be freed twice. (ZyX)

Solution: Do not call the garbage collector when running out of memory.

Files: src/misc2.c

Patch 7.4.917

Problem: Compiler warning for comparing signed and unsigned.

Solution: Add a type cast.

Files: src/hangulin.c

Patch 7.4.918

Problem: A digit in an option name has problems.

Solution: Rename 'python3dll' to **'pythonthreedll'**.

Files: src/option.c, src/option.h, runtime/doc/options.txt

Patch 7.4.919

Problem: The dll options are not in the options window.

Solution: Add the dll options. And other fixes.

Files: runtime/optwin.vim

Patch 7.4.920

Problem: The rubydll option is not in the options window.
Solution: Add the rubydll option.
Files: runtime/optwin.vim

Patch 7.4.921 (after 7.4.906)

Problem: Missing proto file update. (Randall W. Morris)
Solution: Add the missing line for mch_ishidden.
Files: src/proto/os_win32.pro

Patch 7.4.922

Problem: Leaking memory with ":helpt {dir-not-exists}".
Solution: Free dirname. (Dominique Pelle)
Files: src/ex_cmds.c

Patch 7.4.923

Problem: Prototypes not always generated.
Solution: Change #if to OR with PROTO.
Files: src/window.c

Patch 7.4.924

Problem: DEVELOPER_DIR gets reset by configure.
Solution: Do not reset DEVELOPER_DIR when there is no --with-developer-dir argument. (Kazuki Sakamoto, closes #482)
Files: src/configure.in, src/auto/configure

Patch 7.4.925

Problem: User may yank or put using the register being recorded in.
Solution: Add the recording register in the message. (Christian Brabandt, closes #470)
Files: runtime/doc/options.txt, runtime/doc/repeat.txt, src/ops.c, src/option.h, src/screen.c

Patch 7.4.926

Problem: Completing the longest match doesn't work properly with multibyte characters.
Solution: When using multibyte characters use another way to find the longest match. (Hirohito Higashi)
Files: src/ex_getln.c, src/testdir/test_utf8.in, src/testdir/test_utf8.ok

Patch 7.4.927

Problem: Ruby crashes when there is a runtime error.
Solution: Use ruby_options() instead of ruby_process_options(). (Damien)
Files: src/if_ruby.c

Patch 7.4.928

Problem: A clientserver message interrupts handling keys of a mapping.
Solution: Have mch_inchar() send control back to WaitForChar when it is interrupted by server message. (James Kolb)
Files: src/os_unix.c

Patch 7.4.929

Problem: "gv" after paste selects one character less if 'selection' is

"exclusive".
Solution: Increment the end position. (Christian Brabandt)
Files: src/normal.c, src/testdir/test94.in, src/testdir/test94.ok

Patch 7.4.930

Problem: MS-Windows: Most users appear not to like the window border.
Solution: Remove WS_EX_CLIENTEDGE. (Ian Halliday)
Files: src/gui_w32.c

Patch 7.4.931 (after 7.4.929)

Problem: Test 94 fails on some systems.
Solution: Set '**encoding**' to utf-8.
Files: src/testdir/test94.in

Patch 7.4.932 (after 7.4.926)

Problem: test_utf8 has confusing dummy command.
Solution: Use a real command instead of a colon.
Files: src/testdir/test_utf8.in

Patch 7.4.933 (after 7.4.926)

Problem: Crash when using longest completion match.
Solution: Fix array index.
Files: src/ex_getln.c

Patch 7.4.934

Problem: Appveyor also builds on a tag push.
Solution: Add a skip_tags line. (Kenichi Ito, closes #489)
Files: appveyor.yml

Patch 7.4.935 (after 7.4.932)

Problem: test_utf8 fails on MS-Windows when executed with gvim.
Solution: Use the insert flag on feedkeys() to put the string before the ":" that was already read when checking for available chars.
Files: src/testdir/test_utf8.in

Patch 7.4.936

Problem: Crash when dragging with the mouse.
Solution: Add safety check for NULL pointer. Check mouse position for valid value. (Hirohito Higashi)
Files: src/window.c, src/term.c

Patch 7.4.937

Problem: Segfault reading uninitialized memory.
Solution: Do not read match \z0, it does not exist. (Marius Gedminas, closes #497)
Files: src/regexp_nfa.c

Patch 7.4.938

Problem: X11 and GTK have more mouse buttons than Vim supports.
Solution: Recognize more mouse buttons. (Benoit Pierre, closes #498)
Files: src/gui_gtk_x11.c, src/gui_x11.c

Patch 7.4.939

Problem: Memory leak when encountering a syntax error.

Solution: Free the memory. (Dominique Pelle)
Files: src/ex_docmd.c

Patch 7.4.940

Problem: vt52 terminal codes are not correct.
Solution: Move entries outside of #if. (Random) Adjustments based on documented codes.
Files: src/term.c

Patch 7.4.941

Problem: There is no way to ignore case only for tag searches.
Solution: Add the '**tagcase**' option. (Gary Johnson)
Files: runtime/doc/options.txt, runtime/doc/quickref.txt,
runtime/doc/tagsrch.txt, runtime/doc/usr_29.txt,
runtime/optwin.vim, src/Makefile, src/buffer.c, src/option.c,
src/option.h, src/structs.h, src/tag.c,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/test_tagcase.in, src/testdir/test_tagcase.ok

Patch 7.4.942 (after 7.4.941)

Problem: test_tagcase breaks for small builds.
Solution: Bail out of the test early. (Hirohito Higashi)
Files: src/testdir/test_tagcase.in

Patch 7.4.943

Problem: Tests are not run.
Solution: Add test_writefile to makefiles. (Ken Takata)
Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.944

Problem: Writing tests for Vim script is hard.
Solution: Add assertEquals(), assertFalse() and assertTrue() functions. Add the v:errors variable. Add the runtest script. Add a first new style test script.
Files: src/eval.c, src/vim.h, src/misc2.c, src/testdir/Makefile,
src/testdir/runtest.vim, src/testdir/test_assert.vim,
runtime/doc/eval.txt

Patch 7.4.945 (after 7.4.944)

Problem: New style testing is incomplete.
Solution: Add the runtest script to the list of distributed files.
Add the new functions to the function overview.
Rename the functions to match Vim function style.
Move undolevels testing into a new style test script.
Files: Filelist, runtime/doc/usr_41.txt, runtime/doc/eval.txt,
src/testdir/test_assert.vim, src/testdir/Makefile,
src/testdir/test_undolevels.vim, src/testdir/test100.in,
src/testdir/test100.ok

Patch 7.4.946 (after 7.4.945)

Problem: Missing changes in source file.
Solution: Include changes to the eval.c file.
Files: src/eval.c

Patch 7.4.947

Problem: Test_listchars fails with MingW. (Michael Soyka)
Solution: Add the test to the ones that need the fileformat fixed.
(Christian Brabandt)
Files: src/testdir/Make_ming.mak

Patch 7.4.948

Problem: Can't build when the insert_expand feature is disabled.
Solution: Add #ifdefs. (Dan Pasanen, closes #499)
Files: src/eval.c, src/fileio.c

Patch 7.4.949

Problem: When using '**colorcolumn**' and there is a sign with a fullwidth character the highlighting is wrong. (Andrew Stewart)
Solution: Only increment vcol when in the right state. (Christian Brabandt)
Files: src/screen.c, src/testdir/test_listlbr_utf8.in,
src/testdir/test_listlbr_utf8.ok

Patch 7.4.950

Problem: v:errors is not initialized.
Solution: Initialize it to an empty list. (Thinca)
Files: src/eval.c

Patch 7.4.951

Problem: Sorting number strings does not work as expected. (Luc Hermitte)
Solution: Add the "N" argument to sort()
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test_alot.vim,
src/testdir/test_sort.vim, src/testdir/Makefile

Patch 7.4.952

Problem: '**lispwords**' is tested in the old way.
Solution: Make a new style test for '**lispwords**'.
Files: src/testdir/test_alot.vim, src/testdir/test_lispwords.vim,
src/testdir/test100.in, src/testdir/test100.ok,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.953

Problem: When a test script navigates to another buffer the .res file is created with the wrong name.
Solution: Use the "testname" for the .res file. (Damien)
Files: src/testdir/runtest.vim

Patch 7.4.954

Problem: When using Lua there may be a crash. (issue #468)
Solution: Avoid using an uninitialized tv. (Yukihiro Nakadaira)
Files: src/if_lua.c

Patch 7.4.955

Problem: Vim doesn't recognize .pl6 and .pod6 files.
Solution: Recognize them as perl6 and pod6. (Mike Eve, closes #511)
Files: runtime/filetype.vim

Patch 7.4.956
Problem: A few more file name extensions not recognized.
Solution: Add .asciidoc, .bzl, .gradle, etc.
Files: runtime/filetype.vim

Patch 7.4.957
Problem: Test_tagcase fails when using another language than English.
Solution: Set the messages language to C. (Kenichi Ito)
Files: src/testdir/test_tagcase.in

Patch 7.4.958
Problem: Vim checks if the directory "\$TMPDIR" exists.
Solution: Do not check if the name starts with "\$".
Files: src/fileio.c

Patch 7.4.959
Problem: When setting 'term' the clipboard ownership is lost.
Solution: Do not call clip_init(). (James McCoy)
Files: src/term.c

Patch 7.4.960
Problem: Detecting every version of nmake is clumsy.
Solution: Use a tiny C program to get the version of _MSC_VER. (Ken Takata)
Files: src/Make_mvc.mak

Patch 7.4.961
Problem: Test107 fails in some circumstances.
Solution: When using "zt", "zb" and "z=" recompute the fraction.
Files: src/normal.c, src/window.c, src/proto/window.pro

Patch 7.4.962
Problem: Cannot run the tests with gvim. Cannot run individual new tests.
Solution: Add the -f flag. Add new test targets in Makefile.
Files: src/Makefile, src/testdir/Makefile

Patch 7.4.963
Problem: test_listlbr_utf8 sometimes fails.
Solution: Don't use a literal multibyte character but <C-V>uXXXX. Do not dump the screen highlighting. (Christian Brabandt, closes #518)
Files: src/testdir/test_listlbr_utf8.in, src/testdir/test_listlbr_utf8.ok

Patch 7.4.964
Problem: Test 87 doesn't work in a shadow directory.
Solution: Handle the extra subdirectory. (James McCoy, closes #515)
Files: src/testdir/test87.in

Patch 7.4.965
Problem: On FreeBSD /dev/fd/ files are special.
Solution: Use is_dev_fd_file() also for FreeBSD. (Derek Schrock, closes #521)
Files: src/fileio.c

Patch 7.4.966

Problem: Configure doesn't work with a space in a path.
Solution: Put paths in quotes. (James McCoy, closes #525)
Files: src/configure.in, src/auto/configure

Patch 7.4.967

Problem: Cross compilation on MS-windows doesn't work well.
Solution: Tidy up cross compilation across architectures with Visual Studio.
(Mike Williams)
Files: src/Make_mvc.mak

Patch 7.4.968

Problem: test86 and test87 are flaky in Appveyor.
Solution: Reduce the count from 8 to 7. (suggested by ZyX)
Files: src/testdir/test86.in, src/testdir/test87.in

Patch 7.4.969

Problem: Compiler warnings on Windows x64 build.
Solution: Add type casts. (Mike Williams)
Files: src/option.c

Patch 7.4.970

Problem: Rare crash in getvcol(). (Timo Mihaljov)
Solution: Check for the buffer being NULL in init_preedit_start_col.
(Hirohito Higashi, Christian Brabandt)
Files: src/mbyte.c

Patch 7.4.971

Problem: The asin() function can't be used.
Solution: Sort the function table properly. (Watiko)
Files: src/eval.c

Patch 7.4.972

Problem: Memory leak when there is an error in setting an option.
Solution: Free the saved value (Christian Brabandt)
Files: src/option.c

Patch 7.4.973

Problem: When pasting on the command line line breaks result in literal
<CR> characters. This makes pasting a long file name difficult.
Solution: Skip the characters.
Files: src/ex_getln.c, src/ops.c

Patch 7.4.974

Problem: When using :diffsplit the cursor jumps to the first line.
Solution: Put the cursor on the line related to where the cursor was before
the split.
Files: src/diff.c

Patch 7.4.975

Problem: Using ":sort" on a very big file sometimes causes text to be
corrupted. (John Beckett)
Solution: Copy the line into a buffer before calling ml_append().

Files: src/ex_cmds.c

Patch 7.4.976

Problem: When compiling Vim for MSYS2 (linked with msys-2.0.dll), the Win32 clipboard is not enabled.

Solution: Recognize MSYS like CYGWIN. (Ken Takata)

Files: src/configure.in, src/auto/configure

Patch 7.4.977

Problem: **'linebreak'** does not work properly when using "space" in **'listchars'**.

Solution: (Hirohito Higashi, Christian Brabandt)

Files: src/screen.c, src/testdir/test_listlbr.in, src/testdir/test_listlbr.ok

Patch 7.4.978

Problem: test_cdo fails when using another language than English.

Solution: Set the language to C. (Dominique Pelle, Kenichi Ito)

Files: src/testdir/test_cdo.in

Patch 7.4.979

Problem: When changing the crypt key the blocks read from disk are not decrypted.

Solution: Also call ml_decrypt_data() when mf_old_key is set. (Ken Takata)

Files: src/memfile.c

Patch 7.4.980

Problem: Tests for :cdo, :ldo, etc. are outdated.

Solution: Add new style tests for these commands. (Yegappan Lakshmanan)

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/test_cdo.in, src/testdir/test_cdo.ok, src/testdir/test_cdo.vim

Patch 7.4.981

Problem: An error in a test script goes unnoticed.

Solution: Source the test script inside try/catch. (Hirohito Higashi)

Files: src/testdir/runtest.vim

Patch 7.4.982

Problem: Keeping the list of tests updated is a hassle.

Solution: Move the list to a separate file, so that it only needs to be updated in one place.

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/Make_all.mak

Patch 7.4.983

Problem: Executing one test after "make testclean" doesn't work.

Solution: Add a dependency on test1.out.

Files: src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,

src/testdir/Make_vms.mms, src/testdir/Makefile,
src/testdir/Make_all.mak

Patch 7.4.984

Problem: searchpos() always starts searching in the first column, which is not what some people expect. (Brett Stahlman)

Solution: Add the 'z' flag: start at the specified column.

Files: src/vim.h, src/eval.c, src/search.c,
src/testdir/test_searchpos.vim, src/testdir/test_alot.vim,
runtime/doc/eval.txt

Patch 7.4.985

Problem: Can't build with Ruby 2.3.0.

Solution: Use the new TypedData_XXX macro family instead of Data_XXX. Use TypedData. (Ken Takata)

Files: src/if_ruby.c

Patch 7.4.986

Problem: Test49 doesn't work on MS-Windows. test70 is listed twice.

Solution: Move test49 to the group not used on Amiga and MS-Windows.
Remove test70 from SCRIPTS_WIN32.

Files: src/testdir/Make_all.mak, src/testdir/Make_dos.mak

Patch 7.4.987 (after 7.4.985)

Problem: Can't build with Ruby 1.9.2.

Solution: Require Rub 2.0 for defining USE_TYPEDDATA.

Files: src/if_ruby.c

Patch 7.4.988 (after 7.4.982)

Problem: Default test target is test49.out.

Solution: Add a build rule before including Make_all.mak.

Files: src/testdir/Make_dos.mak, src/testdir/Make_amiga.mak,
src/testdir/Make_ming.mak, src/testdir/Make_os2.mak,
src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.989

Problem: Leaking memory when hash_add() fails. Coverity error 99126.

Solution: When hash_add() fails free the memory.

Files: src/eval.c

Patch 7.4.990

Problem: Test 86 fails on AppVeyor.

Solution: Do some registry magic. (Ken Takata)

Files: appveyor.yml

Patch 7.4.991

Problem: When running new style tests the output is not visible.

Solution: Add the testdir/messages file and show it. Update the list of test names.

Files: src/Makefile, src/testdir/Makefile, src/testdir/runtest.vim

Patch 7.4.992

Problem: Makefiles for MS-Windows in src/po are outdated.

Solution: Make them work. (Ken Takata, Taro Muraoka)

Files: src/po/Make_cyg.mak, src/po/Make_ming.mak, src/po/Make_mvc.mak,
src/po/README_mingw.txt, src/po/README_mvc.txt

Patch 7.4.993

Problem: Test 87 is flaky on AppVeyor.
Solution: Reduce the minimum background thread count.
Files: src/testdir/test86.in, src/testdir/test87.in

Patch 7.4.994

Problem: New style tests are not run on MS-Windows.
Solution: Add the new style tests.
Files: src/testdir/Make_dos.mak

Patch 7.4.995

Problem: gdk_pixbuf_new_from_inline() is deprecated.
Solution: Generate auto/gui_gtk_gresources.c. (Kazunobu Kuriyama,
closes #507)
Files: src/Makefile, src/auto/configure, src/config.h.in,
src/config.mk.in, src/configure.in, src/gui_gtk.c,
src/gui_gtk_gresources.xml, src/gui_gtk_x11.c,
src/proto/gui_gtk_gresources.pro,
pixmaps/stock_vim_build_tags.png, pixmaps/stock_vim_find_help.png,
pixmaps/stock_vim_save_all.png,
pixmaps/stock_vim_session_load.png,
pixmaps/stock_vim_session_new.png,
pixmaps/stock_vim_session_save.png, pixmaps/stock_vim_shell.png,
pixmaps/stock_vim_window_maximize.png,
pixmaps/stock_vim_window_maximize_width.png,
pixmaps/stock_vim_window_minimize.png,
pixmaps/stock_vim_window_minimize_width.png,
pixmaps/stock_vim_window_split.png,
pixmaps/stock_vim_window_split_vertical.png

Patch 7.4.996

Problem: New GDK files and testdir/Make_all.mak missing from distribution.
PC build instructions are outdated.
Solution: Add the file to the list. Update PC build instructions.
Files: Filelist, Makefile

Patch 7.4.997

Problem: "make shadow" was sometimes broken.
Solution: Add a test for it. (James McCoy, closes #520)
Files: .travis.yml

Patch 7.4.998

Problem: Running tests in shadow directory fails. Test 49 fails.
Solution: Link more files for the shadow directory. Make test 49 ends up in
the right buffer.
Files: src/Makefile, src/testdir/test49.in

Patch 7.4.999

Problem: "make shadow" creates a broken link. (Tony Mechelynck)
Solution: Remove vimrc.unix from the list.
Files: src/Makefile

Patch 7.4.1000

Problem: Test 49 is slow and doesn't work on MS-Windows.
Solution: Start moving parts of test 49 to test_viml.
Files: src/Makefile, src/testdir/runtest.vim, src/testdir/test_viml.vim,
src/testdir/test49.vim, src/testdir/test49.ok

Patch 7.4.1001 (after 7.4.1000)

Problem: test_viml isn't run.
Solution: Include change in makefile.
Files: src/testdir/Make_all.mak

Patch 7.4.1002

Problem: Cannot run an individual test on MS-Windows.
Solution: Move the rule to run test1 downwards. (Ken Takata)
Files: src/testdir/Make_dos.mak

Patch 7.4.1003

Problem: Travis could check a few more things.
Solution: Run autoconf on one of the builds. (James McCoy, closes #510)
Also build with normal features.
Files: .travis.yml

Patch 7.4.1004

Problem: Using Makefile when auto/config.mk does not exist results in warnings.
Solution: Use default values for essential variables.
Files: src/Makefile

Patch 7.4.1005

Problem: Vim users are not always happy.
Solution: Make them happy.
Files: src/ex_cmds.h, src/ex_cmds.c, src/proto/ex_cmds.pro

Patch 7.4.1006

Problem: The fix in patch 7.3.192 is not tested.
Solution: Add a test, one for each regexp engine. (Elias Diem)
Files: src/testdir/test44.in, src/testdir/test44.ok,
src/testdir/test99.in, src/testdir/test99.ok

Patch 7.4.1007

Problem: When a symbolic link points to a file in the root directory, the swapfile is not correct.
Solution: Do not try getting the full name of a file in the root directory. (Milly, closes #501)
Files: src/os_unix.c

Patch 7.4.1008

Problem: The OS/2 code pollutes the source while nobody uses it these days.
Solution: Drop the support for OS/2.
Files: src/feature.h, src/globals.h, src/macros.h, src/option.h,
src/os_unix.c, src/os_unix.h, src/proto/os_unix.pro, src/vim.h,
src/digraph.c, src/eval.c, src/ex_cmds.c, src/ex_docmd.c,
src/ex_getln.c, src/fileio.c, src/getchar.c, src/memline.c,

src/misc1.c, src/misc2.c, src/netbeans.c, src/option.c,
src/term.c, src/ui.c, src/window.c, src/os_os2_cfg.h,
src/Make_os2.mak, src/testdir/Make_os2.mak, src/testdir/os2.vim,
src/INSTALL, runtime/doc/os_os2.txt

Patch 7.4.1009

Problem: There are still #ifdefs for ARCHIE.
Solution: Remove references to ARCHIE, the code was removed in Vim 5.
Files: src/ex_cmds.c, src/ex_docmd.c, src/fileio.c, src/main.c,
src/memline.c, src/option.c, src/term.c

Patch 7.4.1010

Problem: Some developers are unhappy while running tests.
Solution: Add a test and some color.
Files: src/ex_cmds.c, src/testdir/test_assert.vim

Patch 7.4.1011

Problem: Can't build with Strawberry Perl.
Solution: Include stdbool.h. (Ken Takata, closes #328)
Files: Filelist, src/Make_mvc.mak, src/if_perl_msvc/stdbool.h

Patch 7.4.1012

Problem: Vim overwrites the value of \$PYTHONHOME.
Solution: Do not set \$PYTHONHOME if it is already set. (Kazuki Sakamoto,
closes #500)
Files: src/if_python.c, src/if_python3.c

Patch 7.4.1013

Problem: The local value of **'errorformat'** is not used for ":lexpr" and
":cexpr".
Solution: Use the local value if it exists. (Christian Brabandt) Adjust the
help for this.
Files: runtime/doc/quickfix.txt, src/quickfix.c

Patch 7.4.1014

Problem: `fnamemodify('.', ':.') returns an empty string in Cygwin.
Solution: Use CCP_RELATIVE in the call to cygwin_conv_path. (Jacob Niehus,
closes #505)
Files: src/os_unix.c

Patch 7.4.1015

Problem: The column is not restored properly when the matchparen plugin is
used in Insert mode and the cursor is after the end of the line.
Solution: Set the curswant flag. (Christian Brabandt). Also fix
highlighting the match of the character before the cursor.
Files: src/eval.c, runtime/plugin/matchparen.vim

Patch 7.4.1016

Problem: Still a few OS/2 pieces remain.
Solution: Delete more.
Files: Filelist, README_os2.txt, testdir/todos.vim, src/xxd/Make_os2.mak

Patch 7.4.1017

Problem: When there is a backslash in an option ":set -=" doesn't work.

Solution: Handle a backslash better. (Jacob Niehus) Add a new test, merge in old test.
Files: src/testdir/test_cdo.vim, src/testdir/test_set.vim, src/testdir/test_alot.vim, src/option.c, src/testdir/test_set.in, src/testdir/test_set.ok, src/Makefile

Patch 7.4.1018 (after 7.4.1017)
Problem: Failure running tests.
Solution: Add missing change to list of old style tests.
Files: src/testdir/Make_all.mak

Patch 7.4.1019
Problem: Directory listing of "src" is too long.
Solution: Rename the resources file to make it shorter.
Files: src/gui_gtk_gresources.xml, src/gui_gtk_res.xml, src/Makefile, Filelist

Patch 7.4.1020
Problem: On MS-Windows there is no target to run tests with gvim.
Solution: Add the testgvim target.
Files: src/Make_mvc.mak

Patch 7.4.1021
Problem: Some makefiles are outdated.
Solution: Add a [note](#) to warn developers.
Files: src/Make_manx.mak, src/Make_bc3.mak, src/Make_bc5.mak, src/Make_djg.mak, src/Make_w16.mak

Patch 7.4.1022
Problem: The README file contains some outdated information.
Solution: Update the information about supported systems.
Files: README.txt, README.md

Patch 7.4.1023
Problem: The distribution files for MS-Windows use CR-LF, which is inconsistent with what one gets from github.
Solution: Use LF in the distribution files.
Files: Makefile

Patch 7.4.1024
Problem: Interfaces for MS-Windows are outdated.
Solution: Use Python 2.7.10, Python 3.4.4, Perl 5.22, TCL 8.6.
Files: src/bigvim.bat

Patch 7.4.1025
Problem: Version in installer needs to be updated manually.
Solution: Generate a file with the version number. (Guopeng Wen)
Files: Makefile, nsis/gvim.nsi, nsis/gvim_version.nsh

Patch 7.4.1026
Problem: When using MingW the tests do not clean up all files. E.g. test 17 leaves Xdir1 behind. (Michael Soyka)
Solution: Also delete directories, like Make_dos.mak. Delete files after directories to reduce warnings.

Files: src/testdir/Make_ming.mak, src/testdir/Make_dos.mak

Patch 7.4.1027

Problem: No support for binary numbers.

Solution: Add "bin" to **'nrformats'**. (Jason Schulz)

Files: runtime/doc/change.txt, runtime/doc/eval.txt,
runtime/doc/version7.txt, src/charset.c, src/eval.c,
src/ex_cmds.c, src/ex_getln.c, src/misc2.c, src/ops.c,
src/option.c, src/proto/charset.pro, src/spell.c,
src/testdir/test57.in, src/testdir/test57.ok,
src/testdir/test58.in, src/testdir/test58.ok,
src/testdir/test_increment.in, src/testdir/test_increment.ok,
src/vim.h

Patch 7.4.1028

Problem: Nsis version file missing from the distribution.

Solution: Add the file to the list.

Files: Filelist

Patch 7.4.1029 (after 7.4.1027)

Problem: test_increment fails on systems with 32 bit long.

Solution: Only test with 32 bits.

Files: src/testdir/test_increment.in, src/testdir/test_increment.ok

Patch 7.4.1030

Problem: test49 is still slow.

Solution: Move more tests from old to new style.

Files: src/testdir/test_viml.vim, src/testdir/test49.vim,
src/testdir/test49.ok, src/testdir/runtest.vim

Patch 7.4.1031

Problem: Can't build with Python interface using MingW.

Solution: Update the Makefile. (Yasuhiro Matsumoto)

Files: src/INSTALLpc.txt, src/Make_cyg_ming.mak

Patch 7.4.1032

Problem: message from assert_false() does not look nice.

Solution: Handle missing sourcing_name. Use right number of spaces. (Watiko)
Don't use line number if it's zero.

Files: src/eval.c

Patch 7.4.1033

Problem: Memory use on MS-Windows is very conservative.

Solution: Use the global memory status to estimate amount of memory.
(Mike Williams)

Files: src/os_win32.c, src/os_win32.h, src/proto/os_win32.pro

Patch 7.4.1034

Problem: There is no test for the **'backspace'** option behavior.

Solution: Add a test. (Hirohito Higashi)

Files: src/testdir/test_alot.vim, src/testdir/test_backspace_opt.vim

Patch 7.4.1035

Problem: An Ex range gets adjusted for folded lines even when the range is

not using line numbers.
Solution: Only adjust line numbers for folding. (Christian Brabandt)
Files: runtime/doc/fold.txt, src/ex_docmd.c

Patch 7.4.1036

Problem: Only terminals with up to 256 colors work properly.
Solution: Use the 256 color behavior for all terminals with 256 or more colors. (Robert de Bath, closes #504)
Files: src/syntax.c

Patch 7.4.1037

Problem: Using "q!" when there is a modified hidden buffer does not unload the current buffer, resulting in the need to abandon it again.
Solution: When using "q!" unload the current buffer when needed. (Yasuhiro Matsumoto, Hirohito Higashi)
Files: src/testdir/test31.in, src/testdir/test31.ok, runtime/doc/editing.txt, src/ex_cmds2.c, src/ex_docmd.c, src/gui.c, src/gui_gtk_x11.c, src/os_unix.c, src/proto/ex_cmds2.pro

Patch 7.4.1038

Problem: Still get a warning for a deprecated function with gdk-pixbuf 2.31.
Solution: Change minimum minor version from 32 to 31.
Files: src/configure.in, src/auto/configure

Patch 7.4.1039 (after 7.4.1037)

Problem: Test 31 fails with small build.
Solution: Bail out for small build. (Hirohito Higashi)
Files: src/testdir/test31.in

Patch 7.4.1040

Problem: The tee command is not available on MS-Windows.
Solution: Adjust tee.c for MSVC and add a makefile. (Yasuhiro Matsumoto)
Files: src/tee/tee.c, src/tee/Make_mvc.mak, src/Make_mvc.mak

Patch 7.4.1041

Problem: Various small things.
Solution: Add file to list of distributed files. Adjust README. Fix typo.
Files: Filelist, src/testdir/README.txt, src/testdir/test_charsearch.in, src/INSTALLmac.txt

Patch 7.4.1042

Problem: g-CTRL-G shows the word count, but there is no way to get the word count in a script.
Solution: Add the wordcount() function. (Christian Brabandt)
Files: runtime/doc/editing.txt, runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/eval.c, src/normal.c, src/ops.c, src/proto/ops.pro, src/testdir/test_wordcount.in, src/testdir/test_wordcount.ok, src/testdir/Make_all.mak

Patch 7.4.1043

Problem: Another small thing.
Solution: Now really update the Mac install text.

Files: src/INSTALLmac.txt

Patch 7.4.1044 (after 7.4.1042)

Problem: Can't build without the +eval feature.

Solution: Add #ifdef.

Files: src/ops.c

Patch 7.4.1045

Problem: Having shadow and coverage on the same build results in the source files not being available in the coverage view.

Solution: Move using shadow to the normal build.

Files: .travis.yml

Patch 7.4.1046

Problem: No test coverage for menus.

Solution: Load the standard menus and check there is no error.

Files: src/testdir/test_menu.vim, src/testdir/test_alot.vim

Patch 7.4.1047 (after patch 7.4.1042)

Problem: Tests fail on MS-Windows.

Solution: Set '**selection**' to inclusive.

Files: src/testdir/test_wordcount.in

Patch 7.4.1048 (after patch 7.4.1047)

Problem: Wordcount test still fail on MS-Windows.

Solution: Set '**fileformat**' to "unix".

Files: src/testdir/test_wordcount.in

Patch 7.4.1049 (after patch 7.4.1048)

Problem: Wordcount test still fails on MS-Windows.

Solution: Set '**fileformats**' to "unix".

Files: src/testdir/test_wordcount.in

Patch 7.4.1050

Problem: Warning for unused var with tiny features. (Tony Mechelynck)

Solution: Add #ifdef. Use vim_snprintf(). Reduce number of statements.

Files: src/ops.c

Patch 7.4.1051

Problem: Segfault when unletting "count".

Solution: Check for readonly and locked first. (Dominique Pelle)
Add a test.

Files: src/eval.c, src/testdir/test_alot.vim, src/testdir/test_unlet.vim

Patch 7.4.1052

Problem: Illegal memory access with weird syntax command. (Dominique Pelle)

Solution: Check for column past end of line.

Files: src/syntax.c

Patch 7.4.1053

Problem: Insufficient testing for quickfix commands.

Solution: Add a new style quickfix test. (Yegappan Lakshmanan)

Files: src/testdir/Make_all.mak, src/testdir/test_quickfix.vim

Patch 7.4.1054

Problem: Illegal memory access.
Solution: Check for missing pattern. (Dominique Pelle)
Files: src/syntax.c

Patch 7.4.1055

Problem: Running "make newtests" in src/testdir has no output.
Solution: List the messages file when a test fails. (Christian Brabandt)
Update the list of tests.
Files: src/Makefile, src/testdir/Makefile

Patch 7.4.1056

Problem: Don't know why finding spell suggestions is slow.
Solution: Add some code to gather profiling information.
Files: src/spell.c

Patch 7.4.1057

Problem: Typos in the :options window.
Solution: Fix the typos. (Dominique Pelle)
Files: runtime/optwin.vim

Patch 7.4.1058

Problem: It is not possible to test code that is only reached when memory allocation fails.
Solution: Add the alloc_fail() function. Try it out with :vimgrep.
Files: runtime/doc/eval.txt, src/globals.h, src/eval.c, src/quickfix.c, src/misc2.c, src/proto/misc2.pro, src/testdir/test_quickfix.vim

Patch 7.4.1059

Problem: Code will never be executed.
Solution: Remove the code.
Files: src/quickfix.c

Patch 7.4.1060

Problem: Instructions for writing tests are outdated.
Solution: Mention Make_all.mak. Add steps for new style tests.
Files: src/testdir/README.txt

Patch 7.4.1061

Problem: Compiler warning for ignoring return value of fwrite().
Solution: Do use the return value. (idea: Charles Campbell)
Files: src/misc2.c, src/proto/misc2.pro

Patch 7.4.1062

Problem: Building with Ruby on MS-Windows requires a lot of arguments.
Solution: Make it simpler. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 7.4.1063

Problem: TCL_VER_LONG and DYNAMIC_TCL_VER are not set when building with Cygwin and MingW.
Solution: Add TCL_VER_LONG and DYNAMIC_TCL_VER to the makefile. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.1064

Problem: When a spell file has single letter compounding creating suggestions takes an awful long time.
Solution: Add the NOCOMPOUNDSUGS flag.
Files: runtime/doc/spell.txt, src/spell.c

Patch 7.4.1065

Problem: Cannot use the "dll" options on MS-Windows.
Solution: Support the options on all platforms. Use the built-in name as the default, so that it's clear what Vim is looking for.
Files: src/if_python.c, src/if_python3.c, src/if_lua.c, src/if_perl.xs, src/if_ruby.c, src/option.c, runtime/doc/options.txt, src/Makefile

Patch 7.4.1066 (after 7.4.1065)

Problem: Build fails on MS-Windows.
Solution: Adjust the #ifdefs for "dll" options.
Files: src/option.h

Patch 7.4.1067 (after 7.4.1065)

Problem: Can't build with MingW and Python on MS-Windows.
Solution: Move the build flags to CFLAGS.
Files: src/Make_cyg_ming.mak

Patch 7.4.1068

Problem: Wrong way to check for unletting internal variables.
Solution: Use a better way. (Olaf Dabrunz)
Files: src/testdir/test_unlet.c, src/eval.c

Patch 7.4.1069

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/misc2.c

Patch 7.4.1070

Problem: The Tcl interface can't be loaded dynamically on Unix.
Solution: Make it possible to load it dynamically. (Ken Takata)
Files: runtime/doc/if_tcl.txt, runtime/doc/options.txt, runtime/doc/quickref.txt, runtime/optwin.vim, src/Makefile, src/config.h.in, src/configure.in, src/auto/configure, src/if_tcl.c, src/option.c, src/option.h

Patch 7.4.1071

Problem: New style tests are executed in arbitrary order.
Solution: Sort the test function names. (Hirohito Higashi)
Fix the quickfix test that depended on the order.
Files: src/testdir/runtest.vim, src/testdir/test_quickfix.vim

Patch 7.4.1072

Problem: Increment test is old style.
Solution: Make the increment test a new style test. (Hirohito Higashi)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_increment.in, src/testdir/test_increment.ok, src/testdir/test_increment.vim

Patch 7.4.1073

Problem: Alloc_id depends on numbers, may use the same one twice. It's not clear from the number what it's for.

Solution: Use an enum. Add a function to lookup the enum value from the name.

Files: src/misc2.c, src/vim.h, src/alloc.h, src/globals.h,
src/testdir/runtest.vim, src/proto/misc2.pro,
src/testdir/test_quickfix.vim

Patch 7.4.1074

Problem: Warning from VC2015 compiler.

Solution: Add a type cast. (Mike Williams)

Files: src/gui_dwrite.cpp

Patch 7.4.1075

Problem: Crash when using an invalid command.

Solution: Fix generating the error message. (Dominique Pelle)

Files: src/ex_docmd.c

Patch 7.4.1076

Problem: **CTRL-A** does not work well in right-left mode.

Solution: Remove reversing the line, add a test. (Hirohito Higashi)

Files: src/ops.c, src/testdir/test_increment.vim

Patch 7.4.1077

Problem: The build instructions for MS-Windows are incomplete.

Solution: Add explanations for how to build with various interfaces. (Ken Takata)

Files: src/INSTALLpc.txt

Patch 7.4.1078

Problem: MSVC: "make clean" doesn't cleanup in the tee directory.

Solution: Add the commands to cleanup tee. (Erich Ritz)

Files: src/Make_mvc.mak

Patch 7.4.1079 (after 7.4.1073)

Problem: New include file missing from distribution. Missing changes to quickfix code.

Solution: Add alloc.h to the list of distributed files. Use the enum in quickfix code.

Files: Filelist, src/quickfix.c

Patch 7.4.1080

Problem: VS2015 has a function HandleToLong() that is shadowed by the macro that Vim defines.

Solution: Do not define HandleToLong() for MSVC version 1400 and later. (Mike Williams)

Files: src/gui_w32.c

Patch 7.4.1081

Problem: No test for what previously caused a crash.

Solution: Add test for unletting errmsg.

Files: src/testdir/test_unlet.vim

Patch 7.4.1082

Problem: The Tcl interface is always skipping memory free on exit.
Solution: Only skip for dynamically loaded Tcl.
Files: src/if_tcl.c

Patch 7.4.1083

Problem: Building GvimExt with VS2015 may fail.
Solution: Adjust the makefile. (Mike Williams)
Files: src/GvimExt/Makefile

Patch 7.4.1084

Problem: Using "." to repeat **CTRL-A** in Visual mode increments the wrong numbers.
Solution: Append right size to the redo buffer. (Ozaki Kiichi)
Files: src/normal.c, src/testdir/test_increment.vim

Patch 7.4.1085

Problem: The **CTRL-A** and **CTRL-X** commands do not update the '[' and ']' marks.
Solution: (Yukihiro Nakadaira)
Files: src/ops.c, src/testdir/test_marks.in, src/testdir/test_marks.ok

Patch 7.4.1086

Problem: Crash with an extremely long buffer name.
Solution: Limit the return value of vim_snprintf(). (Dominique Pelle)
Files: src/buffer.c

Patch 7.4.1087

Problem: **CTRL-A** and **CTRL-X** do not work properly with blockwise visual selection if there is a mix of Tab and spaces.
Solution: Add OP_NR_ADD and OP_NR_SUB. (Hirohito Higashi)
Files: src/testdir/test_increment.vim, src/normal.c, src/ops.c, src/proto/ops.pro, src/vim.h

Patch 7.4.1088

Problem: Coverity warns for uninitialized variables. Only one is an actual problem.
Solution: Move the conditions. Don't use endpos if handling an error.
Files: src/ops.c

Patch 7.4.1089

Problem: Repeating **CTRL-A** doesn't work.
Solution: Call prep_redo_cmd(). (Hirohito Higashi)
Files: src/normal.c, src/testdir/test_increment.vim

Patch 7.4.1090

Problem: No tests for :hardcopy and related options.
Solution: Add test_hardcopy.
Files: src/testdir/test_hardcopy.vim, src/Makefile, src/testdir/Make_all.mak

Patch 7.4.1091

Problem: When making a change while need_wait_return is set there is a two second delay.
Solution: Do not assume the ATTENTION prompt was given when need_wait_return

was set already.
Files: src/misc1.c

Patch 7.4.1092

Problem: It is not simple to test for an exception and give a proper error message.

Solution: Add assert_exception().

Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.1093

Problem: Typo in test goes unnoticed.

Solution: Fix the typo. Give error for wrong arguments to cursor(). (partly by Hirohito Higashi) Add a test for cursor().

Files: src/testdir/test_searchpos.vim, src/testdir/test_cursor_func.vim, src/eval.c, src/testdir/test_alot.vim

Patch 7.4.1094

Problem: Test for :hardcopy fails on MS-Windows.

Solution: Check for the +postscript feature.

Files: src/testdir/test_hardcopy.vim

Patch 7.4.1095

Problem: Can't build GvimExt with SDK 7.1.

Solution: Support using setenv.bat instead of vcvars32.bat. (Ken Takata)

Files: src/Make_mvc.mak, src/GvimExt/Makefile

Patch 7.4.1096

Problem: Need several lines to verify a command produces an error.

Solution: Add assert_fails(). (suggested by Nikolai Pavlov)
Make the quickfix alloc test actually work.

Files: src/testdir/test_quickfix.vim, src/eval.c, runtime/doc/eval.txt, src/misc2.c, src/alloc.h

Patch 7.4.1097

Problem: Looking up the alloc ID for tests fails.

Solution: Fix the line computation. Use assert_fails() for unlet test.

Files: src/testdir/runtest.vim, src/testdir/test_unlet.vim

Patch 7.4.1098

Problem: Still using old style C function declarations.

Solution: Always define __ARGS() to include types. Turn a few functions into ANSI style to find out if this causes problems for anyone.

Files: src/vim.h, src/os_unix.h, src/eval.c, src/main.c

Patch 7.4.1099

Problem: It's not easy to know if Vim supports blowfish. (Smu Johnson)

Solution: Add has('crypt-blowfish') and has('crypt-blowfish2').

Files: src/eval.c

Patch 7.4.1100

Problem: Cygwin makefiles are unused.

Solution: Remove them.

Files: src/GvimExt/Make_ming.mak, src/GvimExt/Make_cyg.mak, src/xxd/Make_ming.mak, src/xxd/Make_cyg.mak

Patch 7.4.1101

Problem: With **'rightleft'** and concealing the cursor may move to the wrong position.
Solution: Compute the column differently when **'rightleft'** is set. (Hirohito Higashi)
Files: src/screen.c

Patch 7.4.1102

Problem: Debugger has no stack backtrace support.
Solution: Add "backtrace", "frame", "up" and "down" commands. (Alberto Fanjul, closes #433)
Files: runtime/doc/repeat.txt, src/eval.c, src/ex_cmds2.c, src/globals.h, src/testdir/Make_all.mak, src/testdir/test108.in, src/testdir/test108.ok

Patch 7.4.1103 (after 7.4.1100)

Problem: Removed file still in distribution.
Solution: Remove Make_cyg.mak from the list of files.
Files: Filelist

Patch 7.4.1104

Problem: Various problems building with MzScheme/Racket.
Solution: Make it work with new versions of Racket. (Yukihiro Nakadaira, Ken Takata)
Files: runtime/doc/if_mzsch.txt, src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/auto/configure, src/configure.in, src/if_mzsch.c

Patch 7.4.1105

Problem: When using slices there is a mixup of variable name and namespace.
Solution: Recognize variables that can't be a namespace. (Hirohito Higashi)
Files: src/eval.c, src/testdir/test_eval.in, src/testdir/test_eval.ok

Patch 7.4.1106

Problem: The nsis script can't be used from the appveyor build.
Solution: Add "ifndef" to allow for variables to be set from the command line. Remove duplicate SetCompressor command. Support using other gettext binaries. (Ken Takata) Update build instructions to use libintl-8.dll.
Files: Makefile, nsis/gvim.nsi, src/os_win32.c, src/proto/os_win32.pro, src/main.c, os_w32exe.c

Patch 7.4.1107

Problem: Vim can create a directory but not delete it.
Solution: Add an argument to delete() to make it possible to delete a directory, also recursively.
Files: src/fileio.c, src/eval.c, src/proto/fileio.pro, src/testdir/test_delete.vim, src/testdir/test_alot.vim, runtime/doc/eval.txt

Patch 7.4.1108

Problem: Expanding "~" halfway a file name.
Solution: Handle the file name as one name. (Marco Hinz) Add a test.

Closes #564.

Files: src/testdir/test27.in, src/testdir/test27.ok,
src/testdir/test_expand.vim, src/testdir/test_alot.vim,
src/Makefile, src/misc2.c

Patch 7.4.1109 (after 7.4.1107)
Problem: MS-Windows doesn't have rmdir().
Solution: Add mch_rmdir().
Files: src/os_win32.c, src/proto/os_win32.pro

Patch 7.4.1110
Problem: Test 108 fails when language is French.
Solution: Force English messages. (Dominique Pelle)
Files: src/testdir/test108.in

Patch 7.4.1111
Problem: test_expand fails on MS-Windows.
Solution: Always use forward slashes. Remove references to test27.
Files: src/testdir/runtest.vim, src/testdir/test_expand.vim,
src/testdir/Make_dos.mak, src/testdir/Make_all.mak,
src/testdir/Make_amiga.mak, src/testdir/Make_ming.mak

Patch 7.4.1112
Problem: When using ":next" with an illegal file name no error is reported.
Solution: Give an error message.
Files: src/ex_cmds2.c

Patch 7.4.1113 (after 7.4.1105)
Problem: Using {ns} in variable name does not work. (lilydjwg)
Solution: Fix recognizing colon. Add a test.
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1114 (after 7.4.1107)
Problem: delete() does not work well with symbolic links.
Solution: Recognize symbolic links.
Files: src/eval.c, src/fileio.c, src/os_unix.c, src/proto/os_unix.pro,
src/testdir/test_delete.vim, runtime/doc/eval.txt

Patch 7.4.1115
Problem: MS-Windows: make clean in testdir doesn't clean everything.
Solution: Add command to delete X* directories. (Ken Takata)
Files: src/testdir/Make_dos.mak

Patch 7.4.1116
Problem: delete(x, 'rf') does not delete files starting with a dot.
Solution: Also delete files starting with a dot.
Files: src/misc1.c, src/fileio.c, src/vim.h

Patch 7.4.1117 (after 7.4.1116)
Problem: No longer get "." and ".." in directory list.
Solution: Do not skip "." and ".." unless EW_DODOT is set.
Files: src/misc1.c

Patch 7.4.1118

Problem: Tests hang in 24 line terminal.
Solution: Set the **'more'** option off.
Files: src/testdir/runtest.vim

Patch 7.4.1119

Problem: argidx() has a wrong value after ":%argdelete". (Yegappan Lakshmanan)
Solution: Correct the value of w_arg_idx. Add a test.
Files: src/ex_cmds2.c, src/testdir/test_arglist.vim, src/testdir/Make_all.mak

Patch 7.4.1120

Problem: delete(x, **'rf'**) fails if a directory is empty. (Lcd)
Solution: Ignore not finding matches in an empty directory.
Files: src/fileio.c, src/misc1.c, src/vim.h, src/testdir/test_delete.vim

Patch 7.4.1121

Problem: test_expand leaves files behind.
Solution: Edit another file before deleting, otherwise the swap file remains.
Files: src/testdir/test_expand.vim

Patch 7.4.1122

Problem: Test 92 and 93 fail when using gvim on a system with a non utf-8 locale.
Solution: Avoid using .gvimrc by adding -U NONE. (Yukihiro Nakadaira)
Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 7.4.1123

Problem: Using ":argadd" when there are no arguments results in the second argument to be the current one. (Yegappan Lakshmanan)
Solution: Correct the w_arg_idx value.
Files: src/ex_cmds2.c, src/testdir/test_arglist.vim

Patch 7.4.1124

Problem: MS-Windows: dead key behavior is not ideal.
Solution: Handle dead keys differently when not in Insert or Select mode. (John Wellesz, closes #399)
Files: src/gui_w48.c

Patch 7.4.1125

Problem: There is no perlevel().
Solution: Add perlevel(). (Damien)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/eval.c, src/if_perl.xs, src/proto/if_perl.pro, src/testdir/Make_all.mak, src/testdir/test_perl.vim

Patch 7.4.1126

Problem: Can only get the directory of the current window.
Solution: Add window and tab arguments to getcwd() and haslocaldir(). (Thinca, Hirohito Higashi)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_getcwd.in, src/testdir/test_getcwd.ok,

runtime/doc/eval.txt, patching file src/eval.c

Patch 7.4.1127

Problem: Both old and new style tests for Perl.
Solution: Merge the old tests with the new style tests.
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_perl.in,
src/testdir/test_perl.ok, src/testdir/test_perl.vim

Patch 7.4.1128

Problem: MS-Windows: delete() does not recognize junctions.
Solution: Add mch_isreaddir() for MS-Windows. Update mch_is_symbolic_link().
(Ken Takata)
Files: src/fileio.c, src/os_win32.c, src/proto/os_win32.pro

Patch 7.4.1129

Problem: Python None value can't be converted to a Vim value.
Solution: Just use zero. (Damien)
Files: src/if_py_both.h, src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok,

Patch 7.4.1130

Problem: Memory leak in :vimgrep.
Solution: Call FreeWild(). (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 7.4.1131

Problem: New lines in the viminfo file are dropped.
Solution: Copy lines starting with "|". Fix that when using :rviminfo in a
function global variables were restored as function-local
variables.
Files: src/eval.c, src/structs.h, src/ex_cmds.c, src/misc2.c,
src/proto/misc2.pro, src/testdir/test_viminfo.vim,
src/testdir/Make_all.mak, src/testdir/test74.in,
src/testdir/test74.ok

Patch 7.4.1132

Problem: Old style tests for the argument list.
Solution: Add more new style tests. (Yegappan Lakshmanan)
Files: src/testdir/test_arglist.vim, src/testdir/test_argument_0count.in,
src/testdir/test_argument_0count.ok,
src/testdir/test_argument_count.in, src/Makefile,
src/testdir/test_argument_count.ok, src/testdir/Make_all.mak

Patch 7.4.1133

Problem: Generated function prototypes still have __ARGS().
Solution: Generate function prototypes without __ARGS().
Files: src/Makefile, src/if_ruby.c, src/os_win32.c,
src/proto/blowfish.pro, src/proto/buffer.pro,
src/proto/charset.pro, src/proto/crypt.pro,
src/proto/crypt_zip.pro, src/proto/diff.pro,
src/proto/digraph.pro, src/proto/edit.pro, src/proto/eval.pro,
src/proto/ex_cmds2.pro, src/proto/ex_cmds.pro,
src/proto/ex_docmd.pro, src/proto/ex_eval.pro,
src/proto/ex_getln.pro, src/proto/fileio.pro, src/proto/fold.pro,

src/proto/getchar.pro, src/proto/gui_athena.pro,
src/proto/gui_beval.pro, src/proto/gui_gtk_gresources.pro,
src/proto/gui_gtk.pro, src/proto/gui_gtk_x11.pro,
src/proto/gui_mac.pro, src/proto/gui_motif.pro,
src/proto/gui_photon.pro, src/proto/gui.pro,
src/proto/gui_w16.pro, src/proto/gui_w32.pro,
src/proto/gui_x11.pro, src/proto/gui_xmdl.pro,
src/proto/hangulin.pro, src/proto/hardcopy.pro,
src/proto/hashtab.pro, src/proto/if_cscope.pro,
src/proto/if_lua.pro, src/proto/if_mzsch.pro,
src/proto/if_ole.pro, src/proto/if_perl.pro,
src/proto/if_perlsfio.pro, src/proto/if_python3.pro,
src/proto/if_python.pro, src/proto/if_ruby.pro,
src/proto/if_tcl.pro, src/proto/if_xcmdsrv.pro,
src/proto/main.pro, src/proto/mark.pro, src/proto/mbyte.pro,
src/proto/memfile.pro, src/proto/memline.pro, src/proto/menu.pro,
src/proto/message.pro, src/proto/misc1.pro, src/proto/misc2.pro,
src/proto/move.pro, src/proto/netbeans.pro, src/proto/normal.pro,
src/proto/ops.pro, src/proto/option.pro, src/proto/os_amiga.pro,
src/proto/os_beos.pro, src/proto/os_mac_conv.pro,
src/proto/os_msdos.pro, src/proto/os_mswin.pro,
src/proto/os_qnx.pro, src/proto/os_unix.pro, src/proto/os_vms.pro,
src/proto/os_win16.pro, src/proto/os_win32.pro,
src/proto/popupmnu.pro, src/proto/pty.pro, src/proto/quickfix.pro,
src/proto/regexp.pro, src/proto/screen.pro, src/proto/search.pro,
src/proto/sha256.pro, src/proto/spell.pro, src/proto/syntax.pro,
src/proto/tag.pro, src/proto/termlib.pro, src/proto/term.pro,
src/proto/ui.pro, src/proto/undo.pro, src/proto/version.pro,
src/proto/winclip.pro, src/proto/window.pro,
src/proto/workshop.pro

Patch 7.4.1134

Problem: The arglist test fails on MS-Windows.
Solution: Only check for failure of argedit on Unix.
Files: src/testdir/test_arglist.vim

Patch 7.4.1135

Problem: One more arglist test fails on MS-Windows.
Solution: Don't edit "Y" after editing "y".
Files: src/testdir/test_arglist.vim

Patch 7.4.1136

Problem: Wrong argument to assert_exception() causes a crash. (reported by Coverity)
Solution: Check for NULL pointer. Add a test.
Files: src/eval.c, src/testdir/test_assert.vim

Patch 7.4.1137

Problem: Illegal memory access when using :copen and :cclose.
Solution: Avoid that curbuf is invalid. (suggestion by Justin M. Keyes)
Add a test.
Files: src/window.c, src/testdir/test_quickfix.vim

Patch 7.4.1138

Problem: When running gvim in the foreground some icons are missing.
(Taylor Venable)
Solution: Move the call to gui_gtk_register_resource(). (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 7.4.1139

Problem: MS-Windows: getftype() returns "file" for symlink to directory.
Solution: Make it return "dir". (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.1140

Problem: Recognizing <sid> does not work when the language is Turkish.
(Christian Brabandt)
Solution: Use MB_STNICMP() instead of STNICMP().
Files: src/eval.c

Patch 7.4.1141

Problem: Using searchpair() with a skip expression that uses syntax
highlighting sometimes doesn't work. (David Fishburn)
Solution: Reset next_match_idx. (Christian Brabandt)
Files: src/syntax.c

Patch 7.4.1142

Problem: Cannot define keyword characters for a syntax file.
Solution: Add the ":syn iskeyword" command. (Christian Brabandt)
Files: runtime/doc/options.txt, runtime/doc/syntax.txt, src/buffer.c,
src/option.c, src/structs.h, src/syntax.c,
src/testdir/Make_all.mak, src/testdir/test_syntax.vim

Patch 7.4.1143

Problem: Can't sort on floating point numbers.
Solution: Add the "f" flag to ":sort". (Alex Jakushev) Also add the "f"
flag to sort().
Files: runtime/doc/change.txt, src/ex_cmds.c, src/testdir/test_sort.vim,
src/testdir/test57.in, src/testdir/test57.ok, src/eval.c

Patch 7.4.1144 (after 7.4.1143)

Problem: Can't build on several systems.
Solution: Include float.h. (Christian Robinson, closes #570 #571)
Files: src/ex_cmds.c

Patch 7.4.1145

Problem: Default features are conservative.
Solution: Make the default feature set for most of today's systems "huge".
Files: src/feature.h, src/configure.in, src/auto/configure

Patch 7.4.1146

Problem: Can't build with Python 3 interface using MingW.
Solution: Update the Makefile. (Yasuhiro Matsumoto, Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.1147

Problem: Conflict for "chartab". (Kazunobu Kuriyama)
Solution: Rename the global one to something less obvious. Move it into

src/chartab.c.
Files: src/macros.h, src/globals.h, src/charset.c, src/main.c,
src/option.c, src/screen.c, src/vim.h

Patch 7.4.1148

Problem: Default for MingW and Cygwin is still "normal".
Solution: Use "huge" as default. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 7.4.1149 (after 7.4.1013)

Problem: Using the local value of '**errorformat**' causes more problems than it solves.
Solution: Revert 7.4.1013.
Files: runtime/doc/quickfix.txt, src/quickfix.c

Patch 7.4.1150

Problem: '**langmap**' applies to the first character typed in Select mode. (David Watson)
Solution: Check for SELECTMODE. (Christian Brabandt, closes #572)
Add the 'x' flag to feedkeys().
Files: src/getchar.c, src/normal.c, src/testdir/test_langmap.vim,
src/ex_docmd.c, src/proto/ex_docmd.pro, src/testdir/Make_all.mak,
runtime/doc/eval.txt

Patch 7.4.1151 (after 7.4.1150)

Problem: Missing change to eval.c
Solution: Also change feedkeys().
Files: src/eval.c

Patch 7.4.1152

Problem: Langmap test fails with normal build.
Solution: Check for +langmap feature.
Files: src/testdir/test_langmap.vim

Patch 7.4.1153

Problem: Autocommands triggered by quickfix cannot always get the current title value.
Solution: Call qf_fill_buffer() later. (Christian Brabandt)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1154

Problem: No support for JSON.
Solution: Add jsonencode() and jsondecode(). Also add v:false, v:true, v:null and v:none.
Files: src/json.c, src/eval.c, src/proto.h, src/structs.h, src/vim.h,
src/if_lua.c, src/if_mzsch.c, src/if_ruby.c, src/if_py_both.h,
src/globals.h, src/Makefile, src/Make_bc3.mak, src/Make_bc5.mak,
src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak,
src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak,
src/Make_sas.mak, src/Make_vms.mms, src/proto/json.pro,
src/proto/eval.pro, src/testdir/test_json.vim,
src/testdir/test_alot.vim, Filelist, runtime/doc/eval.txt

Patch 7.4.1155

Problem: Build with normal features fails.
Solution: Always define dict_lookup().
Files: src/eval.c

Patch 7.4.1156

Problem: Coverity warns for NULL pointer and ignoring return value.
Solution: Check for NULL pointer. When dict_add() returns FAIL free the item.
Files: src/json.c

Patch 7.4.1157

Problem: type() does not work for v:true, v:none, etc.
Solution: Add new type numbers.
Files: src/eval.c, src/testdir/test_json.vim, src/testdir/test_viml.vim

Patch 7.4.1158

Problem: Still using __ARGS().
Solution: Remove __ARGS() from eval.c
Files: src/eval.c

Patch 7.4.1159

Problem: Automatically generated function prototypes use __ARGS.
Solution: Remove __ARGS from osdef.sh.
Files: src/osdef.sh, src/osdef1.h.in, src/osdef2.h.in

Patch 7.4.1160

Problem: No error for jsondecode('').
Solution: Give an error message for missing double quote.
Files: src/json.c

Patch 7.4.1161

Problem: ":argadd" without argument is supposed to add the current buffer name to the arglist.
Solution: Make it work as documented. (Coot, closes #577)
Files: src/ex_cmds.h, src/ex_cmds2.c, src/testdir/test_arglist.vim

Patch 7.4.1162

Problem: Missing error number in MzScheme. (Dominique Pelle)
Solution: Add a proper error number.
Files: src/if_mzsch.c

Patch 7.4.1163

Problem: Expressions "% + v:true" and "" . v:true" cause an error.
Solution: Return something sensible when using a special variable as a number or as a string. (suggested by Damien)
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1164

Problem: No tests for comparing special variables. Error in jsondecode() not reported. test_json does not work with Japanese system.
Solution: Set scriptencoding. (Ken Takata) Add a few more tests. Add error.
Files: src/json.c, src/testdir/test_viml.vim, src/testdir/test_json.vim

Patch 7.4.1165

Problem: When defining DYNAMIC_ICONV_DLL in the makefile, the build fails.

Solution: Add #ifdef's. (Taro Muraoka) Try the newer version first.
Files: src/mbyte.c, src/os_win32.c

Patch 7.4.1166

Problem: Can't encode a Funcref into JSON. jsonencode() doesn't handle the same list or dict twice properly. (Nikolai Pavlov)

Solution: Give an error. Reset copyID when the list or dict is finished.

Files: src/json.c, src/proto/json.pro, src/testdir/test_json.vim

Patch 7.4.1167

Problem: No tests for "is" and "isnot" with the new variables.

Solution: Add tests.

Files: src/testdir/test_viml.vim

Patch 7.4.1168

Problem: This doesn't give the right result: eval(string(v:true)). (Nikolai Pavlov)

Solution: Make the string "v:true" instead of "true".

Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1169

Problem: The socket I/O is intertwined with the netbeans code.

Solution: Start refactoring the netbeans communication to split off the socket I/O. Add the +channel feature.

Files: src/channel.c, src/netbeans.c, src/proto/channel.pro, src/proto/netbeans.pro, src/proto/gui_w32.pro, src/gui_w32.c, src/eval.c, src/os_mswin.c, src/ui.c, src/macros.h, Makefile, src/proto.h, src/feature.h, src/os_unix.c, src/vim.h, src/configure.in, src/auto/configure, src/config.mk.in, src/config.aap.in, src/config.h.in, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 7.4.1170 (after 7.4.1169)

Problem: Missing changes in src/Makefile, Filelist.

Solution: Add the missing changes.

Files: Filelist, src/Makefile

Patch 7.4.1171

Problem: Makefile dependencies are outdated.

Solution: Run "make depend". Add GTK resource dependencies.

Files: src/Makefile

Patch 7.4.1172 (after 7.4.1169)

Problem: Configure is overly positive.

Solution: Insert "test".

Files: src/configure.in, src/auto/configure

Patch 7.4.1173 (after 7.4.1168)

Problem: No test for new behavior of v:true et al.

Solution: Add a test.

Files: src/testdir/test_viml.vim

Patch 7.4.1174

Problem: Netbeans contains dead code inside #ifndef INIT_SOCKETS.

Solution: Remove the dead code.
Files: src/netbeans.c

Patch 7.4.1175 (after 7.4.1169)
Problem: Can't build with Mingw and Cygwin.
Solution: Remove extra "endif". (Christian J. Robinson)
Files: src/Make_cyg_ming.mak

Patch 7.4.1176
Problem: Missing change to proto file.
Solution: Update the proto file. (Charles Cooper)
Files: src/proto/gui_w32.pro

Patch 7.4.1177
Problem: The +channel feature is not in :version output. (Tony Mechelynck)
Solution: Add the feature string.
Files: src/version.c

Patch 7.4.1178
Problem: empty() doesn't work for the new special variables.
Solution: Make empty() work. (Damien)
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1179
Problem: test_writefile and test_viml do not delete the tempfile.
Solution: Delete the tempfile. (Charles Cooper) Add DeleteTheScript().
Files: src/testdir/test_writefile.in, src/testdir/test_viml.vim

Patch 7.4.1180
Problem: Crash with invalid argument to glob2regpat().
Solution: Check for NULL. (Justin M. Keyes, closes #596) Add a test.
Files: src/eval.c, src/testdir/test_glob2regpat.vim,
src/testdir/test_alot.vim

Patch 7.4.1181
Problem: free_tv() can't handle special variables. (Damien)
Solution: Add the variable type.
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1182
Problem: Still socket code intertwined with netbeans.
Solution: Move code from netbeans.c to channel.c
Files: src/channel.c, src/netbeans.c, src/proto/channel.pro,
src/proto/netbeans.pro, src/gui.c, src/gui_w48.c

Patch 7.4.1183 (after 7.4.1182)
Problem: MS-Windows build is broken.
Solution: Remove init in wrong place.
Files: src/channel.c

Patch 7.4.1184 (after 7.4.1182)
Problem: MS-Windows build is still broken.
Solution: Change nbsock to ch_fd.
Files: src/channel.c

Patch 7.4.1185

Problem: Can't build with TCL on some systems.
Solution: Rename the channel_ functions.
Files: src/if_tcl.c

Patch 7.4.1186

Problem: Error messages for security context are hard to translate.
Solution: Use one string with %s. (Ken Takata)
Files: src/os_unix.c

Patch 7.4.1187

Problem: MS-Windows channel code only supports one channel. Doesn't build without netbeans support.
Solution: Get the channel index from the socket in the message. Closes #600.
Files: src/channel.c, src/netbeans.c, src/gui_w48.c, src/proto/channel.pro, src/proto/netbeans.pro

Patch 7.4.1188

Problem: Using older JSON standard.
Solution: Update the link. Adjust the text a bit.
Files: src/json.c, runtime/doc/eval.txt

Patch 7.4.1189 (after 7.4.1165)

Problem: Using another language on MS-Windows does not work. (Yongwei Wu)
Solution: Undo the change to try loading libintl-8.dll first.
Files: src/os_win32.c

Patch 7.4.1190

Problem: On OSX the default flag for dlopen() is different.
Solution: Add RTLD_LOCAL in the configure check. (sv99, closes #604)
Files: src/configure.in, src/auto/configure

Patch 7.4.1191

Problem: The channel feature isn't working yet.
Solution: Add the connect(), disconnect(), sendexpr() and sendraw() functions. Add initial documentation. Add a demo server.
Files: src/channel.c, src/eval.c, src/proto/channel.pro, src/proto/eval.pro, runtime/doc/channel.txt, runtime/doc/eval.txt, runtime/doc/Makefile, runtime/tools/demoserver.py

Patch 7.4.1192

Problem: Can't build with FEAT_EVAL but without FEAT_MBYTE. (John Marriott)
Solution: Add #ifdef for FEAT_MBYTE.
Files: src/json.c

Patch 7.4.1193

Problem: Can't build the channel feature on MS-Windows.
Solution: Add #ifdef HAVE_POLL.
Files: src/channel.c

Patch 7.4.1194

Problem: Compiler warning for not using return value of fwrite().

Solution: Return OK/FAIL. (Charles Campbell)
Files: src/channel.c, src/proto/channel.pro

Patch 7.4.1195

Problem: The channel feature does not work in the MS-Windows console.
Solution: Add win32 console support. (Yasuhiro Matsumoto)
Files: src/channel.c, src/gui_w32.c, src/os_mswin.c, src/os_win32.c,
src/proto/gui_w32.pro, src/proto/os_mswin.pro, src/vim.h

Patch 7.4.1196

Problem: Still using __ARGS.
Solution: Remove __ARGS in several files. (script by Hirohito Higashi)
Files: src/arabic.c, src/buffer.c, src/charset.c, src/crypt_zip.c,
src/diff.c, src/digraph.c, src/edit.c, src/ex_cmds.c,
src/ex_cmds2.c, src/ex_docmd.c

Patch 7.4.1197

Problem: Still using __ARGS.
Solution: Remove __ARGS in several files. (script by Hirohito Higashi)
Files: src/ex_eval.c, src/ex_getln.c, src/farsi.c, src/fileio.c,
src/fold.c, src/getchar.c, src/gui.c, src/gui_at_fs.c,
src/gui_at_sb.c, src/gui_athena.c, src/gui_beval.c,
src/gui_motif.c, src/gui_w32.c, src/gui_w48.c

Patch 7.4.1198

Problem: Still using __ARGS.
Solution: Remove __ARGS in several files. (script by Hirohito Higashi)
Also remove use of HAVE_STDARG_H.
Files: src/gui_x11.c, src/hangulin.c, src/hardcopy.c, src/hashtab.c,
src/if_cscope.c, src/if_python3.c, src/if_sniff.c,
src/if_xcmdsrv.c, src/main.c, src/mark.c, src/mbyte.c,
src/memfile.c, src/memfile_test.c, src/memline.c, src/menu.c,
src/message.c, src/misc1.c, src/misc2.c, src/move.c,
src/netbeans.c, src/normal.c

Patch 7.4.1199

Problem: Still using __ARGS.
Solution: Remove __ARGS in several files. (script by Hirohito Higashi)
Files: src/ops.c, src/option.c, src/os_amiga.c, src/os_mac_conv.c,
src/os_unix.c, src/os_vms.c, src/os_w32exe.c, src/popupmnu.c,
src/pty.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c,
src/screen.c, src/search.c, src/sha256.c, src/spell.c,
src/syntax.c, src/tag.c, src/term.c, src/termlib.c, src/ui.c,
src/undo.c, src/version.c, src/window.c

Patch 7.4.1200

Problem: Still using __ARGS.
Solution: Remove __ARGS in several files. (script by Hirohito Higashi)
Files: src/blowfish.c, src/ex_cmds2.c, src/ex_getln.c, src/fold.c,
src/gui_beval.c, src/gui_w32.c, src/os_unix.c, src/os_win16.c,
src/pty.c, src/regexp.c, src/syntax.c, src/xpm_w32.c,
src/ex_cmds.h, src/globals.h, src/gui_at_sb.h, src/gui_beval.h,
src/if_cscope.h, src/if_sniff.h, src/nbdebug.h, src/os_unix.h,
src/proto.h, src/structs.h, src/vim.h, src/xpm_w32.h,

src/if_perl.xs, src/proto/if_lua.pro, src/proto/pty.pro,
runtime/tools/xcmdsrv_client.c,
src/Makefile

Patch 7.4.1201

Problem: One more file still using __ARGS.
Solution: Remove __ARGS in the last file. (script by Hirohito Higashi)
Files: src/gui_at_sb.c

Patch 7.4.1202

Problem: Still one more file still using __ARGS.
Solution: Remove __ARGS in the last file. (script by Hirohito Higashi)
(closes #612)
Files: src/proto/os_mac_conv.pro, src/os_mac_conv.c, src/Makefile

Patch 7.4.1203

Problem: Still more files still using __ARGS.
Solution: Remove __ARGS in really the last files.
Files: src/proto/if_mzsch.pro, src/if_mzsch.c, src/vim.h,
src/proto/gui_gtk_gresources.pro, src/proto/gui_mac.pro,
src/proto/if_ole.pro, src/proto/os_qnx.pro, src/Makefile

Patch 7.4.1204

Problem: Latin1 characters cause encoding conversion.
Solution: Remove the characters.
Files: src/gui_motif.c

Patch 7.4.1205

Problem: Using old style function declarations.
Solution: Change to new style function declarations. (script by Hirohito Higashi)
Files: src/arabic.c, src/blowfish.c, src/buffer.c, src/channel.c,
src/charset.c, src/crypt.c, src/crypt_zip.c, src/diff.c,
src/digraph.c, src/edit.c, src/eval.c

Patch 7.4.1206

Problem: Using old style function declarations.
Solution: Change to new style function declarations. (script by Hirohito Higashi)
Files: src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/ex_getln.c, src/farsi.c, src/fileio.c

Patch 7.4.1207

Problem: Using old style function declarations.
Solution: Change to new style function declarations. (script by Hirohito Higashi)
Files: src/fold.c, src/getchar.c, src/gui_at_fs.c, src/gui_athena.c,
src/gui_at_sb.c, src/gui_beval.c, src/gui.c, src/gui_gtk.c,
src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c

Patch 7.4.1208

Problem: Using old style function declarations.
Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/gui_photon.c, src/gui_w32.c, src/gui_w48.c, src/gui_x11.c, src/hangulin.c, src/hardcopy.c, src/hashtab.c, src/if_cscope.c, src/if_mzsch.c, src/if_perlsfio.c, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_sniff.c, src/if_tcl.c, src/if_xcmdsrv.c, src/integration.c

Patch 7.4.1209 (after 7.4.1207)

Problem: Can't build with Athena. (Elimar Riesebieter)

Solution: Fix function declarations.

Files: src/gui_athena.c, src/gui_x11.c, src/gui_at_sb.c, src/gui_at_fs.c

Patch 7.4.1210

Problem: Using old style function declarations.

Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/main.c, src/mark.c, src/mbyte.c, src/memfile.c, src/memfile_test.c, src/memline.c, src/menu.c, src/message.c

Patch 7.4.1211

Problem: Using old style function declarations.

Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/misc1.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c

Patch 7.4.1212 (after 7.4.1207)

Problem: Can't build with Motif.

Solution: Fix function declaration. (Dominique Pelle)

Files: src/gui_motif.c

Patch 7.4.1213

Problem: Using old style function declarations.

Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/os_amiga.c, src/os_mac_conv.c, src/os_msdos.d, src/os_mswin.c, src/os_qnx.c, src/os_unix.c, src/os_vms.c, src/os_win16.c, src/os_win32.c, src/popupmnu.c, src/pty.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/screen.c

Patch 7.4.1214

Problem: Using old style function declarations.

Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/search.c, src/sha256.c, src/spell.c, src/syntax.c, src/tag.c, src/term.c, src/termllib.c, src/ui.c, src/undo.c

Patch 7.4.1215

Problem: Using old style function declarations.

Solution: Change to new style function declarations. (script by Hirohito Higashi)

Files: src/version.c, src/winclip.c, src/window.c, src/workshop.c, src/xpm_w32.c, runtime/doc/doctags.c, runtime/tools/xcmdsrv_client.c, src/po/sjiscorr.c, src/xxd/xxd.c

Patch 7.4.1216

Problem: Still using HAVE_STDARG_H.
Solution: Assume it's always defined.
Files: src/eval.c, src/misc2.c, src/vim.h, src/proto.h, src/configure.in,
src/auto/configure, config.h.in, src/os_amiga.h, src/os_msdos.h,
src/os_vms_conf.h, src/os_win32.h

Patch 7.4.1217

Problem: Execution of command on channel doesn't work yet.
Solution: Implement the "ex" and "normal" commands.
Files: src/channel.c, src/proto/channel.pro, src/misc2.c, src/eval.c,
src/ex_docmd.c, src/proto/ex_docmd.pro, src/feature.h

Patch 7.4.1218

Problem: Missing change in configure. More changes for function style.
Solution: Avoid the typos.
Files: src/configure.in, src/config.h.in, runtime/tools/ccfilter.c,
src/os_msdos.c

Patch 7.4.1219

Problem: Build fails with +channel but without +float.
Solution: Add #ifdef.
Files: src/ex_cmds.c

Patch 7.4.1220

Problem: Warnings for unused variables in tiny build. (Tony Mechelynck)
Solution: Move declarations inside #ifdef. (Hirohito Higashi)
Files: src/ex_cmds.c

Patch 7.4.1221

Problem: Including netbeans and channel support in small and tiny builds.
Build fails with some interfaces.
Solution: Only include these features in small build and above. Let
configure fail if trying to enable an interface that won't build.
Files: src/configure.in, src/auto/configure

Patch 7.4.1222

Problem: ":normal" command and others missing in tiny build.
Solution: Graduate FEAT_EX_EXTRA.
Files: src/feature.h, src/charset.c, src/eval.c, src/ex_cmds.c,
src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/getchar.c,
src/normal.c, src/ui.c, src/version.c, src/globals.h

Patch 7.4.1223

Problem: Crash when setting v:errors to a number.
Solution: Free the typval without assuming its type. (Yasuhiro Matsumoto)
Files: src/eval.c, src/testdir/test_assert.vim

Patch 7.4.1224

Problem: Build problems with GTK on BSD. (Mike Williams)
Solution: Don't use "\$<". Skip building gui_gtk_gresources.h when it doesn't
work. (Kazunobu Kuriyama)
Files: src/Makefile

Patch 7.4.1225

Problem: Still a few old style function declarations.
Solution: Make them new style. (Hirohito Higashi)
Files: runtime/tools/blink.c, src/eval.c, src/ex_cmds2.c, src/ex_getln.c,
src/fileio.c, src/gui_w32.c, src/gui_x11.c, src/if_perl.xs,
src/os_unix.c, src/po/sjiscorr.c, src/pty.c

Patch 7.4.1226

Problem: GRESOURCE_HDR is unused.
Solution: Remove it. (Kazunobu Kuriyama)
Files: src/configure.in, src/auto/configure, src/config.mk.in

Patch 7.4.1227

Problem: Compiler warnings.
Solution: Add UNUSED. Add type cast. (Yegappan Lakshmanan)
Files: src/getchar.c, src/os_macosx.m

Patch 7.4.1228

Problem: copy() and deepcopy() fail with special variables. (Nikolai Pavlov)
Solution: Make it work. Add a test. Closes #614.
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1229

Problem: "eval" and "expr" channel commands don't work yet.
Solution: Implement them. Update the error numbers. Also add "redraw".
Files: src/channel.c, src/eval.c, src/json.c, src/ex_docmd.c,
src/proto/channel.pro, src/proto/json.pro, src/proto/ex_docmd.pro,
runtime/doc/channel.txt

Patch 7.4.1230

Problem: Win32: opening a channel may hang. Not checking for messages while waiting for characters.
Solution: Add a zero timeout. Call parse_queued_messages(). (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.1231

Problem: JSON messages are not parsed properly.
Solution: Queue received messages.
Files: src/eval.c, src/channel.c, src/json.c, src/proto/eval.pro,
src/proto/channel.pro, src/proto/json.pro, src/structs.h

Patch 7.4.1232

Problem: Compiler warnings when the Sniff feature is enabled.
Solution: Add UNUSED.
Files: src/gui_gtk_x11.c

Patch 7.4.1233

Problem: Channel command may cause a crash.
Solution: Check for NULL argument. (Damien)
Files: src/channel.c

Patch 7.4.1234

Problem: Demo server only runs with Python 2.
Solution: Make it run with Python 3 as well. (Ken Takata)
Files: runtime/tools/demoserver.py

Patch 7.4.1235 (after 7.4.1231)
Problem: Missing change to eval.c.
Solution: Include that change.
Files: src/eval.c

Patch 7.4.1236
Problem: When "syntax manual" was used switching between buffers removes the highlighting.
Solution: Set the syntax option without changing the value. (Anton Lindqvist)
Files: runtime/syntax/manual.vim

Patch 7.4.1237
Problem: Can't translate message without adding a line break.
Solution: Join the two parts of the message.
Files: src/memline.c

Patch 7.4.1238
Problem: Can't handle two messages right after each other.
Solution: Find the end of the JSON. Read more when incomplete. Add a C test for the JSON decoding.
Files: src/channel.c, src/json.c, src/proto/json.pro, src/eval.c, src/Makefile, src/json_test.c, src/memfile_test.c, src/structs.h

Patch 7.4.1239
Problem: JSON message after the first one is dropped.
Solution: Put remainder of message back in the queue.
Files: src/channel.c

Patch 7.4.1240
Problem: Visual Studio tools are noisy.
Solution: Suppress startup info. (Mike Williams)
Files: src/GvimExt/Makefile, src/Make_mvc.mak, src/tee/Make_mvc.mak

Patch 7.4.1241 (after 7.4.1238)
Problem: Missing change in Makefile due to diff mismatch
Solution: Update the list of object files.
Files: src/Makefile

Patch 7.4.1242 (after 7.4.1238)
Problem: json_test fails without the eval feature.
Solution: Add #ifdef.
Files: src/json_test.c

Patch 7.4.1243
Problem: Compiler warning for uninitialized variable.
Solution: Initialize it. (Elias Diem)
Files: src/json.c

Patch 7.4.1244

Problem: The channel functions don't sort together.
Solution: Use a common "ch_" prefix.
Files: src/eval.c, runtime/doc/eval.txt, runtime/tools/demoserver.py

Patch 7.4.1245

Problem: File missing from distribution.
Solution: Add json_test.c.
Files: Filelist

Patch 7.4.1246

Problem: The channel functionality isn't tested.
Solution: Add a test using a Python test server.
Files: src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, src/testdir/test_channel.py,
src/testdir/Make_all.mak

Patch 7.4.1247

Problem: The channel test doesn't run on MS-Windows.
Solution: Make it work on the MS-Windows console. (Ken Takata)
Files: src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1248

Problem: Can't reliably stop the channel test server. Can't start the server if the python file is not executable.
Solution: Use "pkill" instead of "killall". Run the python file as an argument instead of as an executable.
Files: src/testdir/test_channel.vim

Patch 7.4.1249

Problem: Crash when the process a channel is connected to exits.
Solution: Use the file descriptor properly. Add a test. (Damien)
Also add a test for eval().
Files: src/channel.c, src/testdir/test_channel.py,
src/testdir/test_channel.vim

Patch 7.4.1250

Problem: Running tests in shadow directory fails.
Solution: Also link testdir/*.py
Files: src/Makefile

Patch 7.4.1251

Problem: New test file missing from distribution.
Solution: Add src/testdir/*.py.
Files: Filelist

Patch 7.4.1252

Problem: The channel test server may receive two messages concatenated.
Solution: Split the messages.
Files: src/testdir/test_channel.py

Patch 7.4.1253

Problem: Python test server not displaying second of two commands. Solaris doesn't have "pkill --full".
Solution: Also echo the second command. Use "pkill -f".

Files: src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1254

Problem: Opening a second channel causes a crash. (Ken Takata)

Solution: Don't re-allocate the array with channels.

Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel.py

Patch 7.4.1255

Problem: Crash for channel "eval" command without third argument.

Solution: Check for missing argument.

Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel.py

Patch 7.4.1256

Problem: On Mac sys.exit(0) doesn't kill the test server.

Solution: Use self.server.shutdown(). (Jun Takimoto)

Files: src/testdir/test_channel.py

Patch 7.4.1257

Problem: Channel test fails in some configurations.

Solution: Add check for the +channel feature.

Files: src/testdir/test_channel.vim

Patch 7.4.1258

Problem: The channel test can fail if messages arrive later.

Solution: Add a short sleep. (Jun Takimoto)

Files: src/testdir/test_channel.vim

Patch 7.4.1259

Problem: No test for what patch 7.3.414 fixed.

Solution: Add a test. (Elias Diem)

Files: src/testdir/test_increment.vim

Patch 7.4.1260

Problem: The channel feature doesn't work on Win32 GUI.

Solution: Use WSAGetLastError(). (Ken Takata)

Files: src/channel.c, src/testdir/test_channel.vim, src/vim.h

Patch 7.4.1261

Problem: Pending channel messages are garbage collected. Leaking memory in ch_sendexpr(). Leaking memory for a decoded JSON string.

Solution: Mark the message list as used. Free the encoded JSON. Don't save the JSON string.

Files: src/eval.c, src/channel.c, src/json.c, src/proto/channel.pro

Patch 7.4.1262

Problem: The channel callback is not invoked.

Solution: Make a list of pending callbacks.

Files: src/eval.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim

Patch 7.4.1263

Problem: ch_open() hangs when the server isn't running.

Solution: Add a timeout. Use a dict to pass arguments. (Yasuhiro Matsumoto)
Files: runtime/doc/eval.txt, runtime/doc/channel.txt, src/channel.c, src/eval.c, src/netbeans.c, src/os_win32.c, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 7.4.1264

Problem: Crash when receiving an empty array.
Solution: Check for array with wrong number of arguments. (Damien)
Files: src/channel.c, src/eval.c, src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1265

Problem: Not all channel commands are tested.
Solution: Add a test for "normal", "expr" and "redraw".
Files: src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1266

Problem: A BufAdd autocommand may cause an ml_get error (Christian Brabandt)
Solution: Increment RedrawingDisabled earlier.
Files: src/ex_cmds.c

Patch 7.4.1267

Problem: Easy to miss handling all types of variables.
Solution: Change the variable type into an enum.
Files: src/structs.h, src/eval.c

Patch 7.4.1268

Problem: Waittime is used as seconds instead of milliseconds. (Hirohito Higashi)
Solution: Divide by 1000.
Files: src/channel.c

Patch 7.4.1269

Problem: Encoding {'key':v:none} to JSON doesn't give an error (Tyru)
Solution: Give an error.
Files: src/json.c, src/testdir/test_json.vim

Patch 7.4.1270

Problem: Warnings for missing values in switch.
Solution: Change switch to if-else or add values.
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 7.4.1271

Problem: assert_false(v:false) reports an error. (Nikolai Pavlov)
Solution: Recognize v:true and v:false. (Closes #625)
Files: src/eval.c, src/testdir/test_assert.vim

Patch 7.4.1272 (after 7.4.1270)

Problem: Using future enum value.
Solution: Remove it.
Files: src/if_python.c, src/if_python3.c

Patch 7.4.1273 (after 7.4.1271)

Problem: assert_false(v:false) still fails.
Solution: Fix the typo.
Files: src/eval.c

Patch 7.4.1274

Problem: Cannot run a job.
Solution: Add job_start(), job_status() and job_stop(). Currently only works for Unix.
Files: src/eval.c, src/structs.h, runtime/doc/eval.txt, src/os_unix.c, src/proto/os_unix.pro, src/feature.h, src/version.c, src/testdir/test_channel.vim

Patch 7.4.1275 (after 7.4.1274)

Problem: Build fails on MS-Windows.
Solution: Fix wrong #ifdef.
Files: src/eval.c

Patch 7.4.1276

Problem: Warning for not using return value of fcntl().
Solution: Explicitly ignore the return value.
Files: src/fileio.c, src/channel.c, src/memfile.c, src/memline.c

Patch 7.4.1277

Problem: Compiler can complain about missing enum value in switch with some combination of features.
Solution: Remove #ifdefs around case statements.
Files: src/eval.c

Patch 7.4.1278

Problem: When jsonencode() fails it still returns something.
Solution: Return an empty string on failure.
Files: src/json.c, src/channel.c, src/testdir/test_json.vim, src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1279

Problem: jsonencode() is not producing strict JSON.
Solution: Add jsencode() and jsdecode(). Make jsonencode() and jsondecode() strict.
Files: src/json.c, src/json_test.c, src/proto/json.pro, src/channel.c, src/proto/channel.pro, src/eval.c, src/vim.h, src/structs.h, runtime/doc/eval.txt, runtime/doc/channel.txt, src/testdir/test_json.vim

Patch 7.4.1280

Problem: Missing case value.
Solution: Add VAR_JOB.
Files: src/if_python.c, src/if_python3.c

Patch 7.4.1281

Problem: No test for skipping over code that isn't evaluated.
Solution: Add a test with code that would fail when not skipped.
Files: src/testdir/test_viml.vim

Patch 7.4.1282

Problem: Crash when evaluating the pattern of ":catch" causes an error.
(Dominique Pelle)
Solution: Block error messages at this point.
Files: src/ex_eval.c

Patch 7.4.1283

Problem: The job feature isn't available on MS-Windows.
Solution: Add the job feature. Fix argument of job_stop(). (Yasuhiro
Matsumoto)
Files: src/eval.c, src/feature.h, src/os_win32.c, src/proto/os_win32.pro

Patch 7.4.1284 (after 7.4.1282)

Problem: Test 49 fails.
Solution: Check for a different error message.
Files: src/testdir/test49.vim

Patch 7.4.1285

Problem: Cannot measure elapsed time.
Solution: Add reltimefloat().
Files: src/ex_cmds2.c, src/eval.c, src/proto/ex_cmds2.pro,
src/testdir/test_reftime.vim, src/testdir/test_alot.vim

Patch 7.4.1286

Problem: ch_open() with a timeout doesn't work correctly.
Solution: Change how select() is used. Don't give an error on timeout.
Add a test for ch_open() failing.
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1287 (after 7.4.1286)

Problem: Channel test fails.
Solution: Use reltimefloat().
Files: src/testdir/test_channel.vim

Patch 7.4.1288

Problem: ch_sendexpr() does not use JS encoding.
Solution: Use the encoding that fits the channel mode. Refuse using
ch_sendexpr() on a raw channel.
Files: src/channel.c, src/proto/channel.pro, src/eval.c

Patch 7.4.1289

Problem: Channel test fails on MS-Windows, connect() takes too long.
Solution: Adjust the test for MS-Windows using "waittime".
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1290

Problem: Coverity complains about unnecessary check for NULL.
Solution: Remove the check.
Files: src/eval.c

Patch 7.4.1291

Problem: On MS-Windows the channel test server doesn't quit.
Solution: Use return instead of break. (Ken Takata)
Files: src/testdir/test_channel.py

Patch 7.4.1292

Problem: Some compilers complain about uninitialized variable, even though all possible cases are handled. (Dominique Pelle)
Solution: Add a default initialization.
Files: src/eval.c

Patch 7.4.1293

Problem: Sometimes a channel may hang waiting for a message that was already discarded. (Ken Takata)
Solution: Store the ID of the message blocking on in the channel.
Files: src/channel.c

Patch 7.4.1294

Problem: job_stop() only kills the started process.
Solution: Send the signal to the process group. (Olaf Dabrunz)
Files: src/os_unix.c

Patch 7.4.1295

Problem: string(job) doesn't work well on MS-Windows.
Solution: Use the process ID. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.1296

Problem: Cursor changes column with up motion when the matchparen plugin saves and restores the cursor position. (Martin Kunev)
Solution: Make sure curswant is updated before invoking the autocommand.
Files: src/edit.c

Patch 7.4.1297

Problem: On Mac test_channel leaves python instances running.
Solution: Use a small waittime to make ch_open() work. (Ozaki Kiichi)
Files: src/testdir/test_channel.vim

Patch 7.4.1298

Problem: When the channel test fails in an unexpected way the server keeps running.
Solution: Use try/catch. (Ozaki Kiichi)
Files: src/testdir/test_channel.vim

Patch 7.4.1299

Problem: When the server sends a message with ID zero the channel handler is not invoked. (Christian J. Robinson)
Solution: Recognize zero value for the request ID. Add a test for invoking the channel handler.
Files: src/channel.c, src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1300

Problem: Cannot test CursorMovedI because there is typeahead.
Solution: Add disable_char_avail_for_testing().
Files: src/eval.c, src/getchar.c, src/globals.h, src/testdir/test_cursor_func.vim, src/testdir/README.txt

Patch 7.4.1301

Problem: Missing options in ch_open().
Solution: Add s:chopt like in the other calls. (Ozaki Kiichi)
Files: src/testdir/test_channel.vim

Patch 7.4.1302

Problem: Typo in struct field name. (Ken Takata)
Solution: Rename jf_pi to jv_pi.
Files: src/eval.c, src/os_win32.c, src/structs.h

Patch 7.4.1303

Problem: A Funcref is not accepted as a callback.
Solution: Make a Funcref work. (Damien)
Files: src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1304

Problem: Function names are difficult to read.
Solution: Rename jsonencode to json_encode, jsondecode to json_decode, jsencode to js_encode and jsdecode to js_decode.
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test_json.vim

Patch 7.4.1305

Problem: "\\%1l^#.*" does not match on a line starting with "#".
Solution: Do not clear the start-of-line flag. (Christian Brabandt)
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test36.in, src/testdir/test36.ok

Patch 7.4.1306

Problem: Job control doesn't work well on MS-Windows.
Solution: Various fixes. (Ken Takata, Ozaki Kiichi, Yukihiro Nakadaira, Yasuhiro Matsumoto)
Files: src/Make_mvc.mak, src/eval.c, src/os_unix.c, src/os_win32.c, src/proto/os_unix.pro, src/proto/os_win32.pro, src/structs.h

Patch 7.4.1307

Problem: Some channel tests fail on MS-Windows.
Solution: Disable the failing tests temporarily.
Files: src/testdir/test_channel.vim

Patch 7.4.1308 (after 7.4.1307)

Problem: Typo in test.
Solution: Change endf to endif.
Files: src/testdir/test_channel.vim

Patch 7.4.1309

Problem: When a test fails not all relevant info is listed.
Solution: Add the errors to the messages.
Files: src/testdir/runtest.vim

Patch 7.4.1310

Problem: Jobs don't open a channel.
Solution: Create pipes and add them to the channel. Add ch_logfile(). Only Unix for now.
Files: src/channel.c, src/eval.c, src/os_unix.c, src/structs.h, src/gui_w48.c, src/proto/channel.pro, src/testdir/test_channel.vim,

src/testdir/test_channel_pipe.py, runtime/doc/eval.txt

Patch 7.4.1311 (after 7.4.1310)

Problem: sock_T is defined too late.

Solution: Move it up.

Files: src/vim.h

Patch 7.4.1312 (after 7.4.1311)

Problem: sock_T is not defined without the +channel feature.

Solution: Always define it.

Files: src/vim.h

Patch 7.4.1313

Problem: MS-Windows: Using socket after it was closed causes an exception.

Solution: Don't give an error when handling WM_NETBEANS. Re-enable tests for MS-Windows.

Files: src/gui_w48.c, src/testdir/test_channel.vim

Patch 7.4.1314

Problem: Warning for uninitialized variable.

Solution: Initialize it. (Dominique Pelle)

Files: src/channel.c

Patch 7.4.1315

Problem: Using a channel handle does not allow for freeing it when unused.

Solution: Add the Channel variable type.

Files: src/structs.h, src/channel.c, src/misc2.c, src/eval.c, src/if_python.c, src/if_python3.c, src/json.c, src/gui_w48.c, src/netbeans.c, src/proto/channel.pro, src/os_unix.c, src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1316

Problem: Can't build MS-Windows console version. (Tux)

Solution: Add #ifdefs.

Files: src/eval.c

Patch 7.4.1317

Problem: MS-Windows: channel test fails.

Solution: Temporarily disable Test_connect_waittime().

Files: src/testdir/test_channel.vim

Patch 7.4.1318

Problem: Channel with pipes doesn't work in GUI.

Solution: Register input handlers for pipes.

Files: src/structs.h, src/feature.h, src/channel.c, src/eval.c, src/os_unix.c, src/os_win32.c, src/gui_w48.c, src/proto/channel.pro

Patch 7.4.1319 (after 7.4.1318)

Problem: Tests fail on MS-Windows and on Unix with GUI.

Solution: Fix unregistering.

Files: src/structs.h, src/channel.c, src/os_unix.c, src/os_win32.c, src/proto/channel.pro

Patch 7.4.1320

Problem: Building with Cygwin or MingW with channel but without Netbeans doesn't work.
Solution: Set NETBEANS to "no" when not used.
Files: src/Make_cyg_ming.mak

Patch 7.4.1321

Problem: Compiler complains about missing statement.
Solution: Add an empty statement. (Andrei Olsen)
Files: src/os_win32.c

Patch 7.4.1322

Problem: Crash when unletting the variable that holds the channel in a callback function. (Christian Robinson)
Solution: Increase the reference count while invoking the callback.
Files: src/eval.c, src/channel.c, src/proto/eval.pro, src/testdir/test_channel.vim

Patch 7.4.1323

Problem: Do not get warnings when building with MingW.
Solution: Remove the -w flag. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.1324

Problem: Channels with pipes don't work on MS-Windows.
Solution: Add pipe I/O support. (Yasuhiro Matsumoto)
Files: src/channel.c, src/os_win32.c, src/proto/channel.pro, src/structs.h, src/vim.h, src/testdir/test_channel.vim

Patch 7.4.1325

Problem: Channel test fails on difference between Unix and DOS line endings.
Solution: Strip off CR. Make assert show difference better.
Files: src/eval.c, src/channel.c

Patch 7.4.1326

Problem: Build rules are bit too complicated.
Solution: Remove -lwsock32 from Netbeans, it's already added for the channel feature that it depends on. (Tony Mechelynck)
Files: src/Make_cyg_ming.mak

Patch 7.4.1327

Problem: Channel test doesn't work if Python executable is python.exe.
Solution: Find py.exe or python.exe. (Ken Takata)
Files: src/testdir/test_channel.vim

Patch 7.4.1328

Problem: Can't compile with +job but without +channel. (John Marriott)
Solution: Add more #ifdefs.
Files: src/os_unix.c

Patch 7.4.1329

Problem: Crash when using channel that failed to open.
Solution: Check for NULL. Update messages. (Yukihiro Nakadaira)
Files: src/channel.c, src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1330

Problem: fd_read() has an unused argument.
Solution: Remove the timeout. (Yasuhiro Matsumoto)
Files: src/channel.c

Patch 7.4.1331

Problem: Crash when closing the channel in a callback. (Christian J. Robinson)
Solution: Take the callback out of the list before invoking it.
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1332

Problem: Problem using Python3 when compiled with MingW.
Solution: Define PYTHON3_HOME as a wide character string. (Yasuhiro Matsumoto)
Files: src/Make_cyg_ming.mak

Patch 7.4.1333

Problem: Channel test fails on non-darwin builds.
Solution: Add the "osx" feature and test for that. (Kazunobu Kuriyama)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1334

Problem: Many compiler warnings with MingW.
Solution: Add type casts. (Yasuhiro Matsumoto)
Files: src/channel.c, src/dosinst.h, src/eval.c, src/ex_cmds2.c, src/ex_getln.c, src/fileio.c, src/if_cscope.c, src/if_perl.xs, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/main.c, src/mbyte.c, src/misc1.c, src/option.c, src/os_mswin.c, src/os_win32.c

Patch 7.4.1335

Problem: Can't build on MS-Windows with +job but without +channel. (Cesar Romani)
Solution: Add #ifdefs. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.1336

Problem: Channel NL mode is not supported yet.
Solution: Add NL mode support to channels.
Files: src/channel.c, src/netbeans.c, src/structs.h, src/os_win32.c, src/proto/channel.pro, src/proto/os_unix.pro, src/proto/os_win32.pro, src/testdir/test_channel.vim, src/testdir/test_channel_pipe.py

Patch 7.4.1337 (after 7.4.1336)

Problem: Part of the change is missing.
Solution: Add changes to eval.c
Files: src/eval.c

Patch 7.4.1338 (after 7.4.1336)

Problem: Another part of the change is missing.
Solution: Type os_unix.c right this time.

Files: src/os_unix.c

Patch 7.4.1339

Problem: Warnings when building the GUI with MingW. (Cesar Romani)

Solution: Add type casts. (Yasuhiro Matsumoto)

Files: src/edit.c, src/gui_w32.c, src/gui_w48.c, src/os_mswin.c,
src/os_win32.c

Patch 7.4.1340 (after 7.4.1339)

Problem: Merge left extra #endif behind.

Solution: Remove the #endif

Files: src/os_win32.c

Patch 7.4.1341

Problem: It's difficult to add more arguments to ch_senddraw() and
ch_sendexpr().

Solution: Make the third option a dictionary.

Files: src/eval.c, src/structs.h, src/channel.c, src/os_unix.c,
src/os_win32.c, src/proto/channel.pro,
src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 7.4.1342

Problem: On Mac OS/X the waittime must be > 0 for connect to work.

Solution: Use select() in a different way. (partly by Kazunobu Kuriyama)
Always use a waittime of 1 or more.

Files: src/eval.c, src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1343

Problem: Can't compile with +job but without +channel. (Andrei Olsen)

Solution: Move get_job_options up and adjust #ifdef.

Files: src/eval.c

Patch 7.4.1344

Problem: Can't compile Win32 GUI with tiny features.

Solution: Add #ifdef. (Christian Brabandt)

Files: src/gui_w32.c

Patch 7.4.1345

Problem: A few more compiler warnings. (Axel Bender)

Solution: Add type casts.

Files: src/gui_w32.c, src/gui_w48.c

Patch 7.4.1346

Problem: Compiler warnings in build with -O2.

Solution: Add initializations.

Files: src/eval.c

Patch 7.4.1347

Problem: When there is any error Vim will use a non-zero exit code.

Solution: When using ":silent!" do not set the exit code. (Yasuhiro
Matsumoto)

Files: src/message.c

Patch 7.4.1348

Problem: More compiler warnings. (John Marriott)
Solution: Add type casts, remove unused variable.
Files: src/gui_w32.c

Patch 7.4.1349

Problem: And some more MingW compiler warnings. (Cesar Romani)
Solution: Add type casts.
Files: src/if_mzsch.c

Patch 7.4.1350

Problem: When the test server fails to start Vim hangs.
Solution: Check that there is actually something to read from the tty fd.
Files: src/os_unix.c

Patch 7.4.1351

Problem: When the port isn't opened yet when ch_open() is called it may fail instead of waiting for the specified time.
Solution: Loop when select() succeeds but when connect() failed. Also use channel logging for jobs. Add ch_log().
Files: src/channel.c, src/eval.c, src/netbeans.c, src/proto/channel.pro, src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1352

Problem: The test script lists all functions before executing them.
Solution: Only list the function currently being executed.
Files: src/testdir/runtest.vim

Patch 7.4.1353

Problem: Test_connect_waittime is skipped for MS-Windows.
Solution: Add the test back, it works now.
Files: src/testdir/test_channel.vim

Patch 7.4.1354

Problem: MS-Windows: Mismatch between default compile options and what the code expects.
Solution: Change the default WINVER from 0x0500 to 0x0501. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 7.4.1355

Problem: Win32 console and GUI handle channels differently.
Solution: Consolidate code between Win32 console and GUI.
Files: src/channel.c, src/eval.c, src/gui_w48.c, src/os_win32.c, src/proto/channel.pro

Patch 7.4.1356

Problem: Job and channel options parsing is scattered.
Solution: Move all option value parsing to get_job_options();
Files: src/channel.c, src/eval.c, src/structs.h, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 7.4.1357 (after 7.4.1356)

Problem: Error for returning value from void function.
Solution: Don't do that.
Files: src/eval.c

Patch 7.4.1358

Problem: Compiler warning when not building with +crypt.
Solution: Add #ifdef. (John Marriott)
Files: src/undo.c

Patch 7.4.1359 (after 7.4.1356)

Problem: Channel test ch_sendexpr() times out.
Solution: Increase the timeout
Files: src/testdir/test_channel.vim

Patch 7.4.1360

Problem: Can't remove a callback with ch_setoptions().
Solution: When passing zero or an empty string remove the callback.
Files: src/channel.c, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 7.4.1361

Problem: Channel test fails on Solaris.
Solution: Use the 1 msec waittime for all systems.
Files: src/channel.c

Patch 7.4.1362 (after 7.4.1356)

Problem: Using uninitialized value.
Solution: Initialize jo_set.
Files: src/eval.c

Patch 7.4.1363

Problem: Compiler warnings with tiny build.
Solution: Add #ifdefs.
Files: src/gui_w48.c, src/gui_w32.c

Patch 7.4.1364

Problem: The Win 16 code is not maintained and unused.
Solution: Remove the Win 16 support.
Files: src/gui_w16.c, src/gui_w32.c, src/gui_w48.c, src/Make_w16.mak, src/Makefile, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/proto/gui_w16.pro, src/proto/os_win16.pro, src/guiw16rc.h, src/vim16.rc, src/vim16.def, src/tools16.bmp, src/eval.c, src/gui.c, src/misc2.c, src/option.c, src/os_msdos.c, src/os_mswin.c, src/os_win16.c, src/os_win16.h, src/version.c, src/winclip.c, src/feature.h, src/proto.h, src/vim.h, Filelist

Patch 7.4.1365

Problem: Cannot execute a single test function.
Solution: Add an argument to filter the functions with. (Yasuhiro Matsumoto)
Files: src/testdir/runtest.vim

Patch 7.4.1366

Problem: Typo in test and resulting error in test result.
Solution: Fix the typo and correct the result. (James McCoy, closes #650)
Files: src/testdir/test_charsearch.in, src/testdir/test_charsearch.ok

Patch 7.4.1367

Problem: Compiler warning for unreachable code.

Solution: Remove a "break". (Danek Duvall)
Files: src/json.c

Patch 7.4.1368

Problem: One more Win16 file remains.
Solution: Delete it.
Files: src/proto/os_win16.pro

Patch 7.4.1369

Problem: Channels don't have a queue for stderr.
Solution: Have a queue for each part of the channel.
Files: src/channel.c, src/eval.c, src/structs.h, src/netbeans.c,
src/gui_w32.c, src/proto/channel.pro

Patch 7.4.1370

Problem: The Python test script may keep on running.
Solution: Join the threads. (Yasuhiro Matsumoto)
Files: src/testdir/test_channel.py

Patch 7.4.1371

Problem: X11 GUI callbacks don't specify the part of the channel.
Solution: Pass the fd instead of the channel ID.
Files: src/channel.c

Patch 7.4.1372

Problem: channel read implementation is incomplete.
Solution: Add ch_read() and options for ch_readraw().
Files: src/channel.c, src/eval.c, src/structs.h, src/proto/channel.pro,
src/testdir/test_channel.vim

Patch 7.4.1373

Problem: Calling a Vim function over a channel requires turning the
arguments into a string.
Solution: Add the "call" command. (Damien) Also merge "expr" and "eval"
into one.
Files: src/channel.c, src/testdir/test_channel.py,
src/testdir/test_channel.vim

Patch 7.4.1374

Problem: Channel test hangs on MS-Windows.
Solution: Disable the ch_read() that is supposed to time out.
Files: src/testdir/test_channel.vim

Patch 7.4.1375

Problem: Still some Win16 code.
Solution: Remove FEAT_GUI_W16. (Hirohito Higashi)
Files: src/eval.c, src/ex_cmds.h, src/feature.h, src/gui.h, src/menu.c,
src/misc1.c, src/option.c, src/proto.h, src/structs.h, src/term.c,
src/vim.h, runtime/doc/gui_w16.txt

Patch 7.4.1376

Problem: ch_setoptions() cannot set all options.
Solution: Support more options.
Files: src/channel.c, src/eval.c, src/structs.h, runtime/doc/channel.txt,

src/testdir/test_channel.vim

Patch 7.4.1377

Problem: Test_connect_waittime() is flaky.
Solution: Ignore the "Connection reset by peer" error.
Files: src/testdir/test_channel.vim

Patch 7.4.1378

Problem: Can't change job settings after it started.
Solution: Add job_setoptions() with the "stoponexit" flag.
Files: src/eval.c, src/main.c, src/structs.h, src/proto/eval.pro,
src/testdir/test_channel.vim

Patch 7.4.1379

Problem: Channel test fails on Win32 console.
Solution: Don't sleep when timeout is zero. Call channel_wait() before
channel_read(). Channels are not polled during ":sleep". (Yukihiro
Nakadaira)
Files: src/channel.c, src/misc2.c, src/gui_w32.c, src/os_win32.c

Patch 7.4.1380

Problem: The job exit callback is not implemented.
Solution: Add the "exit-cb" option.
Files: src/structs.h, src/eval.c, src/channel.c, src/proto/eval.pro,
src/misc2.c, src/macros.h, src/testdir/test_channel.vim

Patch 7.4.1381 (after 7.4.1380)

Problem: Exit value not available on MS-Windows.
Solution: Set the exit value.
Files: src/structs.h, src/os_win32.c

Patch 7.4.1382

Problem: Can't get the job of a channel.
Solution: Add ch_getjob().
Files: src/eval.c, runtime/doc/channel.txt, runtime/doc/eval.txt

Patch 7.4.1383

Problem: GvimExt only loads the old libintl.dll.
Solution: Also try loading libintl-8.dll. (Ken Takata, closes #608)
Files: src/GvimExt/gvimext.cpp, src/GvimExt/gvimext.h

Patch 7.4.1384

Problem: It is not easy to use a set of plugins and their dependencies.
Solution: Add packages, ":loadplugin", '**packpath**'.
Files: src/main.c, src/ex_cmds2.c, src/option.c, src/option.h,
src/ex_cmds.h, src/eval.c, src/version.c, src/proto/ex_cmds2.pro,
runtime/doc/repeat.txt, runtime/doc/options.txt,
runtime/optwin.vim

Patch 7.4.1385

Problem: Compiler warning for using array.
Solution: Use the right member name. (Yegappan Lakshmanan)
Files: src/eval.c

Patch 7.4.1386

Problem: When the Job exit callback is invoked, the job may be freed too soon. (Yasuhiro Matsumoto)
Solution: Increase refcount.
Files: src/eval.c

Patch 7.4.1387

Problem: Win16 docs still referenced.
Solution: Remove Win16 files from the docs Makefile. (Kenichi Ito)
Files: runtime/doc/Makefile

Patch 7.4.1388

Problem: Compiler warning. (Cesar Romani)
Solution: Initialize variable.
Files: src/ex_cmds2.c

Patch 7.4.1389

Problem: Incomplete function declaration.
Solution: Add "void". (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.1390

Problem: When building with GTK and glib-compile-resources cannot be found building Vim fails. (Michael Gehring)
Solution: Make GLIB_COMPILE_RESOURCES empty instead of leaving it at "no". (nuko8, closes #655)
Files: src/configure.in, src/auto/configure

Patch 7.4.1391

Problem: Warning for uninitialized variable.
Solution: Set it to zero. (Christian Brabandt)
Files: src/eval.c

Patch 7.4.1392

Problem: Some tests fail for Win32 console version.
Solution: Move the tests to SCRIPTS_MORE2. Pass VIMRUNTIME. (Christian Brabandt)
Files: src/testdir/Make_all.mak

Patch 7.4.1393

Problem: Starting a job hangs in the GUI. (Takuya Fujiwara)
Solution: Don't check if ch_job is NULL when checking for an error. (Yasuhiro Matsumoto)
Files: src/channel.c

Patch 7.4.1394

Problem: Can't sort inside a sort function.
Solution: Use a struct to store the sort parameters. (Jacob Niehus)
Files: src/eval.c, src/testdir/test_sort.vim

Patch 7.4.1395

Problem: Using DETACH in quotes is not compatible with the Netbeans interface. (Xavier de Gaye)
Solution: Remove the quotes, only use them for JSON and JS mode.

Files: src/netbeans.c, src/channel.c

Patch 7.4.1396

Problem: Compiler warnings for conversions.

Solution: Add type cast.

Files: src/ex_cmds2.c

Patch 7.4.1397

Problem: Sort test fails on MS-Windows.

Solution: Correct the compare function.

Files: src/testdir/test_sort.vim

Patch 7.4.1398

Problem: The close-cb option is not implemented yet.

Solution: Implement close-cb. (Yasuhiro Matsumoto)

Files: src/channel.c, src/eval.c, src/structs.h, src/proto/channel.pro, src/testdir/test_channel.py, src/testdir/test_channel.vim

Patch 7.4.1399

Problem: The MS-DOS code does not build.

Solution: Remove the old MS-DOS code.

Files: Filelist, src/Make_bc3.mak, src/Make_bc5.mak, src/Make_djg.mak, src/Makefile, src/blowfish.c, src/buffer.c, src/diff.c, src/digraph.c, src/dosinst.h, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/feature.h, src/fileio.c, src/getchar.c, src/globals.h, src/macros.h, src/main.c, src/mbyte.c, src/memfile.c, src/memline.c, src/misc1.c, src/misc2.c, src/netbeans.c, src/option.c, src/option.h, src/os_msdos.c, src/os_msdos.h, src/proto.h, src/proto/os_msdos.pro, src/regexp.c, src/screen.c, src/structs.h, src/syntax.c, src/term.c, src/undo.c, src/uninstal.c, src/version.c, src/vim.h, src/window.c, src/xxd/Make_bc3.mak, src/xxd/Make_djg.mak

Patch 7.4.1400

Problem: Perl eval doesn't work properly on 64-bit big-endian machine.

Solution: Use 32 bit type for the key. (Danek Duvall)

Files: src/if_perl.xs

Patch 7.4.1401

Problem: Having **'autochdir'** set during startup and using diff mode doesn't work. (Axel Bender)

Solution: Don't use **'autochdir'** while still starting up. (Christian Brabandt)

Files: src/buffer.c

Patch 7.4.1402

Problem: GTK 3 is not supported.

Solution: Add GTK 3 support. (Kazunobu Kuriyama)

Files: runtime/doc/eval.txt, runtime/doc/gui.txt, runtime/doc/gui_x11.txt, src/auto/configure, src/channel.c, src/config.h.in, src/configure.in, src/eval.c, src/gui.h, src/gui_beval.c, src/gui_beval.h, src/gui_gtk.c, src/gui_gtk_f.c,

src/gui_gtk_f.h, src/gui_gtk_x11.c, src/if_mzsch.c, src/mbyte.c,
src/netbeans.c, src/structs.h, src/version.c

Patch 7.4.1403

Problem: Can't build without the quickfix feature.
Solution: Add #ifdefs. Call ex_ni() for unimplemented commands. (Yegappan Lakshmanan)
Files: src/ex_cmds2.c, src/popupmnu.c

Patch 7.4.1404

Problem: ch_read() doesn't time out on MS-Windows.
Solution: Instead of WM_NETBEANS use select(). (Yukihiro Nakadaira)
Files: src/channel.c, src/gui_w32.c, src/os_win32.c, src/structs.h,
src/testdir/test_channel.vim, src/vim.h

Patch 7.4.1405

Problem: Completion menu flickers.
Solution: Delay showing the popup menu. (Shougo Matsu, Justin M. Keyes, closes #656)
Files: src/edit.c

Patch 7.4.1406

Problem: Leaking memory in cs_print_tags_priv().
Solution: Free tbuf. (idea by Forrest Fleming)
Files: src/if_cscope.c

Patch 7.4.1407

Problem: json_encode() does not handle NaN and inf properly. (David Barnett)
Solution: For JSON turn them into "null". For JS use "NaN" and "Infinity". Add isnan().
Files: src/eval.c, src/json.c, src/testdir/test_json.vim

Patch 7.4.1408

Problem: MS-Windows doesn't have isnan() and isinf().
Solution: Use _isnan() and _isinf().
Files: src/eval.c, src/json.c

Patch 7.4.1409 (after 7.4.1402)

Problem: Configure includes GUI despite --disable-gui flag.
Solution: Add SKIP_GTK3. (Kazunobu Kuriyama)
Files: src/configure.in, src/auto/configure

Patch 7.4.1410

Problem: Leaking memory in cscope interface.
Solution: Free memory when no tab is found. (Christian Brabandt)
Files: src/if_cscope.c

Patch 7.4.1411

Problem: Compiler warning for indent. (Ajit Thakkar)
Solution: Indent normally.
Files: src/ui.c

Patch 7.4.1412

Problem: Compiler warning for indent. (Dominique Pelle)
Solution: Fix the indent.
Files: src/farsi.c

Patch 7.4.1413

Problem: When calling ch_close() the close callback is invoked, even though the docs say it isn't. (Christian J. Robinson)
Solution: Don't call the close callback.
Files: src/eval.c, src/channel.c, src/netbeans.c, src/proto/channel.pro

Patch 7.4.1414

Problem: Appveyor only builds one feature set.
Solution: Build a combination of features and GUI/console. (Christian Brabandt)
Files: appveyor.yml, src/appveyor.bat

Patch 7.4.1415 (after 7.4.1414)

Problem: Dropped the skip-tags setting.
Solution: Put it back.
Files: appveyor.yml

Patch 7.4.1416

Problem: Using "u_char" instead of "char_u", which doesn't work everywhere. (Jörg Plate)
Solution: Use "char_u" always.
Files: src/integration.c, src/macros.h

Patch 7.4.1417 (after 7.4.1414)

Problem: Missing appveyor.bat from the distribution.
Solution: Add it to the list of files.
Files: Filelist

Patch 7.4.1418

Problem: job_stop() on MS-Windows does not really stop the job.
Solution: Make the default to stop the job forcefully. (Ken Takata)
Make MS-Windows and Unix more similar.
Files: src/os_win32.c, src/os_unix.c, runtime/doc/eval.txt

Patch 7.4.1419

Problem: Tests slowed down because of the "not a terminal" warning.
Solution: Add the --not-a-term command line argument.
Files: src/main.c, src/testdir/Makefile, src/Make_all.mak, src/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, runtime/doc/starting.txt

Patch 7.4.1420 (after 7.4.1419)

Problem: Missing makefile.
Solution: Type the path correctly.
Files: src/testdir/Make_all.mak

Patch 7.4.1421

Problem: May free a channel when a callback may need to be invoked.
Solution: Keep the channel when refcount is zero.

Files: src/eval.c, src/channel.c, src/proto/channel.pro

Patch 7.4.1422

Problem: Error when reading fails uses wrong errno. Keeping channel open after job stops results in test failing.

Solution: Move the error up. Add ch_job_killed.

Files: src/channel.c, src/eval.c, src/structs.h

Patch 7.4.1423

Problem: Channel test fails on MS-Windows.

Solution: Do not give an error message when reading fails, assume the other end exited.

Files: src/channel.c

Patch 7.4.1424

Problem: Not using --not-a-term when running tests on MS-Windows.

Solution: Use NO_PLUGIN. (Christian Brabandt)

Files: src/testdir/Make_dos.mak

Patch 7.4.1425

Problem: There are still references to MS-DOS support.

Solution: Remove most of the help txt and install instructions. (Ken Takata)

Files: src/INSTALLpc.txt, runtime/doc/os_msdos.txt, csdpmi4b.zip, Filelist

Patch 7.4.1426

Problem: The "out-io" option for jobs is not implemented yet.

Solution: Implement the "buffer" value: append job output to a buffer.

Files: src/eval.c, src/channel.c, src/structs.h, src/netbeans.c, runtime/doc/channel.txt

Patch 7.4.1427

Problem: Trailing comma in enums is not ANSI C.

Solution: Remove the trailing commas.

Files: src/alloc.h, src/gui_mac.c

Patch 7.4.1428

Problem: Compiler warning for non-virtual destructor.

Solution: Make it virtual. (Yasuhiro Matsumoto)

Files: src/gui_dwrite.cpp

Patch 7.4.1429

Problem: On MS-Windows, when not use renderoptions=type:directx, drawing emoji will be broken.

Solution: Fix usage of unicodepdy. (Yasuhiro Matsumoto)

Files: src/gui_w32.c

Patch 7.4.1430

Problem: When encoding JSON, turning NaN and Infinity into null without giving an error is not useful.

Solution: Pass NaN and Infinity on. If the receiver can't handle them it will generate the error.

Files: src/json.c, src/testdir/test_json.vim, runtime/doc/eval.txt

Patch 7.4.1431

Problem: Including header files twice.
Solution: Remove the extra includes.
Files: src/if_csscope.h

Patch 7.4.1432

Problem: Typo in button text.
Solution: Fix the typo. (Dominique Pelle)
Files: src/gui_gtk.c

Patch 7.4.1433

Problem: The Sniff interface is no longer useful, the tool has not been available for many years.
Solution: Delete the Sniff interface and related code.
Files: src/if_sniff.c, src/if_sniff.h, src/charset.c, src/edit.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/gui_gtk_x11.c, src/gui_w32.c, src/gui_x11.c, src/normal.c, src/os_unix.c, src/os_win32.c, src/term.c, src/ui.c, src/version.c, src/ex_cmds.h, src/feature.h, src/keymap.h, src/structs.h, src/vim.h, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/configure.in, src/auto/configure, src/config.h.in, src/config.mk.in, runtime/doc/if_sniff.txt, src/config.aap.in, src/main.aap

Patch 7.4.1434

Problem: JSON encoding doesn't handle surrogate pair.
Solution: Improve multibyte handling of JSON. (Yasuhiro Matsumoto)
Files: src/json.c, src/testdir/test_json.vim

Patch 7.4.1435

Problem: It is confusing that ch_sendexpr() and ch_sendraw() wait for a response.
Solution: Add ch_evalexpr() and ch_evalraw().
Files: src/eval.c, runtime/doc/channel.txt, runtime/doc/eval.txt, src/testdir/test_channel.vim

Patch 7.4.1436 (after 7.4.1433)

Problem: Sniff files still referenced in distribution.
Solution: Remove sniff files from distribution.
Files: Filelist

Patch 7.4.1437

Problem: Old system doesn't have isinf() and NAN. (Ben Fritz)
Solution: Adjust #ifdefs. Detect isnan() and isinf() functions with configure. Use a replacement when missing. (Kazunobu Kuriyama)
Files: src/eval.c, src/json.c, src/macros.h, src/message.c, src/config.h.in, src/configure.in, src/auto/configure

Patch 7.4.1438

Problem: Can't get buffer number of a channel.
Solution: Add ch_getbufnr().
Files: src/eval.c, src/channel.c, src/testdir/test_channel.vim, runtime/doc/channel.txt, runtime/doc/eval.txt

Patch 7.4.1439 (after 7.4.1434)

Problem: Using uninitialized variable.

Solution: Initialize vc_type.

Files: src/json.c

Patch 7.4.1440 (after 7.4.1437)

Problem: Can't build on Windows.

Solution: Change #ifdefs. Only define isnan when used.

Files: src/macros.h, src/eval.c, src/json.c

Patch 7.4.1441

Problem: Using empty name instead of no name for channel buffer.

Solution: Remove the empty name.

Files: src/channel.c

Patch 7.4.1442

Problem: MS-Windows: more compilation warnings for destructor.

Solution: Add "virtual". (Ken Takata)

Files: src/if_ole.cpp

Patch 7.4.1443

Problem: Can't build GTK3 with small features.

Solution: Use gtk_widget_get_window(). Fix typos. (Dominique Pelle)

Files: src/gui_gtk_x11.c

Patch 7.4.1444

Problem: Can't build with JSON but without multibyte.

Solution: Fix pointer name.

Files: src/json.c

Patch 7.4.1445

Problem: Memory corruption when 'encoding' is not utf-8.

Solution: Convert decoded string later.

Files: src/json.c

Patch 7.4.1446

Problem: Crash when using json_decode().

Solution: Terminate string with a NUL byte.

Files: src/json.c

Patch 7.4.1447

Problem: Memory leak when using ch_read(). (Dominique Pelle)

No log message when stopping a job and a few other situations.

Too many "Nothing to read" messages. Channels are not freed.

Solution: Free the listtv. Add more log messages. Remove "Nothing to read" message. Remove the channel from the job when its refcount becomes zero.

Files: src/eval.c, src/channel.c

Patch 7.4.1448

Problem: JSON tests fail if 'encoding' is not utf-8.

Solution: Force encoding to utf-8.

Files: src/testdir/test_json.vim

Patch 7.4.1449

Problem: Build fails with job feature but without channel feature.
Solution: Add #ifdef.
Files: src/eval.c

Patch 7.4.1450

Problem: Json encoding still fails when encoding is not utf-8.
Solution: Set '**encoding**' before :scriptencoding. Run the json test separately to avoid affecting other tests.
Files: src/testdir/test_json.vim, src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 7.4.1451

Problem: Vim hangs when a channel has a callback but isn't referenced.
Solution: Have channel_unref() only return TRUE when the channel was actually freed.
Files: src/eval.c, src/channel.c, src/proto/channel.pro

Patch 7.4.1452

Problem: When a callback adds a syntax item either the redraw doesn't happen right away or in the GUI the cursor is in the wrong position for a moment. (Jakson Alves de Aquino)
Solution: Redraw after the callback was invoked.
Files: src/channel.c

Patch 7.4.1453

Problem: Missing --not-a-term.
Solution: Add the argument.
Files: src/testdir/Make_amiga.mak

Patch 7.4.1454

Problem: The exit callback test is flaky.
Solution: Loop to wait for a short time up to a second.
Files: src/testdir/test_channel.vim

Patch 7.4.1455

Problem: JSON decoding test for surrogate pairs is in the wrong place.
Solution: Move the test lines. (Ken Takata)
Files: src/testdir/test_json.vim

Patch 7.4.1456

Problem: Test 87 fails with Python 3.5.
Solution: Work around difference. (Taro Muraoka)
Files: src/testdir/test87.in

Patch 7.4.1457

Problem: Opening a channel with select() is not done properly.
Solution: Also used read-fds. Use getsockopt() to check for errors. (Ozaki Kiichi)
Files: src/channel.c

Patch 7.4.1458

Problem: When a JSON channel has a callback it may never be cleared.
Solution: Do not write "DETACH" into a JS or JSON channel.

Files: src/channel.c

Patch 7.4.1459 (after 7.4.1457)

Problem: MS-Windows doesn't know socklen_t.

Solution: Use previous method for WIN32.

Files: src/channel.c

Patch 7.4.1460

Problem: Syntax error in rarely used code.

Solution: Fix the mch_rename() declaration. (Ken Takata)

Files: src/os_unix.c, src/proto/os_unix.pro

Patch 7.4.1461

Problem: When starting job on MS-Windows all parts of the command are put in quotes.

Solution: Only use quotes when needed. (Yasuhiro Matsumoto)

Files: src/eval.c

Patch 7.4.1462

Problem: Two more rarely used functions with errors.

Solution: Add proper argument types. (Dominique Pelle)

Files: src/misc2.c, src/termlib.c

Patch 7.4.1463

Problem: Configure doesn't find isinf() and isnan() on some systems.

Solution: Use a configure check that includes math.h.

Files: src/configure.in, src/auto/configure

Patch 7.4.1464

Problem: When the argument of sort() is zero or empty it fails.

Solution: Make zero work as documented. (suggested by Yasuhiro Matsumoto)

Files: src/eval.c, src/testdir/test_sort.vim

Patch 7.4.1465

Problem: Coverity reported possible use of NULL pointer when using buffer output with JSON mode.

Solution: Make it actually possible to use JSON mode with a buffer. Re-encode the JSON to append it to the buffer.

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1466

Problem: Coverity reports dead code.

Solution: Remove the two lines.

Files: src/channel.c

Patch 7.4.1467

Problem: Can't build without the float feature.

Solution: Add #ifdefs. (Nick Owens, closes #667)

Files: src/eval.c, src/json.c

Patch 7.4.1468

Problem: Sort test doesn't test with "1" argument.

Solution: Also test ignore-case sorting. (Yasuhiro Matsumoto)

Files: src/testdir/test_sort.vim

Patch 7.4.1469

Problem: Channel test sometimes fails, especially on OS/X. (Kazunobu Kuriyama)
Solution: Change the && into ||, call getsockopt() in more situations. (Ozaki Kiichi)
Files: src/channel.c

Patch 7.4.1470

Problem: Coverity reports missing restore.
Solution: Move json_encode() call up.
Files: src/channel.c

Patch 7.4.1471

Problem: Missing out-of-memory check. And Coverity warning.
Solution: Bail out when msg is NULL.
Files: src/channel.c

Patch 7.4.1472

Problem: Coverity warning for not using return value.
Solution: Add "(void)".
Files: src/os_unix.c

Patch 7.4.1473

Problem: Can't build without the autocommand feature.
Solution: Add #ifdefs. (Yegappan Lakshmanan)
Files: src/edit.c, src/main.c, src/syntax.c

Patch 7.4.1474

Problem: Compiler warnings without the float feature.
Solution: Move #ifdefs. (John Marriott)
Files: src/eval.c

Patch 7.4.1475

Problem: When using hangulininput with utf-8 a CSI character is misinterpreted.
Solution: Convert CSI to K_CSI. (SungHyun Nam)
Files: src/ui.c

Patch 7.4.1476

Problem: Function arguments marked as unused while they are not.
Solution: Remove UNUSED. (Yegappan Lakshmanan)
Files: src/diff.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/window.c

Patch 7.4.1477

Problem: Test_reftime is flaky, it depends on timing.
Solution: When it fails run it a second time.
Files: src/testdir/runtest.vim

Patch 7.4.1478

Problem: ":loadplugin" doesn't take care of ftdetect files.
Solution: Also load ftdetect scripts when appropriate.
Files: src/ex_cmds2.c

Patch 7.4.1479

Problem: No testfor ":loadplugin".
Solution: Add a test. Fix how option is being set.
Files: src/ex_cmds2.c, src/testdir/test_loadplugin.vim,
src/testdir/Make_all.mak

Patch 7.4.1480

Problem: Cannot add a pack directory without loading a plugin.
Solution: Add the :packadd command.
Files: src/ex_cmds.h, src/ex_cmds2.c, src/proto/ex_cmds2.pro,
src/testdir/test_loadplugin.vim, runtime/doc/repeat.txt

Patch 7.4.1481

Problem: Can't build with small features.
Solution: Add #ifdef.
Files: src/ex_cmds2.c

Patch 7.4.1482

Problem: "timeout" option not supported on ch_eval*().
Solution: Get and use the timeout option from the argument.
Files: src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1483

Problem: A one-time callback is not used for a raw channel.
Solution: Use a one-time callback when it exists.
Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel.py

Patch 7.4.1484

Problem: Channel "err-io" value "out" is not supported.
Solution: Connect stderr to stdout if wanted.
Files: src/os_unix.c, src/os_win32.c, src/testdir/test_channel.vim,
src/testdir/test_channel_pipe.py

Patch 7.4.1485

Problem: Job input from buffer is not implemented.
Solution: Implement it. Add "in-top" and "in-bot" options.
Files: src/structs.h, src/eval.c, src/channel.c, src/proto/channel.pro,
src/os_unix.c, src/os_win32.c, src/testdir/test_channel.vim

Patch 7.4.1486

Problem: ":loadplugin" is not optimal, some people find it confusing.
Solution: Only use ":packadd" with an optional "!".
Files: src/ex_cmds.h, src/ex_cmds2.c, src/testdir/test_loadplugin.vim,
src/testdir/test_packadd.vim, src/testdir/Make_all.mak,
runtime/doc/repeat.txt

Patch 7.4.1487

Problem: For WIN32 isinf() is defined as a macro.
Solution: Define it as an inline function. (ZyX)
Files: src/macros.h

Patch 7.4.1488 (after 7.4.1475)

Problem: Not using key when result from `hangul_string_convert()` is NULL.
Solution: Fall back to not converted string.
Files: `src/ui.c`

Patch 7.4.1489 (after 7.4.1487)

Problem: "inline" is not supported by old MSVC.
Solution: use "`__inline`". (Ken Takata)
Files: `src/macros.h`

Patch 7.4.1490

Problem: Compiler warning for unused function.
Solution: Add `#ifdef`. (Dominique Pelle)
Files: `src/gui_gtk_x11.c`

Patch 7.4.1491

Problem: Visual-block shift breaks multibyte characters.
Solution: Compute column differently. (Yasuhiro Matsumoto) Add a test.
Files: `src/ops.c`, `src/testdir/test_visual.vim`, `src/testdir/Make_all.mak`

Patch 7.4.1492

Problem: No command line completion for `":packadd"`.
Solution: Implement completion. (Hirohito Higashi)
Files: `src/ex_docmd.c`, `src/ex_getln.c`, `src/testdir/test_packadd.vim`,
`src/vim.h`

Patch 7.4.1493

Problem: Wrong callback invoked for zero-id messages.
Solution: Don't use the first one-time callback when the sequence number
doesn't match.
Files: `src/channel.c`, `src/testdir/test_channel.vim`,
`src/testdir/test_channel.py`

Patch 7.4.1494

Problem: `clr_history()` does not work properly.
Solution: Increment `hisptr`. Add a test. (Yegappan Lakshmanan)
Files: `src/ex_getln.c`, `src/testdir/test_history.vim`,
`src/testdir/Make_all.mak`

Patch 7.4.1495

Problem: Compiler warnings when building on Unix with the job feature but
without the channel feature.
Solution: Move `#ifdefs`. (Dominique Pelle)
Files: `src/os_unix.c`

Patch 7.4.1496

Problem: Crash when built with GUI but it's not active. (Dominique Pelle)
Solution: Check `gui.in_use`.
Files: `src/channel.c`

Patch 7.4.1497

Problem: Cursor drawing problem with GTK 3.
Solution: Handle blinking differently. (Kazunobu Kuriyama)
Files: `src/gui_gtk_x11.c`

Patch 7.4.1498

Problem: Error for locked item when using json_decode(). (Shougo Matsu)
Solution: Initialize v_lock.
Files: src/json.c

Patch 7.4.1499

Problem: No error message when :packadd does not find anything.
Solution: Add an error message. (Hirohito Higashi)
Files: runtime/doc/repeat.txt, src/ex_cmds.h, src/ex_cmds2.c,
src/globals.h, src/testdir/test_packadd.vim

Patch 7.4.1500

Problem: Should_free flag set to FALSE.
Solution: Set it to TRUE. (Neovim 4415)
Files: src/ex_eval.c

Patch 7.4.1501

Problem: Garbage collection with an open channel is not tested.
Solution: Call garbagecollect() in the test.
Files: src/testdir/test_channel.vim

Patch 7.4.1502

Problem: Writing last-but-one line of buffer to a channel isn't implemented yet.
Solution: Implement it. Fix leaving a swap file behind.
Files: src/channel.c, src/structs.h, src/memline.c, src/proto/channel.pro

Patch 7.4.1503

Problem: Crash when using ch_getjob(). (Damien)
Solution: Check for a NULL job.
Files: src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1504 (after 7.4.1502)

Problem: No test for reading last-but-one line.
Solution: Add a test.
Files: src/testdir/test_channel.vim

Patch 7.4.1505

Problem: When channel log is enabled get too many "looking for messages" log entries.
Solution: Only give the message after another message.
Files: src/channel.c

Patch 7.4.1506

Problem: Job cannot read from a file.
Solution: Implement reading from a file for Unix.
Files: src/eval.c, src/os_unix.c, src/os_win32.c,
src/testdir/test_channel.vim

Patch 7.4.1507

Problem: Crash when starting a job fails.
Solution: Check for the channel to be NULL. (idea by Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.1508

Problem: Can't build GvimExt with MingW.
Solution: Adjust the makefile. (Ben Fritz)
Files: src/GvimExt/Make_ming.mak

Patch 7.4.1509

Problem: Keeping both a variable for a job and the channel it refers to is a hassle.
Solution: Allow passing the job where a channel is expected. (Damien)
Files: src/eval.c, src/testdir/test_channel.vim

Patch 7.4.1510

Problem: Channel test fails on AppVeyor.
Solution: Wait longer than 10 msec if needed.
Files: src/testdir/test_channel.vim

Patch 7.4.1511

Problem: Statusline highlighting is sometimes wrong.
Solution: Check for Highlight type. (Christian Brabandt)
Files: src/buffer.c

Patch 7.4.1512

Problem: Channel input from file not supported on MS-Windows.
Solution: Implement it. (Yasuhiro Matsumoto)
Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 7.4.1513

Problem: "J" fails if there are not enough lines. (Christian Neukirchen)
Solution: Reduce the count, only fail on the last line.
Files: src/normal.c, src/testdir/test_join.vim, src/testdir/test_alot.vim

Patch 7.4.1514

Problem: Channel output to file not implemented yet.
Solution: Implement it for Unix.
Files: src/os_unix.c, src/testdir/test_channel.vim,
src/testdir/test_channel_pipe.py

Patch 7.4.1515

Problem: Channel test is a bit flaky.
Solution: Instead of a fixed sleep time wait until an expression evaluates to true.
Files: src/testdir/test_channel.vim

Patch 7.4.1516

Problem: Cannot change file permissions.
Solution: Add setfperm().
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test_alot.vim,
src/testdir/test_file_perm.vim

Patch 7.4.1517

Problem: Compiler warning with 64bit compiler.
Solution: Add typecast. (Mike Williams)
Files: src/channel.c

Patch 7.4.1518

Problem: Channel with disconnected in/out/err is not supported.
Solution: Implement it for Unix.
Files: src/eval.c, src/os_unix.c, src/structs.h,
src/testdir/test_channel.vim, src/testdir/test_channel_pipe.py

Patch 7.4.1519 (after 7.4.1514)

Problem: Channel output to file not implemented for MS-Windows.
Solution: Implement it. (Yasuhiro Matsumoto)
Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 7.4.1520

Problem: Channel test: Waiting for a file to appear doesn't work.
Solution: In waitFor() ignore errors.
Files: src/testdir/test_channel.vim

Patch 7.4.1521 (after 7.4.1516)

Problem: File permission test fails on MS-Windows.
Solution: Expect a different permission.
Files: src/testdir/test_file_perm.vim

Patch 7.4.1522

Problem: Cannot write channel err to a buffer.
Solution: Implement it.
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1523

Problem: Writing channel to a file fails on MS-Windows.
Solution: Disable it for now.
Files: src/testdir/test_channel.vim

Patch 7.4.1524

Problem: Channel test fails on BSD.
Solution: Break out of the loop when connect() succeeds. (Ozaki Kiichi)
Files: src/channel.c

Patch 7.4.1525

Problem: On a high resolution screen the toolbar icons are too small.
Solution: Add "huge" and "giant" to '**toolbariconsize**'. (Brian Gix)
Files: src/gui_gtk_x11.c, src/option.h

Patch 7.4.1526

Problem: Writing to file and not connecting a channel doesn't work for MS-Windows.
Solution: Make it work. (Yasuhiro Matsumoto)
Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 7.4.1527

Problem: Channel test is flaky on MS-Windows.
Solution: Limit the select() timeout to 50 msec and try with a new socket if it fails.
Files: src/channel.c

Patch 7.4.1528

Problem: Using "ever" for packages is confusing.
Solution: Use "start", as it's related to startup.
Files: src/ex_cmds2.c, runtime/doc/repeat.txt

Patch 7.4.1529

Problem: Specifying buffer number for channel not implemented yet.
Solution: Implement passing a buffer number.
Files: src/structs.h, src/channel.c, src/eval.c,
src/testdir/test_channel.vim

Patch 7.4.1530

Problem: MS-Windows job_start() closes wrong handle.
Solution: Close hThread on the process info. (Ken Takata)
Files: src/os_win32.c

Patch 7.4.1531

Problem: Compiler warning for uninitialized variable. (Dominique Pelle)
Solution: Always give the variable a value.
Files: src/channel.c

Patch 7.4.1532

Problem: MS-Windows channel leaks file descriptor.
Solution: Use CreateFile with the right options. (Yasuhiro Matsumoto)
Files: src/os_win32.c

Patch 7.4.1533

Problem: Using feedkeys() with an empty string disregards 'x' option.
Solution: Make 'x' work with an empty string. (Thinca)
Files: src/eval.c, src/testdir/test_alot.vim,
src/testdir/test_feedkeys.vim

Patch 7.4.1534

Problem: Compiler warning for shadowed variable. (Kazunobu Kuriyama)
Solution: Rename it.
Files: src/eval.c

Patch 7.4.1535

Problem: The feedkeys test has a one second delay.
Solution: Avoid need_wait_return() to delay. (Hirohito Higashi)
Files: src/eval.c

Patch 7.4.1536

Problem: Cannot re-use a channel for another job.
Solution: Add the "channel" option to job_start().
Files: src/channel.c, src/eval.c, src/structs.h, src/os_unix.c,
src/os_win32.c, src/proto/channel.pro,
src/testdir/test_channel.vim

Patch 7.4.1537

Problem: Too many feature flags for pipes, jobs and channels.
Solution: Only use FEAT_JOB_CHANNEL.
Files: src/structs.h, src/feature.h, src/configure.in,
src/auto/configure, src/config.h.in, src/channel.c, src/eval.c,
src/gui.c, src/main.c, src/memline.c, src/misc2.c, src/os_mswin.c,

src/os_unix.c, src/os_win32.c, src/ui.c, src/version.c,
src/macros.h, src/proto.h, src/vim.h, src/Make_cyg_ming.mak,
src/Make_bc5.mak, src/Make_mvc.mak

Patch 7.4.1538

Problem: Selection with the mouse does not work in command line mode.
Solution: Use cairo functions. (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 7.4.1539

Problem: Too much code in eval.c.
Solution: Move job and channel code to channel.c.
Files: src/eval.c, src/channel.c, src/proto/channel.pro,
src/proto/eval.pro

Patch 7.4.1540

Problem: Channel test is a bit flaky.
Solution: Increase expected wait time.
Files: src/testdir/test_channel.vim

Patch 7.4.1541

Problem: Missing job_info().
Solution: Implement it.
Files: src/eval.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 7.4.1542

Problem: job_start() with a list is not tested.
Solution: Call job_start() with a list.
Files: src/testdir/test_channel.vim

Patch 7.4.1543

Problem: Channel log methods are not tested.
Solution: Log job activity and check it.
Files: src/testdir/test_channel.vim

Patch 7.4.1544

Problem: On Win32 escaping the command does not work properly.
Solution: Reset '**ssl**' when escaping the command. (Yasuhiro Matsumoto)
Files: src/channel.c

Patch 7.4.1545

Problem: GTK3: horizontal cursor movement in Visual selection not good.
Solution: Make it work better. (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 7.4.1546

Problem: Sticky type checking is more annoying than useful.
Solution: Remove the error for changing a variable type.
Files: src/eval.c, src/testdir/test_assign.vim,
src/testdir/test_alot.vim, runtime/doc/eval.txt

Patch 7.4.1547

Problem: Getting a cterm highlight attribute that is not set results in the

string "-1".
Solution: Return an empty string. (Taro Muraoka)
Files: src/syntax.c, src/testdir/test_alot.vim,
src/testdir/test_syn_attr.vim

Patch 7.4.1548 (after 7.4.1546)

Problem: Two tests fail.
Solution: Adjust the expected error number. Remove check for type.
Files: src/testdir/test101.ok, src/testdir/test55.in,
src/testdir/test55.ok

Patch 7.4.1549 (after 7.4.1547)

Problem: Test for syntax attributes fails in Win32 GUI.
Solution: Use an existing font name.
Files: src/testdir/test_syn_attr.vim

Patch 7.4.1550

Problem: Cannot load packages early.
Solution: Add the ":packloadall" command.
Files: src/ex_cmds.h, src/ex_cmds2.c, src/main.c,
src/proto/ex_cmds2.pro, src/testdir/test_packadd.vim

Patch 7.4.1551

Problem: Cannot generate help tags in all doc directories.
Solution: Make ":helptags ALL" work.
Files: src/ex_cmds2.c, src/proto/ex_cmds2.pro, src/ex_cmds.c, src/vim.h
src/testdir/test_packadd.vim

Patch 7.4.1552

Problem: ":colorscheme" does not use **'packpath'**.
Solution: Also use in "start" and "opt" directories in **'packpath'**.
Files: src/ex_cmds2.c, src/gui.c, src/hardcopy.c, src/os_mswin.c,
src/spell.c, src/tag.c, src/if_py_both.h, src/vim.h,
src/digraph.c, src/eval.c, src/ex_docmd.c, src/main.c,
src/option.c, src/syntax.c, src/testdir/test_packadd.vim

Patch 7.4.1553

Problem: ":runtime" does not use **'packpath'**.
Solution: Add "what" argument.
Files: src/ex_cmds2.c, src/vim.h, runtime/doc/repeat.txt,
src/testdir/test_packadd.vim

Patch 7.4.1554

Problem: Completion for :colorscheme does not use **'packpath'**.
Solution: Make it work, add a test. (Hirohito Higashi)
Files: src/ex_getln.c, src/testdir/test_packadd.vim

Patch 7.4.1555

Problem: List of test targets incomplete.
Solution: Add newly added tests.
Files: src/Makefile

Patch 7.4.1556

Problem: "make install" changes the help tags file, causing it to differ

from the repository.
Solution: Move it aside and restore it.
Files: src/Makefile

Patch 7.4.1557

Problem: Windows cannot be identified.
Solution: Add a unique window number to each window and functions to use it.
Files: src/structs.h, src/window.c, src/eval.c, src/proto/eval.pro,
src/proto/window.pro, src/testdir/test_window_id.vim,
src/testdir/Make_all.mak, runtime/doc/eval.txt

Patch 7.4.1558

Problem: It is not easy to find out what windows display a buffer.
Solution: Add win_findbuf().
Files: src/eval.c, src/window.c, src/proto/window.pro,
src/testdir/test_window_id.vim, runtime/doc/eval.txt

Patch 7.4.1559

Problem: Passing cookie to a callback is clumsy.
Solution: Change function() to take arguments and return a partial.
Files: src/structs.h, src/channel.c, src/eval.c, src/if_python.c,
src/if_python3.c, src/if_py_both.h, src/json.c,
src/proto/eval.pro, src/testdir/test_partial.vim,
src/testdir/test_alot.vim, runtime/doc/eval.txt

Patch 7.4.1560

Problem: Dict options with a dash are more difficult to use.
Solution: Use an underscore, so that dict.err_io can be used.
Files: src/channel.c, src/structs.h, src/testdir/test_channel.vim,
runtime/doc/channel.txt

Patch 7.4.1561 (after 7.4.1559)

Problem: Missing update to proto file.
Solution: Change the proto file.
Files: src/proto/channel.pro

Patch 7.4.1562

Problem: ":helptags ALL" crashes. (Lcd)
Solution: Don't free twice.
Files: src/ex_cmds.c

Patch 7.4.1563

Problem: Partial test fails on windows.
Solution: Return 1 or -1 from compare function.
Files: src/testdir/test_partial.vim

Patch 7.4.1564

Problem: An empty list in function() causes an error.
Solution: Handle an empty list like there is no list of arguments.
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1565

Problem: Crash when assert_equal() runs into a NULL string.
Solution: Check for NULL. (Dominique) Add a test.

Files: src/eval.c, src/testdir/test_assert.vim

Patch 7.4.1566

Problem: Compiler warning for shadowed variable. (Kazunobu Kuriyama)
Solution: Remove the inner one.
Files: src/eval.c

Patch 7.4.1567

Problem: Crash in assert_fails().
Solution: Check for NULL. (Dominique Pelle) Add a test.
Files: src/eval.c, src/testdir/test_assert.vim

Patch 7.4.1568

Problem: Using **CTRL-]** in help on option in parentheses doesn't work.
Solution: Skip the "(" in "("'. (Hirohito Higashi)
Files: src/ex_cmds.c

Patch 7.4.1569

Problem: Using old style tests for quickfix.
Solution: Change them to new style tests. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak, src/testdir/test106.in,
src/testdir/test106.ok, src/testdir/test_qf_title.in,
src/testdir/test_qf_title.ok, src/testdir/test_quickfix.vim

Patch 7.4.1570

Problem: There is no way to avoid the message when editing a file.
Solution: Add the "F" flag to '**shortmess**'. (Shougo Matsu, closes #686)
Files: runtime/doc/options.txt, src/buffer.c, src/ex_cmds.c,
src/option.h

Patch 7.4.1571

Problem: No test for ":help".
Solution: Add a test for what 7.4.1568 fixed. (Hirohito Higashi)
Files: src/testdir/test_alot.vim, src/testdir/test_help_tagjump.vim

Patch 7.4.1572

Problem: Setting '**compatible**' in test influences following tests.
Solution: Turn '**compatible**' off again.
Files: src/testdir/test_backspace_opt.vim

Patch 7.4.1573

Problem: Tests get stuck at the more prompt.
Solution: Move the backspace test out of test_alot.
Files: src/testdir/test_alot.vim, src/testdir/Make_all.mak

Patch 7.4.1574

Problem: ":undo 0" does not work. (Florent Fayolle)
Solution: Make it undo all the way. (closes #688)
Files: src/undo.c, src/testdir/test_undolevels.vim,
src/testdir/test_ex_undo.vim, src/testdir/test_alot.vim

Patch 7.4.1575

Problem: Using wrong size for struct.
Solution: Use the size for wide API. (Ken Takata)

Files: src/gui_w32.c

Patch 7.4.1576

Problem: Write error of viminfo file is not handled properly. (Christian Neukirchen)

Solution: Check the return value of fclose(). (closes #682)

Files: src/ex_cmds.c

Patch 7.4.1577

Problem: Cannot pass "dict.Myfunc" around as a partial.

Solution: Create a partial when expected.

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1578

Problem: There is no way to invoke a function later or periodically.

Solution: Add timer support.

Files: src/eval.c, src/ex_cmds2.c, src/screen.c, src/ex_docmd.c, src/feature.h, src/gui.c, src/proto/eval.pro, src/proto/ex_cmds2.pro, src/proto/screen.pro, src/structs.h, src/version.c, src/testdir/test_alot.vim, src/testdir/test_timers.vim, runtime/doc/eval.txt

Patch 7.4.1579 (after 7.4.1578)

Problem: Missing changes in channel.c

Solution: Include the changes.

Files: src/channel.c

Patch 7.4.1580

Problem: Crash when using function reference. (Luchr)

Solution: Set initial refcount. (Ken Takata, closes #690)

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1581

Problem: Using ":call dict.func()" where the function is a partial does not work. Using "dict.func()" where the function does not take a Dictionary does not work.

Solution: Handle partial properly in ":call". (Yasuhiro Matsumoto)

Files: src/eval.c, src/testdir/test_partial.vim, src/testdir/test55.ok

Patch 7.4.1582

Problem: Get E923 when using function(dict.func, [], dict). (Kent Sibilev) Storing a function with a dict in a variable drops the dict if the function is script-local.

Solution: Translate the function name. Use dict arg if present.

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1583

Problem: Warning for uninitialized variable.

Solution: Initialize it. (Dominique)

Files: src/ex_cmds2.c

Patch 7.4.1584

Problem: Timers don't work for Win32 console.

Solution: Add check_due_timer() in WaitForChar().

Files: src/os_win32.c

Patch 7.4.1585

Problem: Partial is not recognized everywhere.

Solution: Check for partial in trans_function_name(). (Yasuhiro Matsumoto)
Add a test.

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1586

Problem: Nesting partials doesn't work.

Solution: Append arguments. (Ken Takata)

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1587

Problem: Compiler warnings with 64 bit compiler.

Solution: Add type casts. (Mike Williams)

Files: src/ex_cmds2.c

Patch 7.4.1588

Problem: Old style test for quickfix.

Solution: Turn test 96 into a new style test.

Files: src/testdir/Make_all.mak, src/testdir/test96.in,
src/testdir/test96.ok, src/testdir/test_quickfix.vim

Patch 7.4.1589

Problem: Combining dict and args with partial doesn't always work.

Solution: Use the arguments from the partial.

Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1590

Problem: Warning for shadowed variable. (Christian Brabandt)

Solution: Move the variable into a local block.

Files: src/eval.c

Patch 7.4.1591

Problem: The quickfix title is truncated.

Solution: Save the command before it is truncated. (Anton Lindqvist)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1592

Problem: Quickfix code using memory after being freed. (Dominique Pelle)

Solution: Detect that the window was closed. (Hirohito Higashi)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1593

Problem: Using channel timeout instead of request timeout. (Coverity)

Solution: Remove the extra assignment.

Files: src/channel.c

Patch 7.4.1594

Problem: Timers don't work on Unix.

Solution: Add missing code.

Files: src/os_unix.c

Patch 7.4.1595

Problem: Not checking for failed open(). (Coverity)
Solution: Check file descriptor not being negative.
Files: src/os_unix.c

Patch 7.4.1596

Problem: Memory leak. (Coverity)
Solution: Free the pattern.
Files: src/ex_cmds2.c

Patch 7.4.1597

Problem: Memory leak when out of memory. (Coverity)
Solution: Free the name.
Files: src/eval.c

Patch 7.4.1598

Problem: When starting the GUI fails a swap file is left behind. (Joerg Plate)
Solution: Preserve files before exiting. (closes #692)
Files: src/main.c, src/gui.c

Patch 7.4.1599

Problem: No link to Coverity.
Solution: Add Coverity badge in README.
Files: README.md

Patch 7.4.1600

Problem: libs directory is not useful.
Solution: Remove arp.library, it was only for very old Amiga versions.
Files: libs/arp.library, Filelist

Patch 7.4.1601

Problem: README files take a lot of space in the top directory.
Solution: Move most of them to "READMEdir".
Files: Filelist, Makefile, README.txt.info, README_ami.txt, README_ami.txt.info, README_amibin.txt, README_amibin.txt.info, README_amisrc.txt, README_amisrc.txt.info, README_bindos.txt, README_dos.txt, README_extra.txt, README_mac.txt, README_ole.txt, README_os2.txt, README_os390.txt, README_src.txt, README_srcdos.txt, README_unix.txt, README_vms.txt, README_w32s.txt, READMEdir/README.txt.info, READMEdir/README_ami.txt, READMEdir/README_ami.txt.info, READMEdir/README_amibin.txt, READMEdir/README_amibin.txt.info, READMEdir/README_amisrc.txt, READMEdir/README_amisrc.txt.info, READMEdir/README_bindos.txt, READMEdir/README_dos.txt, READMEdir/README_extra.txt, READMEdir/README_mac.txt, READMEdir/README_ole.txt, READMEdir/README_os2.txt, READMEdir/README_os390.txt, READMEdir/README_src.txt, READMEdir/README_srcdos.txt, READMEdir/README_unix.txt, READMEdir/README_vms.txt, READMEdir/README_w32s.txt,

Patch 7.4.1602

Problem: Info files take space in the top directory.
Solution: Move them to "READMEdir".

Files: Filelist, src.info, Contents.info, runtime.info, vimdir.info, Vim.info, Xxd.info, READMEDir/src.info, READMEDir/Contents.info, READMEDir/runtime.info, READMEDir/vimdir.info, READMEDir/Vim.info, READMEDir/Xxd.info

Patch 7.4.1603

Problem: Timer with an ":echo" command messes up display.
Solution: Redraw depending on the mode. (Hirohito Higashi) Avoid the more prompt being used recursively.
Files: src/screen.c, src/message.c

Patch 7.4.1604

Problem: Although emoji characters are ambiguous width, best is to treat them as full width.
Solution: Update the Unicode character tables. Add the 'emoji' options. (Yasuhiro Matsumoto)
Files: runtime/doc/options.txt, runtime/optwin.vim, runtime/tools/unicode.vim, src/mbyte.c, src/option.c, src/option.h

Patch 7.4.1605

Problem: Catching exception that won't be thrown.
Solution: Remove try/catch.
Files: src/testdir/test55.in

Patch 7.4.1606

Problem: Having type() handle a Funcref that is or isn't a partial differently causes problems for existing scripts.
Solution: Make type() return the same value. (Thinca)
Files: src/eval.c, src/testdir/test_viml.vim

Patch 7.4.1607

Problem: Comparing a function that exists on two dicts is not backwards compatible. (Thinca)
Solution: Only compare the function, not what the partial adds.
Files: src/eval.c, src/testdir/test_alot.vim, src/testdir/test_expr.vim

Patch 7.4.1608

Problem: string() doesn't handle a partial.
Solution: Make a string from a partial.
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1609

Problem: Contents file is only for Amiga distro.
Solution: Move it to "READMEDir". Update some info.
Files: Filelist, Contents, READMEDir/Contents

Patch 7.4.1610

Problem: Compiler warnings for non-virtual destructor.
Solution: Mark the classes final. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/gui_dwrite.cpp, src/if_ole.cpp

Patch 7.4.1611

Problem: The versplit feature makes the code unnecessary complicated.
Solution: Remove FEAT_VERTSPLIT, always support vertical splits when

FEAT_WINDOWS is defined.
Files: src/buffer.c, src/charset.c, src/eval.c, src/ex_cmds.c,
src/ex_docmd.c, src/ex_getln.c, src/gui.c, src/if_lua.c,
src/if_mzsch.c, src/if_ruby.c, src/main.c, src/misc1.c,
src/misc2.c, src/move.c, src/normal.c, src/option.c,
src/quickfix.c, src/screen.c, src/syntax.c, src/term.c, src/ui.c,
src/window.c, src/globals.h, src/gui.h, src/if_py_both.h,
src/option.h, src/structs.h, src/term.h
src/feature.h, src/vim.h, src/version.c

Patch 7.4.1612 (after 7.4.1611)
Problem: Can't build with small features.
Solution: Move code and #ifdefs.
Files: src/ex_getln.c

Patch 7.4.1613 (after 7.4.1612)
Problem: Still can't build with small features.
Solution: Adjust #ifdefs.
Files: src/ex_getln.c

Patch 7.4.1614
Problem: Still quickfix test in old style.
Solution: Turn test 10 into a new style test.
Files: src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/main.aap, src/testdir/test10.in,
src/testdir/test10.ok, src/testdir/test_quickfix.vim,
src/testdir/test10a.in, src/testdir/test10a.ok

Patch 7.4.1615
Problem: Build fails with tiny features.
Solution: Adjust #ifdefs.
Files: src/normal.c, src/window.c

Patch 7.4.1616
Problem: Malformed channel request causes a hang.
Solution: Drop malformed message. (Damien)
Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel.py

Patch 7.4.1617
Problem: When a JSON message is split it isn't decoded.
Solution: Wait a short time for the rest of the message to arrive.
Files: src/channel.c, src/json.c, src/structs.h,
src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1618
Problem: Starting job with output to buffer changes options in the current
buffer.
Solution: Set "curbuf" earlier. (Yasuhiro Matsumoto)
Files: src/channel.c

Patch 7.4.1619
Problem: When **'fileformats'** is set in the vimrc it applies to new buffers
but not the initial buffer.

Solution: Set **'fileformat'** when starting up. (Mike Williams)
Files: src/option.c

Patch 7.4.1620

Problem: Emoji characters are not considered as a kind of word character.
Solution: Give emoji characters a word class number. (Yasuhiro Matsumoto)
Files: src/mbyte.c

Patch 7.4.1621

Problem: Channel test doesn't work with Python 2.6.
Solution: Add number in formatting placeholder. (Wiredool)
Files: src/testdir/test_channel.py

Patch 7.4.1622

Problem: Channel demo doesn't work with Python 2.6.
Solution: Add number in formatting placeholder
Files: runtime/tools/demoserver.py

Patch 7.4.1623

Problem: All Channels share the message ID, it keeps getting bigger.
Solution: Use a message ID per channel.
Files: src/channel.c, src/proto/channel.pro, src/structs.h

Patch 7.4.1624

Problem: Can't get info about a channel.
Solution: Add ch_info().
Files: src/eval.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 7.4.1625

Problem: Trying to close file descriptor that isn't open.
Solution: Check for negative number.
Files: src/os_unix.c

Patch 7.4.1626 (after 7.4.1624)

Problem: Missing changes to structs.
Solution: Include the changes.
Files: src/structs.h

Patch 7.4.1627

Problem: Channel out_cb and err_cb are not tested.
Solution: Add a test.
Files: src/testdir/test_channel.vim

Patch 7.4.1628

Problem: 64-bit Compiler warning.
Solution: Change type of variable. (Mike Williams)
Files: src/channel.c

Patch 7.4.1629

Problem: Handling emoji characters as full width has problems with backwards compatibility.
Solution: Remove ambiguous and double width characters from the emoji table.
Use a separate table for the character class.

(partly by Yasuhiro Matsumoto)
Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 7.4.1630

Problem: Unicode table for double width is outdated.
Solution: Update to the latest Unicode standard.
Files: src/mbyte.c

Patch 7.4.1631

Problem: Compiler doesn't understand switch on all enum values. (Tony Mechelynck)
Solution: Initialize variable.
Files: src/channel.c

Patch 7.4.1632

Problem: List of test targets is outdated.
Solution: Update to current list of test targets.
Files: src/Makefile

Patch 7.4.1633

Problem: If the help tags file was removed "make install" fails. (Tony Mechelynck)
Solution: Only try moving the file if it exists.
Files: src/Makefile

Patch 7.4.1634

Problem: Vertical movement after **CTRL-A** ends up in the wrong column. (Urtica Dioica)
Solution: Set curswant when appropriate. (Hirohito Higashi)
Files: src/ops.c, src/testdir/test_increment.vim

Patch 7.4.1635

Problem: Channel test is a bit flaky.
Solution: Remove 'DETACH' if it's there.
Files: src/testdir/test_channel.vim

Patch 7.4.1636

Problem: When 'F' is in '**shortmess**' the prompt for the encryption key isn't displayed. (Toothpik)
Solution: Reset msg_silent.
Files: src/ex_getln.c

Patch 7.4.1637

Problem: Can't build with older MinGW compiler.
Solution: Change option from c++11 to gnu++11. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.1638

Problem: When binding a function to a dict the reference count is wrong.
Solution: Decrement dict reference count, only reference the function when actually making a copy. (Ken Takata)
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1639

Problem: Invoking garbage collection may cause a double free.
Solution: Don't free the dict in a partial when recursive is FALSE.
Files: src/eval.c

Patch 7.4.1640

Problem: Crash when an autocommand changes a quickfix list. (Dominique)
Solution: Check whether an entry is still valid. (Yegappan Lakshmanan, Hirohito Higashi)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1641

Problem: Using unterminated string.
Solution: Add NUL before calling vim_strsave_shellescape(). (James McCoy)
Files: src/eval.c, src/testdir/test105.in, src/testdir/test105.ok

Patch 7.4.1642

Problem: Handling emoji characters as full width has problems with backwards compatibility.
Solution: Only put characters in the 1f000 range in the emoji table.
Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 7.4.1643 (after 7.4.1641)

Problem: Terminating file name has side effects.
Solution: Restore the character. (mostly by James McCoy, closes #713)
Files: src/eval.c, src/testdir/test105.in, src/testdir/test105.ok

Patch 7.4.1644

Problem: Using string() on a partial that exists in the dictionary it binds results in an error. (Nikolai Pavlov)
Solution: Make string() not fail on a recursively nested structure. (Ken Takata)
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1645

Problem: When a dict contains a partial it can't be redefined as a function. (Nikolai Pavlov)
Solution: Remove the partial when overwriting with a function.
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1646

Problem: Using Python vim.bindeval() on a partial doesn't work. (Nikolai Pavlov)
Solution: Add VAR_PARTIAL support in Python.
Files: src/if_py_both.h, src/testdir/test_partial.vim

Patch 7.4.1647

Problem: Using freed memory after setqflist() and ":caddbuffer". (Dominique)
Solution: Set qf_ptr when adding the first item to the quickfix list.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1648

Problem: Compiler has a problem copying a string into di_key[]. (Yegappan Lakshmanan)
Solution: Add dictitem16_T.

Files: src/structs.h, src/eval.c

Patch 7.4.1649

Problem: The matchit plugin needs to be copied to be used.

Solution: Put the matchit plugin in an optional package.

Files: Filelist, runtime/macros/matchit.vim, runtime/macros/matchit.txt, runtime/macros/README.txt, src/Makefile, runtime/pack/dist/opt/matchit/plugin/matchit.vim, runtime/pack/dist/opt/matchit/doc/matchit.txt, runtime/pack/dist/opt/matchit/doc/tags, runtime/doc/usr_05.txt, runtime/doc/usr_toc.txt

Patch 7.4.1650

Problem: Quickfix test fails.

Solution: Accept any number of matches.

Files: src/testdir/test_quickfix.vim

Patch 7.4.1651

Problem: Some dead (MSDOS) code remains.

Solution: Remove the unused lines. (Ken Takata)

Files: src/misc1.c

Patch 7.4.1652

Problem: Old style test for fnamemodify().

Solution: Turn it into a new style test.

Files: src/testdir/test105.in, src/testdir/test105.ok, src/testdir/test_fnamemodify.vim, src/testdir/test_alot.vim, src/testdir/Make_all.mak

Patch 7.4.1653 (after 7.4.1649)

Problem: Users who loaded matchit.vim manually have to change their startup. (Gary Johnson)

Solution: Add a file in the old location that loads the package.

Files: runtime/macros/matchit.vim, Filelist

Patch 7.4.1654

Problem: Crash when using expand('%:S') in a buffer without a name.

Solution: Don't set a NUL. (James McCoy, closes #714)

Files: src/eval.c, src/testdir/test_fnamemodify.vim

Patch 7.4.1655

Problem: remote_expr() hangs. (Ramel)

Solution: Check for messages in the waiting loop.

Files: src/if_xcmdsrv.c

Patch 7.4.1656

Problem: Crash when using partial with a timer.

Solution: Increment partial reference count. (Hirohito Higashi)

Files: src/eval.c, src/testdir/test_timers.vim

Patch 7.4.1657

Problem: On Unix in a terminal: channel messages are not handled right away. (Jackson Alves de Aquino)

Solution: Break the loop for timers when something was received.

Files: src/os_unix.c

Patch 7.4.1658

Problem: A plugin does not know when VimEnter autocommands were already triggered.

Solution: Add the v:vim_did_enter variable.

Files: src/eval.c, src/main.c, src/vim.h, src/testdir/test_autocmd.vim, src/testdir/test_alot.vim, runtime/doc/autocmd.txt, runtime/doc/eval.txt

Patch 7.4.1659 (after 7.4.1657)

Problem: Compiler warning for argument type. (Manuel Ortega)

Solution: Remove "&".

Files: src/os_unix.c

Patch 7.4.1660

Problem: has('patch-7.4.1') doesn't work.

Solution: Fix off-by-one error. (Thinca)

Files: src/eval.c, src/testdir/test_expr.vim, src/testdir/test60.in, src/testdir/test60.ok

Patch 7.4.1661

Problem: No test for special characters in channel eval command.

Solution: Testing sending and receiving text with special characters.

Files: src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1662

Problem: No test for an invalid Ex command on a channel.

Solution: Test handling an invalid command gracefully. Avoid getting an error message, do write it to the channel log.

Files: src/channel.c, src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 7.4.1663

Problem: In tests it's often useful to check if a pattern matches.

Solution: Add assert_match().

Files: src/eval.c, src/testdir/test_assert.vim, src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 7.4.1664

Problem: Crash in :cgetexpr.

Solution: Check for NULL pointer. (Dominique) Add a test.

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1665

Problem: Crash when calling job_start() with a NULL string. (Dominique)

Solution: Check for an invalid argument.

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1666

Problem: When reading JSON from a channel all readahead is used.

Solution: Use the fill function to reduce overhead.

Files: src/channel.c, src/json.c, src/structs.h

Patch 7.4.1667

Problem: Win32: waiting on a pipe with fixed sleep time.
Solution: Start with a short delay and increase it when looping.
Files: src/channel.c

Patch 7.4.1668

Problem: channel_get_all() does multiple allocations.
Solution: Compute the size and allocate once.
Files: src/channel.c

Patch 7.4.1669

Problem: When writing buffer lines to a pipe Vim may block.
Solution: Avoid blocking, write more lines later.
Files: src/channel.c, src/misc2.c, src/os_unix.c, src/structs.h,
src/vim.h, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 7.4.1670

Problem: Completion doesn't work well for a variable containing "#".
Solution: Recognize the "#". (Watiko)
Files: src/eval.c

Patch 7.4.1671

Problem: When help exists in multiple languages, adding @ab while "ab" is
the default help language is unnecessary.
Solution: Leave out "@ab" when not needed. (Ken Takata)
Files: src/ex_getln.c

Patch 7.4.1672

Problem: The Dvorak support is a bit difficult to install.
Solution: Turn it into an optional package.
Files: runtime/macros/dvorak, runtime/macros/README.txt,
runtime/pack/dist/opt/dvorak/plugin/dvorak.vim,
runtime/pack/dist/opt/dvorak/dvorak/enable.vim,
runtime/pack/dist/opt/dvorak/dvorak/disable.vim

Patch 7.4.1673

Problem: The justify plugin has to be copied or sourced to be used.
Solution: Turn it into a package.
Files: runtime/macros/justify.vim, runtime/macros/README.txt,
runtime/pack/dist/opt/justify/plugin/justify.vim, Filelist

Patch 7.4.1674

Problem: The editexisting plugin has to be copied or sourced to be used.
Solution: Turn it into a package.
Files: runtime/macros/editexisting.vim, runtime/macros/README.txt,
runtime/pack/dist/opt/editexisting/plugin/editexisting.vim,
Filelist

Patch 7.4.1675

Problem: The swapmous plugin has to be copied or sourced to be used.
Solution: Turn it into the swapmouse package.
Files: runtime/macros/swapmous.vim, runtime/macros/README.txt,
runtime/pack/dist/opt/swapmouse/plugin/swapmouse.vim, Filelist

Patch 7.4.1676

Problem: The shellmenu plugin has to be copied or sourced to be used.
Solution: Turn it into a package.
Files: runtime/macros/shellmenu.vim, runtime/macros/README.txt,
runtime/pack/dist/opt/shellmenu/plugin/shellmenu.vim, Filelist

Patch 7.4.1677

Problem: A reference to the removed file_select plugin remains.
Solution: Remove it.
Files: runtime/macros/README.txt

Patch 7.4.1678

Problem: Warning for unused argument.
Solution: Add UNUSED. (Dominique Pelle)
Files: src/if_mzsch.c

Patch 7.4.1679

Problem: Coverity: copying value of v_lock without initializing it.
Solution: Init v_lock in rettv_list_alloc() and rettv_dict_alloc().
Files: src/eval.c

Patch 7.4.1680

Problem: Coverity warns for not checking name length (false positive).
Solution: Only copy the characters we know are there.
Files: src/channel.c

Patch 7.4.1681

Problem: Coverity warns for fixed size buffer length (false positive).
Solution: Add a check for the name length.
Files: src/eval.c

Patch 7.4.1682

Problem: Coverity: no check for NULL.
Solution: Add check for invalid argument to assert_match().
Files: src/eval.c

Patch 7.4.1683

Problem: Generated .bat files do not support --nofork.
Solution: Add check for --nofork. Also add "setlocal". (Kevin Cantú,
closes #659)
Files: src/dosinst.c

Patch 7.4.1684

Problem: README text is slightly outdated.
Solution: Mention the READMEdir directory.
Files: README.md, README.txt

Patch 7.4.1685

Problem: There is no easy way to get all the information about a match.
Solution: Add matchstrpos(). (Ozaki Kiichi)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/eval.c,
src/testdir/test_alot.vim, src/testdir/test_matchstrpos.vim

Patch 7.4.1686

Problem: When running tests \$HOME/.viminfo is written. (James McCoy)
Solution: Add **'nviminfo'** to the **'viminfo'** option. (closes #722)
Files: src/testdir/test_backspace_opt.vim, src/testdir/test_viminfo.vim,
src/testdir/runtest.vim.

Patch 7.4.1687

Problem: The channel close_cb option does not work.
Solution: Use jo_close_partial instead of jo_err_partial. (Damien)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1688

Problem: MzScheme does not support partial.
Solution: Add minimal partial support. (Ken Takata)
Files: src/if_mzsch.c

Patch 7.4.1689

Problem: Ruby interface has inconsistent coding style.
Solution: Fix the coding style. (Ken Takata)
Files: src/if_ruby.c

Patch 7.4.1690

Problem: Can't compile with the conceal feature but without multibyte.
Solution: Adjust #ifdef. (Owen Leibman)
Files: src/eval.c, src/window.c

Patch 7.4.1691

Problem: When switching to a new buffer and an autocommand applies syntax highlighting an ml_get error may occur.
Solution: Check "syn_buf" against the buffer in the window. (Alexander von Buddenbrock, closes #676)
Files: src/syntax.c

Patch 7.4.1692

Problem: feedkeys('i', 'x') gets stuck, waits for a character to be typed.
Solution: Behave like ":normal". (Yasuhiro Matsumoto)
Files: src/eval.c, src/testdir/test_feedkeys.vim

Patch 7.4.1693

Problem: Building the Perl interface gives compiler warnings.
Solution: Remove a pragma. Add noreturn attributes. (Damien)
Files: src/if_perl.xs

Patch 7.4.1694

Problem: Win32 gvim doesn't work with "dvorakj" input method.
Solution: Wait for QS_ALLINPUT instead of QS_ALLEVENTS. (Yukihiro Nakadaira)
Files: src/gui_w32.c

Patch 7.4.1695

Problem: ":syn reset" clears the effect ":syn iskeyword". (James McCoy)
Solution: Remove clearing the syntax keywords.
Files: src/syntax.c

Patch 7.4.1696

Problem: When using :stopinsert in a silent mapping the "INSERT" message

isn't cleared. (Coacher)
Solution: Always clear the message. (Christian Brabandt, closes #718)
Files: src/ex_docmd.c, src/proto/screen.pro, src/screen.c

Patch 7.4.1697

Problem: Display problems when the **'ambiwidth'** and **'emoji'** options are not set properly or the terminal doesn't behave as expected.
Solution: After drawing an ambiguous width character always position the cursor.
Files: src/mbyte.c, src/screen.c, src/proto/mbyte.pro

Patch 7.4.1698

Problem: Two tests fail when running tests with MinGW. (Michael Soyka)
Solution: Convert test_getcwd.ok test_wordcount.ok to unix fileformat.
Files: src/testdir/Make_ming.mak

Patch 7.4.1699

Problem: :packadd does not work the same when used early or late.
Solution: Always load plugins matching "plugin/**/*.vim".
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 7.4.1700

Problem: Equivalence classes are not properly tested.
Solution: Add tests for multibyte and latin1. Fix an error. (Owen Leibman)
Files: src/regexp.c, src/testdir/Make_all.mak,
src/testdir/test_alot_latin.vim, src/testdir/test_alot_utf8.vim,
src/testdir/test_regexp_latin.vim,
src/testdir/test_regexp_utf8.vim

Patch 7.4.1701

Problem: Equivalence classes still tested in old style tests.
Solution: Remove the duplicate.
Files: src/testdir/test44.in, src/testdir/test44.ok,
src/testdir/test99.in, src/testdir/test99.ok

Patch 7.4.1702

Problem: Using freed memory when parsing **'printoptions'** fails.
Solution: Save the old options and restore them in case of an error.
(Dominique)
Files: src/hardcopy.c, src/testdir/test_hardcopy.vim

Patch 7.4.1703

Problem: Can't assert for not equal and not matching.
Solution: Add assert_notmatch() and assert_notequal().
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test_assert.vim

Patch 7.4.1704

Problem: Using freed memory with "wincmd p". (Dominique Pelle)
Solution: Also clear "prevwin" in other tab pages.
Files: src/window.c

Patch 7.4.1705

Problem: The **'guifont'** option does not allow for a quality setting.
Solution: Add the "q" item, supported on MS-Windows. (Yasuhiro Matsumoto)

Files: runtime/doc/options.txt, src/gui_w32.c, src/os_mswin.c,
src/proto/os_mswin.pro

Patch 7.4.1706

Problem: Old style function declaration breaks build.
Solution: Remove __ARGS().
Files: src/proto/os_mswin.pro

Patch 7.4.1707

Problem: Cannot use empty dictionary key, even though it can be useful.
Solution: Allow using an empty dictionary key.
Files: src/hashtab.c, src/eval.c, src/testdir/test_expr.vim

Patch 7.4.1708

Problem: New regexp engine does not work properly with EBCDIC.
Solution: Define equivalence class characters. (Owen Leibman)
Files: src/regexp_nfa.c

Patch 7.4.1709

Problem: Mistake in #ifdef.
Solution: Change PROOF_QUALITY to DRAFT_QUALITY. (Ken Takata)
Files: src/os_mswin.c

Patch 7.4.1710

Problem: Not all output of an external command is read.
Solution: Avoid timing out when the process has exited. (closes #681)
Files: src/os_unix.c

Patch 7.4.1711

Problem: When using try/catch in '**statusline**' it is still considered an error and the status line will be disabled.
Solution: Check did_emsg instead of called_emsg. (haya14busa, closes #729)
Files: src/screen.c, src/testdir/test_statusline.vim,
src/testdir/test_alot.vim

Patch 7.4.1712

Problem: For plugins in packages, plugin authors need to take care of all dependencies.
Solution: When loading "start" packages and for :packloadall, first add all directories to '**runtimepath**' before sourcing plugins.
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 7.4.1713

Problem: GTK GUI doesn't work on Wayland.
Solution: Specify that only the X11 backend is allowed. (Simon McVittie)
Files: src/gui_gtk_x11.c

Patch 7.4.1714

Problem: Non-GUI specific settings in the gvimrc_example file.
Solution: Move some settings to the vimrc_example file. Remove setting '**hlsearch**' again. (suggested by Hirohito Higashi)
Files: runtime/vimrc_example.vim, runtime/gvimrc_example.vim

Patch 7.4.1715

Problem: Double free when a partial is in a cycle with a list or dict.
(Nikolai Pavlov)
Solution: Do not free a nested list or dict used by the partial.
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1716

Problem: **'autochdir'** doesn't work for the first file. (Rob Hoelz)
Solution: Call DO_AUTOCHDIR after startup. (Christian Brabandt, closes #704)
Files: src/main.c

Patch 7.4.1717

Problem: Leaking memory when opening a channel fails.
Solution: Unreference partials in job options.
Files: src/eval.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim

Patch 7.4.1718

Problem: Coverity: not using return value of set_ref_in_item().
Solution: Use the return value.
Files: src/eval.c

Patch 7.4.1719

Problem: Leaking memory when there is a cycle involving a job and a partial.
Solution: Add a copyID to job and channel. Set references in items referred by them. Go through all jobs and channels to find unreferenced items. Also, decrement reference counts when garbage collecting.
Files: src/eval.c, src/channel.c, src/netbeans.c, src/globals.h,
src/ops.c, src/regexp.c, src/tag.c, src/proto/channel.pro,
src/proto/eval.pro, src/testdir/test_partial.vim, src/structs.h

Patch 7.4.1720

Problem: Tests fail without the job feature.
Solution: Skip tests when the job feature is not present.
Files: src/testdir/test_partial.vim

Patch 7.4.1721

Problem: The vimtbar files are unused.
Solution: Remove them. (Ken Takata)
Files: src/vimtbar.dll, src/vimtbar.h, src/vimtbar.lib, Filelist

Patch 7.4.1722

Problem: Crash when calling garbagecollect() after starting a job.
Solution: Set the copyID on job and channel. (Hirohito Higashi, Ozaki Kiichi)
Files: src/eval.c

Patch 7.4.1723

Problem: When using try/catch in **'tabline'** it is still considered an error and the tabline will be disabled.
Solution: Check did_emsg instead of called_emsg. (haya14busa, closes #746)
Files: src/screen.c, src/testdir/test_tabline.vim,
src/testdir/test_alot.vim

Patch 7.4.1724 (after 7.4.1723)

Problem: Tabline test fails in GUI.
Solution: Remove 'e' from **'guioptions'**.
Files: src/testdir/test_tabline.vim

Patch 7.4.1725

Problem: Compiler errors for non-ANSI compilers.
Solution: Remove // comment. Remove comma at end of enum. (Michael Jarvis)
Files: src/eval.c

Patch 7.4.1726

Problem: ANSI compiler complains about string length.
Solution: Split long string in two parts. (Michael Jarvis)
Files: src/ex_cmds.c

Patch 7.4.1727

Problem: Cannot detect a crash in tests when caused by garbagecollect().
Solution: Add garbagecollect_for_testing(). Do not free a job if is still useful.
Files: src/channel.c, src/eval.c, src/getchar.c, src/main.c, src/vim.h,
src/proto/eval.pro, src/testdir/runtest.vim,
src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 7.4.1728

Problem: The help for functions require a space after the "(".
Solution: Make **CTRL-]** on a function name ignore the arguments. (Hirohito Higashi)
Files: src/ex_cmds.c, src/testdir/test_help_tagjump.vim,
runtime/doc/eval.txt

Patch 7.4.1729

Problem: The Perl interface cannot use **'print'** operator for writing directly in standard IO.
Solution: Add a minimal implementation of PerlIO Layer feature and try to use it for STDOUT/STDERR. (Damien)
Files: src/if_perl.xs, src/testdir/test_perl.vim

Patch 7.4.1730

Problem: It is not easy to get a character out of a string.
Solution: Add strgetchar() and strcharpart().
Files: src/eval.c, src/testdir/test_expr.vim

Patch 7.4.1731

Problem: Python: turns partial into simple funcref.
Solution: Use partials like partials. (Nikolai Pavlov, closes #734)
Files: runtime/doc/if_pyth.txt, src/eval.c, src/if_py_both.h,
src/if_python.c, src/if_python3.c, src/proto/eval.pro,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.1732

Problem: Folds may close when using autocompleate. (Anmol Sethi)
Solution: Increment/decrement disable_fold. (Christian Brabandt, closes #643)

Files: src/edit.c, src/fold.c, src/globals.h

Patch 7.4.1733

Problem: "make install" doesn't know about cross-compiling. (Christian Neukirchen)

Solution: Add CROSS_COMPILING. (closes #740)

Files: src/configure.in, src/auto/configure, src/config.mk.in, src/Makefile

Patch 7.4.1734 (after 7.4.1730)

Problem: Test fails when not using utf-8.

Solution: Split test in regular and utf-8 part.

Files: src/testdir/test_expr.vim, src/testdir/test_expr_utf8.vim, src/testdir/test_alot_utf8.vim

Patch 7.4.1735

Problem: It is not possible to only see part of the message history. It is not possible to clear messages.

Solution: Add a count to ":messages" and a clear argument. (Yasuhiro Matsumoto)

Files: runtime/doc/message.txt, src/ex_cmds.h, src/message.c, src/testdir/test_messages.vim, src/testdir/test_alot.vim

Patch 7.4.1736 (after 7.4.1731)

Problem: Unused variable.

Solution: Remove it. (Yasuhiro Matsumoto)

Files: src/if_py_both.h

Patch 7.4.1737

Problem: Argument marked as unused is used.

Solution: Remove UNUSED.

Files: src/message.c

Patch 7.4.1738

Problem: Count for ":messages" depends on number of lines.

Solution: Add ADDR_OTHER address type.

Files: src/ex_cmds.h

Patch 7.4.1739

Problem: Messages test fails on MS-Windows.

Solution: Adjust the asserts. Skip the "messages maintainer" line if not showing all messages.

Files: src/message.c, src/testdir/test_messages.vim

Patch 7.4.1740

Problem: syn_cchar defined with matchadd() does not appear if there are no other syntax definitions which matches buffer text.

Solution: Check for startcol. (Ozaki Kiichi, haya14busa, closes #757)

Files: src/screen.c, src/testdir/Make_all.mak, src/testdir/test_alot_utf8.vim, src/testdir/test_match_conceal.in, src/testdir/test_match_conceal.ok, src/testdir/test_matchadd_conceal.vim, src/testdir/test_matchadd_conceal_utf8.vim, src/testdir/test_undolevels.vim

Patch 7.4.1741

Problem: Not testing utf-8 characters.
Solution: Move the right asserts to the test_expr_utf8 test.
Files: src/testdir/test_expr.vim, src/testdir/test_expr_utf8.vim

Patch 7.4.1742

Problem: strgetchar() does not work correctly.
Solution: use mb_cptr2len(). Add a test. (Naruhiko Nishino)
Files: src/eval.c, src/testdir/test_expr_utf8.vim

Patch 7.4.1743

Problem: Clang warns for uninitialized variable. (Michael Jarvis)
Solution: Initialize it.
Files: src/if_py_both.h

Patch 7.4.1744

Problem: Python: Converting a sequence may leak memory.
Solution: Decrement a reference. (Nikolai Pavlov)
Files: src/if_py_both.h

Patch 7.4.1745

Problem: README file is not clear about where to get Vim.
Solution: Add links to github, releases and the Windows installer.
(Suggested by Christian Brabandt)
Files: README.md, README.txt

Patch 7.4.1746

Problem: Memory leak in Perl.
Solution: Decrement the reference count. Add a test. (Damien)
Files: src/if_perl.xs, src/testdir/test_perl.vim

Patch 7.4.1747

Problem: Coverity: missing check for NULL pointer.
Solution: Check for out of memory.
Files: src/if_py_both.h

Patch 7.4.1748

Problem: "gD" does not find match in first column of first line. (Gary Johnson)
Solution: Accept match at the cursor.
Files: src/normal.c, src/testdir/test_goto.vim, src/testdir/test_alot.vim

Patch 7.4.1749

Problem: When using GTK 3.20 there are a few warnings.
Solution: Use new functions when available. (Kazunobu Kuriyama)
Files: src/gui_beval.c src/gui_gtk_x11.c

Patch 7.4.1750

Problem: When a buffer gets updated while in command line mode, the screen may be messed up.
Solution: Postpone the redraw when the screen is scrolled.
Files: src/channel.c

Patch 7.4.1751

Problem: Crash when **'tagstack'** is off. (Dominique Pelle)
Solution: Fix it. (Hirohito Higashi)
Files: src/tag.c, src/testdir/test_alot.vim, src/testdir/test_tagjump.vim

Patch 7.4.1752

Problem: When adding to the quickfix list the current position is reset.
Solution: Do not reset the position when not needed. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1753

Problem: "noinsert" in **'completeopt'** is sometimes ignored.
Solution: Set the variables when the **'completeopt'** was set. (Ozaki Kiichi)
Files: src/edit.c, src/option.c, src/proto/edit.pro

Patch 7.4.1754

Problem: When **'filetype'** was set and reloading a buffer which does not cause it to be set, the syntax isn't loaded. (KillTheMule)
Solution: Remember whether the FileType event was fired and fire it if not. (Anton Lindqvist, closes #747)
Files: src/fileio.c, src/testdir/test_syntax.vim

Patch 7.4.1755

Problem: When using getreg() on a non-existing register a NULL list is returned. (Bjorn Linse)
Solution: Allocate an empty list. Add a test.
Files: src/eval.c, src/testdir/test_expr.vim

Patch 7.4.1756

Problem: "dll" options are not expanded.
Solution: Expand environment variables. (Ozaki Kiichi)
Files: src/option.c, src/testdir/test_alot.vim, src/testdir/test_expand_dllpath.vim

Patch 7.4.1757

Problem: When using complete() it may set **'modified'** even though nothing was inserted.
Solution: Use Down/Up instead of Next/Previous match. (Shougo Matsu, closes #745)
Files: src/edit.c

Patch 7.4.1758

Problem: Triggering CursorHoldI when in **CTRL-X** mode causes problems.
Solution: Do not trigger CursorHoldI in **CTRL-X** mode. Add "!" flag to feedkeys() (test with that didn't work though).
Files: src/edit.c, src/eval.c

Patch 7.4.1759

Problem: When using feedkeys() in a timer the inserted characters are not used right away.
Solution: Break the wait loop when characters have been added to typebuf. use this for testing CursorHoldI.
Files: src/gui.c, src/os_win32.c, src/os_unix.c, src/testdir/test_autocmd.vim

Patch 7.4.1760 (after 7.4.1759)

Problem: Compiler warning for unused variable.
Solution: Add #ifdef. (John Marriott)
Files: src/os_win32.c

Patch 7.4.1761

Problem: Coverity complains about ignoring return value.
Solution: Add "(void)" to get rid of the warning.
Files: src/eval.c

Patch 7.4.1762

Problem: Coverity: useless assignments.
Solution: Remove them.
Files: src/search.c

Patch 7.4.1763

Problem: Coverity: useless assignment.
Solution: Add #if 0.
Files: src/spell.c

Patch 7.4.1764

Problem: C++ style comment. (Ken Takata)
Solution: Finish the work started here: don't call perror() when stderr
isn't working.
Files: src/os_unix.c

Patch 7.4.1765

Problem: Undo options are not together in the options window.
Solution: Put them together. (Gary Johnson)
Files: runtime/optwin.vim

Patch 7.4.1766

Problem: Building instructions for MS-Windows are outdated.
Solution: Mention setting SDK_INCLUDE_DIR. (Ben Franklin, closes #771) Move
outdated instructions further down.
Files: src/INSTALLpc.txt

Patch 7.4.1767

Problem: When installing Vim on a GTK system the icon cache is not updated.
Solution: Update the GTK icon cache when possible. (Kazunobu Kuriyama)
Files: src/Makefile, src/configure.in, src/config.mk.in,
src/auto/configure

Patch 7.4.1768

Problem: Arguments of setqflist() are not checked properly.
Solution: Add better checks, add a test. (Nikolai Pavlov, Hirohito Higashi,
closes #661)
Files: src/eval.c, src/testdir/test_quickfix.vim

Patch 7.4.1769

Problem: No "closed", "errors" and "encoding" attribute on Python output.
Solution: Add attributes and more tests. (Roland Puntaier, closes #622)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c,

src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.1770

Problem: Cannot use true color in the terminal.
Solution: Add the **'guicolors'** option. (Nikolai Pavlov)
Files: runtime/doc/options.txt, runtime/doc/term.txt,
runtime/doc/various.txt, src/auto/configure, src/config.h.in,
src/configure.in, src/eval.c, src/globals.h, src/hardcopy.c,
src/option.c, src/option.h, src/proto/term.pro, src/screen.c,
src/structs.h, src/syntax.c, src/term.c, src/term.h,
src/version.c, src/vim.h

Patch 7.4.1771 (after 7.4.1768)

Problem: Warning for unused variable.
Solution: Add #ifdef. (John Marriott)
Files: src/eval.c

Patch 7.4.1772 (after 7.4.1767)

Problem: Installation fails when \$GTK_UPDATE_ICON_CACHE is empty.
Solution: Add quotes. (Kazunobu Kuriyama)
Files: src/Makefile

Patch 7.4.1773 (after 7.4.1770)

Problem: Compiler warnings. (Dominique Pelle)
Solution: Add UNUSED. Add type cast. Avoid a buffer overflow.
Files: src/syntax.c, src/term.c

Patch 7.4.1774 (after 7.4.1770)

Problem: Cterm true color feature has warnings.
Solution: Add type casts.
Files: src/screen.c, src/syntax.c, src/term.c

Patch 7.4.1775

Problem: The rgb.txt file is not installed.
Solution: Install the file. (Christian Brabandt)
Files: src/Makefile

Patch 7.4.1776

Problem: Using wrong buffer length.
Solution: use the right name. (Kazunobu Kuriyama)
Files: src/term.c

Patch 7.4.1777

Problem: Newly added features can escape the sandbox.
Solution: Add checks for restricted and secure. (Yasuhiro Matsumoto)
Files: src/eval.c

Patch 7.4.1778

Problem: When using the term truecolor feature, the t_8f and t_8b termcap options are not set by default.
Solution: Move the values to before BT_EXTRA_KEYS. (Christian Brabandt)
Files: src/term.c

Patch 7.4.1779

Problem: Using negative index in strchrpart(). (Yegappan Lakshmanan)
Solution: Assume single byte when using a negative index.
Files: src/eval.c

Patch 7.4.1780

Problem: Warnings reported by cppcheck.
Solution: Fix the warnings. (Dominique Pelle)
Files: src/ex_cmds2.c, src/json.c, src/misc1.c, src/ops.c,
src/regexp_nfa.c

Patch 7.4.1781

Problem: synIDattr() does not respect **'guicolors'**.
Solution: Change the condition for the mode. (Christian Brabandt)
Files: src/eval.c

Patch 7.4.1782

Problem: strchrpart() does not work properly with some multibyte characters.
Solution: Use mb_cptr2len() instead of mb_char2len(). (Hirohito Higashi)
Files: src/eval.c, src/testdir/test_expr_utf8.vim

Patch 7.4.1783

Problem: The old regexp engine doesn't handle character classes correctly. (Manuel Ortega)
Solution: Use regmbc() instead of regc(). Add a test.
Files: src/regexp.c, src/testdir/test_regexp_utf8.vim

Patch 7.4.1784

Problem: The termtruecolor feature is enabled differently from many other features.
Solution: Enable the termtruecolor feature for the big build, not through configure.
Files: src/configure.in, src/config.h.in, src/auto/configure,
src/feature.h

Patch 7.4.1785 (after 7.4.1783)

Problem: Regexp test fails on windows.
Solution: set **'isprint'** to the right value for testing.
Files: src/testdir/test_regexp_utf8.vim

Patch 7.4.1786

Problem: Compiled-in colors do not match rgb.txt.
Solution: Use the rgb.txt colors. (Kazunobu Kuriyama)
Files: src/term.c

Patch 7.4.1787

Problem: When a job ends the close callback is invoked before other callbacks. On Windows the close callback is not called.
Solution: First invoke out/err callbacks before the close callback. Make the close callback work on Windows.
Files: src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, src/testdir/test_channel_pipe.py

Patch 7.4.1788

Problem: NSIS script is missing packages.
Solution: Add the missing directories. (Ken Takata)
Files: nsis/gvim.nsi

Patch 7.4.1789

Problem: Cannot use ch_read() in the close callback.
Solution: Do not discard the channel if there is readahead. Do not discard readahead if there is a close callback.
Files: src/eval.c, src/channel.c, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 7.4.1790

Problem: Leading white space in a job command matters. (Andrew Stewart)
Solution: Skip leading white space.
Files: src/os_unix.c

Patch 7.4.1791

Problem: Channel could be garbage collected too early.
Solution: Don't free a channel or remove it from a job when it is still useful.
Files: src/channel.c

Patch 7.4.1792

Problem: Color name decoding is implemented several times.
Solution: Move it to term.c. (Christian Brabandt)
Files: src/gui_mac.c, src/gui_photon.c, src/gui_w32.c, src/proto/term.pro, src/term.c

Patch 7.4.1793

Problem: Some character classes may differ between systems. On OS/X the regexp test fails.
Solution: Make this less dependent on the system. (idea by Kazunobu Kuriyama)
Files: src/regexp.c, src/regexp_nfa.c

Patch 7.4.1794 (after 7.4.1792)

Problem: Can't build on MS-Windows.
Solution: Add missing declaration.
Files: src/gui_w32.c

Patch 7.4.1795

Problem: Compiler warning for redefining RGB. (John Marriott)
Solution: Rename it to TORGB.
Files: src/term.c

Patch 7.4.1796 (after 7.4.1795)

Problem: Colors are wrong on MS-Windows. (Christian Robinson)
Solution: Use existing RGB macro if it exists. (Ken Takata)
Files: src/term.c

Patch 7.4.1797

Problem: Warning from Windows 64 bit compiler.
Solution: Change int to size_t. (Mike Williams)
Files: src/term.c

Patch 7.4.1798

Problem: Still compiler warning for unused return value. (Charles Campbell)
Solution: Assign to ignoredp.
Files: src/term.c

Patch 7.4.1799

Problem: **'guicolors'** is a confusing option name.
Solution: Use **'termguicolors'** instead. (Hirohito Higashi, Ken Takata)
Files: runtime/doc/options.txt, runtime/doc/term.txt,
runtime/doc/variables.txt, runtime/syntax/dircolors.vim, src/eval.c,
src/feature.h, src/globals.h, src/hardcopy.c, src/option.c,
src/option.h, src/proto/term.pro, src/screen.c, src/structs.h,
src/syntax.c, src/term.c, src/version.c, src/vim.h

Patch 7.4.1800 (after 7.4.1799)

Problem: Unnecessary #ifdef.
Solution: Just use USE_24BIT. (Ken Takata)
Files: src/syntax.c

Patch 7.4.1801

Problem: Make uninstall leaves file behind.
Solution: Delete rgb.txt. (Kazunobu Kuriyama)
Files: src/Makefile

Patch 7.4.1802

Problem: Quickfix doesn't handle long lines well, they are split.
Solution: Drop characters after a limit. (Anton Lindqvist)
Files: src/quickfix.c, src/testdir/test_quickfix.vim,
src/testdir/samples/quickfix.txt

Patch 7.4.1803

Problem: GTK3 doesn't handle menu separators properly.
Solution: Use gtk_separator_menu_item_new(). (Kazunobu Kuriyama)
Files: src/gui_gtk.c

Patch 7.4.1804

Problem: Can't use Vim as MANPAGER.
Solution: Add manpager.vim. (Enno Nagel, closes #491)
Files: runtime/doc/filetype.txt, runtime/plugin/manpager.vim

Patch 7.4.1805

Problem: Running tests in shadow dir fails.
Solution: Link the samples directory
Files: src/Makefile

Patch 7.4.1806

Problem: **'termguicolors'** option missing from the options window.
Solution: Add the entry.
Files: runtime/optwin.vim

Patch 7.4.1807

Problem: Test_out_close_cb sometimes fails.
Solution: Always write DETACH to out, not err.

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1808 (after 7.4.1806)

Problem: Using wrong feature name to check for 'termguicolors'.

Solution: Use the right feature name. (Ken Takata)

Files: runtime/optwin.vim

Patch 7.4.1809 (after 7.4.1808)

Problem: Using wrong short option name for 'termguicolors'.

Solution: Use the option name.

Files: runtime/optwin.vim

Patch 7.4.1810

Problem: Sending DETACH after a channel was closed isn't useful.

Solution: Only add DETACH for a netbeans channel.

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1811

Problem: Netbeans channel gets garbage collected.

Solution: Set reference in nb_channel.

Files: src/eval.c, src/netbeans.c, src/proto/netbeans.pro

Patch 7.4.1812

Problem: Failure on startup with Athena and Motif.

Solution: Check for INVALIDCOLOR. (Kazunobu Kuriyama)

Files: src/syntax.c, src/vim.h

Patch 7.4.1813

Problem: Memory access error when running test_quickfix.

Solution: Allocate one more byte. (Yegappan Lakshmanan)

Files: src/quickfix.c

Patch 7.4.1814

Problem: A channel may be garbage collected while it's still being used by a job. (James McCoy)

Solution: Mark the channel as used if the job is still used. Do the same for channels that are still used.

Files: src/eval.c, src/channel.c, src/proto/channel.pro

Patch 7.4.1815

Problem: Compiler warnings for unused variables. (Ajit Thakkar)

Solution: Add a dummy initialization. (Yasuhiro Matsumoto)

Files: src/quickfix.c

Patch 7.4.1816

Problem: Looping over a null list throws an error.

Solution: Skip over the for loop.

Files: src/eval.c, src/testdir/test_expr.vim

Patch 7.4.1817

Problem: The screen is not updated if a callback is invoked when closing a channel.

Solution: Invoke redraw_after_callback().

Files: src/channel.c

Patch 7.4.1818

Problem: Help completion adds @en to all matches except the first one.
Solution: Remove "break", go over all items.
Files: src/ex_getln.c

Patch 7.4.1819

Problem: Compiler warnings when sprintf() is a macro.
Solution: Don't interrupt sprintf() with an #ifdef. (Michael Jarvis, closes #788)
Files: src/fileio.c, src/tag.c, src/term.c

Patch 7.4.1820

Problem: Removing language from help tags too often.
Solution: Only remove @en when not needed. (Hirohito Higashi)
Files: src/ex_getln.c, src/testdir/test_help_tagjump.vim

Patch 7.4.1821 (after 7.4.1820)

Problem: Test fails on MS-Windows.
Solution: Sort the completion results.
Files: src/testdir/test_help_tagjump.vim

Patch 7.4.1822

Problem: Redirecting stdout of a channel to "null" doesn't work. (Nicola)
Solution: Correct the file descriptor number.
Files: src/os_unix.c

Patch 7.4.1823

Problem: Warning from 64 bit compiler.
Solution: Add type cast. (Mike Williams)
Files: src/quickfix.c

Patch 7.4.1824

Problem: When a job is no longer referenced and does not have an exit callback the process may hang around in defunct state. (Nicola)
Solution: Call job_status() if the job is running and won't get freed because it might still be useful.
Files: src/channel.c

Patch 7.4.1825

Problem: When job writes to buffer nothing is written. (Nicola)
Solution: Do not discard a channel before writing is done.
Files: src/channel.c

Patch 7.4.1826

Problem: Callbacks are invoked when it's not safe. (Andrew Stewart)
Solution: When a channel is to be closed don't invoke callbacks right away, wait for a safe moment.
Files: src/structs.h, src/channel.c

Patch 7.4.1827

Problem: No error when invoking a callback when it's not safe.
Solution: Add an error message. Avoid the error when freeing a channel.
Files: src/structs.h, src/channel.c

Patch 7.4.1828

Problem: May try to access buffer that's already freed.
Solution: When freeing a buffer remove it from any channel.
Files: src/buffer.c, src/channel.c, src/proto/channel.pro

Patch 7.4.1829 (after 7.4.1828)

Problem: No message on channel log when buffer was freed.
Solution: Log a message.
Files: src/channel.c

Patch 7.4.1830

Problem: non-antialiased misnamed.
Solution: Use NONANTIALIASED and NONANTIALIASED_QUALITY. (Kim Brouer, closes #793)
Files: src/os_mswin.c, runtime/doc/options.txt

Patch 7.4.1831

Problem: When timer_stop() is called with a string there is no proper error message.
Solution: Require getting a number. (Bjorn Linse)
Files: src/eval.c

Patch 7.4.1832

Problem: Memory leak in debug commands.
Solution: Free memory before overwriting the pointer. (hint by Justin Keyes)
Files: src/ex_cmds2.c

Patch 7.4.1833

Problem: Cannot use an Ex command for **'keywordprg'**.
Solution: Accept an Ex command. (Nelo-Thara Wallus)
Files: src/normal.c, runtime/doc/options.txt

Patch 7.4.1834

Problem: Possible crash when conceal is active.
Solution: Check for the screen to be valid when redrawing a line.
Files: src/screen.c

Patch 7.4.1835

Problem: When splitting and closing a window the status height changes.
Solution: Compute the frame height correctly. (Hirohito Higashi)
Files: src/window.c, src/testdir/test_alot.vim, src/testdir/test_window_cmd.vim

Patch 7.4.1836

Problem: When using a partial on a dictionary it always gets bound to that dictionary.
Solution: Make a difference between binding a function to a dictionary explicitly or automatically.
Files: src/structs.h, src/eval.c, src/testdir/test_partial.vim, runtime/doc/eval.txt

Patch 7.4.1837

Problem: The BufUnload event is triggered twice, when :bunload is used with

``bufhidden`` set to ``unload`` or ``delete``.
Solution: Do not trigger the event when `ml_mfp` is NULL. (Hirohito Higashi)
Files: `src/buffer.c`, `src/testdir/test_autocmd.vim`

Patch 7.4.1838

Problem: Functions specifically for testing do not sort together.
Solution: Rename `garbagecollect_for_testing()` to `test_garbagecollect_now()`.
Add `test_null_list()`, `test_null_dict()`, etc.
Files: `src/eval.c`, `src/testdir/test_expr.vim`,
`src/testdir/test_channel.vim`, `runtime/doc/eval.txt`

Patch 7.4.1839

Problem: Cannot get the items stored in a partial.
Solution: Support using `get()` on a partial.
Files: `src/eval.c`, `src/testdir/test_partial.vim`, `runtime/doc/eval.txt`

Patch 7.4.1840

Problem: When using packages an "after" directory cannot be used.
Solution: Add the "after" directory of the package to `'runtimepath'` if it exists.
Files: `src/ex_cmds2.c`, `src/testdir/test_packadd.vim`

Patch 7.4.1841

Problem: The code to reallocate the buffer used for quickfix is repeated.
Solution: Move the code to a function. (Yegappan Lakshmanan, closes #831)
Files: `src/quickfix.c`, `src/testdir/test_quickfix.vim`

Patch 7.4.1842 (after 7.4.1839)

Problem: `get()` works for Partial but not for Funcref.
Solution: Accept Funcref. Also return the function itself. (Nikolai Pavlov)
Files: `src/eval.c`, `src/testdir/test_partial.vim`, `runtime/doc/eval.txt`

Patch 7.4.1843

Problem: Tests involving Python are flaky.
Solution: Set the `pt_auto` field. Add tests. (Nikolai Pavlov)
Files: `runtime/doc/if_pyth.txt`, `src/if_py_both.h`, `src/testdir/test86.in`,
`src/testdir/test86.ok`, `src/testdir/test87.in`,
`src/testdir/test87.ok`

Patch 7.4.1844

Problem: Using old function name in comment. More functions should start with `test_`.
Solution: Rename function in comment. (Hirohito Higashi) Rename `disable_char_avail_for_testing()` to `test_disable_char_avail()`.
And `alloc_fail()` to `test_alloc_fail()`.
Files: `src/eval.c`, `src/getchar.c`, `src/testdir/runtest.vim`,
`src/testdir/test_cursor_func.vim`, `src/testdir/test_quickfix.vim`,
`runtime/doc/eval.txt`

Patch 7.4.1845

Problem: Mentioning NetBeans when reading from channel. (Ramel Eshed)
Solution: Make the text more generic.
Files: `src/channel.c`

Patch 7.4.1846

Problem: Ubsan detects a multiplication overflow.
Solution: Don't use orig_mouse_time when it's zero. (Dominique Pelle)
Files: src/term.c

Patch 7.4.1847

Problem: Getting an item from a NULL dict crashes. Setting a register to a NULL list crashes. (Nikolai Pavlov, issue #768) Comparing a NULL dict with a NULL dict fails.
Solution: Properly check for NULL.
Files: src/eval.c, src/testdir/test_expr.vim

Patch 7.4.1848

Problem: Can't build with Strawberry Perl 5.24.
Solution: Define S_SvREFCNT_dec() if needed. (Damien, Ken Takata)
Files: src/if_perl.xs

Patch 7.4.1849

Problem: Still trying to read from channel that is going to be closed. (Ramel Eshed)
Solution: Check if ch_to_be_closed is set.
Files: src/channel.c

Patch 7.4.1850

Problem: GUI freezes when using a job. (Shougo Matsu)
Solution: Unregister the channel when there is an input error.
Files: src/channel.c

Patch 7.4.1851

Problem: test_syn_attr fails when using the GUI. (Dominique Pelle)
Solution: Escape the font name properly.
Files: src/testdir/test_syn_attr.vim

Patch 7.4.1852

Problem: Unix: Cannot run all tests with the GUI.
Solution: Add the "testgui" target.
Files: src/Makefile, src/testdir/Makefile

Patch 7.4.1853

Problem: Crash when job and channel are in the same dict while using partials. (Luc Hermitte)
Solution: Do not decrement the channel reference count too early.
Files: src/channel.c

Patch 7.4.1854

Problem: When setting **'termguicolors'** the Ignore highlighting doesn't work. (Charles Campbell)
Solution: Handle the color names "fg" and "bg" when the GUI isn't running and no colors are specified, fall back to black and white.
Files: src/syntax.c

Patch 7.4.1855

Problem: Valgrind reports memory leak for job that is not freed.
Solution: Free all jobs on exit. Add test for failing job.

Files: src/channel.c, src/misc2.c, src/proto/channel.pro,
src/testdir/test_partial.vim

Patch 7.4.1856 (after 7.4.1855)

Problem: failing job test fails on MS-Windows.
Solution: Expect "fail" status instead of "dead".
Files: src/testdir/test_partial.vim

Patch 7.4.1857

Problem: When a channel appends to a buffer that is '**nomodifiable**' there is an error but appending is done anyway.
Solution: Add the '**modifiable**' option. Refuse to write to a '**nomodifiable**' when the value is 1.
Files: src/structs.h, src/channel.c, src/testdir/test_channel.vim,
runtime/doc/channel.txt

Patch 7.4.1858

Problem: When a channel writes to a buffer it doesn't find a buffer by the short name but re-uses it anyway.
Solution: Find buffer also by the short name.
Files: src/channel.c, src/buffer.c, src/vim.h

Patch 7.4.1859

Problem: Cannot use a function reference for "exit_cb".
Solution: Use get_callback(). (Yegappan Lakshmanan)
Files: src/channel.c, src/structs.h

Patch 7.4.1860

Problem: Using a partial for timer_start() may cause a crash.
Solution: Set the copyID in timer objects. (Ozaki Kiichi)
Files: src/testdir/test_timers.vim, src/eval.c, src/ex_cmds2.c,
src/proto/ex_cmds2.pro

Patch 7.4.1861

Problem: Compiler warnings with 64 bit compiler.
Solution: Change int to size_t. (Mike Williams)
Files: src/ex_cmds2.c

Patch 7.4.1862

Problem: string() with repeated argument does not give a result usable by eval().
Solution: Refactor echo_string and tv2string(), moving the common part to echo_string_core(). (Ken Takata)
Files: src/eval.c, src/testdir/test_viml.vim, src/testdir/test86.ok,
src/testdir/test87.ok

Patch 7.4.1863

Problem: Compiler warnings on Win64.
Solution: Adjust types, add type casts. (Ken Takata)
Files: src/if_mzsch.c, src/if_perl.xs, src/if_ruby.c, src/version.c

Patch 7.4.1864

Problem: Python: encoding error with Python 2.
Solution: Use "getcwdu" instead of "getcwd". (Ken Takata)

Files: src/if_py_both.h

Patch 7.4.1865

Problem: Memory leaks in test49. (Dominique Pelle)

Solution: Use NULL instead of an empty string.

Files: src/eval.c

Patch 7.4.1866

Problem: Invalid memory access when exiting with EXITFREE defined.
(Dominique Pelle)

Solution: Set "really_exiting" and skip error messages.

Files: src/misc2.c, src/eval.c

Patch 7.4.1867

Problem: Memory leak in test_matchstrpos.

Solution: Free the string before overwriting. (Yegappan Lakshmanan)

Files: src/eval.c

Patch 7.4.1868

Problem: Setting really_exiting causes memory leaks to be reported.

Solution: Add the in_free_all_mem flag.

Files: src/globals.h, src/misc2.c, src/eval.c

Patch 7.4.1869

Problem: Can't build with old version of Perl.

Solution: Define PERLIO_FUNCS_DECL. (Tom G. Christensen)

Files: src/if_perl.xs

Patch 7.4.1870 (after 7.4.1863)

Problem: One more Win64 compiler warning.

Solution: Change declared argument type. (Ken Takata)

Files: src/if_mzsch.c

Patch 7.4.1871

Problem: Appending to the quickfix list while the quickfix window is open
is very slow.

Solution: Do not delete all the lines, only append the new ones. Avoid
using a window while updating the list. (closes #841)

Files: src/quickfix.c

Patch 7.4.1872

Problem: Still build problem with old version of Perl.

Solution: Also define SvREFCNT_inc_void_NN if needed. (Tom G. Christensen)

Files: src/if_perl.xs

Patch 7.4.1873

Problem: When a callback adds a timer the GUI doesn't use it until later.
(Ramel Eshed)

Solution: Return early if a callback adds a timer.

Files: src/ex_cmds2.c, src/gui_gtk_x11.c, src/gui_w32.c, src/gui_x11.c,
src/globals.h

Patch 7.4.1874

Problem: Unused variable in Win32 code.

Solution: Remove it. (Mike Williams)
Files: src/gui_w32.c

Patch 7.4.1875

Problem: Comparing functions and partials doesn't work well.
Solution: Add tests. (Nikolai Pavlov) Compare the dict and arguments in the partial. (closes #813)
Files: src/eval.c, src/testdir/test_partial.vim

Patch 7.4.1876

Problem: Typing "k" at the hit-enter prompt has no effect.
Solution: Don't assume recursive use of the prompt if a character was typed. (Hirohito Higashi)
Files: src/message.c

Patch 7.4.1877

Problem: No test for invoking "close_cb" when writing to a buffer.
Solution: Add using close_cb to a test case.
Files: src/testdir/test_channel.vim

Patch 7.4.1878

Problem: Whether a job has exited isn't detected until a character is typed. After calling exit_cb the cursor is in the wrong place.
Solution: Don't wait forever for a character to be typed when there is a pending job. Update the screen if needed after calling exit_cb.
Files: src/os_unix.c, src/channel.c, src/proto/channel.pro

Patch 7.4.1879 (after 7.4.1877)

Problem: Channel test is flaky.
Solution: Wait for close_cb to be invoked.
Files: src/testdir/test_channel.vim

Patch 7.4.1880

Problem: MS-Windows console build defaults to not having +channel.
Solution: Include the channel feature if building with huge features.
Files: src/Make_mvc.mak

Patch 7.4.1881

Problem: Appending to a long quickfix list is slow.
Solution: Add qf_last.
Files: src/quickfix.c

Patch 7.4.1882

Problem: Check for line break at end of line wrong. (Dominique Pelle)
Solution: Correct the logic.
Files: src/quickfix.c

Patch 7.4.1883

Problem: Cppcheck found 2 incorrect printf formats.
Solution: Use %ld and %lx. (Dominique Pelle)
Files: src/VisVim/Commands.cpp, src/gui_mac.c

Patch 7.4.1884

Problem: Updating marks in a quickfix list is very slow when the list is

long.
Solution: Only update marks if the buffer has a quickfix entry.
Files: src/structs.h, src/quickfix.c

Patch 7.4.1885

Problem: MinGW console build defaults to not having +channel.
Solution: Include the channel feature if building with huge features. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.1886

Problem: When waiting for a character is interrupted by receiving channel data and the first character of a mapping was typed, the mapping times out. (Ramel Eshed)
Solution: When dealing with channel data don't return from mch_inchar().
Files: src/getchar.c, src/proto/getchar.pro, src/os_unix.c

Patch 7.4.1887

Problem: When receiving channel data '**updatetime**' is not respected.
Solution: Recompute the waiting time after being interrupted.
Files: src/os_unix.c

Patch 7.4.1888

Problem: Wrong computation of remaining wait time in RealWaitForChar()
Solution: Remember the original waiting time.
Files: src/os_unix.c

Patch 7.4.1889

Problem: When umask is set to 0177 Vim can't create temp files. (Lcd)
Solution: Also correct umask when using mkdtemp().
Files: src/fileio.c

Patch 7.4.1890

Problem: GUI: When channel data is received the cursor blinking is interrupted. (Ramel Eshed)
Solution: Don't update the cursor when it is blinking.
Files: src/screen.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/gui_mac.c, src/proto/gui_mac.pro, src/gui_photon.c, src/proto/gui_photon.pro, src/gui_w32.c, src/proto/gui_w32.pro, src/gui_x11.c, src/proto/gui_x11.pro

Patch 7.4.1891

Problem: Channel reading very long lines is slow.
Solution: Collapse multiple buffers until a NL is found.
Files: src/channel.c, src/netbeans.c, src/proto/channel.pro, src/structs.h

Patch 7.4.1892

Problem: balloon eval only gets the window number, not the ID.
Solution: Add v:beval_winid.
Files: src/eval.c, src/gui_beval.c, src/vim.h

Patch 7.4.1893

Problem: Cannot easily get the window ID for a buffer.

Solution: Add bufwinid().
Files: src/eval.c, runtime/doc/eval.txt

Patch 7.4.1894

Problem: Cannot get the window ID for a mouse click.
Solution: Add v:mouse_winid.
Files: src/eval.c, src/vim.h, runtime/doc/eval.txt

Patch 7.4.1895

Problem: Cannot use a window ID where a window number is expected.
Solution: Add LOWEST_WIN_ID, so that the window ID can be used where a number is expected.
Files: src/window.c, src/eval.c, src/vim.h, runtime/doc/eval.txt, src/testdir/test_window_id.vim

Patch 7.4.1896

Problem: Invoking mark_adjust() when adding a new line below the last line is pointless.
Solution: Skip calling mark_adjust() when appending below the last line.
Files: src/misc1.c, src/ops.c

Patch 7.4.1897

Problem: Various typos, long lines and style mistakes.
Solution: Fix the typos, wrap lines, improve style.
Files: src/buffer.c, src/ex_docmd.c, src/getchar.c, src/option.c, src/main.aap, src/testdir/README.txt, src/testdir/test_reftime.vim, src/testdir/test_tagjump.vim, src/INSTALL, src/config.aap.in, src/if_mzsch.c

Patch 7.4.1898

Problem: User commands don't support modifiers.
Solution: Add the <mods> item. (Yegappan Lakshmanan, closes #829)
Files: runtime/doc/map.txt, src/ex_docmd.c, src/testdir/Make_all.mak, src/testdir/test_usercommands.vim

Patch 7.4.1899

Problem: GTK 3: cursor blinking doesn't work well.
Solution: Instead of gui_gtk_window_clear() use gui_mch_clear_block(). (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 7.4.1900

Problem: Using **CTRL-]** in the help on "{address}." doesn't work.
Solution: Recognize an item in {}. (Hirohito Higashi, closes #814)
Files: src/ex_cmds.c, src/testdir/test_help_tagjump.vim

Patch 7.4.1901

Problem: Win32: the "Disabled" menu items would appear enabled.
Solution: Use submenu_id if there is a parent. (Shane Harper, closes #834)
Files: src/gui_w32.c

Patch 7.4.1902

Problem: No test for collapsing buffers for a channel. Some text is lost.
Solution: Add a simple test. Set rq_bufrlen correctly.

Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel_pipe.py

Patch 7.4.1903

Problem: When writing viminfo merging current history with history in viminfo may drop recent history entries.

Solution: Add new format for viminfo lines, use it for history entries. Use a timestamp for ordering the entries. Add test_settime(). Add the viminfo version. Does not do merging on timestamp yet.

Files: src/eval.c, src/ex_getln.c, src/ex_cmds.c, src/structs.h,
src/globals.h, src/proto/ex_cmds.pro, src/proto/ex_getln.pro,
src/testdir/test_viminfo.vim

Patch 7.4.1904 (after 7.4.1903)

Problem: Build fails.

Solution: Add missing changes.

Files: src/vim.h

Patch 7.4.1905 (after 7.4.1903)

Problem: Some compilers can't handle a double semicolon.

Solution: Remove one semicolon.

Files: src/ex_cmds.c

Patch 7.4.1906

Problem: Collapsing channel buffers and searching for NL does not work properly. (Xavier de Gaye, Ramel Eshed)

Solution: Do not assume the buffer contains a NUL or not. Change NUL bytes to NL to avoid the string is truncated.

Files: src/channel.c, src/netbeans.c, src/proto/channel.pro

Patch 7.4.1907

Problem: Warnings from 64 bit compiler.

Solution: Change type to size_t. (Mike Williams)

Files: src/ex_cmds.c

Patch 7.4.1908

Problem: Netbeans uses uninitialized pointer and freed memory.

Solution: Set "buffer" at the right place (hint by Ken Takata)

Files: src/netbeans.c

Patch 7.4.1909

Problem: Doubled semicolons.

Solution: Reduce to one. (Dominique Pelle)

Files: src/dosinst.c, src/fold.c, src/gui_gtk_x11.c, src/gui_w32.c,
src/main.c, src/misc2.c

Patch 7.4.1910

Problem: Tests using external command to delete directory.

Solution: Use delete().

Files: src/testdir/test17.in, src/testdir/test73.in,
src/testdir/test_getcwd.in

Patch 7.4.1911

Problem: Recent history lines may be lost when exiting Vim.

Solution: Merge history using the timestamp.
Files: src/ex_getln.c, src/ex_cmds.c, src/vim.h, src/proto/ex_getln.pro, src/testdir/test_viminfo.vim

Patch 7.4.1912

Problem: No test for using setqflist() on an older quickfix list.
Solution: Add a couple of tests.
Files: src/testdir/test_quickfix.vim

Patch 7.4.1913

Problem: When ":doautocmd" is used modelines are used even when no autocommands were executed. (Daniel Hahler)
Solution: Skip processing modelines. (closes #854)
Files: src/fileio.c, src/ex_cmds.c, src/ex_docmd.c, src/proto/fileio.pro

Patch 7.4.1914

Problem: Executing autocommands while using the signal stack has a high chance of crashing Vim.
Solution: Don't invoke autocommands when on the signal stack.
Files: src/os_unix.c

Patch 7.4.1915

Problem: The effect of the PopupMenu autocommand isn't directly visible.
Solution: Call gui_update_menus() before displaying the popup menu. (Shane Harper, closes #855)
Files: src/menu.c

Patch 7.4.1916 (after 7.4.1906)

Problem: No proper test for what 7.4.1906 fixes.
Solution: Add a test for reading many lines.
Files: src/testdir/test_channel.vim

Patch 7.4.1917

Problem: History lines read from viminfo in different encoding than when writing are not converted.
Solution: Convert the history lines.
Files: src/ex_cmds.c, src/testdir/test_viminfo.vim

Patch 7.4.1918

Problem: Not enough testing for parsing viminfo lines.
Solution: Add test with viminfo lines in bad syntax. Fix memory leak.
Files: src/ex_cmds.c, src/ex_getln.c, src/testdir/test_viminfo.vim

Patch 7.4.1919

Problem: Register contents is not merged when writing viminfo.
Solution: Use timestamps for register contents.
Files: src/ops.c, src/ex_getln.c, src/ex_cmds.c, src/proto/ex_cmds.pro, src/proto/ex_getln.pro, src/proto/ops.pro, src/vim.h

Patch 7.4.1920 (after 7.4.1919)

Problem: Missing test changes.
Solution: Update viminfo test.
Files: src/testdir/test_viminfo.vim

Patch 7.4.1921 (after 7.4.1919)

Problem: vim_time() not included when needed.
Solution: Adjust #ifdef.
Files: src/ex_cmds.c

Patch 7.4.1922

Problem: Ruby 2.4.0 unifies Fixnum and Bignum into Integer.
Solution: Use rb_cInteger. (Weiyong Mao)
Files: src/if_ruby.c

Patch 7.4.1923

Problem: Command line editing is not tested much.
Solution: Add tests for expanding the file name and 'wildmenu'.
Files: src/testdir/test_cmdline.vim, src/testdir/Make_all.mak

Patch 7.4.1924

Problem: Missing "void" for functions without argument.
Solution: Add "void". (Hirohito Higashi)
Files: src/channel.c, src/edit.c, src/ex_cmds2.c, src/ops.c, src/screen.c

Patch 7.4.1925

Problem: Viminfo does not merge file marks properly.
Solution: Use a timestamp. Add the :clearjumps command.
Files: src/mark.c, src/ex_cmds.c, src/ex_docmd.c, src/proto/mark.pro,
src/structs.h, src/vim.h, src/ex_cmds.h,
src/testdir/test_viminfo.vim

Patch 7.4.1926

Problem: Possible crash with many history items.
Solution: Avoid the index going past the last item.
Files: src/ex_getln.c

Patch 7.4.1927

Problem: Compiler warning for signed/unsigned.
Solution: Add type cast.
Files: src/if_mzsch.c

Patch 7.4.1928

Problem: Overwriting pointer argument.
Solution: Assign to what it points to. (Dominique Pelle)
Files: src/fileio.c

Patch 7.4.1929

Problem: Inconsistent indenting and weird name.
Solution: Fix indent, make name all upper case. (Ken Takata)
Files: src/if_ruby.c

Patch 7.4.1930

Problem: Can't build without +spell but with +quickfix. (Charles)
Solution: Add better #ifdef around ml_append_buf(). (closes #864)
Files: src/memline.c

Patch 7.4.1931

Problem: Using both old and new style file mark lines from viminfo.

Solution: Skip the old style lines if the viminfo file was written with a [Vim version that](#) supports the new style.
Files: src/ex_cmds.c

Patch 7.4.1932

Problem: When writing viminfo the jumplist is not merged with the one in the viminfo file.
Solution: Merge based on timestamp.
Files: src/mark.c, src/testdir/test_viminfo.vim

Patch 7.4.1933

Problem: Compiler warning about uninitialized variable. (Yegappan)
Solution: Give it a dummy value.
Files: src/ex_getln.c

Patch 7.4.1934

Problem: New style tests not executed with MinGW compiler.
Solution: Add new style test support. (Yegappan Lakshmanan)
Files: src/testdir/Make_ming.mak

Patch 7.4.1935

Problem: When using the GUI search/replace a second match right after the replacement is skipped.
Solution: Add the SEARCH_START flag. (Mleddy)
Files: src/gui.c

Patch 7.4.1936

Problem: Off-by-one error in bounds check. (Coverity)
Solution: Check register number properly.
Files: src/ops.c

Patch 7.4.1937

Problem: No test for directory stack in quickfix.
Solution: Add a test. (Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim

Patch 7.4.1938

Problem: When writing viminfo numbered marks were duplicated.
Solution: Check for duplicates between current numbered marks and the ones read from viminfo.
Files: src/mark.c

Patch 7.4.1939

Problem: Memory access error when reading viminfo. (Dominique Pelle)
Solution: Correct index in jumplist when at the end.
Files: src/mark.c, src/testdir/test_viminfo.vim

Patch 7.4.1940

Problem: "gd" hangs in some situations. (Eric Biggers)
Solution: Remove the SEARCH_START flag when looping. Add a test.
Files: src/normal.c, src/testdir/test_goto.vim

Patch 7.4.1941

Problem: Not all quickfix tests are also done with the location lists.

Solution: Test more quickfix code. Use user commands instead of "exe".
(Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim

Patch 7.4.1942

Problem: Background is not drawn properly when 'termguicolors' is set.
Solution: Check cterm_normal_bg_color. (Jacob Niehus, closes #805)
Files: src/screen.c

Patch 7.4.1943

Problem: Coverity warns for unreachable code.
Solution: Remove the code that won't do anything.
Files: src/mark.c

Patch 7.4.1944

Problem: Win32: Cannot compile with XPM feature using VC2015
Solution: Add XPM libraries compiled with VC2015, and enable to build
gvim.exe which supports XPM using VC2015. (Ken Takata)
Files: src/Make_mvc.mak, src/xpm/x64/lib-vc14/libXpm.lib,
src/xpm/x86/lib-vc14/libXpm.lib

Patch 7.4.1945

Problem: The Man plugin doesn't work that well.
Solution: Use "g:ft_man_open_mode" to be able open man pages in vert split
or separate tab. Set nomodifiable for buffer with man content. Add
a test. (Andrey Starodubtsev, closes #873)
Files: runtime/ftplugin/man.vim, src/testdir/test_man.vim,
src/testdir/Make_all.mak

Patch 7.4.1946 (after 7.4.1944)

Problem: File list does not include new XPM libraries.
Solution: Add the file list entries.
Files: Filelist

Patch 7.4.1947

Problem: Viminfo continuation line with wrong length isn't skipped. (Marius
Gedminas)
Solution: Skip a line when encountering an error, but not two lines.
Files: src/ex_cmds.c

Patch 7.4.1948

Problem: Using Ctrl-A with double-byte encoding may result in garbled text.
Solution: Skip to the start of a character. (Hirohito Higashi)
Files: src/ops.c

Patch 7.4.1949

Problem: Minor problems with the quickfix code.
Solution: Fix the problems. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1950

Problem: Quickfix long lines test not executed for buffer.
Solution: Call the function to test long lines. (Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim

Patch 7.4.1951

Problem: Ruby test is old style.

Solution: Convert to a new style test. (Ken Takata)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_ruby.in,
src/testdir/test_ruby.ok, src/testdir/test_ruby.vim

Patch 7.4.1952

Problem: Cscope interface does not support finding assignments.

Solution: Add the "a" command. (ppetina, closes #882)

Files: runtime/doc/if_cscop.txt, src/if_cscope.c

Patch 7.4.1953

Problem: Not all parts of the quickfix code are tested.

Solution: Add more tests. (Yegappan Lakshmanan)

Files: src/testdir/samples/quickfix.txt,
src/testdir/test_quickfix.vim

Patch 7.4.1954 (after 7.4.1948)

Problem: No test for what 7.4.1948 fixes.

Solution: Add a test. (Hirohito Higashi, closes #880)

Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/test_increment_dbcs.vim

Patch 7.4.1955

Problem: Using 32-bit Perl with 64-bit time_t causes memory corruption.
(Christian Brabandt)

Solution: Use time_T instead of time_t for global variables. (Ken Takata)

Files: src/ex_cmds.c, src/globals.h, src/misc2.c, src/proto/ex_cmds.pro,
src/proto/misc2.pro, src/structs.h, src/vim.h

Patch 7.4.1956

Problem: When using **CTRL-W** f and pressing "q" at the ATTENTION dialog the
newly opened window is not closed.

Solution: Close the window and go back to the original one. (Norio Takagi,
Hirohito Higashi)

Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 7.4.1957

Problem: Perl interface has obsolete workaround.

Solution: Remove the workaround added by 7.3.623. (Ken Takata)

Files: src/if_perl.xs

Patch 7.4.1958

Problem: Perl interface preprocessor statements not nicely indented.

Solution: Improve the indenting. (Ken Takata)

Files: src/if_perl.xs

Patch 7.4.1959

Problem: Crash when running test_channel.vim on Windows.

Solution: Check for NULL pointer result from FormatMessage(). (Christian
Brabandt)

Files: src/channel.c

Patch 7.4.1960

Problem: Unicode standard 9 was released.
Solution: Update the character property tables. (Christian Brabandt)
Files: src/mbyte.c

Patch 7.4.1961

Problem: When **'insertmode'** is reset while doing completion the popup menu remains even though Vim is in Normal mode.
Solution: Ignore stop_insert_mode when the popup menu is visible. Don't set stop_insert_mode when **'insertmode'** was already off. (Christian Brabandt)
Files: src/edit.c, src/option.c, src/Makefile, src/testdir/test_alot.vim, src/testdir/test_popup.vim

Patch 7.4.1962

Problem: Two test files for increment/decrement.
Solution: Move the old style test into the new style test. (Hirohito Higashi, closes #881)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/main.aap, src/testdir/test35.in, src/testdir/test35.ok, src/testdir/test_increment.vim

Patch 7.4.1963

Problem: Running Win32 Vim in mintty does not work.
Solution: Detect mintty and give a helpful error message. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/iscygpty.c, src/iscygpty.h, src/main.c, Filelist

Patch 7.4.1964

Problem: The quickfix init function is too big.
Solution: Factor out parsing **'errorformat'** to a separate function. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 7.4.1965

Problem: When using a job in raw mode to append to a buffer garbage characters are added.
Solution: Do not replace the trailing NUL with a NL. (Ozaki Kiichi)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.1966

Problem: Coverity reports a resource leak.
Solution: Close "fd" also when bailing out.
Files: src/quickfix.c

Patch 7.4.1967

Problem: Falling back from NFA to old regexp engine does not work properly. (fritzophrenic)
Solution: Do not restore nfa_match. (Christian Brabandt, closes #867)
Files: src/regexp_nfa.c, src/testdir/test64.in, src/testdir/test64.ok

Patch 7.4.1968

Problem: Invalid memory access with "**<C-**".
Solution: Do not recognize this as a special character. (Dominique Pelle)

Files: src/misc2.c, src/testdir/test_expr.vim

Patch 7.4.1969

Problem: When the netbeans channel is closed consuming the buffer may cause a crash.

Solution: Check for nb_channel not to be NULL. (Xavier de Gaye)

Files: src/netbeans.c

Patch 7.4.1970

Problem: Using ":insert" in an empty buffer sets the jump mark. (Ingo Karkat)

Solution: Don't adjust marks when replacing the empty line in an empty buffer. (closes #892)

Files: src/ex_cmds.c, src/testdir/test_jumps.vim, src/testdir/test_alot.vim

Patch 7.4.1971

Problem: It is not easy to see unrecognized error lines below the current error position.

Solution: Add ":clist +count".

Files: src/quickfix.c, runtime/doc/quickfix.txt

Patch 7.4.1972

Problem: On Solaris select() does not work as expected when there is typeahead.

Solution: Add ICANON when sleeping. (Ozaki Kiichi)

Files: src/os_unix.c

Patch 7.4.1973

Problem: On MS-Windows the package directory may be added at the end because of forward/backward slash differences. (Matthew Desjardins)

Solution: Ignore slash differences.

Files: src/ex_cmds2.c

Patch 7.4.1974

Problem: GUI has a problem with some termcodes.

Solution: Handle negative numbers. (Kazunobu Kuriyama)

Files: src/gui.c

Patch 7.4.1975

Problem: On MS-Windows large files (> 2Gbyte) cause problems.

Solution: Use "off_T" instead of "off_t". Use "stat_T" instead of "struct stat". Use 64 bit system functions if available. (Ken Takata)

Files: src/Makefile, src/buffer.c, src/diff.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c, src/fileio.c, src/gui.c, src/gui_at_fs.c, src/if_cscope.c, src/main.c, src/memfile.c, src/memline.c, src/misc1.c, src/misc2.c, src/netbeans.c, src/os_mswin.c, src/os_win32.c, src/proto/fileio.pro, src/proto/memline.pro, src/proto/os_mswin.pro, src/pty.c, src/quickfix.c, src/spell.c, src/structs.h, src/tag.c, src/testdir/Make_all.mak, src/testdir/test_largefile.vim, src/testdir/test_stat.vim, src/undo.c, src/vim.h

Patch 7.4.1976

Problem: Number variables are not 64 bits while they could be.
Solution: Add the num64 feature. (Ken Takata, Yasuhiro Matsumoto)
Files: runtime/doc/eval.txt, runtime/doc/various.txt,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/charset.c,
src/eval.c, src/ex_cmds.c, src/ex_getln.c, src/feature.h,
src/fileio.c, src/fold.c, src/json.c, src/message.c, src/misc1.c,
src/misc2.c, src/ops.c, src/option.c, src/proto/charset.pro,
src/proto/eval.pro, src/quickfix.c, src/structs.h,
src/testdir/test_viml.vim, src/version.c

Patch 7.4.1977

Problem: With 64 bit changes don't need three calls to sprintf().
Solution: Simplify the code, use vim_snprintf(). (Ken Takata)
Files: src/fileio.c

Patch 7.4.1978 (after 7.4.1975)

Problem: Large file test does not delete its output.
Solution: Delete the output. Check size properly when possible. (Ken Takata)
Files: src/testdir/test_largefile.vim

Patch 7.4.1979 (after 7.4.1976)

Problem: Getting value of binary option is wrong. (Kent Sibilev)
Solution: Fix type cast. Add a test.
Files: src/option.c, src/testdir/test_expr.vim

Patch 7.4.1980

Problem: **'errorformat'** is parsed for every call to ":caddexpr". Can't add
to two location lists asynchronously.
Solution: Keep the previously parsed data when appropriate. (mostly by
Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.1981

Problem: No testing for Farsi code.
Solution: Add a minimal test. Clean up Farsi code.
Files: src/farsi.c, src/Makefile, src/charset.c, src/normal.c,
src/proto/main.pro, src/testdir/Make_all.mak,
src/testdir/test_farsi.vim

Patch 7.4.1982

Problem: Viminfo file contains duplicate change marks.
Solution: Drop duplicate marks.
Files: src/mark.c

Patch 7.4.1983

Problem: farsi.c and arabic.c are included in a strange way.
Solution: Build them like other files.
Files: src/main.c, src/farsi.c, src/arabic.c, src/proto.h,
src/proto/main.pro, src/proto/farsi.pro, src/proto/arabic.pro,
src/Makefile, src/Make_bc5.mak, src/Make_cyg_ming.mak,
src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak,
src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak,
Filelist

Patch 7.4.1984

Problem: Not all quickfix features are tested.
Solution: Add a few more tests. (Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim

Patch 7.4.1985 (after 7.4.1983)

Problem: Missing changes in VMS build file.
Solution: Use the right file name.
Files: src/Make_vms.mms

Patch 7.4.1986

Problem: Compiler warns for loss of data.
Solution: Use size_t instead of int. (Christian Brabandt)
Files: src/ex_cmds2.c

Patch 7.4.1987

Problem: When copying unrecognized lines for viminfo, end up with useless continuation lines.
Solution: Skip continuation lines.
Files: src/ex_cmds.c

Patch 7.4.1988

Problem: When updating viminfo with file marks there is no time order.
Solution: Remember the time when a buffer was last used, store marks for the most recently used buffers.
Files: src/buffer.c, src/structs.h, src/mark.c, src/main.c,
src/ex_cmds.c, src/proto/mark.pro, src/testdir/test_viminfo.vim

Patch 7.4.1989

Problem: filter() and map() only accept a string argument.
Solution: Implement using a Funcref argument (Yasuhiro Matsumoto, Ken Takata)
Files: runtime/doc/eval.txt, src/Makefile, src/eval.c,
src/testdir/test_alot.vim, src/testdir/test_filter_map.vim,
src/testdir/test_partial.vim

Patch 7.4.1990 (after 7.4.1952)

Problem: Cscope items are not sorted.
Solution: Put the new "a" command first. (Ken Takata)
Files: src/if_cscope.c

Patch 7.4.1991

Problem: glob() does not add a symbolic link when there are no wildcards.
Solution: Remove the call to mch_getperm().
Files: src/misc1.c

Patch 7.4.1992

Problem: Values for true and false can be confusing.
Solution: Update the documentation. Add a test. Make v:true evaluate to TRUE for a non-zero-arg.
Files: runtime/doc/eval.txt, src/eval.c, src/Makefile,
src/testdir/test_true_false.vim, src/testdir/test_alot.vim

Patch 7.4.1993

Problem: Not all TRUE and FALSE arguments are tested.
Solution: Add a few more tests.
Files: src/testdir/test_true_false.vim

Patch 7.4.1994 (after 7.4.1993)

Problem: True-false test fails.
Solution: Filter the dict to only keep the value that matters.
Files: src/testdir/test_true_false.vim

Patch 7.4.1995

Problem: GUI: cursor drawn in wrong place if a timer callback causes a screen update. (David Samvelyan)
Solution: Also redraw the cursor when it's blinking and on.
Files: src/gui_gtk_x11.c, src/gui_mac.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/screen.c, src/proto/gui_gtk_x11.pro, src/proto/gui_mac.pro, src/proto/gui_photon.pro, src/proto/gui_w32.pro, src/proto/gui_x11.pro

Patch 7.4.1996

Problem: Capturing the output of a command takes a few commands.
Solution: Add evalcmd().
Files: src/eval.c, runtime/doc/eval.txt, src/testdir/test_alot.vim, src/Makefile, src/testdir/test_evalcmd.vim

Patch 7.4.1997

Problem: Cannot easily scroll the quickfix window.
Solution: Add ":cbottom".
Files: src/ex_cmds.h, src/quickfix.c, src/proto/quickfix.pro, src/ex_docmd.c, src/testdir/test_quickfix.vim, runtime/doc/quickfix.txt

Patch 7.4.1998

Problem: When writing buffer lines to a job there is no NL to NUL conversion.
Solution: Make it work symmetrical with writing lines from a job into a buffer.
Files: src/channel.c, src/proto/channel.pro, src/netbeans.c

Patch 7.4.1999

Problem: evalcmd() doesn't work recursively.
Solution: Use redir_evalcmd instead of redir_vname.
Files: src/message.c, src/eval.c, src/globals.h, src/proto/eval.pro, src/testdir/test_evalcmd.vim

Patch 7.4.2000 (after 7.4.1999)

Problem: Evalcmd test fails.
Solution: Add missing piece.
Files: src/ex_docmd.c

Patch 7.4.2001 (after 7.4.2000)

Problem: Tiny build fails. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 7.4.2002

Problem: Crash when passing number to filter() or map().
Solution: Convert to a string. (Ozaki Kiichi)
Files: src/eval.c, src/testdir/test_filter_map.vim

Patch 7.4.2003

Problem: Still cursor flickering when a callback updates the screen. (David Samvelyan)
Solution: Put the cursor in the right position after updating the screen.
Files: src/screen.c

Patch 7.4.2004

Problem: GUI: cursor displayed in the wrong position.
Solution: Correct screen_cur_col and screen_cur_row.
Files: src/screen.c

Patch 7.4.2005

Problem: After using evalcmd() message output is in the wrong position. (Christian Brabandt)
Solution: Reset msg_col.
Files: src/eval.c

Patch 7.4.2006

Problem: Crash when using tabnext in BufUnload autocmd. (Norio Takagi)
Solution: First check that the current buffer is the right one. (Hirohito Higashi)
Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 7.4.2007

Problem: Running the tests leaves a viminfo file behind.
Solution: Make the viminfo option empty.
Files: src/testdir/runtest.vim

Patch 7.4.2008

Problem: evalcmd() has a confusing name.
Solution: Rename to execute(). Make silent optional. Support a list of commands.
Files: src/eval.c, src/ex_docmd.c, src/message.c, src/globals.h, src/proto/eval.pro, src/Makefile, src/testdir/test_evalcmd.vim, src/testdir/test_execute_func.vim, src/testdir/test_alot.vim, runtime/doc/eval.txt

Patch 7.4.2009 (after 7.4.2008)

Problem: Messages test fails.
Solution: Don't set redir_execute before returning. Add missing version number.
Files: src/eval.c

Patch 7.4.2010

Problem: There is a :cbottom command but no :lbottom command.
Solution: Add :lbottom. (Yegappan Lakshmanan)
Files: runtime/doc/index.txt, runtime/doc/quickfix.txt, src/ex_cmds.h, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.2011

Problem: It is not easy to get a list of command arguments.
Solution: Add getcompletion(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/eval.c, src/ex_docmd.c,
src/proto/ex_docmd.pro, src/testdir/test_cmdline.vim

Patch 7.4.2012 (after 7.4.2011)

Problem: Test for getcompletion() does not pass on all systems.
Solution: Only test what is supported.
Files: src/testdir/test_cmdline.vim

Patch 7.4.2013

Problem: Using "noinsert" in '**completeopt**' breaks redo.
Solution: Set compl_curr_match. (Shougo Matsu, closes #874)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 7.4.2014

Problem: Using "noinsert" in '**completeopt**' does not insert match.
Solution: Set compl_enter_selects. (Shougo Matsu, closes #875)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 7.4.2015

Problem: When a file gets a name when writing it '**acd**' is not effective.
(Dan Church)
Solution: Invoke DO_AUTOCHDIR after writing the file. (Allen Haim, closes
#777, closes #803) Add test_autochdir() to enable '**acd**' before
"starting" is reset.
Files: src/ex_cmds.c, src/buffer.c, src/eval.c, src/globals.h,
src/Makefile, src/testdir/test_autochdir.vim,
src/testdir/Make_all.mak

Patch 7.4.2016

Problem: Warning from MinGW about _WIN32_WINNT redefined. (John Marriott)
Solution: First undefine it. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.2017

Problem: When there are many errors adding them to the quickfix list takes
a long time.
Solution: Add BLN_NOOPT. Don't call buf_valid() in buf_copy_options().
Remember the last file name used. When going through the buffer
list start from the end of the list. Only call buf_valid() when
autocommands were executed.
Files: src/buffer.c, src/option.c, src/quickfix.c, src/vim.h

Patch 7.4.2018

Problem: buf_valid() can be slow when there are many buffers.
Solution: Add bufref_valid(), only go through the buffer list when a buffer
was freed.
Files: src/structs.h, src/buffer.c, src/quickfix.c, src/proto/buffer.pro

Patch 7.4.2019

Problem: When ignoring case utf_fold() may consume a lot of time.

Solution: Optimize for ASCII.
Files: src/mbyte.c

Patch 7.4.2020

Problem: Can't build without +autocmd feature.
Solution: Adjust #ifdefs.
Files: src/buffer.c

Patch 7.4.2021

Problem: Still too many buf_valid() calls.
Solution: Make au_new_curbuf a bufref. Use bufref_valid() in more places.
Files: src/ex_cmds.c, src/buffer.c, src/globals.h

Patch 7.4.2022

Problem: Warnings from 64 bit compiler.
Solution: Add type casts. (Mike Williams)
Files: src/eval.c

Patch 7.4.2023

Problem: buflist_findname_stat() may find a dummy buffer.
Solution: Set the BF_DUMMY flag after loading a dummy buffer. Start finding buffers from the end of the list.
Files: src/quickfix.c, src/buffer.c

Patch 7.4.2024

Problem: More buf_valid() calls can be optimized.
Solution: Use bufref_valid() instead.
Files: src/buffer.c, src/ex_cmds.c, src/structs.h, src/channel.c, src/diff.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/main.c, src/misc2.c, src/netbeans.c, src/quickfix.c, src/spell.c, src/term.c, src/if_py_both.h, src/window.c, src/proto/buffer.pro, src/proto/window.pro

Patch 7.4.2025

Problem: The cursor blinking stops or is irregular when receiving data over a channel and writing it in a buffer, and when updating the status line. (Ramel Eshed)
Solution: Make it a bit better by flushing GUI output. Don't redraw the cursor after updating the screen if the blink state is off.
Files: src/gui_gtk_x11.c, src/screen.c

Patch 7.4.2026

Problem: Reference counting for callbacks isn't right.
Solution: Add free_callback(). (Ken Takata) Fix reference count.
Files: src/channel.c, src/eval.c, src/ex_cmds2.c, src/proto/eval.pro

Patch 7.4.2027

Problem: Can't build with +eval but without +menu.
Solution: Add #ifdef. (John Marriott)
Files: src/eval.c

Patch 7.4.2028

Problem: cppcheck warns for using index before limits check.

Solution: Swap the expressions. (Dominique Pelle)
Files: src/mbyte.c

Patch 7.4.2029

Problem: printf() does not work with 64 bit numbers.
Solution: use the "L" length modifier. (Ken Takata)
Files: src/message.c, src/testdir/test_expr.vim

Patch 7.4.2030

Problem: ARCH must be set properly when using MinGW.
Solution: Detect the default value of ARCH from the current compiler. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 7.4.2031

Problem: The list_lbr_utf8 test fails if ~/.vim/syntax/c.vim sets **'textwidth'** to a non-zero value. (Oyvind A. Holm)
Solution: Add a setup.vim file that sets **'runtimepath'** and \$HOME to a safe value. (partly by Christian Brabandt, closes #912)
Files: src/testdir/setup.vim, src/testdir/amiga.vim, src/testdir/dos.vim, src/testdir/unix.vim, src/testdir/vms.vim, src/testdir/runtest.vim

Patch 7.4.2032 (after 7.4.2030)

Problem: Build fails with 64 bit MinGW. (Axel Bender)
Solution: Handle dash vs. underscore. (Ken Takata, Hirohito Higashi)
Files: src/Make_cyg_ming.mak

Patch 7.4.2033

Problem: **'cscopequickfix'** option does not accept new value "a".
Solution: Adjust list of command characters. (Ken Takata)
Files: src/option.h, src/Makefile, src/testdir/test_cscope.vim, src/testdir/Make_all.mak

Patch 7.4.2034 (after 7.4.2032)

Problem: Build fails with some version of MinGW. (illusorypan)
Solution: Recognize mingw32. (Ken Takata, closes #921)
Files: src/Make_cyg_ming.mak

Patch 7.4.2035

Problem: On Solaris with ZFS the ACL may get removed.
Solution: Always restore the ACL for Solaris ZFS. (Danek Duvall)
Files: src/fileio.c

Patch 7.4.2036

Problem: Looking up a buffer by number is slow if there are many.
Solution: Use a hashtable.
Files: src/structs.h, src/buffer.c

Patch 7.4.2037 (after 7.4.2036)

Problem: Small build fails.
Solution: Adjust #ifdefs.
Files: src/hashtab.c

Patch 7.4.2038 (after 7.4.2036)

Problem: Small build still fails.
Solution: Adjust more #ifdefs.
Files: src/globals.h, src/buffer.c

Patch 7.4.2039

Problem: The Netbeans integration is not tested.
Solution: Add a first Netbeans test.
Files: src/testdir/test_netbeans.vim, src/testdir/test_netbeans.py,
src/testdir/Make_all.mak, src/Makefile,
src/testdir/test_channel.vim, src/testdir/shared.vim

Patch 7.4.2040

Problem: New files missing from distribution.
Solution: Add new test scripts.
Files: Filelist

Patch 7.4.2041

Problem: Netbeans file authentication not tested.
Solution: Add a test.
Files: src/testdir/test_netbeans.vim

Patch 7.4.2042

Problem: GTK: display updating is not done properly and can be slow.
Solution: Use gdk_display_flush() instead of gdk_display_sync(). Don't call
gdk_window_process_updates(). (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 7.4.2043

Problem: setbuvfar() causes a screen redraw.
Solution: Only use aucmd_prepbuf() for options.
Files: src/eval.c

Patch 7.4.2044

Problem: filter() and map() either require a string or defining a function.
Solution: Support lambda, a short way to define a function that evaluates an
expression. (Yasuhiro Matsumoto, Ken Takata)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_alot.vim,
src/Makefile, src/testdir/test_channel.vim,
src/testdir/test_lambda.vim

Patch 7.4.2045

Problem: Memory leak when using a function callback.
Solution: Don't save the function name when it's in the partial.
Files: src/channel.c

Patch 7.4.2046

Problem: The qf_init_ext() function is too big.
Solution: Refactor it. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 7.4.2047

Problem: Compiler warning for initializing a struct.
Solution: Initialize in another way. (Anton Lindqvist)
Files: src/quickfix.c

Patch 7.4.2048

Problem: There is still code and help for unsupported systems.
Solution: Remove the code and text. (Hirohito Higashi)
Files: runtime/doc/eval.txt, runtime/lang/menu_sk_sk.vim,
runtime/menu.vim, runtime/optwin.vim, src/Make_bc5.mak,
src/ex_docmd.c, src/feature.h, src/fileio.c, src/globals.h,
src/main.c, src/memfile.c, src/memline.c, src/misc1.c,
src/misc2.c, src/option.c, src/option.h, src/os_unix.c,
src/os_unix.h, src/proto.h, src/term.c, src/undo.c, src/version.c,
src/vim.h, src/xxd/xxd.c

Patch 7.4.2049

Problem: There is no way to get a list of the error lists.
Solution: Add ":chistory" and ":lhistory".
Files: src/ex_cmds.h, src/quickfix.c, src/ex_docmd.c, src/message.c,
src/proto/quickfix.pro, src/testdir/test_quickfix.vim

Patch 7.4.2050

Problem: When using ":vimgrep" may end up with duplicate buffers.
Solution: When adding an error list entry pass the buffer number if possible.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.2051

Problem: No proper testing of trunc_string().
Solution: Add a unittest for message.c.
Files: src/Makefile, src/message.c, src/message_test.c, src/main.c,
src/proto/main.pro, src/structs.h

Patch 7.4.2052

Problem: Coverage report is messed up by the unittests.
Solution: Add a separate test target for script tests. Use that when
collecting coverage information.
Files: src/Makefile

Patch 7.4.2053

Problem: Can't run scripttests in the top directory.
Solution: Add targets to the top Makefile.
Files: Makefile

Patch 7.4.2054 (after 7.4.2048)

Problem: Wrong part of #ifdef removed.
Solution: Use the right part. (Hirohito Higashi)
Files: src/os_unix.c

Patch 7.4.2055

Problem: eval.c is too big
Solution: Move Dictionary functions to dict.c
Files: src/eval.c, src/dict.c, src/vim.h, src/globals.h,
src/proto/eval.pro, src/proto/dict.pro, src/Makefile, Filelist

Patch 7.4.2056 (after 7.4.2055)

Problem: Build fails.
Solution: Add missing changes.

Files: src/proto.h

Patch 7.4.2057

Problem: eval.c is too big.

Solution: Move List functions to list.c

Files: src/eval.c, src/dict.c, src/list.c, src/proto.h, src/Makefile, src/globals.h, src/proto/eval.pro, src/proto/list.pro, Filelist

Patch 7.4.2058

Problem: eval.c is too big.

Solution: Move user functions to userfunc.c

Files: src/userfunc.c, src/eval.c, src/vim.h, src/globals.h, src/structs.h, src/proto.h, src/Makefile, src/proto/eval.pro, src/proto/userfunc.pro, Filelist

Patch 7.4.2059

Problem: Non-Unix builds fail.

Solution: Update Makefiles for new files.

Files: src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak

Patch 7.4.2060 (after 7.4.2059)

Problem: Wrong file name.

Solution: Fix typo.

Files: src/Make_mvc.mak

Patch 7.4.2061

Problem: qf_init_ext() is too big.

Solution: Move code to qf_parse_line() (Yegappan Lakshmanan)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.2062

Problem: Using dummy variable to compute struct member offset.

Solution: Use offsetof().

Files: src/globals.h, src/macros.h, src/vim.h, src/spell.c

Patch 7.4.2063

Problem: eval.c is still too big.

Solution: Split off internal functions to evalfunc.c.

Files: src/eval.c, src/evalfunc.c, src/list.c, src/proto.h, src/globals.h, src/vim.h, src/proto/eval.pro, src/proto/evalfunc.pro, src/proto/list.pro, src/Makefile, Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak

Patch 7.4.2064

Problem: Coverity warns for possible buffer overflow.

Solution: Use vim_strcat() instead of strcat().

Files: src/quickfix.c

Patch 7.4.2065

Problem: Compiler warns for uninitialized variable. (John Marriott)

Solution: Set lnum to the right value.
Files: src/evalfunc.c

Patch 7.4.2066

Problem: getcompletion() not well tested.
Solution: Add more testing.
Files: src/testdir/test_cmdline.vim

Patch 7.4.2067

Problem: Compiler warning for char/char_u conversion. (Tony Mechelynck)
Inefficient code.
Solution: Use more lines to fill with spaces. (Nikolai Pavlov) Add type cast.
Files: src/quickfix.c

Patch 7.4.2068

Problem: Not all arguments of trunc_string() are tested. Memory access
error when running the message tests.
Solution: Add another test case. (Yegappan Lakshmanan) Make it easy to run
unittests with valgrind. Fix the access error.
Files: src/message.c, src/message_test.c, src/Makefile

Patch 7.4.2069

Problem: spell.c is too big.
Solution: Split it in spell file handling and spell checking.
Files: src/spell.c, src/spellfile.c, src/spell.h, src/Makefile,
src/proto/spell.pro, src/proto/spellfile.pro, src/proto.h
Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak,
src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak,
src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak

Patch 7.4.2070 (after 7.4.2069)

Problem: Missing change to include file.
Solution: Include the spell header file.
Files: src/vim.h

Patch 7.4.2071

Problem: The return value of type() is difficult to use.
Solution: Define v:t_constants. (Ken Takata)
Files: runtime/doc/eval.txt, src/eval.c, src/evalfunc.c,
src/testdir/test_channel.vim, src/testdir/test_viml.vim, src/vim.h

Patch 7.4.2072

Problem: substitute() does not support a Funcref argument.
Solution: Support a Funcref like it supports a string starting with "\=".
Files: src/evalfunc.c, src/regexp.c, src/eval.c, src/proto/eval.pro,
src/proto/regexp.pro, src/testdir/test_expr.vim

Patch 7.4.2073

Problem: rgb.txt is read for every color name.
Solution: Load rgb.txt once. (Christian Brabandt) Add a test.
Files: runtime/rgb.txt, src/term.c, src/testdir/test_syn_attr.vim

Patch 7.4.2074

Problem: One more place using a dummy variable.

Solution: Use `offsetof()`. (Ken Takata)
Files: `src/userfunc.c`

Patch 7.4.2075

Problem: No autocmd event to initialize a window or tab page.
Solution: Add `WinNew` and `TabNew` events. (partly by Felipe Morales)
Files: `src/fileio.c`, `src/window.c`, `src/vim.h`,
`src/testdir/test_autocmd.vim`, `runtime/doc/autocmd.txt`

Patch 7.4.2076

Problem: Syntax error when dict has `'>'` key.
Solution: Check for `endchar`. (Ken Takata)
Files: `src/userfunc.c`, `src/testdir/test_lambda.vim`

Patch 7.4.2077

Problem: Cannot update **'tabline'** when a tab was closed.
Solution: Add the `TabClosed` autocmd event. (partly by Felipe Morales)
Files: `src/fileio.c`, `src/window.c`, `src/vim.h`,
`src/testdir/test_autocmd.vim`, `runtime/doc/autocmd.txt`

Patch 7.4.2078

Problem: Running checks in `po` directory fails.
Solution: Add colors used in `syntax.c` to the builtin color table.
Files: `src/term.c`

Patch 7.4.2079

Problem: Netbeans test fails on non-Unix systems.
Solution: Only do the permission check on Unix systems.
Files: `src/testdir/test_netbeans.vim`

Patch 7.4.2080

Problem: When using `PERROR()` on some systems `assert_fails()` does not see the error.
Solution: Make `PERROR()` always report the error.
Files: `src/vim.h`, `src/message.c`, `src/proto/message.pro`

Patch 7.4.2081

Problem: Line numbers in the error list are not always adjusted.
Solution: Set `b_has_qf_entry` properly. (Yegappan Lakshmanan)
Files: `src/quickfix.c`, `src/structs.h`, `src/testdir/test_quickfix.vim`

Patch 7.4.2082

Problem: Not much test coverage for digraphs.
Solution: Add a new style digraph test. (Christian Brabandt)
Files: `src/Makefile`, `src/testdir/test_alot.vim`,
`src/testdir/test_digraph.vim`

Patch 7.4.2083

Problem: Coverity complains about not restoring a value.
Solution: Restore the value, although it's not really needed. Change return to jump to cleanup, might leak memory.
Files: `src/userfunc.c`

Patch 7.4.2084

Problem: New digraph test makes testing hang.
Solution: Don't set "nocp".
Files: src/testdir/test_digraph.vim

Patch 7.4.2085

Problem: Digraph tests fails on some systems.
Solution: Run it separately and set **'encoding'** early.
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_digraph.vim

Patch 7.4.2086

Problem: Using the system default encoding makes tests unpredictable.
Solution: Always use utf-8 or latin1 in the new style tests. Remove setting encoding and scriptencoding where it is not needed.
Files: src/testdir/runtest.vim, src/testdir/test_channel.vim,
src/testdir/test_digraph.vim, src/testdir/test_expand_dllpath.vim,
src/testdir/test_expr_utf8.vim, src/testdir/test_json.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_regexp_utf8.vim, src/testdir/test_visual.vim,
src/testdir/test_alot_utf8.vim,

Patch 7.4.2087

Problem: Digraph code test coverage is still low.
Solution: Add more tests. (Christian Brabandt)
Files: src/testdir/test_digraph.vim

Patch 7.4.2088 (after 7.4.2087)

Problem: Keymap test fails with normal features.
Solution: Bail out if the keymap feature is not supported.
Files: src/testdir/test_digraph.vim

Patch 7.4.2089

Problem: Color handling of X11 GUIs is too complicated.
Solution: Simplify the code. Use RGBA where appropriate. (Kazunobu Kuriyama)
Files: src/gui.h, src/gui_beval.c, src/gui_gtk_x11.c, src/netbeans.c

Patch 7.4.2090

Problem: Using submatch() in a lambda passed to substitute() is verbose.
Solution: Use a static list and pass it as an optional argument to the function. Fix memory leak.
Files: src/structs.h, src/list.c, src/userfunc.c, src/channel.c,
src/eval.c, src/evalfunc.c, src/ex_cmds2.c, src/regexp.c,
src/proto/list.pro, src/proto/userfunc.pro,
src/testdir/test_expr.vim, runtime/doc/eval.txt

Patch 7.4.2091

Problem: Coverity reports a resource leak when out of memory.
Solution: Close the file before returning.
Files: src/term.c

Patch 7.4.2092

Problem: GTK 3 build fails with older GTK version.
Solution: Check the pango version. (Kazunobu Kuriyama)

Files: src/gui_beval.c

Patch 7.4.2093

Problem: Netbeans test fails once in a while. Leaving log file behind.
Solution: Add it to the list of flaky tests. Disable logfile.
Files: src/testdir/runtest.vim, src/testdir/test_channel.vim

Patch 7.4.2094

Problem: The color allocation in X11 is overly complicated.
Solution: Remove find_closest_color(), XAllocColor() already does this.
(Kazunobu Kuriyama)
Files: src/gui_x11.c

Patch 7.4.2095

Problem: Man test fails when run with the GUI.
Solution: Adjust for different behavior of GUI. Add assert_inrange().
Files: src/eval.c, src/evalfunc.c, src/proto/eval.pro,
src/testdir/test_assert.vim, src/testdir/test_man.vim,
runtime/doc/eval.txt

Patch 7.4.2096

Problem: Lambda functions show up with completion.
Solution: Don't show lambda functions. (Ken Takata)
Files: src/userfunc.c, src/testdir/test_cmdline.vim

Patch 7.4.2097

Problem: Warning from 64 bit compiler.
Solution: use size_t instead of int. (Mike Williams)
Files: src/message.c

Patch 7.4.2098

Problem: Text object tests are old style.
Solution: Turn them into new style tests. (James McCoy, closes #941)
Files: src/testdir/Make_all.mak, src/testdir/test_textobjects.in,
src/testdir/test_textobjects.ok, src/testdir/test_textobjects.vim,
src/Makefile

Patch 7.4.2099

Problem: When a keymap is active only "(lang)" is displayed. (Ilya Dogolazky)
Solution: Show the keymap name. (Dmitri Vereshchagin, closes #933)
Files: src/buffer.c, src/proto/screen.pro, src/screen.c

Patch 7.4.2100

Problem: "cgn" and "dgn" do not work correctly with a single character match and the replacement includes the searched pattern. (John Beckett)
Solution: If the match is found in the wrong column try in the next column. Turn the test into new style. (Christian Brabandt)
Files: src/search.c, src/testdir/Make_all.mak, src/Makefile,
src/testdir/test53.in, src/testdir/test53.ok,
src/testdir/test_gn.vim

Patch 7.4.2101

Problem: Looping over windows, buffers and tab pages is inconsistent.
Solution: Use FOR_ALL_ macros everywhere. (Yegappan Lakshmanan)
Files: src/buffer.c, src/diff.c, src/edit.c, src/eval.c, src/evalfunc.c,
src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/fileio.c,
src/globals.h, src/gui.c, src/gui_mac.c, src/if_lua.c,
src/if_mzsch.c, src/if_perl.xs, src/if_ruby.c, src/if_tcl.c,
src/main.c, src/mark.c, src/memfile.c, src/memline.c, src/misc1.c,
src/move.c, src/netbeans.c, src/normal.c, src/option.c,
src/quickfix.c, src/screen.c, src/spell.c, src/term.c,
src/window.c, src/workshop.c

Patch 7.4.2102 (after 7.4.2101)

Problem: Tiny build with GUI fails.
Solution: Revert one FOR_ALL_ change.
Files: src/gui.c

Patch 7.4.2103

Problem: Can't have "augroup END" right after ":au!".
Solution: Check for the bar character before the command argument.
Files: src/fileio.c, src/testdir/test_autocmd.vim,
runtime/doc/autocmd.txt

Patch 7.4.2104

Problem: Code duplication when unreferencing a function.
Solution: De-duplicate.
Files: src/userfunc.c

Patch 7.4.2105

Problem: Configure reports default features to be "normal" while it is
"huge".
Solution: Change the default text. Build with newer autoconf.
Files: src/configure.in, src/auto/configure

Patch 7.4.2106

Problem: Clang warns about missing field in initializer.
Solution: Define COMMA and use it. (Kazunobu Kuriyama)
Files: src/ex_cmds.c, src/globals.h, src/vim.h

Patch 7.4.2107 (after 7.4.2106)

Problem: Misplaced equal sign.
Solution: Remove it.
Files: src/globals.h

Patch 7.4.2108

Problem: Netbeans test is flaky.
Solution: Wait for the cursor to be positioned.
Files: src/testdir/test_netbeans.vim

Patch 7.4.2109

Problem: Setting '**display**' to "lastline" is a drastic change, while
omitting it results in lots of "@" lines.
Solution: Add "truncate" to show "@@@" for a truncated line.
Files: src/option.h, src/screen.c, runtime/doc/options.txt

Patch 7.4.2110

Problem: When there is an CmdUndefined autocmd then the error for a missing command is E464 instead of E492. (Manuel Ortega)
Solution: Don't let the pointer be NULL.
Files: src/ex_docmd.c, src/testdir/test_usercommands.vim

Patch 7.4.2111

Problem: Defaults are very conservative.
Solution: Move settings from vimrc_example.vim to defaults.vim. Load defaults.vim if no .vimrc was found.
Files: src/main.c, src/version.c, src/os_amiga.h, src/os_dos.h, src/os_mac.h, src/os_unix.h, src/feature.h, src/Makefile, runtime/vimrc_example.vim, runtime/defaults.vim, runtime/evim.vim, Filelist, runtime/doc/starting.txt

Patch 7.4.2112

Problem: getcompletion(.., 'dir') returns a match with trailing "*" when there are no matches. (Chdiza)
Solution: Return an empty list when there are no matches. Add a trailing slash to directories. (Yegappan Lakshmanan) Add tests for no matches. (closes #947)
Files: src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 7.4.2113

Problem: Test for undo is flaky.
Solution: Turn it into a new style test. Use test_settime() to avoid flakiness.
Files: src/Makefile, src/undo.c, src/testdir/test61.in, src/testdir/test61.ok, src/testdir/test_undo.vim, src/testdir/test_undolevels.vim, src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 7.4.2114

Problem: Tiny build fails.
Solution: Always include vim_time().
Files: src/ex_cmds.c

Patch 7.4.2115

Problem: Loading defaults.vim with -C argument.
Solution: Don't load the defaults script with -C argument. Test sourcing the defaults script. Set 'display' to "truncate".
Files: src/main.c, src/Makefile, runtime/defaults.vim, src/testdir/test_startup.vim, src/testdir/Make_all.mak

Patch 7.4.2116

Problem: The default vimrc for Windows is very conservative.
Solution: Use the defaults.vim in the Windows installer.
Files: src/dosinst.c

Patch 7.4.2117

Problem: Deleting an augroup that still has autocmds does not give a warning. The next defined augroup takes its place.
Solution: Give a warning and prevent the index being used for another group name.

Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 7.4.2118

Problem: Mac: can't build with tiny features.

Solution: Don't define FEAT_CLIPBOARD unconditionally. (Kazunobu Kuriyama)

Files: src/vim.h

Patch 7.4.2119

Problem: Closures are not supported.

Solution: Capture variables in lambdas from the outer scope. (Yasuhiro Matsumoto, Ken Takata)

Files: runtime/doc/eval.txt, src/eval.c, src/ex_cmds2.c, src/globals.h,
src/proto/eval.pro, src/proto/userfunc.pro,
src/testdir/test_lambda.vim, src/userfunc.c

Patch 7.4.2120

Problem: User defined functions can't be a closure.

Solution: Add the "closure" argument. Allow using :unlet on a bound variable. (Yasuhiro Matsumoto, Ken Takata)

Files: runtime/doc/eval.txt, src/testdir/test_lambda.vim, src/userfunc.c,
src/eval.c src/proto/userfunc.pro

Patch 7.4.2121

Problem: No easy way to check if lambda and closure are supported.

Solution: Add the +lambda feature.

Files: src/evalfunc.c, src/version.c, src/testdir/test_lambda.vim

Patch 7.4.2122 (after 7.4.2118)

Problem: Mac: don't get +clipboard in huge build.

Solution: Move #define down below including feature.h

Files: src/vim.h

Patch 7.4.2123

Problem: No new style test for diff mode.

Solution: Add a test. Check that folds are in sync.

Files: src/Makefile, src/testdir/test_diffmode.vim,
src/testdir/Make_all.mak, src/testdir/test47.in,
src/testdir/test47.ok

Patch 7.4.2124

Problem: diffmode test leaves files behind, breaking another test.

Solution: Delete the files.

Files: src/testdir/test_diffmode.vim

Patch 7.4.2125

Problem: Compiler warning for loss of data.

Solution: Add a type cast. (Christian Brabandt)

Files: src/message.c

Patch 7.4.2126

Problem: No tests for :diffget and :diffput

Solution: Add tests.

Files: src/testdir/test_diffmode.vim

Patch 7.4.2127

Problem: The short form of ":noswapfile" is ":noswap" instead of ":nos".
(Kent Sibilev)
Solution: Only require three characters. Add a test for the short forms.
Files: src/ex_docmd.c, src/testdir/test_usercommands.vim

Patch 7.4.2128

Problem: Memory leak when saving for undo fails.
Solution: Free allocated memory. (Hirohito Higashi)
Files: src/ex_cmds.c

Patch 7.4.2129

Problem: Memory leak when using timer_start(). (Dominique Pelle)
Solution: Don't copy the callback when using a partial.
Files: src/evalfunc.c

Patch 7.4.2130

Problem: Pending timers cause false memory leak reports.
Solution: Free all timers on exit.
Files: src/ex_cmds2.c, src/proto/ex_cmds2.pro, src/misc2.c

Patch 7.4.2131

Problem: More memory leaks when using partial, e.g. for "exit-cb".
Solution: Don't copy the callback when using a partial.
Files: src/channel.c

Patch 7.4.2132

Problem: test_partial has memory leaks reported.
Solution: Add a [note](#) about why this happens.
Files: src/testdir/test_partial.vim

Patch 7.4.2133 (after 7.4.2128)

Problem: Can't build with tiny features.
Solution: Add #ifdef.
Files: src/ex_cmds.c

Patch 7.4.2134

Problem: No error for using function() badly.
Solution: Check for passing wrong function name. (Ken Takata)
Files: src/eval.c, src/evalfunc.c, src/proto/userfunc.pro, src/testdir/test_expr.vim, src/userfunc.c, src/vim.h

Patch 7.4.2135

Problem: Various tiny issues.
Solution: Update comments, white space, etc.
Files: src/diff.c, src/digraph.c, src/testdir/test80.in, src/testdir/test_channel.vim, src/testdir/Makefile, runtime/menu.vim, src/INSTALLpc.txt, src/xpm/README.txt

Patch 7.4.2136

Problem: Closure function fails.
Solution: Don't reset uf_scoped when it points to another funccal.
Files: src/userfunc.c, src/testdir/test_lambda.vim

Patch 7.4.2137

Problem: Using function() with a name will find another function when it is redefined.

Solution: Add funcref(). Refer to lambda using a partial. Fix several reference counting issues.

Files: src/vim.h, src/structs.h, src/userfunc.c, src/eval.c, src/evalfunc.c, src/channel.c, src/proto/eval.pro, src/proto/userfunc.pro, src/if_mzsch.c, src/regexp.c, src/misc2.c, src/if_py_both.h, src/testdir/test_expr.vim, runtime/doc/eval.txt

Patch 7.4.2138

Problem: Test 86 and 87 fail.

Solution: Call func_ref() also for regular functions.

Files: src/if_py_both.h

Patch 7.4.2139

Problem: :delfunction causes illegal memory access.

Solution: Correct logic when deciding to free a function.

Files: src/userfunc.c, src/testdir/test_lambda.vim

Patch 7.4.2140

Problem: Tiny build fails.

Solution: Add dummy typedefs.

Files: src/structs.h

Patch 7.4.2141

Problem: Coverity reports bogus NULL check.

Solution: When checking for a variable in the funccal scope don't pass the varname.

Files: src/userfunc.c, src/proto/userfunc.pro, src/eval.c

Patch 7.4.2142

Problem: Leaking memory when redefining a function.

Solution: Don't increment the function reference count when it's found by name. Don't remove the wrong function from the hashtable. More reference counting fixes.

Files: src/structs.h, src/userfunc.c

Patch 7.4.2143

Problem: A funccal is garbage collected while it can still be used.

Solution: Set copyID in all referenced functions. Do not list lambda functions with ":function".

Files: src/userfunc.c, src/proto/userfunc.pro, src/eval.c, src/testdir/test_lambda.vim

Patch 7.4.2144

Problem: On MS-Windows quickfix does not handle a line with 1023 bytes ending in CR-LF properly.

Solution: Don't consider CR a line break. (Ken Takata)

Files: src/quickfix.c

Patch 7.4.2145

Problem: Win32: Using CreateThread/ExitThread is not safe.

Solution: Use _beginthreadex and return from the thread. (Ken Takata)

Files: src/os_win32.c

Patch 7.4.2146

Problem: Not enough testing for popup menu. **CTRL-E** does not always work properly.

Solution: Add more tests. When using **CTRL-E** check if the popup menu is visible. (Christian Brabandt)

Files: src/edit.c, src/testdir/test_popup.vim

Patch 7.4.2147 (after 7.4.2146)

Problem: test_alot fails.

Solution: Close window.

Files: src/testdir/test_popup.vim

Patch 7.4.2148

Problem: Not much testing for cscope.

Solution: Add a test that uses the cscope program. (Christian Brabandt)

Files: src/testdir/test_cscope.vim

Patch 7.4.2149

Problem: If a test leaves a window open a following test may fail.

Solution: Always close extra windows after running a test.

Files: src/testdir/runtest.vim, src/testdir/test_popup.vim

Patch 7.4.2150

Problem: Warning with MinGW 64. (John Marriott)

Solution: Change return type. (Ken Takata)

Files: src/os_win32.c

Patch 7.4.2151

Problem: Quickfix test fails on MS-Windows.

Solution: Close the help window. (Christian Brabandt)

Files: src/testdir/test_quickfix.vim

Patch 7.4.2152

Problem: No proper translation of messages with a count.

Solution: Use ngettext(). (Sergey Alyoshin)

Files: src/evalfunc.c, src/fold.c, src/os_win32.c, src/screen.c, src/vim.h

Patch 7.4.2153

Problem: GUI test isn't testing much.

Solution: Turn into a new style test. Execute a shell command.

Files: src/testdir/test_gui.vim, src/testdir/test16.in,
src/testdir/test16.ok, src/testdir/Make_all.mak, src/Makefile,
src/testdir/Make_vms.mms

Patch 7.4.2154

Problem: Test_communicate() fails sometimes.

Solution: Add it to the flaky tests.

Files: src/testdir/runtest.vim

Patch 7.4.2155

Problem: Quotes make GUI test fail on MS-Windows.

Solution: Remove quotes, strip white space.

Files: src/testdir/test_gui.vim

Patch 7.4.2156

Problem: Compiler warning.
Solution: Add type cast. (Ken Takata, Mike Williams)
Files: src/os_win32.c

Patch 7.4.2157

Problem: Test_job_start_fails() is expected to report memory leaks, making it hard to see other leaks in test_partial.
Solution: Move Test_job_start_fails() to a separate test file.
Files: src/testdir/test_partial.vim, src/testdir/test_job_fails.vim, src/Makefile, src/testdir/Make_all.mak

Patch 7.4.2158

Problem: Result of getcompletion('', 'cscope') depends on previous completion. (Christian Brabandt)
Solution: Call set_context_in_cscope_cmd().
Files: src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 7.4.2159

Problem: Insufficient testing for cscope.
Solution: Add more tests. (Dominique Pelle)
Files: src/testdir/test_cscope.vim

Patch 7.4.2160

Problem: setmatches() mixes up values. (Nikolai Pavlov)
Solution: Save the string instead of reusing a shared buffer.
Files: src/dict.c, src/evalfunc.c, src/testdir/test_expr.vim,

Patch 7.4.2161 (after 7.4.2160)

Problem: Expression test fails without conceal feature.
Solution: Only check "conceal" with the conceal feature.
Files: src/testdir/test_expr.vim

Patch 7.4.2162

Problem: Result of getcompletion('', 'sign') depends on previous completion.
Solution: Call set_context_in_sign_cmd(). (Dominique Pelle)
Files: src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 7.4.2163

Problem: match() and related functions tested with old style test.
Solution: Convert to new style test. (Hirohito Higashi)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test63.in, src/testdir/test63.ok, src/testdir/test_alot.vim, src/testdir/test_match.vim, src/testdir/test_matchstrpos.vim

Patch 7.4.2164

Problem: It is not possible to use plugins in an "after" directory to tune the behavior of a package.
Solution: First load plugins from non-after directories, then packages and finally plugins in after directories.
Reset '**loadplugins**' before executing --cmd arguments.

Files: src/main.c, src/vim.h, src/ex_cmds2.c, src/testdir/Makefile,
src/testdir/shared.vim, src/testdir/test_startup.vim,
src/testdir/setup.vim, runtime/doc/starting.txt

Patch 7.4.2165 (after 7.4.2164)

Problem: Startup test fails on MS-Windows.
Solution: Don't check output if RunVim() returns zero.
Files: src/testdir/test_startup.vim

Patch 7.4.2166 (after 7.4.2164)

Problem: Small build can't run startup test.
Solution: Skip the test.
Files: src/testdir/test_startup.vim

Patch 7.4.2167 (after 7.4.2164)

Problem: Small build can't run tests.
Solution: Don't try setting '**packpath**'.
Files: src/testdir/setup.vim

Patch 7.4.2168

Problem: Not running the startup test on MS-Windows.
Solution: Write vimcmd.
Files: src/testdir/Make_ming.mak, src/testdir/Make_dos.mak

Patch 7.4.2169 (after 7.4.2168)

Problem: Startup test gets stuck on MS-Windows.
Solution: Use double quotes.
Files: src/testdir/shared.vim, src/testdir/test_startup.vim

Patch 7.4.2170

Problem: Cannot get information about timers.
Solution: Add timer_info().
Files: src/evalfunc.c, src/ex_cmds2.c, src/proto/ex_cmds2.pro,
runtime/doc/eval.txt

Patch 7.4.2171 (after 7.4.2170)

Problem: MS-Windows build fails.
Solution: Add QueryPerformanceCounter().
Files: src/ex_cmds2.c

Patch 7.4.2172

Problem: No test for "vim --help".
Solution: Add a test.
Files: src/testdir/test_startup.vim, src/testdir/shared.vim

Patch 7.4.2173 (after 7.4.2172)

Problem: Can't test help on MS-Windows.
Solution: Skip the test.
Files: src/testdir/test_startup.vim

Patch 7.4.2174

Problem: Adding duplicate flags to '**whichwrap**' leaves commas behind.
Solution: Also remove the commas. (Naruhiko Nishino)
Files: src/Makefile, src/option.c, src/testdir/Make_all.mak,

src/testdir/test_alot.vim, src/testdir/test_options.in,
src/testdir/test_options.ok, src/testdir/test_options.vim

Patch 7.4.2175

Problem: Insufficient testing of cscope.
Solution: Add more tests. (Dominique Pelle)
Files: src/testdir/test_cscope.vim

Patch 7.4.2176

Problem: #ifdefs in main() are complicated.
Solution: Always define vim_main2(). Move params to the file level.
(suggested by Ken Takata)
Files: src/main.c, src/structs.h, src/vim.h, src/if_mzsch.c,
src/proto/if_mzsch.pro

Patch 7.4.2177

Problem: No testing for -C and -N command line flags, file arguments,
startuptime.
Solution: Add tests.
Files: src/testdir/test_startup.vim, src/testdir/shared.vim

Patch 7.4.2178

Problem: No test for reading from stdin.
Solution: Add a test.
Files: src/testdir/test_startup.vim, src/testdir/shared.vim

Patch 7.4.2179 (after 7.4.2178)

Problem: Reading from stdin test fails on MS-Windows.
Solution: Strip the extra space.
Files: src/testdir/test_startup.vim

Patch 7.4.2180

Problem: There is no easy way to stop all timers. There is no way to
temporary pause a timer.
Solution: Add timer_stopall() and timer_pause().
Files: src/evalfunc.c, src/ex_cmds2.c, src/proto/ex_cmds2.pro,
src/structs.h, src/testdir/test_timers.vim,
src/testdir/shared.vim, runtime/doc/eval.txt

Patch 7.4.2181

Problem: Compiler warning for unused variable.
Solution: Remove it. (Dominique Pelle)
Files: src/ex_cmds2.c

Patch 7.4.2182

Problem: Color Grey40 used in startup but not in the short list.
Solution: Add Grey40 to the builtin colors.
Files: src/term.c

Patch 7.4.2183

Problem: Sign tests are old style.
Solution: Turn them into new style tests. (Dominique Pelle)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_signs.in,
src/testdir/test_signs.ok, src/testdir/test_signs.vim,

Patch 7.4.2184

Problem: Tests that use RunVim() do not actually perform the test.
Solution: Use "return" instead of "call". (Ken Takata)
Files: src/testdir/shared.vim

Patch 7.4.2185

Problem: Test glob2regpat does not test much.
Solution: Add a few more test cases. (Dominique Pelle)
Files: src/testdir/test_glob2regpat.vim

Patch 7.4.2186

Problem: Timers test is flaky.
Solution: Relax the sleep time check.
Files: src/testdir/test_timers.vim

Patch 7.4.2187 (after 7.4.2185)

Problem: glob2regpat test fails on Windows.
Solution: Remove the checks that use backslashes.
Files: src/testdir/test_glob2regpat.vim

Patch 7.4.2188 (after 7.4.2146)

Problem: Completion does not work properly with some plugins.
Solution: Revert the part related to typing **CTRL-E**. (closes #972)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 7.4.2189

Problem: Cannot detect encoding in a fifo.
Solution: Extend the stdin way of detecting encoding to fifo. Add a test for detecting encoding on stdin and fifo. (Ken Takata)
Files: src/buffer.c, src/fileio.c, src/Makefile, src/testdir/Make_all.mak, src/testdir/test_startup_utf8.vim, src/vim.h

Patch 7.4.2190

Problem: When startup test fails it's not easy to find out why. GUI test fails with Gnome.
Solution: Add the help entry matches to a list an assert that. Set \$HOME for Gnome to create .gnome2 directory.
Files: src/testdir/test_startup.vim, src/testdir/test_gui.vim

Patch 7.4.2191

Problem: No automatic prototype for vim_main2().
Solution: Move the #endif. (Ken Takata)
Files: src/main.c, src/vim.h, src/proto/main.pro

Patch 7.4.2192

Problem: Generating prototypes with Cygwin doesn't work well.
Solution: Change #ifdefs. (Ken Takata)
Files: src/gui.h, src/gui_w32.c, src/ops.c, src/proto/fileio.pro, src/proto/message.pro, src/proto/normal.pro, src/proto/ops.pro, src/vim.h

Patch 7.4.2193

Problem: With Gnome when the GUI can't start test_startup hangs.
Solution: Call gui_mch_early_init_check(). (Hirohito Higashi)
Files: src/gui.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro

Patch 7.4.2194

Problem: Sign tests don't cover enough.
Solution: Add more test cases. (Dominique Pelle)
Files: src/testdir/test_signs.vim

Patch 7.4.2195

Problem: MS-Windows: The vimrun program does not support Unicode.
Solution: Use GetCommandLineW(). Cleanup old #ifdefs. (Ken Takata)
Files: src/vimrun.c

Patch 7.4.2196

Problem: glob2regpat test doesn't test everything on MS-Windows.
Solution: Add patterns with backslash handling.
Files: src/testdir/test_glob2regpat.vim

Patch 7.4.2197

Problem: All functions are freed on exit, which may hide leaks.
Solution: Only free named functions, not reference counted ones.
Files: src/userfunc.c

Patch 7.4.2198

Problem: Test alot sometimes fails under valgrind. (Dominique Pelle)
Solution: Avoid passing a callback with the wrong number of arguments.
Files: src/testdir/test_partial.vim

Patch 7.4.2199

Problem: In the GUI the cursor is hidden when redrawing any window, causing flicker.
Solution: Only undraw the cursor when updating the window it's in.
Files: src/screen.c, src/gui.c, src/proto/gui.pro, src/gui_gtk_x11.c

Patch 7.4.2200

Problem: Cannot get all information about a quickfix list.
Solution: Add an optional argument to get/set loc/qf list(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/quickfix.pro, src/quickfix.c, src/tag.c, src/testdir/test_quickfix.vim

Patch 7.4.2201

Problem: The sign column disappears when the last sign is deleted.
Solution: Add the '**signcolumn**' option. (Christian Brabandt)
Files: runtime/doc/options.txt, runtime/optwin.vim, src/edit.c, src/move.c, src/option.c, src/option.h, src/proto/option.pro, src/screen.c, src/structs.h, src/testdir/test_options.vim

Patch 7.4.2202

Problem: Build fails with small features.
Solution: Correct option initialization.
Files: src/option.c

Patch 7.4.2203

Problem: Test fails with normal features.
Solution: Check if signs are supported.
Files: src/testdir/test_options.vim

Patch 7.4.2204

Problem: It is not easy to get information about buffers, windows and tabpages.
Solution: Add getbufinfo(), getwininfo() and gettabinfo(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/dict.c, src/evalfunc.c, src/option.c, src/proto/dict.pro, src/proto/option.pro, src/proto/window.pro, src/testdir/Make_all.mak, src/testdir/test_bufwintabinfo.vim, src/window.c, src/Makefile

Patch 7.4.2205

Problem: **'wildignore'** always applies to getcompletion().
Solution: Add an option to use **'wildignore'** or not. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 7.4.2206

Problem: Warning for unused function.
Solution: Put the function inside #ifdef. (John Marriott)
Files: src/evalfunc.c

Patch 7.4.2207

Problem: The +xpm feature is not sorted properly in :version output.
Solution: Move it up. (Tony Mechelynck)
Files: src/version.c

Patch 7.4.2208

Problem: Test for mappings is old style.
Solution: Convert the test to new style.
Files: src/testdir/test_mapping.vim, src/testdir/test_mapping.in, src/testdir/test_mapping.ok, src/Makefile, src/testdir/test_alot.vim, src/testdir/Make_all.mak

Patch 7.4.2209

Problem: Cannot map <M->. (Stephen Riehm)
Solution: Solve the memory access problem in another way. (Dominique Pelle)
Allow for using <M-\> in a string.
Files: src/eval.c, src/gui_mac.c, src/misc2.c, src/option.c, src/proto/misc2.pro, src/syntax.c, src/term.c, src/testdir/test_mapping.vim

Patch 7.4.2210

Problem: On OSX configure mixes up a Python framework and the Unix layout.
Solution: Make configure check properly. (Tim D. Smith, closes #980)
Files: src/configure.in, src/auto/configure

Patch 7.4.2211

Problem: Mouse support is not automatically enabled with simple term.
Solution: Recognize "st" and other names. (Manuel Schiller, closes #963)

Files: src/os_unix.c

Patch 7.4.2212

Problem: Mark " is not set when closing a window in another tab. (Guraga)

Solution: Check all tabs for the window to be valid. (based on patch by Hirohito Higashi, closes #974)

Files: src/window.c, src/proto/window.pro, src/buffer.c,
src/testdir/test_viminfo.vim

Patch 7.4.2213

Problem: Cannot highlight the "~" lines at the end of a window differently.

Solution: Add the EndOfBuffer highlighting. (Marco Hinz, James McCoy)

Files: runtime/doc/options.txt, runtime/doc/syntax.txt, src/option.c,
src/screen.c, src/syntax.c, src/vim.h

Patch 7.4.2214

Problem: A font that uses ligatures messes up the screen display.

Solution: Put spaces between characters when building the glyph table.
(based on a patch from Manuel Schiller)

Files: src/gui_gtk_x11.c

Patch 7.4.2215

Problem: It's not easy to find out if a window is a quickfix or location
list window.

Solution: Add "loclist" and "quickfix" entries to the dict returned by
getwininfo(). (Yegappan Lakshmanan)

Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_bufwintabinfo.vim

Patch 7.4.2216 (after 7.4.2215)

Problem: Test fails without the +sign feature.

Solution: Only check for signcolumn with the +sign feature.

Files: src/testdir/test_bufwintabinfo.vim

Patch 7.4.2217

Problem: When using matchaddpos() a character after the end of the line can
be highlighted.

Solution: Only highlight existing characters. (Hirohito Higashi)

Files: src/screen.c, src/structs.h, src/testdir/test_match.vim

Patch 7.4.2218

Problem: Can't build with +timers when +digraph is not included.

Solution: Change #ifdef for e_number_exp. (Damien)

Files: src/globals.h

Patch 7.4.2219

Problem: Recursive call to substitute gets stuck in sandbox. (Nikolai
Pavlov)

Solution: Handle the recursive call. (Christian Brabandt, closes #950)
Add a test.

Files: src/ex_cmds.c, src/testdir/test_regexp_latin.vim

Patch 7.4.2220

Problem: printf() gives an error when the argument for %s is not a string.

(Ozaki Kiichi)
Solution: Behave like invoking string() on the argument. (Ken Takata)
Files: runtime/doc/eval.txt, src/message.c, src/testdir/test_expr.vim

Patch 7.4.2221

Problem: printf() does not support binary format.
Solution: Add %b and %B. (Ozaki Kiichi)
Files: runtime/doc/eval.txt, src/message.c, src/testdir/test_expr.vim

Patch 7.4.2222

Problem: Sourcing a script where a character has 0x80 as a second byte does not work. (Filipe L B Correia)
Solution: Turn 0x80 into K_SPECIAL KS_SPECIAL KE_FILLER. (Christian Brabandt, closes #728) Add a test case.
Files: src/getchar.c, src/proto/getchar.pro, src/misc1.c, src/testdir/test_regexp_utf8.vim

Patch 7.4.2223

Problem: Buffer overflow when using latin1 character with feedkeys().
Solution: Check for an illegal character. Add a test.
Files: src/testdir/test_regexp_utf8.vim, src/testdir/test_source_utf8.vim, src/testdir/test_alot_utf8.vim, src/Makefile, src/getchar.c, src/macros.h, src/evalfunc.c, src/os_unix.c, src/os_win32.c, src/spell.c,

Patch 7.4.2224

Problem: Compiler warnings with older compiler and 64 bit numbers.
Solution: Add "LL" to large values. (Mike Williams)
Files: src/eval.c, src/evalfunc.c

Patch 7.4.2225

Problem: Crash when placing a sign in a deleted buffer.
Solution: Check for missing buffer name. (Dominique Pelle). Add a test.
Files: src/ex_cmds.c, src/testdir/test_signs.vim

Patch 7.4.2226

Problem: The field names used by getbufinfo(), gettabinfo() and getwininfo() are not consistent.
Solution: Use bufnr, winnr and tabnr. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_bufwintabinfo.vim

Patch 7.4.2227

Problem: Tab page tests are old style.
Solution: Change into new style tests. (Hirohito Higashi)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test62.in, src/testdir/test62.ok, src/testdir/test_alot.vim, src/testdir/test_tabpage.vim

Patch 7.4.2228

Problem: Test files have inconsistent modelines.
Solution: Don't set 'tabstop' to 2, use 'sts' and 'sw'.
Files: src/testdir/README.txt, src/testdir/test_backspace_opt.vim, src/testdir/test_digraph.vim, src/testdir/test_gn.vim

src/testdir/test_help_tagjump.vim,
src/testdir/test_increment_dbcs.vim,
src/testdir/test_increment.vim, src/testdir/test_match.vim,
src/testdir/test_tagjump.vim, src/testdir/test_window_cmd.vim,
src/testdir/test_regexp_latin.vim, src/testdir/test_timers.vim

Patch 7.4.2229

Problem: Startup test fails on Solaris.
Solution: Recognize a character device. (Danek Duvall)
Files: src/buffer.c, src/fileio.c, src/proto/fileio.pro, src/vim.h

Patch 7.4.2230

Problem: There is no equivalent of **'smartcase'** for a tag search.
Solution: Add value "followscs" and "smart" to **'tagcase'**. (Christian Brabandt, closes #712) Turn tagcase test into new style.
Files: runtime/doc/options.txt, runtime/doc/tagsrch.txt, src/option.h,
src/tag.c, src/search.c, src/proto/search.pro,
src/testdir/test_tagcase.in, src/testdir/test_tagcase.ok,
src/testdir/test_tagcase.vim, src/Makefile,
src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 7.4.2231

Problem: ":oldfiles" output is a very long list.
Solution: Add a pattern argument. (Coot, closes #575)
Files: runtime/doc/starting.txt, src/ex_cmds.h, src/eval.c,
src/ex_cmds.c, src/proto/eval.pro, src/proto/ex_cmds.pro,
src/testdir/test_viminfo.vim

Patch 7.4.2232

Problem: The default ttimeoutlen is very long.
Solution: Use "100". (Hirohito Higashi)
Files: runtime/defaults.vim

Patch 7.4.2233

Problem: Crash when using funcref() with invalid name. (Dominique Pelle)
Solution: Check for NULL translated name.
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 7.4.2234

Problem: Can't build with +eval but without +quickfix. (John Marriott)
Solution: Move skip_vimgrep_pat() to separate #ifdef block.
Files: src/quickfix.c

Patch 7.4.2235

Problem: submatch() does not check for a valid argument.
Solution: Give an error if the argument is out of range. (Dominique Pelle)
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 7.4.2236

Problem: The **'langnoremap'** option leads to double negatives. And it does not work for the last character of a mapping.
Solution: Add **'langremap'** with the opposite value. Keep **'langnoremap'** for backwards compatibility. Make it work for the last character of a mapping. Make the test work.

Files: runtime/doc/options.txt, runtime/defaults.vim, src/option.c,
src/option.h, src/macros.h, src/testdir/test_mapping.vim

Patch 7.4.2237

Problem: Can't use "." and "\$" with ":tab".
Solution: Support a range for ":tab". (Hirohito Higashi)
Files: runtime/doc/tabpage.txt, src/ex_docmd.c,
src/testdir/test_tabpage.vim

Patch 7.4.2238

Problem: With SGR mouse reporting (suckless terminal) the mouse release and
scroll up/down is confused.
Solution: Don't see a release as a scroll up/down. (Ralph Eastwood)
Files: src/term.c

Patch 7.4.2239

Problem: Warning for missing declaration of skip_vimgrep_pat(). (John
Marriott)
Solution: Move it to another file.
Files: src/quickfix.c, src/proto/quickfix.pro, src/ex_cmds.c,
src/proto/ex_cmds.pro

Patch 7.4.2240

Problem: Tests using the sleep time can be flaky.
Solution: Use reltime() if available. (Partly by Shane Harper)
Files: src/testdir/shared.vim, src/testdir/test_timers.vim

Patch 7.4.2241 (after 7.4.2240)

Problem: Timer test sometimes fails.
Solution: Increase the maximum time for repeating timer.
Files: src/testdir/test_timers.vim

Patch 7.4.2242 (after 7.4.2240)

Problem: Timer test sometimes fails.
Solution: Increase the maximum time for callback timer test.
Files: src/testdir/test_timers.vim

Patch 7.4.2243

Problem: Warning for assigning negative value to unsigned. (Danek Duvall)
Solution: Make cterm_normal_fg_gui_color and _bg_ guicolor_T, cast to long_u
only when an unsigned is needed.
Files: src/structs.h, src/globals.h, src/screen.c, src/term.c,
src/syntax.c, src/gui_gtk_x11.c, src/gui.c, src/gui_mac.c,
src/gui_photon.c, src/gui_w32.c, src/gui_x11.c,
src/proto/term.pro, src/proto/gui_gtk_x11.pro,
src/proto/gui_mac.pro, src/proto/gui_photon.pro,
src/proto/gui_w32.pro, src/proto/gui_x11.pro

Patch 7.4.2244

Problem: Adding pattern to ":oldfiles" is not a generic solution.
Solution: Add the ":filter /pat/ cmd" command modifier. Only works for some
commands right now.
Files: src/structs.h, src/ex_docmd.c, src/ex_cmds.h, src/message.c,
src/proto/message.pro, runtime/doc/starting.txt,

runtime/doc/various.txt, src/testdir/test_viminfo.vim,
src/testdir/test_alot.vim, src/testdir/test_filter_cmd.vim,
src/Makefile

Patch 7.4.2245 (after 7.4.2244)

Problem: Filter test fails.
Solution: Include missing changes.
Files: src/buffer.c

Patch 7.4.2246 (after 7.4.2244)

Problem: Oldfiles test fails.
Solution: Include missing changes.
Files: src/ex_cmds.c

Patch 7.4.2247 (after 7.4.2244)

Problem: Tiny build fails. (Tony Mechelynck)
Solution: Remove #ifdef.
Files: src/ex_cmds.c

Patch 7.4.2248

Problem: When cancelling the :ptjump prompt a preview window is opened for a following command.
Solution: Reset g_do_tagpreview. (Hirohito Higashi) Add a test. Avoid that the test runner gets stuck in trying to close a window.
Files: src/tag.c, src/testdir/test_tagjump.vim, src/testdir/runtest.vim

Patch 7.4.2249

Problem: Missing colon in error message.
Solution: Add the colon. (Dominique Pelle)
Files: src/userfunc.c

Patch 7.4.2250

Problem: Some error messages cannot be translated.
Solution: Enclose them in _() and N_(). (Dominique Pelle)
Files: src/channel.c, src/evalfunc.c, src/ex_cmds.c, src/spell.c, src/window.c

Patch 7.4.2251

Problem: In rare cases diffing 4 buffers is not enough.
Solution: Raise the limit to 8. (closes #1000)
Files: src/structs.h, runtime/doc/diff.txt

Patch 7.4.2252

Problem: Compiler warnings for signed/unsigned in expression.
Solution: Remove type cast. (Dominique Pelle)
Files: src/vim.h

Patch 7.4.2253

Problem: Check for Windows 3.1 will always return false. (Christian Brabandt)
Solution: Remove the dead code.
Files: src/gui_w32.c, src/evalfunc.c, src/ex_cmds.c, src/option.c, src/os_win32.c, src/version.c, src/proto/gui_w32.pro

Patch 7.4.2254

Problem: Compiler warnings in MzScheme code.
Solution: Add UNUSED. Remove unreachable code.
Files: src/if_mzsch.c

Patch 7.4.2255

Problem: The script that checks translations can't handle plurals.
Solution: Check for plural msgid and msgstr entries. Leave the cursor on the first error.
Files: src/po/check.vim

Patch 7.4.2256

Problem: Coverity complains about null pointer check.
Solution: Remove wrong and superfluous error check.
Files: src/eval.c

Patch 7.4.2257

Problem: Coverity complains about not checking for NULL.
Solution: Check for out of memory.
Files: src/if_py_both.h

Patch 7.4.2258

Problem: Two JSON messages are sent without a separator.
Solution: Separate messages with a NL. (closes #1001)
Files: src/json.c, src/channel.c, src/vim.h, src/testdir/test_channel.py, src/testdir/test_channel.vim, runtime/doc/channel.txt

Patch 7.4.2259

Problem: With **'incsearch'** can only see the next match.
Solution: Make **CTRL-N/CTRL-P** move to the previous/next match. (Christian Brabandt)
Files: runtime/doc/cmdline.txt, src/ex_getln.c, src/testdir/Make_all.mak, src/testdir/test_search.vim, src/Makefile

Patch 7.4.2260 (after 7.4.2258)

Problem: Channel test is flaky.
Solution: Add a newline to separate JSON messages.
Files: src/testdir/test_channel.vim

Patch 7.4.2261 (after 7.4.2259)

Problem: Build fails with small features.
Solution: Move "else" inside the #ifdef.
Files: src/ex_getln.c

Patch 7.4.2262

Problem: Fail to read register content from viminfo if it is 438 characters long. (John Chen)
Solution: Adjust the check for line wrapping. (closes #1010)
Files: src/testdir/test_viminfo.vim, src/ex_cmds.c

Patch 7.4.2263

Problem: `:filter` does not work for many commands. Can only get matching messages.
Solution: Make `:filter` work for `:command`, `:map`, `:list`, `:number` and `:print`.

Make ":filter!" show non-matching lines.
Files: src/getchar.c, src/ex_cmds.c, src/ex_cmds.h, src/ex_docmd.c,
src/message.c, src/structs.h, src/testdir/test_filter_cmd.vim

Patch 7.4.2264

Problem: When adding entries to an empty quickfix list the title is reset.
Solution: Improve handling of the title. (Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim, src/quickfix.c

Patch 7.4.2265

Problem: printf() isn't tested much.
Solution: Add more tests for printf(). (Dominique Pelle)
Files: src/testdir/test_expr.vim

Patch 7.4.2266 (after 7.4.2265)

Problem: printf() test fails on Windows. "-inf" is not used.
Solution: Check for Windows-specific values for "nan". Add sign to "inf"
when appropriate.
Files: src/message.c, src/testdir/test_expr.vim

Patch 7.4.2267 (after 7.4.2266)

Problem: Build fails on MS-Windows.
Solution: Add define to get isinf().
Files: src/message.c

Patch 7.4.2268 (after 7.4.2259)

Problem: Using **CTRL-N** and **CTRL-P** for incsearch shadows completion keys.
Solution: Use **CTRL-T** and **CTRL-G** instead.
Files: runtime/doc/cmdline.txt, src/ex_getln.c,
src/testdir/test_search.vim

Patch 7.4.2269

Problem: Using **'hlsearch'** highlighting instead of matchpos if there is no
search match.
Solution: Pass NULL as last item to next_search_hl() when searching for
'hlsearch' match. (Shane Harper, closes #1013)
Files: src/screen.c, src/testdir/test_match.vim

Patch 7.4.2270

Problem: Insufficient testing for NUL bytes on a raw channel.
Solution: Add a test for writing and reading.
Files: src/testdir/test_channel.vim

Patch 7.4.2271

Problem: Netbeans test doesn't read settings from file.
Solution: Use "-Xnbauth".
Files: src/testdir/test_netbeans.vim

Patch 7.4.2272

Problem: getbufinfo(), getwininfo() and gettabinfo() are inefficient.
Solution: Instead of making a copy of the variables dictionary, use a
reference.
Files: src/evalfunc.c

Patch 7.4.2273

Problem: getwininfo() and getbufinfo() are inefficient.
Solution: Do not make a copy of all window/buffer-local options. Make it possible to get them with gettabwinvar() or getbufvar().
Files: src/evalfunc.c, src/eval.c, src/testdir/test_bufwintabinfo.vim, runtime/doc/eval.txt

Patch 7.4.2274

Problem: Command line completion on "find **/filename" drops sub-directory.
Solution: Handle this case separately. (Harm te Hennepe, closes #932, closes #939)
Files: src/misc1.c, src/testdir/test_cmdline.vim

Patch 7.4.2275

Problem: ":diffoff!" does not remove filler lines.
Solution: Force a redraw and invalidate the cursor. (closes #1014)
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 7.4.2276

Problem: Command line test fails on Windows when run twice.
Solution: Wipe the buffer so that the directory can be deleted.
Files: src/testdir/test_cmdline.vim

Patch 7.4.2277

Problem: Memory leak in getbufinfo() when there is a sign. (Dominique Pelle)
Solution: Remove extra vim_strsave().
Files: src/evalfunc.c

Patch 7.4.2278

Problem: New users have no idea of the '**scrolloff**' option.
Solution: Set '**scrolloff**' in defaults.vim.
Files: runtime/defaults.vim

Patch 7.4.2279

Problem: Starting diff mode with the cursor in the last line might end up only showing one closed fold. (John Beckett)
Solution: Scroll the window to show the same relative cursor position.
Files: src/diff.c, src/window.c, src/proto/window.pro

Patch 7.4.2280

Problem: printf() doesn't handle infinity float values correctly.
Solution: Add a table with possible infinity values. (Dominique Pelle)
Files: src/message.c, src/testdir/test_expr.vim

Patch 7.4.2281

Problem: Timer test fails sometimes.
Solution: Reduce minimum time by 1 msec.
Files: src/testdir/test_timers.vim

Patch 7.4.2282

Problem: When a child process is very fast waiting 10 msec for it is noticeable. (Ramel Eshed)
Solution: Start waiting for 1 msec and gradually increase.

Files: src/os_unix.c

Patch 7.4.2283

Problem: Part of ":oldfiles" command isn't cleared. (Lifepillar)

Solution: Clear the rest of the line. (closes 1018)

Files: src/ex_cmds.c

Patch 7.4.2284

Problem: Comment in scope header file is outdated. (KillTheMule)

Solution: Point to the help instead. (closes #1017)

Files: src/if_scope.h

Patch 7.4.2285

Problem: Generated files are outdated.

Solution: Generate the files. Avoid errors when generating prototypes.

Files: src/if_mzsch.h, src/Makefile, src/option.h, src/os_mac_conv.c,
src/os_amiga.c, src/vim.h, src/structs.h, src/os_win32.c,
src/if_lua.c, src/proto/mbyte.pro

Patch 7.4.2286

Problem: The tee program isn't included. Makefile contains build instructions that don't work.

Solution: Update the Filelist and build instructions. Remove build instructions for DOS and old Windows. Add the tee program.

Files: Filelist, Makefile, nsis/gvim.nsi

Patch 7.4.2287

Problem: The callback passed to ch_sendraw() is not used.

Solution: Pass the read part, not the send part. (haya14busa, closes #1019)

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.2288

Problem: MS-Windows build instructions are clumsy. "dosbin" doesn't build.

Solution: Add rename.bat. Fix building "dosbin".

Files: Makefile, Filelist, rename.bat

Patch 7.4.2289

Problem: When installing and \$DESTDIR is set the icons probably won't be installed.

Solution: Create the icon directories if \$DESTDIR is not empty. (Danek Duvall)

Files: src/Makefile

Patch 7.4.2290

Problem: Compiler warning in tiny build. (Tony Mechelynck)

Solution: Add #ifdef around infinity_str().

Files: src/message.c

Patch 7.4.2291

Problem: printf() handles floats wrong when there is a sign.

Solution: Fix placing the sign. Add tests. (Dominique Pelle)

Files: src/testdir/test_expr.vim, runtime/doc/eval.txt, src/message.c

Patch 7.4.2292 (after 7.4.2291)

Problem: Not all systems understand %F in printf().
Solution: Use %f.
Files: src/message.c

Patch 7.4.2293

Problem: Modelines in source code are inconsistent.
Solution: Use the same line in most files. Add **'noet'**. (Naruhiko Nishino)
Files: src/alloc.h, src/arabic.c, src/arabic.h, src/ascii.h,
src/blowfish.c, src/buffer.c, src/channel.c, src/charset.c,
src/crypt.c, src/crypt_zip.c, src/dict.c, src/diff.c,
src/digraph.c, src/dosinst.c, src/dosinst.h, src/edit.c,
src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds.h,
src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c,
src/farsi.c, src/farsi.h, src/feature.h, src/fileio.c, src/fold.c,
src/getchar.c, src/glbl_ime.cpp, src/glbl_ime.h, src/globals.h,
src/gui.c, src/gui.h, src/gui_at_fs.c, src/gui_at_sb.c,
src/gui_at_sb.h, src/gui_athena.c, src/gui_beval.c,
src/gui_beval.h, src/gui_gtk.c, src/gui_gtk_f.c, src/gui_gtk_f.h,
src/gui_gtk_vms.h, src/gui_gtk_x11.c, src/gui_mac.c,
src/gui_motif.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c,
src/gui_x11_pm.h, src/gui_xmdl.c, src/gui_xmew.c,
src/gui_xmew.h, src/gui_xmewp.h, src/hangulin.c, src/hardcopy.c,
src/hashtab.c, src/if_cscope.c, src/if_cscope.h, src/if_mzsch.c,
src/if_mzsch.h, src/if_ole.cpp, src/if_perl.xs, src/if_perlsfio.c,
src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/if_xcmds.c,
src/integration.c, src/integration.h, src/iscypty.c, src/json.c,
src/json_test.c, src/keymap.h, src/list.c, src/macros.h,
src/main.c, src/mark.c, src/mbyte.c, src/memfile.c,
src/memfile_test.c, src/memline.c, src/menu.c, src/message.c,
src/message_test.c, src/misc1.c, src/misc2.c, src/move.c,
src/nbdebug.c, src/nbdebug.h, src/netbeans.c, src/normal.c,
src/ops.c, src/option.c, src/option.h, src/os_amiga.c,
src/os_amiga.h, src/os_beos.c, src/os_beos.h, src/os_dos.h,
src/os_mac.h, src/os_mac_conv.c, src/os_macosx.m, src/os_mint.h,
src/os_mswin.c, src/os_qnx.c, src/os_qnx.h, src/os_unix.c,
src/os_unix.h, src/os_unixx.h, src/os_vms.c, src/os_w32dll.c,
src/os_w32exe.c, src/os_win32.c, src/os_win32.h, src/popupmnu.c,
src/proto.h, src/pty.c, src/quickfix.c, src/regexp.c,
src/regexp.h, src/regexp_nfa.c, src/screen.c, src/search.c,
src/sha256.c, src/spell.c, src/spell.h, src/spellfile.c,
src/structs.h, src/syntax.c, src/tag.c, src/term.c, src/term.h,
src/termlib.c, src/ui.c, src/undo.c, src/uninstal.c,
src/userfunc.c, src/version.c, src/version.h, src/vim.h,
src/vim.rc, src/vimio.h, src/vimrun.c, src/winclip.c,
src/window.c, src/workshop.c, src/workshop.h, src/wsdebug.c,
src/wsdebug.h, src/xpm_w32.c

Patch 7.4.2294

Problem: Sign test fails on MS-Windows when using the distributed zip archives.
Solution: Create dummy files instead of relying on files in the pixmaps directory.
Files: src/testdir/test_signs.vim

Patch 7.4.2295 (after 7.4.2293)

Problem: Cscope test fails.

Solution: Avoid checking for specific line and column numbers.

Files: src/testdir/test_cscope.vim

Patch 7.4.2296

Problem: No tests for :undolist and "U" command.

Solution: Add tests. (Dominique Pelle)

Files: src/testdir/test_undo.vim

Patch 7.4.2297

Problem: When starting a job that reads from a buffer and reaching the end, the job hangs.

Solution: Close the pipe or socket when all lines were read.

Files: src/channel.c, src/testdir/test_channel.vim

Patch 7.4.2298

Problem: It is not possible to close the "in" part of a channel.

Solution: Add ch_close_in().

Files: src/evalfunc.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, runtime/doc/eval.txt,
runtime/doc/channel.txt

Patch 7.4.2299

Problem: QuickFixCmdPre and QuickFixCmdPost autocommands are not always triggered.

Solution: Also trigger on ":cexpr", ":cbuffer", etc. (Yegappan Lakshmanan)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 7.4.2300

Problem: Get warning for deleting autocommand group when the autocommand using the group is scheduled for deletion. (Pavol Juhas)

Solution: Check for deleted autocommand.

Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 7.4.2301

Problem: MS-Windows: some files remain after testing.

Solution: Close the channel output file. Wait for the file handle to be closed before deleting the file.

Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 7.4.2302

Problem: Default interface versions for MS-Windows are outdated.

Solution: Use Active Perl 5.24, Python 3.5.2. Could only make it work with Ruby 1.9.2.

Files: src/bigvim.bat, src/bigvim64.bat, src/Make_mvc.mak

Patch 7.4.2303

Problem: When using "is" the mode isn't always updated.

Solution: Redraw the command line. (Christian Brabandt)

Files: src/search.c

Patch 7.4.2304

Problem: In a timer callback the timer itself can't be found or stopped.

(Thinca)
Solution: Do not remove the timer from the list, remember whether it was freed.
Files: src/ex_cmds2.c, src/testdir/test_timers.vim

Patch 7.4.2305

Problem: Marks, writefile and nested function tests are old style.
Solution: Turn them into new style tests. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak, src/testdir/test_marks.in, src/testdir/test_marks.ok, src/testdir/test_marks.vim, src/testdir/test_nested_function.in, src/testdir/test_nested_function.ok, src/testdir/test_nested_function.vim, src/testdir/test_writefile.in, src/testdir/test_writefile.ok, src/testdir/test_writefile.vim, src/Makefile

Patch 7.4.2306

Problem: Default value for '**langremap**' is wrong.
Solution: Set the right value. (Jürgen Krämer) Add a test.
Files: src/option.c, src/testdir/test_mapping.vim

Patch 7.4.2307

Problem: Several tests are old style.
Solution: Turn them into new style tests. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak, src/testdir/test102.in, src/testdir/test102.ok, src/testdir/test46.in, src/testdir/test46.ok, src/testdir/test81.in, src/testdir/test81.ok, src/testdir/test_charsearch.in, src/testdir/test_charsearch.ok, src/testdir/test_charsearch.vim, src/testdir/test_fnameescape.vim, src/testdir/test_substitute.vim, src/Makefile

Patch 7.4.2308 (after 7.4.2307)

Problem: Old charsearch test still listed in Makefile.
Solution: Remove the line.
Files: src/testdir/Make_all.mak

Patch 7.4.2309

Problem: Crash when doing tabnext in a BufUnload autocmd. (Dominique Pelle)
Solution: When detecting that the tab page changed, don't just abort but delete the window where w_buffer is NULL.
Files: src/window.c, src/testdir/test_tabpage.vim

Patch 7.4.2310 (after 7.4.2304)

Problem: Accessing freed memory when a timer does not repeat.
Solution: Free after removing it. (Dominique Pelle)
Files: src/ex_cmds2.c

Patch 7.4.2311

Problem: Appveyor 64 bit build still using Python 3.4
Solution: Switch to Python 3.5. (Ken Takata, closes #1032)
Files: appveyor.yml, src/appveyor.bat

Patch 7.4.2312

Problem: Crash when autocommand moves to another tab. (Dominique Pelle)
Solution: When navigating to another window halfway the :edit command go back to the right window.
Files: src/buffer.c, src/ex_cmds.c, src/ex_getln.c, src/ex_docmd.c, src/window.c, src/proto/ex_getln.pro, src/testdir/test_tabpage.vim

Patch 7.4.2313

Problem: Crash when deleting an augroup and listing an autocommand. (Dominique Pelle)
Solution: Make sure deleted_augroup is valid.
Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 7.4.2314

Problem: No error when deleting an augroup while it's the current one.
Solution: Disallow deleting an augroup when it's the current one.
Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 7.4.2315

Problem: Insufficient testing for Normal mode commands.
Solution: Add a big test. (Christian Brabandt, closes #1029)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_normal.vim

Patch 7.4.2316

Problem: Channel sort test is flaky.
Solution: Add a check the output has been read.
Files: src/testdir/test_channel.vim

Patch 7.4.2317 (after 7.4.2315)

Problem: Normal mode tests fail on MS-Windows.
Solution: Do some tests only on Unix. Set '**fileformat**' to "unix".
Files: src/testdir/test_normal.vim

Patch 7.4.2318

Problem: When '**incsearch**' is not set **CTRL-T** and **CTRL-G** are not inserted as before.
Solution: Move #ifdef and don't use goto.
Files: src/ex_getln.c

Patch 7.4.2319

Problem: No way for a system wide vimrc to stop loading defaults.vim. (Christian Hesse)
Solution: Bail out of defaults.vim if skip_defaults_vim was set.
Files: runtime/defaults.vim

Patch 7.4.2320

Problem: Redraw problem when using '**incsearch**'.
Solution: Save the current view when deleting characters. (Christian Brabandt) Fix that the '"' mark is set in the wrong position. Don't change the search start when using BS.
Files: src/ex_getln.c, src/normal.c, src/testdir/test_search.vim

Patch 7.4.2321

Problem: When a test is commented out we forget about it.

Solution: Let a test throw an exception with "Skipped" and list skipped test functions. (Christian Brabandt)
Files: src/testdir/Makefile, src/testdir/runtest.vim, src/testdir/test_popup.vim, src/testdir/README.txt

Patch 7.4.2322

Problem: Access memory beyond the end of the line. (Dominique Pelle)
Solution: Adjust the cursor column.
Files: src/move.c, src/testdir/test_normal.vim

Patch 7.4.2323

Problem: Using freed memory when using '**formatexpr**'. (Dominique Pelle)
Solution: Make a copy of '**formatexpr**' before evaluating it.
Files: src/ops.c, src/testdir/test_normal.vim

Patch 7.4.2324

Problem: Crash when editing a new buffer and BufUnload autocommand wipes out the new buffer. (Norio Takagi)
Solution: Don't allow wiping out this buffer. (partly by Hirohito Higashi)
Move old style test13 into test_autocmd. Avoid ml_get error when editing a file.
Files: src/structs.h, src/buffer.c, src/ex_cmds.c, src/ex_docmd.c, src/window.c, src/testdir/test13.in, src/testdir/test13.ok, src/testdir/test_autocmd.vim, src/testdir/Make_all.mak, src/Makefile

Patch 7.4.2325 (after 7.4.2324)

Problem: Tiny build fails.
Solution: Add #ifdef.
Files: src/buffer.c

Patch 7.4.2326

Problem: Illegal memory access when Visual selection starts in invalid position. (Dominique Pelle)
Solution: Correct position when needed.
Files: src/normal.c, src/misc2.c, src/proto/misc2.pro

Patch 7.4.2327

Problem: Freeing a variable that is on the stack.
Solution: Don't free res_tv or err_tv. (Ozaki Kiichi)
Files: src/channel.c

Patch 7.4.2328

Problem: Crash when BufWinLeave autocmd goes to another tab page. (Hirohito Higashi)
Solution: Make close_buffer() go back to the right window.
Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 7.4.2329

Problem: Error for min() and max() contains %s. (Nikolai Pavlov)
Solution: Pass the function name. (closes #1040)
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 7.4.2330

Problem: Coverity complains about not checking curwin to be NULL.
Solution: Use firstwin to avoid the warning.
Files: src/buffer.c

Patch 7.4.2331

Problem: Using **CTRL-X CTRL-V** to complete a command line from Insert mode does not work after entering an expression on the command line.
Solution: Don't use "ccline" when not actually using a command line. (test by Hirohito Higashi)
Files: src/edit.c, src/ex_getln.c, src/proto/ex_getln.pro, src/testdir/test_popup.vim

Patch 7.4.2332

Problem: Crash when stop_timer() is called in a callback of a callback. Vim hangs when the timer callback uses too much time.
Solution: Set tr_id to -1 when a timer is to be deleted. Don't keep calling callbacks forever. (Ozaki Kiichi)
Files: src/evalfunc.c, src/ex_cmds2.c, src/structs.h, src/proto/ex_cmds2.pro, src/testdir/test_timers.vim

Patch 7.4.2333

Problem: Outdated comments in test.
Solution: Cleanup normal mode test. (Christian Brabandt)
Files: src/testdir/test_normal.vim

Patch 7.4.2334

Problem: On MS-Windows test_getcwd leaves Xtopdir behind.
Solution: Set '**noswapfile**'. (Michael Soyka)
Files: src/testdir/test_getcwd.in

Patch 7.4.2335

Problem: taglist() is slow. (Luc Hermitte)
Solution: Check for **CTRL-C** less often when doing a linear search. (closes #1044)
Files: src/tag.c

Patch 7.4.2336

Problem: Running normal mode tests leave a couple of files behind. (Yegappan Lakshmanan)
Solution: Delete the files. (Christian Brabandt)
Files: src/testdir/test_normal.vim

Patch 7.4.2337

Problem: taglist() is still slow. (Luc Hermitte)
Solution: Check for **CTRL-C** less often when finding duplicates.
Files: src/tag.c

Patch 7.4.2338

Problem: Can't build with small features. (John Marriott)
Solution: Nearly always define FEAT_TAG_BINS.
Files: src/feature.h, src/tag.c

Patch 7.4.2339

Problem: Tab page test fails when run as fake root.

Solution: Check **'buftype'** instead of **'filetype'**. (James McCoy, closes #1042)
Files: src/testdir/test_tabpage.vim

Patch 7.4.2340

Problem: MS-Windows: Building with Ruby uses old version.
Solution: Update to 2.2.X. Use clearer name for the API version. (Ken Takata)
Files: Makefile, src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/bigvim.bat

Patch 7.4.2341

Problem: Tiny things. Test doesn't clean up properly.
Solution: Adjust comment and white space. Restore option value.
Files: src/ex_cmds.c, src/message.c, src/testdir/test_autocmd.vim

Patch 7.4.2342

Problem: Typo in MS-Windows build script.
Solution: change "w2" to "22".
Files: src/bigvim.bat

Patch 7.4.2343

Problem: Too many old style tests.
Solution: Turn several into new style tests. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak, src/testdir/test101.in, src/testdir/test101.ok, src/testdir/test18.in, src/testdir/test18.ok, src/testdir/test2.in, src/testdir/test2.ok, src/testdir/test21.in, src/testdir/test21.ok, src/testdir/test6.in, src/testdir/test6.ok, src/testdir/test_arglist.vim, src/testdir/test_charsearch.vim, src/testdir/test_fnameescape.vim, src/testdir/test_gf.vim, src/testdir/test_hlsearch.vim, src/testdir/test_smartindent.vim, src/testdir/test_tagjump.vim, src/Makefile

Patch 7.4.2344

Problem: The "Reading from channel output..." message can be unwanted. Appending to a buffer leaves an empty first line behind.
Solution: Add the "out_msg" and "err_msg" options. Writing the first line overwrites the first, empty line.
Files: src/structs.h, src/channel.c, src/testdir/test_channel.vim, runtime/doc/channel.txt

Patch 7.4.2345 (after 7.4.2340)

Problem: For MinGW RUBY_API_VER_LONG isn't set correctly. Many default version numbers are outdated.
Solution: Set RUBY_API_VER_LONG to RUBY_VER_LONG. Use latest stable releases for defaults. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 7.4.2346

Problem: Autocommand test fails when run directly, passes when run as part of test_alot.
Solution: Add command to make the cursor move. Close a tab page.
Files: src/testdir/test_autocmd.vim

Patch 7.4.2347

Problem: Crash when closing a buffer while Visual mode is active.
(Dominique Pelle)
Solution: Adjust the position before computing the number of lines.
When closing the current buffer stop Visual mode.
Files: src/buffer.c, src/normal.c, src/testdir/test_normal.vim

Patch 7.4.2348

Problem: Crash on exit when EXITFREE is defined. (Dominique Pelle)
Solution: Don't access curwin when exiting.
Files: src/buffer.c

Patch 7.4.2349

Problem: Valgrind reports using uninitialized memory. (Dominique Pelle)
Solution: Check the length before checking for a NUL.
Files: src/message.c

Patch 7.4.2350

Problem: Test 86 and 87 fail with some version of Python.
Solution: Unify "can't" and "cannot". Unify quotes.
Files: src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test87.in, src/testdir/test87.ok

Patch 7.4.2351

Problem: Netbeans test fails when run from unpacked MS-Windows sources.
Solution: Open README.txt instead of Makefile.
Files: src/testdir/test_netbeans.py, src/testdir/test_netbeans.vim

Patch 7.4.2352

Problem: Netbeans test fails in shadow directory.
Solution: Also copy README.txt to the shadow directory.
Files: src/Makefile

Patch 7.4.2353

Problem: Not enough test coverage for Normal mode commands.
Solution: Add more tests. (Christian Brabandt)
Files: src/testdir/test_normal.vim

Patch 7.4.2354

Problem: The example that explains nested backreferences does not work
properly with the new regexp engine. (Harm te Hennepe)
Solution: Also save the end position when adding a state. (closes #990)
Files: src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 7.4.2355

Problem: Regexp fails to match when using "\>)\|?". (Ramel)
Solution: When a state is already in the list, but addstate_here() is used
and the existing state comes later, add the new state anyway.
Files: src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 7.4.2356

Problem: Reading past end of line when using previous substitute pattern.
(Dominique Pelle)
Solution: Don't set "pat" only set "searchstr".

Files: src/search.c, src/testdir/test_search.vim

Patch 7.4.2357

Problem: Attempt to read history entry while not initialized.

Solution: Skip when the index is negative.

Files: src/ex_getln.c

Patch 7.4.2358

Problem: Compiler warnings with Solaris Studio when using GTK3. (Danek Duvall)

Solution: Define FUNC2GENERIC depending on the system. (Kazunobu Kuriyama)

Files: src/gui.h, src/gui_beval.c, src/gui_gtk_f.c

Patch 7.4.2359

Problem: Memory leak in timer_start().

Solution: Check the right field to be NULL.

Files: src/evalfunc.c, src/testdir/test_timers.vim

Patch 7.4.2360

Problem: Invalid memory access when formatting. (Dominique Pelle)

Solution: Make sure cursor line and column are associated.

Files: src/misc1.c

Patch 7.4.2361

Problem: Checking for last_timer_id to overflow is not reliable. (Ozaki Kiichi)

Solution: Check for the number not going up.

Files: src/ex_cmds2.c

Patch 7.4.2362

Problem: Illegal memory access with ":1@". (Dominique Pelle)

Solution: Correct cursor column after setting the line number. Also avoid calling end_visual_mode() when not in Visual mode.

Files: src/ex_docmd.c, src/buffer.c

Patch 7.4.2363

Problem: Superfluous function prototypes.

Solution: Remove them.

Files: src/regexp.c

Patch 7.4.2364

Problem: Sort test sometimes fails.

Solution: Add it to the list of flaky tests.

Files: src/testdir/runtest.vim

Patch 7.4.2365

Problem: Needless line break. Confusing directory name.

Solution: Remove line break. Prepend "../" to "tools".

Files: Makefile, src/normal.c

Patch 7.4.2366

Problem: MS-Windows gvim.exe does not have DirectX support.

Solution: Add the DIRECTX to the script.

Files: src/bigvim.bat

Patch 7.4.2367 (after 7.4.2364)
Problem: Test runner misses a comma.
Solution: Add the comma.
Files: src/testdir/runtest.vim

VERSION 8.1

[version-8.1](#) [version8.1](#) [vim-8.1](#)

This section is about improvements made between version 8.0 and 8.1.

This release has hundreds of bug fixes, there is a new feature and there are many minor improvements.

The terminal window

[new-terminal-window](#)

You can now open a window which functions as a terminal. You can use it for:

- Running a command, such as "make", while editing in other windows
- Running a shell and execute several commands
- Use the terminal debugger plugin, see [terminal-debugger](#)

All of this is especially useful when running Vim on a remote (ssh) connection, when you can't easily open more terminals.

For more information see [terminal-window](#) .

Changed

[changed-8.1](#)

Internal: A few C99 features are now allowed such as // comments and a comma after the last enum entry. See [style-compiler](#) .

Since patch 8.0.0029 removed support for older MS-Windows systems, only MS-Windows XP and later are supported.

Added

[added-8.1](#)

Various syntax, indent and other plugins were added.

Quickfix improvements (by Yegappan Lakshmanan):

Added support for modifying any quickfix/location list in the quickfix stack.

Added a unique identifier for every quickfix/location list.

Added support for associating any Vim type as a context information to a quickfix/location list.

Enhanced the getqflist(), getloclist(), setqflist() and setloclist() functions to get and set the various quickfix/location list attributes.

Added the QuickFixLine highlight group to highlight the current line

in the quickfix window.

The quickfix buffer `b:changedtick` variable is incremented for every change to the contained quickfix list.

Added a `changedtick` variable to a quickfix/location list which is incremented when the list is modified.

Added support for parsing text using `'errorformat'` without creating a new quickfix list.

Added support for the "module" item to a quickfix entry which can be used for display purposes instead of a long file name.

Added support for freeing all the lists in the quickfix/location stack.

When opening a quickfix window using the `:copen/:``cwindow` commands, the supplied split modifiers are used.

Functions:

All the `term_` functions.

```
assert_beeeps()  
assert_equalfile()  
assert_report()  
balloon_show()  
balloon_split()  
ch_canread()  
getchangelist()  
getjumplist()  
getwinpos()  
pyxeval()  
remote_startserver()  
setbufline()  
test_ignore_error()  
test_override()  
trim()  
win_screenpos()
```

Autocommands:

```
CmdlineChanged  
CmdlineEnter  
CmdlineLeave  
ColorSchemePre  
DirChanged  
ExitPre  
TerminalOpen  
TextChangedP  
TextYankPost
```

Commands:

```
:pyx  
:pythonx  
:pyxdo  
:pyxfile  
:terminal  
:tmapclear  
:tmap  
:tnoremap  
:tunmap
```

Options:

```
'balloonevalterm'  
'imstyle'  
'mzschemedll'  
'mzschemegcdll'  
'makeencoding'  
'pumwidth'  
'pythonhome'  
'pythonthreehome'  
'pyxversion'  
'termwinkey'  
'termwinscroll'  
'termwinsize'  
'viminfofile'  
'winptydll'
```

Patches

patches-8.1

Patch 8.0.0001

Problem: Intro screen still mentions version7. (Paul)
Solution: Change it to version8.
Files: src/version.c

Patch 8.0.0002

Problem: The netrw plugin does not work.
Solution: Make it accept version 8.0.
Files: runtime/autoload/netrw.vim

Patch 8.0.0003

Problem: getwinvar() returns wrong Value of boolean and number options, especially non big endian systems. (James McCoy)
Solution: Cast the pointer to long or int. (closes #1060)
Files: src/option.c, src/testdir/test_bufwintabinfo.vim

Patch 8.0.0004

Problem: A string argument for function() that is not a function name results in an error message with NULL. (Christian Brabandt)
Solution: Use the argument for the error message.
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 8.0.0005

Problem: Netbeans test fails with Python 3. (Jonathonf)
Solution: Encode the string before sending it. (closes #1070)
Files: src/testdir/test_netbeans.py

Patch 8.0.0006

Problem: ":lb" is interpreted as ":lbottom" while the documentation says it means ":lbuffer".
Solution: Adjust the order of the commands. (haya14busa, closes #1093)
Files: src/ex_cmds.h

Patch 8.0.0007

Problem: Vim 7.4 is still mentioned in a few places.
Solution: Update to Vim 8. (Uncle Bill, closes #1094)
Files: src/INSTALLpc.txt, src/vimtutor, uninstal.txt

Patch 8.0.0008

Problem: Popup complete test is disabled.
Solution: Enable the test and change the assert. (Hirohito Higashi)
Files: src/testdir/test_popup.vim

Patch 8.0.0009

Problem: Unnecessary workaround for AppVeyor.
Solution: Revert patch 7.4.990. (Christian Brabandt)
Files: appveyor.yml

Patch 8.0.0010

Problem: Crash when editing file that starts with crypt header. (igor2x)
Solution: Check for length of text. (Christian Brabandt) Add a test.
Files: src/fileio.c, src/testdir/test_crypt.vim, src/Makefile,
src/testdir/Make_all.mak

Patch 8.0.0011

Problem: On OSX Test_pipe_through_sort_all() sometimes fails.
Solution: Add the test to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0012

Problem: Typos in comments.
Solution: Change "its" to "it's". (Matthew Brener, closes #1088)
Files: src/evalfunc.c, src/main.aap, src/nbdebug.c, src/netbeans.c,
src/quickfix.c, src/workshop.c, src/wsdebug.c

Patch 8.0.0013 (after 8.0.0011)

Problem: Missing comma in list.
Solution: Add the comma.
Files: src/testdir/runtest.vim

Patch 8.0.0014

Problem: Crypt tests are old style.
Solution: Convert to new style.
Files: src/testdir/test71.in, src/testdir/test71.ok,
src/testdir/test71a.in, src/testdir/test_crypt.vim, src/Makefile,
src/testdir/Make_all.mak

Patch 8.0.0015

Problem: Can't tell which part of a channel has "buffered" status.
Solution: Add an optional argument to ch_status(). Let ch_info() also
return "buffered" for out_status and err_status.
Files: src/evalfunc.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, runtime/doc/eval.txt

Patch 8.0.0016 (after 8.0.0015)

Problem: Build fails.
Solution: Include missing change.

Files: src/eval.c

Patch 8.0.0017

Problem: Cannot get the number of the current quickfix or location list.

Solution: Use the current list if "nr" in "what" is zero. (Yegappan Lakshmanan) Remove debug command from test.

Files: src/quickfix.c, src/testdir/test_quickfix.vim,
runtime/doc/eval.txt

Patch 8.0.0018

Problem: When using ":sleep" channel input is not handled.

Solution: When there is a channel check for input also when not in raw mode. Check every 100 msec.

Files: src/channel.c, src/proto/channel.pro, src/ui.c, src/proto/ui.pro,
src/ex_docmd.c, src/os_amiga.c, src/proto/os_amiga.pro,
src/os_unix.c, src/proto/os_unix.pro, src/os_win32.c,
src/proto/os_win32.pro

Patch 8.0.0019

Problem: Test_command_count is old style.

Solution: Turn it into a new style test. (Naruhiko Nishino)
Use more assert functions.

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_autocmd.vim, src/testdir/test_command_count.in,
src/testdir/test_command_count.ok,
src/testdir/test_command_count.vim

Patch 8.0.0020

Problem: The regexp engines are not reentrant.

Solution: Add regexec_T and save/restore the state when needed.

Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test_expr.vim,
runtime/doc/eval.txt, runtime/doc/change.txt

Patch 8.0.0021

Problem: In the GUI when redrawing the cursor it may be on the second half of a double byte character.

Solution: Correct the cursor column. (Yasuhiro Matsumoto)

Files: src/screen.c

Patch 8.0.0022

Problem: If a channel in NL mode is missing the NL at the end the remaining characters are dropped.

Solution: When the channel is closed use the remaining text. (Ozaki Kiichi)

Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.0.0023

Problem: "gd" and "gD" may find a match in a comment or string.

Solution: Ignore matches in comments and strings. (Anton Lindqvist)

Files: src/normal.c, src/testdir/test_goto.vim

Patch 8.0.0024

Problem: When the netbeans channel closes, "DETACH" is put in the output part. (Ozaki Kiichi)

Solution: Write "DETACH" in the socket part.

Files: src/channel.c, src/testdir/test_netbeans.vim

Patch 8.0.0025

Problem: Inconsistent use of spaces vs tabs in gd test.

Solution: Use tabs. (Anton Lindqvist)

Files: src/testdir/test_goto.vim

Patch 8.0.0026

Problem: Error format with %W, %C and %Z does not work. (Gerd Wachsmuth)

Solution: Skip code when qf_multiignore is set. (Lcd)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0027

Problem: A channel is closed when reading on stderr or stdout fails, but there may still be something to read on another part.

Solution: Turn ch_to_be_closed into a bitfield. (Ozaki Kiichi)

Files: src/channel.c, src/eval.c, src/structs.h, src/proto/channel.pro, src/testdir/test_channel.vim

Patch 8.0.0028

Problem: Superfluous semicolons.

Solution: Remove them. (Ozaki Kiichi)

Files: src/ex_cmds2.c

Patch 8.0.0029

Problem: Code for MS-Windows is complicated because of the exceptions for old systems.

Solution: Drop support for MS-Windows older than Windows XP. (Ken Takata)

Files: runtime/doc/gui_w32.txt, runtime/doc/os_win32.txt, runtime/doc/todo.txt, src/GvimExt/Makefile, src/Make_mvc.mak, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/gui_w32.c, src/if_cscope.c, src/misc1.c, src/misc2.c, src/option.c, src/os_mswin.c, src/os_win32.c, src/os_win32.h, src/proto/os_mswin.pro, src/proto/os_win32.pro, src/version.c

Patch 8.0.0030

Problem: Mouse mode is not automatically detected for tmux.

Solution: Check for **'term'** to be "tmux". (Michael Henry)

Files: src/os_unix.c

Patch 8.0.0031

Problem: After **":bwipeout"** **'fileformat'** is not set to the right default.

Solution: Get the default from **'fileformats'**. (Mike Williams)

Files: src/option.c, src/Makefile, src/testdir/test_fileformat.vim, src/testdir/test_alot.vim

Patch 8.0.0032

Problem: Tests may change the input file when something goes wrong.

Solution: Avoid writing the input file.

Files: src/testdir/test51.in, src/testdir/test67.in, src/testdir/test97.in, src/testdir/test_tabpage.vim

Patch 8.0.0033

Problem: Cannot use overlapping positions with matchaddpos().

Solution: Check end of match. (Ozaki Kiichi) Add a test (Hirohito Higashi)
Files: src/screen.c, src/testdir/test_match.vim

Patch 8.0.0034

Problem: No completion for ":messages".
Solution: Complete "clear" argument. (Hirohito Higashi)
Files: src/ex_docmd.c, src/ex_getln.c, src/proto/ex_docmd.pro,
src/testdir/test_cmdline.vim, src/vim.h,
runtime/doc/eval.txt, runtime/doc/map.txt

Patch 8.0.0035 (after 7.4.2013)

Problem: Order of matches for 'omnifunc' is messed up. (Danny Su)
Solution: Do not set compl_curr_match when called from complete_check().
(closes #1168)
Files: src/edit.c, src/evalfunc.c, src/proto/edit.pro, src/search.c,
src/spell.c, src/tag.c, src/testdir/test76.in,
src/testdir/test76.ok, src/testdir/test_popup.vim, src/Makefile,
src/testdir/Make_all.mak

Patch 8.0.0036

Problem: Detecting that a job has finished may take a while.
Solution: Check for a finished job more often (Ozaki Kiichi)
Files: src/channel.c, src/os_unix.c, src/os_win32.c,
src/proto/os_unix.pro, src/proto/os_win32.pro,
src/testdir/test_channel.vim

Patch 8.0.0037

Problem: Get E924 when switching tabs. ()
Solution: Use win_valid_any_tab() instead of win_valid(). (Martin Vuille,
closes #1167, closes #1171)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0038

Problem: OPEN_CHR_FILES not defined for FreeBSD using Debian userland
files.
Solution: Check for __FreeBSD_kernel__. (James McCoy, closes #1166)
Files: src/vim.h

Patch 8.0.0039

Problem: When Vim 8 reads an old viminfo and exits, the next time marks are
not read from viminfo. (Ned Batchelder)
Solution: Set a mark when it wasn't set before, even when the timestamp is
zero. (closes #1170)
Files: src/mark.c, src/testdir/test_viminfo.vim

Patch 8.0.0040 (after 8.0.0033)

Problem: Whole line highlighting with matchaddpos() does not work.
Solution: Check for zero length. (Hirohito Higashi)
Files: src/screen.c, src/testdir/test_match.vim

Patch 8.0.0041

Problem: When using Insert mode completion but not actually inserting
anything an undo item is still created. (Tommy Allen)
Solution: Do not call stop_arrow() when not inserting anything.

Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0042 (after 8.0.0041)

Problem: When using Insert mode completion with **'completeopt'** containing "noinsert" change is not saved for undo. (Tommy Allen)

Solution: Call stop_arrow() before inserting for pressing Enter.

Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0043 (after 8.0.0041)

Problem: When using Insert mode completion with **'completeopt'** containing "noinsert" with **CTRL-N** the change is not saved for undo. (Tommy Allen)

Solution: Call stop_arrow() before inserting for any key.

Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0044

Problem: In diff mode the cursor may end up below the last line, resulting in an ml_get error.

Solution: Check the line to be valid.

Files: src/move.c, src/diff.c, src/proto/diff.pro, src/testdir/test_diffmode.vim

Patch 8.0.0045

Problem: Calling job_stop() right after job_start() does not work.

Solution: Block signals while fork is still busy. (Ozaki Kiichi, closes #1155)

Files: src/auto/configure, src/config.h.in, src/configure.in, src/os_unix.c, src/testdir/test_channel.vim

Patch 8.0.0046

Problem: Using NUL instead of NULL.

Solution: Change to NULL. (Dominique Pelle)

Files: src/ex_cmds.c, src/json.c

Patch 8.0.0047

Problem: Crash when using the preview window from an unnamed buffer. (lifepillar)

Solution: Do not clear the wrong buffer. (closes #1200)

Files: src/popupmnu.c

Patch 8.0.0048

Problem: On Windows job_stop() stops cmd.exe, not the processes it runs. (Linwei)

Solution: Iterate over all processes and terminate the one where the parent is the job process. (Yasuhiro Matsumoto, closes #1184)

Files: src/os_win32.c, src/structs.h

Patch 8.0.0049

Problem: When a match ends in part of concealed text highlighting, it might mess up concealing by resetting prev_syntax_id.

Solution: Do not reset prev_syntax_id and add a test to verify. (Christian Brabandt, closes #1092)

Files: src/screen.c, src/testdir/test_matchadd_conceal.vim

Patch 8.0.0050

Problem: An exiting job is detected with a large latency.
Solution: Check for pending job more often. (Ozaki Kiichi) Change the double loop in mch_inchar() into one.
Files: src/channel.c, src/os_unix.c, src/testdir/shared.vim, src/testdir/test_channel.vim

Patch 8.0.0051 (after 8.0.0048)

Problem: New code for job_stop() breaks channel test on AppVeyor.
Solution: Revert the change.
Files: src/os_win32.c, src/structs.h

Patch 8.0.0052 (after 8.0.0049)

Problem: Conceal test passes even without the bug fix.
Solution: Add a redraw command. (Christian Brabandt)
Files: src/testdir/test_matchadd_conceal.vim

Patch 8.0.0053 (after 8.0.0047)

Problem: No test for what 8.0.0047 fixes.
Solution: Add a test. (Hirohito Higashi)
Files: src/testdir/test_popup.vim

Patch 8.0.0054 (after 8.0.0051)

Problem: On Windows job_stop() stops cmd.exe, not the processes it runs. (Linwei)
Solution: Iterate over all processes and terminate the one where the parent is the job process. Now only when there is no job object. (Yasuhiro Matsumoto, closes #1203)
Files: src/os_win32.c

Patch 8.0.0055

Problem: Minor comment and style deficiencies.
Solution: Update comments and fix style.
Files: src/buffer.c, src/misc2.c, src/os_unix.c

Patch 8.0.0056

Problem: When setting **'filetype'** there is no check for a valid name.
Solution: Only allow valid characters in **'filetype'**, **'syntax'** and **'keymap'**.
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0057 (after 8.0.0056)

Problem: Tests fail without the **'keymap'** features.
Solution: Check for feature in test.
Files: src/testdir/test_options.vim

Patch 8.0.0058

Problem: Positioning of the popup menu is not good.
Solution: Position it better. (Hirohito Higashi)
Files: src/popupmnu.c

Patch 8.0.0059

Problem: Vim does not build on VMS systems.
Solution: Various changes for VMS. (Zoltan Arpadffy)
Files: src/json.c, src/macros.h, src/Make_vms.mms, src/os_unix.c,

src/os_unix.h, src/os_vms.c, src/os_vms_conf.h,
src/proto/os_vms.pro, src/testdir/Make_vms.mms

Patch 8.0.0060

Problem: When using an Ex command for **'keywordprg'** it is escaped as with a shell command. (Romain Lafourcade)

Solution: Escape for an Ex command. (closes #1175)

Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.0.0061 (after 8.0.0058)

Problem: Compiler warning for unused variable.

Solution: Add #ifdef. (John Marriott)

Files: src/popupmnu.c

Patch 8.0.0062

Problem: No digraph for HORIZONTAL ELLIPSIS.

Solution: Use ",.". (Hans Ginzel, closes #1226)

Files: src/digraph.c, runtime/doc/digraph.txt

Patch 8.0.0063

Problem: Compiler warning for comparing with unsigned. (Zoltan Arpadffy)

Solution: Change <= to ==.

Files: src/undo.c

Patch 8.0.0064 (after 8.0.0060)

Problem: Normal test fails on MS-Windows.

Solution: Don't try using an illegal file name.

Files: src/testdir/test_normal.vim

Patch 8.0.0065 (after 8.0.0056)

Problem: Compiler warning for unused function in tiny build. (Tony Mechelynck)

Solution: Add #ifdef.

Files: src/option.c

Patch 8.0.0066

Problem: when calling an operator function when **'linebreak'** is set, it is internally reset before calling the operator function.

Solution: Restore **'linebreak'** before calling op_function(). (Christian Brabandt)

Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.0.0067

Problem: VMS has a problem with infinity.

Solution: Avoid an overflow. (Zoltan Arpadffy)

Files: src/json.c, src/macros.h

Patch 8.0.0068

Problem: Checking did_throw after executing autocommands is wrong. (Daniel Hahler)

Solution: Call aborting() instead, and only when autocommands were executed.

Files: src/quickfix.c, src/if_cscope.c, src/testdir/test_quickfix.vim

Patch 8.0.0069

Problem: Compiler warning for self-comparison.
Solution: Define ONE_WINDOW and add #ifdef.
Files: src/globals.h, src/buffer.c, src/ex_docmd.c, src/move.c,
src/screen.c, src/quickfix.c, src/window.c

Patch 8.0.0070

Problem: Tests referred in Makefile that no longer exist.
Solution: Remove test71 and test74 entries. (Michael Soyka)
Files: src/testdir/Mak_ming.mak

Patch 8.0.0071

Problem: Exit value from a shell command is wrong. (Hexchain Tong)
Solution: Do not check for ended jobs while waiting for a shell command.
(ichizok, closes #1196)
Files: src/os_unix.c

Patch 8.0.0072

Problem: MS-Windows: Crash with long font name. (Henry Hu)
Solution: Fix comparing with LF_FACESIZE. (Ken Takata, closes #1243)
Files: src/os_mswin.c

Patch 8.0.0073 (after 8.0.0069)

Problem: More comparisons between firstwin and lastwin.
Solution: Use ONE_WINDOW for consistency. (Hirohito Higashi)
Files: src/buffer.c, src/ex_cmds.c, src/ex_docmd.c, src/option.c,
src/window.c

Patch 8.0.0074

Problem: Cannot make Vim fail on an internal error.
Solution: Add IEMSG() and IEMSG2(). (Dominique Pelle) Avoid reporting an
internal error without mentioning where.
Files: src/globals.h, src/blowfish.c, src/dict.c, src/edit.c, src/eval.c,
src/evalfunc.c, src/ex_eval.c, src/getchar.c, src/gui_beval.c,
src/gui_w32.c, src/hangulin.c, src/hashtab.c, src/if_cscope.c,
src/json.c, src/memfile.c, src/memline.c, src/message.c,
src/misc2.c, src/option.c, src/quickfix.c, src/regexp.c,
src/spell.c, src/undo.c, src/userfunc.c, src/vim.h, src/window.c,
src/proto/misc2.pro, src/proto/message.pro, src/Makefile

Patch 8.0.0075

Problem: Using number for exception type lacks type checking.
Solution: Use an enum.
Files: src/structs.h, src/ex_docmd.c, src/ex_eval.c,
src/proto/ex_eval.pro

Patch 8.0.0076

Problem: Channel log has double parens ()().
Solution: Remove () for write_buf_line. (Yasuhiro Matsumoto)
Files: src/channel.c

Patch 8.0.0077

Problem: The GUI code is not tested by Travis.
Solution: Install the virtual framebuffer.
Files: .travis.yml

Patch 8.0.0078

Problem: Accessing freed memory in quickfix.
Solution: Reset pointer when freeing '**errorformat**'. (Dominique Pelle)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0079

Problem: Accessing freed memory in quickfix. (Dominique Pelle)
Solution: Do not free the current list when adding to it.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0080

Problem: The OS X build fails on Travis.
Solution: Skip the virtual framebuffer on OS X.
Files: .travis.yml

Patch 8.0.0081

Problem: Inconsistent function names.
Solution: Rename do_cscope to ex_cscope. Clean up comments.
Files: src/ex_cmds.h, src/if_cscope.c, src/ex_docmd.c,
src/proto/if_cscope.pro

Patch 8.0.0082

Problem: Extension for configure should be ".ac".
Solution: Rename configure.in to configure.ac. (James McCoy, closes #1173)
Files: src/configure.in, src/configure.ac, Filelist, src/Makefile,
src/blowfish.c, src/channel.c, src/config.h.in, src/main.aap,
src/os_unix.c, src/INSTALL, src/mysign

Patch 8.0.0083

Problem: Using freed memory with win_getid(). (Dominique Pelle)
Solution: For the current tab use curwin.
Files: src/window.c, src/testdir/test_window_id.vim

Patch 8.0.0084

Problem: Using freed memory when adding to a quickfix list. (Dominique Pelle)
Solution: Clear the directory name.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0085

Problem: Using freed memory with recursive function call. (Dominique Pelle)
Solution: Make a copy of the function name.
Files: src/eval.c, src/testdir/test_nested_function.vim

Patch 8.0.0086

Problem: Cannot add a comment after ":hide". (Norio Takagi)
Solution: Make it work, add a test. (Hirohito Higashi)
Files: src/Makefile, src/ex_cmds.h, src/ex_docmd.c,
src/testdir/Make_all.mak, src/testdir/test_hide.vim

Patch 8.0.0087

Problem: When the channel callback gets job info the job may already have been deleted. (lifepillar)

Solution: Do not delete the job when the channel is still useful. (ichizok, closes #1242, closes #1245)
Files: src/channel.c, src/eval.c, src/os_unix.c, src/os_win32.c, src/structs.h, src/testdir/test_channel.vim

Patch 8.0.0088

Problem: When a test fails in Setup or Teardown the problem is not reported.
Solution: Add a try/catch. (Hirohito Higashi)
Files: src/testdir/runtest.vim

Patch 8.0.0089

Problem: Various problems with GTK 3.22.2.
Solution: Fix the problems, add #ifdefs. (Kazunobu Kuriyama)
Files: src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_x11.c

Patch 8.0.0090

Problem: Cursor moved after last character when using **'breakindent'**.
Solution: Fix the cursor positioning. Turn the breakindent test into new style. (Christian Brabandt)
Files: src/screen.c, src/testdir/Make_all.mak, src/testdir/test_breakindent.in, src/testdir/test_breakindent.ok, src/testdir/test_breakindent.vim, src/Makefile

Patch 8.0.0091

Problem: Test_help_complete sometimes fails in MS-Windows console.
Solution: Use getcompletion() instead of feedkeys() and command line completion. (Hirohito Higashi)
Files: src/testdir/test_help_tagjump.vim

Patch 8.0.0092

Problem: C indenting does not support nested namespaces that C++ 17 has.
Solution: Add check that passes double colon inside a name. (Pauli, closes #1214)
Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 8.0.0093

Problem: Not using multiprocess build feature.
Solution: Enable multiprocess build with MSVC 10. (Ken Takata)
Files: src/Make_mvc.mak

Patch 8.0.0094

Problem: When vimrun.exe is not found the error message is not properly encoded.
Solution: Use utf-16 and MessageBoxW(). (Ken Takata)
Files: src/os_win32.c

Patch 8.0.0095

Problem: Problems with GTK 3.22.2 fixed in 3.22.4.
Solution: Adjust the #ifdefs. (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 8.0.0096

Problem: When the input or output is not a tty Vim appears to hang.
Solution: Add the --ttyfail argument. Also add the "ttyin" and "ttyout"

features to be able to check in Vim script.
Files: src/globals.h, src/structs.h, src/main.c, src/evalfunc.c,
runtime/doc/starting.txt, runtime/doc/eval.txt

Patch 8.0.0097
Problem: When a channel callback consumes a lot of time Vim becomes
unresponsive. (skywind)
Solution: Bail out of checking channel readahead after 100 msec.
Files: src/os_unix.c, src/misc2.c, src/vim.h, src/os_win32.c,
src/channel.c

Patch 8.0.0098 (after 8.0.0097)
Problem: Can't build on MS-Windows.
Solution: Add missing parenthesis.
Files: src/vim.h

Patch 8.0.0099
Problem: Popup menu always appears above the cursor when it is in the lower
half of the screen. (Matt Gardner)
Solution: Compute the available space better. (Hirohito Higashi,
closes #1241)
Files: src/popupmnu.c

Patch 8.0.0100
Problem: Options that are a file name may contain non-filename characters.
Solution: Check for more invalid characters.
Files: src/option.c

Patch 8.0.0101
Problem: Some options are not strictly checked.
Solution: Add flags for stricter checks.
Files: src/option.c

Patch 8.0.0102 (after 8.0.0101)
Problem: Cannot set **'dictionary'** to a path.
Solution: Allow for slash and backslash. Add a test (partly by Daisuke
Suzuki, closes #1279, closes #1284)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0103
Problem: May not process channel readahead. (skywind)
Solution: If there is readahead don't block on input.
Files: src/channel.c, src/proto/channel.pro, src/os_unix.c,
src/os_win32.c, src/misc2.c

Patch 8.0.0104
Problem: Value of **'thesaurus'** option not checked properly.
Solution: Add P_NDNAME flag. (Daisuke Suzuki)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0105
Problem: When using ch_read() with zero timeout, can't tell the difference
between reading an empty line and nothing available.
Solution: Add ch_canread().

Files: src/evalfunc.c, src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, src/testdir/shared.vim,
runtime/doc/eval.txt, runtime/doc/channel.txt

Patch 8.0.0106 (after 8.0.0100)

Problem: Cannot use a semicolon in '**backupext**'. (Jeff)
Solution: Allow for a few more characters when "secure" isn't set.
Files: src/option.c

Patch 8.0.0107

Problem: When reading channel output in a timer, messages may go missing.
(Skywind)
Solution: Add the "drop" option. Write error messages in the channel log.
Don't have ch_canread() check for the channel being open.
Files: src/structs.h, src/channel.c, src/message.c, src/evalfunc.c,
src/proto/channel.pro, runtime/doc/channel.txt

Patch 8.0.0108 (after 8.0.0107)

Problem: The channel "drop" option is not tested.
Solution: Add a test.
Files: src/testdir/test_channel.vim

Patch 8.0.0109

Problem: Still checking if memcmp() exists while every system should have
it now.
Solution: Remove vim_memcmp(). (James McCoy, closes #1295)
Files: src/config.h.in, src/configure.ac, src/misc2.c, src/os_vms_conf.h,
src/osdef1.h.in, src/search.c, src/tag.c, src/vim.h

Patch 8.0.0110

Problem: Drop command doesn't use existing window.
Solution: Check the window width properly. (Hirohito Higashi)
Files: src/buffer.c, src/testdir/test_tabpage.vim

Patch 8.0.0111

Problem: The :history command is not tested.
Solution: Add tests. (Dominique Pelle)
Files: runtime/doc/cmdline.txt, src/testdir/test_history.vim

Patch 8.0.0112

Problem: Tests 92 and 93 are old style.
Solution: Make test92 and test93 new style. (Hirohito Higashi, closes #1289)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test92.in, src/testdir/test92.ok,
src/testdir/test93.in, src/testdir/test93.ok,
src/testdir/test_mksession.vim,
src/testdir/test_mksession_utf8.vim

Patch 8.0.0113

Problem: MS-Windows: message box to prompt for saving changes may appear on
the wrong monitor.
Solution: Adjust the CenterWindow function. (Ken Takata)
Files: src/gui_w32.c

Patch 8.0.0114

Problem: Coding style not optimal.
Solution: Add spaces. (Ken Takata)
Files: src/gui_w32.c, src/os_mswin.c

Patch 8.0.0115

Problem: When building with Cygwin libwinpthread isn't found.
Solution: Link winpthread statically. (jmmmerz, closes #1255, closes #1256)
Files: src/Make_cyg_ming.mak

Patch 8.0.0116

Problem: When reading English help and using **CTRL-]** the language from **'helplang'** is used.
Solution: Make help tag jumps keep the language. (Tatsuki, test by Hirohito Higashi, closes #1249)
Files: src/tag.c, src/testdir/test_help_tagjump.vim

Patch 8.0.0117

Problem: Parallel make fails. (J. Lewis Muir)
Solution: Make sure the objects directory exists. (closes #1259)
Files: src/Makefile

Patch 8.0.0118

Problem: "make proto" adds extra function prototype.
Solution: Add #ifdef.
Files: src/misc2.c

Patch 8.0.0119

Problem: No test for using **CTRL-R** on the command line.
Solution: Add a test. (Dominique Pelle) And some more.
Files: src/testdir/test_cmdline.vim

Patch 8.0.0120

Problem: Channel test is still flaky on OS X.
Solution: Set the drop argument to "never".
Files: src/testdir/test_channel.vim

Patch 8.0.0121

Problem: Setting **'cursorline'** changes the curswant column. (Daniel Hahler)
Solution: Add the P_RWINONLY flag. (closes #1297)
Files: src/option.c, src/testdir/test_goto.vim

Patch 8.0.0122

Problem: Channel test is still flaky on OS X.
Solution: Add a short sleep.
Files: src/testdir/test_channel.py

Patch 8.0.0123

Problem: Modern Sun compilers define "__sun" instead of "sun".
Solution: Use __sun. (closes #1296)
Files: src/mbyte.c, src/pty.c, src/os_unixx.h, src/vim.h

Patch 8.0.0124

Problem: Internal error for assert_inrange(1, 1).

Solution: Adjust number of allowed arguments. (Dominique Pelle)
Files: src/evalfunc.c, src/testdir/test_assert.vim

Patch 8.0.0125

Problem: Not enough testing for entering Ex commands.
Solution: Add test for CTRL-\ e {expr}. (Dominique Pelle)
Files: src/testdir/test_cmdline.vim

Patch 8.0.0126

Problem: Display problem with 'foldcolumn' and a wide character.
(esiegerman)
Solution: Don't use "extra" but an allocated buffer. (Christian Brabandt,
closes #1310)
Files: src/screen.c, src/testdir/Make_all.mak, src/Makefile,
src/testdir/test_display.vim

Patch 8.0.0127

Problem: Cancelling completion still inserts text when formatting is done
for 'textwidth'. (lacygoill)
Solution: Don't format when CTRL-E was typed. (Hirohito Higashi,
closes #1312)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0128 (after 8.0.0126)

Problem: Display test fails on MS-Windows.
Solution: Set 'isprint' to "@".
Files: src/testdir/test_display.vim

Patch 8.0.0129

Problem: Parallel make still doesn't work. (Lewis Muir)
Solution: Define OBJ_MAIN.
Files: src/Makefile

Patch 8.0.0130

Problem: Configure uses "ushort" while the Vim code doesn't.
Solution: Use "unsigned short" instead. (Fredrik Fornwall, closes #1314)
Files: src/configure.ac, src/auto/configure

Patch 8.0.0131

Problem: Not enough test coverage for syntax commands.
Solution: Add more tests. (Dominique Pelle)
Files: src/testdir/test_syntax.vim

Patch 8.0.0132 (after 8.0.0131)

Problem: Test fails because of using :finish.
Solution: Change to return.
Files: src/testdir/test_syntax.vim

Patch 8.0.0133

Problem: ";'(" causes ml_get errors in an empty buffer. (Dominique Pelle)
Solution: Check the cursor line earlier.
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.0.0134

Problem: Null pointer access reported by UBSan.
Solution: Check curwin->w_buffer is not NULL. (Yegappan Lakshmanan)
Files: src/ex_cmds.c

Patch 8.0.0135

Problem: An address relative to the current line, ":",+3y", does not work properly on a closed fold. (Efraim Yawitz)
Solution: Correct for including the closed fold. (Christian Brabandt)
Files: src/ex_docmd.c, src/testdir/test_fold.vim, src/testdir/Make_all.mak, src/Makefile

Patch 8.0.0136

Problem: When using indent folding and changing indent the wrong fold is opened. (Jonathan Fudger)
Solution: Open the fold under the cursor a bit later. (Christian Brabandt)
Files: src/ops.c, src/testdir/test_fold.vim

Patch 8.0.0137

Problem: When **'maxfuncdepth'** is set above 200 the nesting is limited to 200. (Brett Stahlman)
Solution: Allow for Ex command recursion depending on **'maxfuncdepth'**.
Files: src/ex_docmd.c, src/testdir/test_nested_function.vim

Patch 8.0.0138 (after 8.0.0137)

Problem: Small build fails.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.0.0139 (after 8.0.0135)

Problem: Warning for unused argument.
Solution: Add UNUSED.
Files: src/ex_docmd.c

Patch 8.0.0140

Problem: Pasting inserted text in Visual mode does not work properly. (Matthew Malcomson)
Solution: Stop Visual mode before stuffing the inserted text. (Christian Brabandt, from neovim #5709)
Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.0.0141 (after 8.0.0137)

Problem: Nested function test fails on AppVeyor.
Solution: Disable the test on Windows for now.
Files: src/testdir/test_nested_function.vim

Patch 8.0.0142

Problem: Normal colors are wrong with **'termguicolors'**.
Solution: Initialize to INVALIDCOLOR instead of zero. (Ben Jackson, closes #1344)
Files: src/syntax.c

Patch 8.0.0143

Problem: Line number of current buffer in getbufinfo() is wrong.
Solution: For the current buffer use the current line number. (Ken Takata)

Files: src/evalfunc.c

Patch 8.0.0144

Problem: When using MSVC the GvimExt directory is cleaned twice.

Solution: Remove the lines. (Ken Takata)

Files: src/Make_mvc.mak

Patch 8.0.0145

Problem: Running tests on MS-Windows is a little bit noisy.

Solution: Redirect some output to "nul". (Ken Takata)

Files: src/testdir/Make_dos.mak

Patch 8.0.0146

Problem: When using **'termguicolors'** on MS-Windows the RGB definition causes the colors to be wrong.

Solution: Undefined RGB and use our own. (Gabriel Barta)

Files: src/term.c

Patch 8.0.0147

Problem: searchpair() does not work when **'magic'** is off. (Chris Paul)

Solution: Add \m in the pattern. (Christian Brabandt, closes #1341)

Files: src/evalfunc.c, src/testdir/test_search.vim

Patch 8.0.0148

Problem: When a C preprocessor statement has two line continuations the following line does not have the right indent. (Ken Takata)

Solution: Add the indent of the previous continuation line. (Hirohito Higashi)

Files: src/misc1.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 8.0.0149

Problem: ":earlier" and ":later" do not work after startup or reading the undo file.

Solution: Use absolute time stamps instead of relative to the Vim start time. (Christian Brabandt, Pavel Juhas, closes #1300, closes #1254)

Files: src/testdir/test_undo.vim, src/undo.c

Patch 8.0.0150

Problem: When the pattern of :filter does not have a separator then completion of the command fails.

Solution: Skip over the pattern. (Ozaki Kiichi, closes #1299)

Files: src/ex_docmd.c, src/testdir/test_filter_cmd.vim

Patch 8.0.0151

Problem: To pass buffer content to system() and systemlist() one has to first create a string or list.

Solution: Allow passing a buffer number. (LemonBoy, closes #1240)

Files: runtime/doc/eval.txt, src/Makefile, src/evalfunc.c, src/testdir/Make_all.mak, src/testdir/test_system.vim

Patch 8.0.0152

Problem: Running the channel test creates channellog.

Solution: Delete the debug line.

Files: src/testdir/test_channel.vim

Patch 8.0.0153 (after 8.0.0151)

Problem: system() test fails on MS-Windows.

Solution: Deal with extra space and CR.

Files: src/testdir/test_system.vim

Patch 8.0.0154 (after 8.0.0151)

Problem: system() test fails on OS/X.

Solution: Deal with leading spaces.

Files: src/testdir/test_system.vim

Patch 8.0.0155

Problem: When sorting zero elements a NULL pointer is passed to qsort(), which ubsan warns for.

Solution: Don't call qsort() if there are no elements. (Dominique Pelle)

Files: src/syntax.c

Patch 8.0.0156

Problem: Several float functions are not covered by tests.

Solution: Add float tests. (Dominique Pelle)

Files: src/Makefile, src/testdir/test_alot.vim,
src/testdir/test_float_func.vim

Patch 8.0.0157

Problem: No command line completion for ":syntax spell" and ":syntax sync".

Solution: Implement the completion. (Dominique Pelle)

Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.0.0158 (after 8.0.0156)

Problem: On MS-Windows some float functions return a different value when passed unusual values. strtod() doesn't work for "inf" and "nan".

Solution: Accept both results. Fix str2float() for MS-Windows. Also reorder assert function arguments.

Files: src/testdir/test_float_func.vim, src/eval.c

Patch 8.0.0159

Problem: Using a NULL pointer when using feedkeys() to trigger drawing a tabline.

Solution: Skip drawing a tabline if TabPageIdxs is NULL. (Dominique Pelle)
Also fix recursing into getcmdline() from the cmd window.

Files: src/screen.c, src/ex_getln.c

Patch 8.0.0160

Problem: EMSG() is sometimes used for internal errors.

Solution: Change them to IEMSG(). (Dominique Pelle) And a few more.

Files: src/regexp_nfa.c, src/channel.c, src/eval.c

Patch 8.0.0161 (after 8.0.0159)

Problem: Build fails when using small features.

Solution: Update #ifdef for using save_ccline. (Hirohito Higashi)

Files: src/ex_getln.c

Patch 8.0.0162

Problem: Build error on Fedora 23 with small features and gnome2.
Solution: Undefine ngettext(). (Hirohito Higashi)
Files: src/gui_gtk.c, src/gui_gtk_x11.c

Patch 8.0.0163

Problem: Ruby 2.4 no longer supports rb_cFixnum.
Solution: move rb_cFixnum into an #ifdef. (Kazuki Sakamoto, closes #1365)
Files: src/if_ruby.c

Patch 8.0.0164

Problem: Outdated and misplaced comments.
Solution: Fix the comments.
Files: src/charset.c, src/getchar.c, src/list.c, src/misc2.c,
src/testdir/README.txt

Patch 8.0.0165

Problem: Ubsan warns for integer overflow.
Solution: Swap two conditions. (Dominique Pelle)
Files: src/regexp_nfa.c

Patch 8.0.0166

Problem: JSON with a duplicate key gives an internal error. (Lcd)
Solution: Give a normal error. Avoid an error when parsing JSON from a
remote client fails.
Files: src/evalfunc.c, src/json.c, src/channel.c,
src/testdir/test_json.vim

Patch 8.0.0167

Problem: str2nr() and str2float() do not always work with negative values.
Solution: Be more flexible about handling signs. (LemonBoy, closes #1332)
Add more tests.
Files: src/evalfunc.c, src/testdir/test_float_func.vim,
src/testdir/test_functions.vim, src/testdir/test_alot.vim,
src/Makefile

Patch 8.0.0168

Problem: Still some float functionality is not covered by tests.
Solution: Add more tests. (Dominique Pelle, closes #1364)
Files: src/testdir/test_float_func.vim

Patch 8.0.0169

Problem: For complicated string json_decode() may run out of stack space.
Solution: Change the recursive solution into an iterative solution.
Files: src/json.c

Patch 8.0.0170 (after 8.0.0169)

Problem: Channel test fails for using freed memory.
Solution: Fix memory use in json_decode().
Files: src/json.c

Patch 8.0.0171

Problem: JS style JSON does not support single quotes.
Solution: Allow for single quotes. (Yasuhiro Matsumoto, closes #1371)
Files: src/json.c, src/testdir/test_json.vim, src/json_test.c,

runtime/doc/eval.txt

Patch 8.0.0172 (after 8.0.0159)

Problem: The command selected in the command line window is not executed.
(Andrey Starodubtsev)

Solution: Save and restore the command line at a lower level. (closes #1370)

Files: src/ex_getln.c, src/testdir/test_history.vim

Patch 8.0.0173

Problem: When compiling with EBCDIC defined the build fails. (Yaroslav Kuzmin)

Solution: Move sortFunctions() to the right file. Avoid warning for redefining __SUSV3.

Files: src/eval.c, src/evalfunc.c, src/os_unixx.h

Patch 8.0.0174

Problem: For completion "locale -a" is executed on MS-Windows, even though it most likely won't work.

Solution: Skip executing "locale -a" on MS-Windows. (Ken Takata)

Files: src/ex_cmds2.c

Patch 8.0.0175

Problem: Setting language in gvim on MS-Windows does not work when libintl.dll is dynamically linked with msvcrt.dll.

Solution: Use putenv() from libintl as well. (Ken Takata, closes #1082)

Files: src/mbyte.c, src/misc1.c, src/os_win32.c, src/proto/os_win32.pro, src/vim.h

Patch 8.0.0176

Problem: Using :change in between :function and :endfunction fails.

Solution: Recognize :change inside a function. (ichizok, closes #1374)

Files: src/userfunc.c, src/testdir/test_viml.vim

Patch 8.0.0177

Problem: When opening a buffer on a directory and inside a try/catch then the BufEnter event is not triggered.

Solution: Return NOTDONE from readfile() for a directory and deal with the three possible return values. (Justin M. Keyes, closes #1375, closes #1353)

Files: src/buffer.c, src/ex_cmds.c, src/ex_docmd.c, src/fileio.c, src/memline.c

Patch 8.0.0178

Problem: test_command_count may fail when a previous test interferes, seen on MS-Windows.

Solution: Run it separately.

Files: src/testdir/test_alot.vim, src/testdir/Make_all.mak

Patch 8.0.0179

Problem: **'formatprg'** is a global option but the value may depend on the type of buffer. (Sung Pae)

Solution: Make **'formatprg'** global-local. (closes #1380)

Files: src/structs.h, src/option.h, src/option.c, src/normal.c, runtime/doc/options.txt, src/testdir/test_normal.vim

Patch 8.0.0180

Problem: Error E937 is used both for duplicate key in JSON and for trying to delete a buffer that is in use.
Solution: Rename the JSON error to E938. (Norio Takagi, closes #1376)
Files: src/json.c, src/testdir/test_json.vim

Patch 8.0.0181

Problem: When '**cursorbind**' and '**cursorcolumn**' are both on, the column highlight in non-current windows is wrong.
Solution: Add validate_cursor(). (Masanori Misono, closes #1372)
Files: src/move.c

Patch 8.0.0182

Problem: When '**cursorbind**' and '**cursorline**' are set, but '**cursorcolumn**' is not, then the cursor line highlighting is not updated. (Hirohito Higashi)
Solution: Call redraw_later() with NOT_VALID.
Files: src/move.c

Patch 8.0.0183

Problem: Ubsan warns for using a pointer that is not aligned.
Solution: First copy the address. (Yegappan Lakshmanan)
Files: src/channel.c

Patch 8.0.0184

Problem: When in Ex mode and an error is caught by try-catch, Vim still exits with a non-zero exit code.
Solution: Don't set ex_exitval when inside a try-catch. (partly by Christian Brabandt)
Files: src/message.c, src/testdir/test_system.vim

Patch 8.0.0185 (after 8.0.0184)

Problem: The system() test fails on MS-Windows.
Solution: Skip the test on MS-Windows.
Files: src/testdir/test_system.vim

Patch 8.0.0186

Problem: The error message from assert_notequal() is confusing.
Solution: Only mention the expected value.
Files: src/eval.c, src/testdir/test_assert.vim

Patch 8.0.0187

Problem: Building with a new Ruby version fails.
Solution: Use ruby_sysinit() instead of NtInitialize(). (Tomas Volf, closes #1382)
Files: src/if_ruby.c

Patch 8.0.0188 (after 8.0.0182)

Problem: Using NOT_VALID for redraw_later() to update the cursor line/column highlighting is not efficient.
Solution: Call validate_cursor() when '**cul**' or '**cuc**' is set.
Files: src/move.c

Patch 8.0.0189

Problem: There are no tests for the :profile command.
Solution: Add tests. (Dominique Pelle, closes #1383)
Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/test_profile.vim

Patch 8.0.0190

Problem: Detecting duplicate tags uses a slow linear search.
Solution: Use a much faster hash table solution. (James McCoy, closes #1046)
But don't add hi_keylen, it makes hash tables 50% bigger.
Files: src/tag.c

Patch 8.0.0191 (after 8.0.0187)

Problem: Some systems do not have ruby_sysinit(), causing the build to fail.
Solution: Clean up how ruby_sysinit() and NtInitialize() are used. (Taro Muraoka)
Files: src/if_ruby.c

Patch 8.0.0192 (after 8.0.0190)

Problem: Build fails with tiny features.
Solution: Change #ifdef for hash_clear(). Avoid warning for unused argument.
Files: src/hashtab.c, src/if_cscope.c

Patch 8.0.0193 (after 8.0.0188)

Problem: Accidentally removed #ifdef.
Solution: Put it back. (Masanori Misono)
Files: src/move.c

Patch 8.0.0194 (after 8.0.0189)

Problem: Profile tests fails if total and self time are equal.
Solution: Make one time optional.
Files: src/testdir/test_profile.vim

Patch 8.0.0195 (after 8.0.0190)

Problem: Jumping to a tag that is a static item in the current file fails. (Kazunobu Kuriyama)
Solution: Make sure the first byte of the tag key is not NUL. (Suggested by James McCoy, closes #1387)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.0.0196 (after 8.0.0194)

Problem: The test for :profile is slow and does not work on MS-Windows.
Solution: Use the "-es" argument. (Dominique Pelle) Swap single and double quotes for system()
Files: src/testdir/test_profile.vim

Patch 8.0.0197

Problem: On MS-Windows the system() test skips a few parts.
Solution: Swap single and double quotes for the command.
Files: src/testdir/test_system.vim

Patch 8.0.0198

Problem: Some syntax arguments take effect even after "if 0". (Taylor Venable)
Solution: Properly skip the syntax statements. Make "syn case" and "syn conceal" report the current state. Fix that "syn clear" didn't reset the conceal flag. Add tests for :syntax skipping properly.
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.0.0199

Problem: Warning for an unused parameter when the libcall feature is disabled. Warning for a function type cast when compiling with -pedantic.
Solution: Add UNUSED. Use a different type cast. (Damien Molinier)
Files: src/evalfunc.c, src/os_unix.c

Patch 8.0.0200

Problem: Some syntax arguments are not tested.
Solution: Add more syntax command tests.
Files: src/testdir/test_syntax.vim

Patch 8.0.0201

Problem: When completing a group name for a highlight or syntax command cleared groups are included.
Solution: Skip groups that have been cleared.
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.0.0202

Problem: No test for invalid syntax group name.
Solution: Add a test for group name error and warning.
Files: src/testdir/test_syntax.vim

Patch 8.0.0203

Problem: Order of complication flags is sometimes wrong.
Solution: Put interface-specific flags before ALL_CFLAGS. (idea by Yousong Zhou, closes #1100)
Files: src/Makefile

Patch 8.0.0204

Problem: Compiler warns for uninitialized variable. (Tony Mechelynck)
Solution: When skipping set "id" to -1.
Files: src/syntax.c

Patch 8.0.0205

Problem: After :undojoin some commands don't work properly, such as :redo. (Matthew Malcomson)
Solution: Don't set curbuf->b_u_curhead. (closes #1390)
Files: src/undo.c, src/testdir/test_undo.vim

Patch 8.0.0206

Problem: Test coverage for :retab insufficient.
Solution: Add test for :retab. (Dominique Pelle, closes #1391)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_retab.vim

Patch 8.0.0207

Problem: Leaking file descriptor when system() cannot find the buffer.

(Coverity)
Solution: Close the file descriptor. (Dominique Pelle, closes #1398)
Files: src/evalfunc.c

Patch 8.0.0208

Problem: Internally used commands for **CTRL-Z** and mouse click end up in history. (Matthew Malcomson)
Solution: Use do_cmdline_cmd() instead of stuffing them in the readahead buffer. (James McCoy, closes #1395)
Files: src/edit.c, src/normal.c

Patch 8.0.0209

Problem: When using :substitute with the "c" flag and '**cursorbind**' is set the cursor is not updated in other windows.
Solution: Call do_check_cursorbind(). (Masanori Misono)
Files: src/ex_cmds.c

Patch 8.0.0210

Problem: Vim does not support bracketed paste, as implemented by xterm and other terminals.
Solution: Add t_BE, t_BD, t_PS and t_PE.
Files: src/term.c, src/term.h, src/option.c, src/misc2.c, src/keymap.h, src/edit.c, src/normal.c, src/evalfunc.c, src/getchar.c, src/vim.h, src/proto/edit.pro, runtime/doc/term.txt

Patch 8.0.0211 (after 8.0.0210)

Problem: Build fails if the multibyte feature is disabled.
Solution: Change #ifdef around ins_char_bytes.
Files: src/misc1.c

Patch 8.0.0212

Problem: The buffer used to store a key name theoretically could be too small. (Coverity)
Solution: Count all possible modifier characters. Add a check for the length just in case.
Files: src/keymap.h, src/misc2.c

Patch 8.0.0213

Problem: The Netbeans "specialKeys" command does not check if the argument fits in the buffer. (Coverity)
Solution: Add a length check.
Files: src/netbeans.c

Patch 8.0.0214

Problem: Leaking memory when syntax cluster id is unknown. (Coverity)
Solution: Free the memory.
Files: src/syntax.c

Patch 8.0.0215

Problem: When a Cscope line contains **CTRL-L** a NULL pointer may be used. (Coverity)
Solution: Don't check for an emacs tag in a cscope line.
Files: src/tag.c

Patch 8.0.0216

Problem: When decoding JSON with a JS style object the JSON test may use a NULL pointer. (Coverity)
Solution: Check for a NULL pointer.
Files: src/json.c, src/json_test.c

Patch 8.0.0217 (after 8.0.0215)

Problem: Build fails without the cscope feature.
Solution: Add #ifdef.
Files: src/tag.c

Patch 8.0.0218

Problem: No command line completion for :cexpr, :cgetexpr, :caddexpr, etc.
Solution: Make completion work. (Yegappan Lakshmanan) Add a test.
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.0.0219

Problem: Ubsan reports errors for integer overflow.
Solution: Define macros for minimum and maximum values. Select an expression based on the value. (Mike Williams)
Files: src/charset.c, src/eval.c, src/evalfunc.c, src/structs.h, src/testdir/test_viml.vim

Patch 8.0.0220

Problem: Completion for :match does not show "none" and other missing highlight names.
Solution: Skip over cleared entries before checking the index to be at the end.
Files: src/syntax.c, src/testdir/test_cmdline.vim

Patch 8.0.0221

Problem: Checking if PROTO is defined inside a function has no effect.
Solution: Remove the check for PROTO. (Hirohito Higashi)
Files: src/misc1.c

Patch 8.0.0222

Problem: When a multibyte character ends in a zero byte, putting blockwise text puts it before the character instead of after it.
Solution: Use int instead of char for the character under the cursor. (Luchr, closes #1403) Add a test.
Files: src/ops.c, src/testdir/test_put.vim, src/Makefile, src/testdir/test_alot.vim

Patch 8.0.0223

Problem: Coverity gets confused by the flags passed to find_tags() and warns about uninitialized variable.
Solution: Disallow using cscope and help tags at the same time.
Files: src/tag.c

Patch 8.0.0224

Problem: When **'fileformats'** is changed in a BufReadPre auto command, it does not take effect in readfile(). (Gary Johnson)
Solution: Check the value of **'fileformats'** after executing auto commands. (Christian Brabandt)

Files: src/fileio.c, src/testdir/test_fileformat.vim

Patch 8.0.0225

Problem: When a block is visually selected and put is used on the end of the selection only one line is changed.

Solution: Check for the end properly. (Christian Brabandt, neovim issue 5781)

Files: src/ops.c, src/testdir/test_put.vim

Patch 8.0.0226

Problem: The test for patch 8.0.0224 misses the CR characters and passes even without the fix. (Christian Brabandt)

Solution: Use double quotes and \

Files: src/testdir/test_fileformat.vim

Patch 8.0.0227

Problem: Crash when **'fileformat'** is forced to "dos" and the first line in the file is empty and does not have a CR character.

Solution: Don't check for CR before the start of the buffer.

Files: src/fileio.c, src/testdir/test_fileformat.vim

Patch 8.0.0228 (after 8.0.0210)

Problem: When pasting test in an xterm on the command line it is surrounded by <PasteStart> and <PasteEnd>. (Johannes Kaltenbach)

Solution: Add missing changes.

Files: src/ex_getln.c, src/term.c

Patch 8.0.0229 (after 8.0.0179)

Problem: When freeing a buffer the local value of the **'formatprg'** option is not cleared.

Solution: Add missing change.

Files: src/buffer.c

Patch 8.0.0230 (after 8.0.0210)

Problem: When using bracketed paste line breaks are not respected.

Solution: Turn CR characters into a line break if the text is being inserted. (closes #1404)

Files: src/edit.c

Patch 8.0.0231

Problem: There are no tests for bracketed paste mode.

Solution: Add a test. Fix repeating with "normal .".

Files: src/edit.c, src/testdir/test_paste.vim, src/Makefile, src/testdir/Make_all.mak

Patch 8.0.0232

Problem: Pasting in Insert mode does not work when bracketed paste is used and **'esckey'** is off.

Solution: When **'esckey'** is off disable bracketed paste in Insert mode.

Files: src/edit.c

Patch 8.0.0233 (after 8.0.0231)

Problem: The paste test fails if the GUI is being used.

Solution: Skip the test in the GUI.

Files: src/testdir/test_paste.vim

Patch 8.0.0234 (after 8.0.0225)

Problem: When several lines are visually selected and one of them is short, using put may cause a crash. (Axel Bender)

Solution: Check for a short line. (Christian Brabandt)

Files: src/ops.c, src/testdir/test_put.vim

Patch 8.0.0235

Problem: Memory leak detected when running tests for diff mode.

Solution: Free p_extra_free.

Files: src/screen.c

Patch 8.0.0236 (after 8.0.0234)

Problem: Gcc complains that a variable may be used uninitialized. Confusion between variable and label name. (John Marriott)

Solution: Initialize it. Rename end to end_lnum.

Files: src/ops.c

Patch 8.0.0237

Problem: When setting wildoptions=tagfile the completion context is not set correctly. (desjardins)

Solution: Check for EXPAND_TAGS_LISTFILES. (Christian Brabandt, closes #1399)

Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.0.0238

Problem: When using bracketed paste autoindent causes indent to be increased.

Solution: Disable 'ai' and set 'paste' temporarily. (Ken Takata)

Files: src/edit.c, src/testdir/test_paste.vim

Patch 8.0.0239

Problem: The address sanitizer sometimes finds errors, but it needs to be run manually.

Solution: Add an environment to Travis with clang and the address sanitizer. (Christian Brabandt) Also include changes only on github.

Files: .travis.yml

Patch 8.0.0240 (after 8.0.0239)

Problem: The clang build on CI fails with one configuration.

Solution: Redo a previous patch that was accidentally reverted.

Files: .travis.yml

Patch 8.0.0241

Problem: Vim defines a mch_memmove() function but it doesn't work, thus is always unused.

Solution: Remove the mch_memmove implementation. (suggested by Dominique Pelle)

Files: src/os_unix.h, src/misc2.c, src/vim.h

Patch 8.0.0242

Problem: Completion of user defined functions is not covered by tests.

Solution: Add tests. Also test various errors of user-defined commands. (Dominique Pelle, closes #1413)

Files: src/testdir/test_usercommands.vim

Patch 8.0.0243

Problem: When making a character lower case with tolower() changes the byte count, it is not made lower case.

Solution: Add strlow_save(). (Dominique Pelle, closes #1406)

Files: src/evalfunc.c, src/misc2.c, src/proto/misc2.pro, src/testdir/test_functions.vim

Patch 8.0.0244

Problem: When the user sets t_BE empty after startup to disable bracketed paste, this has no direct effect.

Solution: When t_BE is made empty write t_BD. When t_BE is made non-empty write the new value.

Files: src/option.c

Patch 8.0.0245

Problem: The generated zh_CN.cp936.po message file is not encoded properly.

Solution: Instead of using zh_CN.po as input, use zh_CN.UTF-8.po.

Files: src/po/Makefile

Patch 8.0.0246

Problem: Compiler warnings for int to pointer conversion.

Solution: Fix macro for mch_memmove(). (John Marriott)

Files: src/vim.h

Patch 8.0.0247

Problem: Under some circumstances, one needs to type Ctrl-N or Ctrl-P twice to have a menu entry selected. (Lifepillar)

Solution: call ins_compl_free(). (Christian Brabandt, closes #1411)

Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0248

Problem: vim_strcat() cannot handle overlapping arguments.

Solution: Use mch_memmove() instead of strcpy(). (Justin M. Keyes, closes #1415)

Files: src/misc2.c

Patch 8.0.0249

Problem: When two submits happen quick after each other, the tests for the first one may error out.

Solution: Use a git depth of 10 instead of 1. (Christian Brabandt)

Files: .travis.yml

Patch 8.0.0250

Problem: When virtcol() gets a column that is not the first byte of a multibyte character the result is unpredictable. (Christian Ludwig)

Solution: Correct the column to the first byte of a multibyte character. Change the utf-8 test to new style.

Files: src/charset.c, src/testdir/test_utf8.in, src/testdir/test_utf8.ok, src/testdir/test_utf8.vim, src/Makefile, src/testdir/Make_all.mak, src/testdir/test_alot_utf8.vim

Patch 8.0.0251

Problem: It is not so easy to write a script that works with both Python 2 and Python 3, even when the Python code works with both.

Solution: Add **'pyxversion'**, :pyx, etc. (Marc Weber, Ken Takata)

Files: Filelist, runtime/doc/eval.txt, runtime/doc/if_pyth.txt, runtime/doc/index.txt, runtime/doc/options.txt, runtime/optwin.vim, runtime/doc/quickref.txt, runtime/doc/usr_41.txt, src/Makefile, src/evalfunc.c, src/ex_cmds.h, src/ex_cmds2.c, src/ex_docmd.c, src/if_python.c, src/if_python3.c, src/option.c, src/option.h, src/proto/ex_cmds2.pro, src/testdir/Make_all.mak, src/testdir/pyxfile/py2_magic.py, src/testdir/pyxfile/py2_shebang.py, src/testdir/pyxfile/py3_magic.py, src/testdir/pyxfile/py3_shebang.py, src/testdir/pyxfile/pyx.py, src/testdir/test_pyx2.vim, src/testdir/test_pyx3.vim src/userfunc.c

Patch 8.0.0252

Problem: Characters below 256 that are not one byte are not always recognized as word characters.

Solution: Make vim_iswordc() and vim_iswordp() work the same way. Add a test for this. (Ozaki Kiichi)

Files: src/Makefile, src/charset.c, src/kword_test.c, src/mbyte.c, src/proto/mbyte.pro

Patch 8.0.0253

Problem: When creating a session when **'winminheight'** is 2 or larger and loading that session gives an error.

Solution: Also set **'winminheight'** before setting **'winheight'** to 1. (Rafael Bodill, neovim #5717)

Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.0.0254

Problem: When using an assert function one can either specify a message or get a message about what failed, not both.

Solution: Concatenate the error with the message.

Files: src/eval.c, src/testdir/test_assert.vim

Patch 8.0.0255

Problem: When calling setpos() with a buffer argument it often is ignored. (Matthew Malcomson)

Solution: Make the buffer argument work for all marks local to a buffer. (neovim #5713) Add more tests.

Files: src/mark.c, src/testdir/test_marks.vim, runtime/doc/eval.txt

Patch 8.0.0256 (after 8.0.0255)

Problem: Tests fail because some changes were not included.

Solution: Add changes to evalfunc.c

Files: src/evalfunc.c

Patch 8.0.0257 (after 8.0.0252)

Problem: The keyword test file is not included in the archive.

Solution: Update the list of files.

Files: Filelist

Patch 8.0.0258 (after 8.0.0253)

Problem: mksession test leaves file behind.

Solution: Delete the file. Rename files to start with "X".

Files: src/testdir/test_mksession.vim

Patch 8.0.0259

Problem: Tab commands do not handle count correctly. (Ken Hamada)

Solution: Add ADDR_TABS_RELATIVE. (Hirohito Higashi)

Files: runtime/doc/tabpage.txt, src/ex_cmds.h, src/ex_docmd.c,
src/testdir/test_tabpage.vim

Patch 8.0.0260

Problem: Build fails with tiny features.

Solution: Move get_tabpage_arg() inside #ifdef.

Files: src/ex_docmd.c

Patch 8.0.0261

Problem: Not enough test coverage for eval functions.

Solution: Add more tests. (Dominique Pelle, closes #1420)

Files: src/testdir/test_functions.vim

Patch 8.0.0262

Problem: Farsi support is barely tested.

Solution: Add more tests for Farsi. Clean up the code.

Files: src/edit.c, src/farsi.c, src/testdir/test_farsi.vim

Patch 8.0.0263

Problem: Farsi support is not tested enough.

Solution: Add more tests for Farsi. Clean up the code.

Files: src/farsi.c, src/testdir/test_farsi.vim

Patch 8.0.0264

Problem: Memory error reported by ubsan, probably for using the string returned by execute().

Solution: NUL terminate the result of execute().

Files: src/evalfunc.c

Patch 8.0.0265

Problem: May get ml_get error when :pydo deletes lines or switches to another buffer. (Nikolai Pavlov, issue #1421)

Solution: Check the buffer and line every time.

Files: src/if_py_both.h, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/Makefile,
src/testdir/Make_all.mak

Patch 8.0.0266

Problem: Compiler warning for using uninitialized variable.

Solution: Set tab_number also when there is an error.

Files: src/ex_docmd.c

Patch 8.0.0267

Problem: A channel test sometimes fails on Mac.

Solution: Add the test to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0268

Problem: May get ml_get error when :luado deletes lines or switches to another buffer. (Nikolai Pavlov, issue #1421)

Solution: Check the buffer and line every time.

Files: src/if_lua.c, src/testdir/test_lua.vim, src/Makefile, src/testdir/Make_all.mak

Patch 8.0.0269

Problem: May get ml_get error when :perlundo deletes lines or switches to another buffer. (Nikolai Pavlov, issue #1421)

Solution: Check the buffer and line every time.

Files: src/if_perl.xs, src/testdir/test_perl.vim

Patch 8.0.0270

Problem: May get ml_get error when :rubydo deletes lines or switches to another buffer. (Nikolai Pavlov, issue #1421)

Solution: Check the buffer and line every time.

Files: src/if_ruby.c, src/testdir/test_ruby.vim

Patch 8.0.0271

Problem: May get ml_get error when :tclundo deletes lines or switches to another buffer. (Nikolai Pavlov, closes #1421)

Solution: Check the buffer and line every time.

Files: src/if_tcl.c, src/testdir/test_tcl.vim, src/Makefile, src/testdir/Make_all.mak

Patch 8.0.0272

Problem: Crash on exit is not detected when running tests.

Solution: Remove the dash before the command. (Dominique Pelle, closes #1425)

Files: src/testdir/Makefile

Patch 8.0.0273

Problem: Dead code detected by Coverity when not using gnome.

Solution: Rearrange the #ifdefs to avoid dead code.

Files: src/gui_gtk_x11.c

Patch 8.0.0274

Problem: When update_single_line() is called recursively, or another screen update happens while it is busy, errors may occur.

Solution: Check and update updating_screen. (Christian Brabandt)

Files: src/screen.c

Patch 8.0.0275

Problem: When checking for **CTRL-C** typed the GUI may detect a screen resize and redraw the screen, causing trouble.

Solution: Set updating_screen in ui_breakcheck().

Files: src/ui.c

Patch 8.0.0276

Problem: Checking for FEAT_GUI_GNOME inside GTK 3 code is unnecessary.

Solution: Remove the #ifdef. (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 8.0.0277

Problem: The GUI test may trigger fontconfig and take a long time.
Solution: Set \$XDG_CACHE_HOME. (Kazunobu Kuriyama)
Files: src/testdir/unix.vim, src/testdir/test_gui.vim

Patch 8.0.0278 (after 8.0.0277)

Problem: GUI test fails on MS-Windows.
Solution: Check that tester_HOME exists.
Files: src/testdir/test_gui.vim

Patch 8.0.0279

Problem: With MSVC 2015 the dll name is vcruntime140.dll.
Solution: Check the MSVC version and use the right dll name. (Ken Takata)
Files: src/Make_mvc.mak

Patch 8.0.0280

Problem: On MS-Windows setting an environment variable with multibyte strings does not work well.
Solution: Use wputenv when possible. (Taro Muraoka, Ken Takata)
Files: src/misc1.c, src/os_win32.c, src/os_win32.h, src/proto/os_win32.pro, src/vim.h

Patch 8.0.0281

Problem: MS-Windows files are still using ARGSUSED while most other files have UNUSED.
Solution: Change ARGSUSED to UNUSED or delete it.
Files: src/os_win32.c, src/gui_w32.c, src/os_mswin.c, src/os_w32exe.c, src/winclip.c

Patch 8.0.0282

Problem: When doing a Visual selection and using "I" to go to insert mode, **CTRL-O** needs to be used twice to go to Normal mode. (Coacher)
Solution: Check for the return value of edit(). (Christian Brabandt, closes #1290)
Files: src/normal.c, src/ops.c

Patch 8.0.0283

Problem: The return value of mode() does not indicate that completion is active in Replace and Insert mode. (Zhen-Huan (Kenny) Hu)
Solution: Add "c" or "x" for two kinds of completion. (Yegappan Lakshmanan, closes #1397) Test some more modes.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_functions.vim, src/testdir/test_mapping.vim

Patch 8.0.0284

Problem: The Test_collapse_buffers() test failed once, looks like it is flaky.
Solution: Add it to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0285 (after 8.0.0277)

Problem: Tests fail with tiny build on Unix.
Solution: Only set g:tester_HOME when build with the +eval feature.
Files: src/testdir/unix.vim

Patch 8.0.0286

Problem: When concealing is active and the screen is resized in the GUI it is not immediately redrawn.
Solution: Use update_prepare() and update_finish() from update_single_line().
Files: src/screen.c

Patch 8.0.0287

Problem: Cannot access the arguments of the current function in debug mode. (Luc Hermitte)
Solution: use get_funcval(). (LemonBoy, closes #1432, closes #1352)
Files: src/userfunc.c

Patch 8.0.0288 (after 8.0.0284)

Problem: Errors reported while running tests.
Solution: Put comma in the right place.
Files: src/testdir/runtest.vim

Patch 8.0.0289

Problem: No test for "ga" and :ascii.
Solution: Add a test. (Dominique Pelle, closes #1429)
Files: src/Makefile, src/testdir/test_alot.vim, src/testdir/test_ga.vim

Patch 8.0.0290

Problem: If a wide character doesn't fit at the end of the screen line, and the line doesn't fit on the screen, then the cursor position may be wrong. (anliting)
Solution: Don't skip over wide character. (Christian Brabandt, closes #1408)
Files: src/screen.c

Patch 8.0.0291 (after 8.0.0282)

Problem: Visual block insertion does not insert in all lines.
Solution: Don't bail out of insert too early. Add a test. (Christian Brabandt, closes #1290)
Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.0.0292

Problem: The stat test is a bit slow.
Solution: Remove a couple of sleep comments and reduce another.
Files: src/testdir/test_stat.vim

Patch 8.0.0293

Problem: Some tests have a one or three second wait.
Solution: Reset the 'showmode' option. Use a test time of one to disable sleep after an error or warning message.
Files: src/misc1.c, src/testdir/runtest.vim, src/testdir/test_normal.vim

Patch 8.0.0294

Problem: Argument list is not stored correctly in a session file. (lgpasquale)

Solution: Use "\$argadd" instead of "argadd". (closes #1434)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.0.0295 (after 8.0.0293)

Problem: test_viml hangs.
Solution: Put resetting 'more' before sourcing the script.
Files: src/testdir/runtest.vim

Patch 8.0.0296

Problem: Bracketed paste can only append, not insert.
Solution: When the cursor is in the first column insert the text.
Files: src/normal.c, src/testdir/test_paste.vim, runtime/doc/term.txt

Patch 8.0.0297

Problem: Double free on exit when using a closure. (James McCoy)
Solution: Split free_al_functions in two parts. (closes #1428)
Files: src/userfunc.c, src/structs.h

Patch 8.0.0298

Problem: Ex command range with repeated search does not work. (Bruce DeVisser)
Solution: Skip over \/, \? and \&.
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.0.0299

Problem: When the GUI window is resized Vim does not always take over the new size. (Luchr)
Solution: Reset new_p_guifont in gui_resize_shell(). Call gui_may_resize_shell() in the main loop.
Files: src/main.c, src/gui.c

Patch 8.0.0300

Problem: Cannot stop diffing hidden buffers. (Daniel Hahler)
Solution: When using :diffoff! make the whole list of diffed buffers empty. (closes #736)
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.0.0301

Problem: No tests for ":set completion" and various errors of the :set command.
Solution: Add more :set tests. (Dominique Pelle, closes #1440)
Files: src/testdir/test_options.vim

Patch 8.0.0302

Problem: Cannot set terminal key codes with :let.
Solution: Make it work.
Files: src/option.c, src/testdir/test_assign.vim

Patch 8.0.0303

Problem: Bracketed paste does not work in Visual mode.
Solution: Delete the text before pasting
Files: src/normal.c, src/ops.c, src/proto/ops.pro, src/testdir/test_paste.vim

Patch 8.0.0304 (after 8.0.0302)

Problem: Assign test fails in the GUI.
Solution: Skip the test for setting t_k1.
Files: src/testdir/test_assign.vim

Patch 8.0.0305

Problem: Invalid memory access when option has duplicate flag.
Solution: Correct pointer computation. (Dominique Pelle, closes #1442)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0306

Problem: mode() not sufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan)
Files: src/testdir/test_functions.vim

Patch 8.0.0307

Problem: Asan detects a memory error when EXITFREE is defined. (Dominique Pelle)
Solution: In getvcol() check for ml_get_buf() returning an empty string. Also skip adjusting the scroll position. Set "exiting" in mch_exit() for all systems.
Files: src/charset.c, src/window.c, src/os_mswin.c, src/os_win32.c, src/os_amiga.c

Patch 8.0.0308

Problem: When using a symbolic link, the package path will not be inserted at the right position in '**runtimepath**'. (Dugan Chen, Norio Takagi)
Solution: Resolve symbolic links when finding the right position in '**runtimepath**'. (Hirohito Higashi)
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 8.0.0309

Problem: Cannot use an empty key in json.
Solution: Allow for using an empty key.
Files: src/json.c, src/testdir/test_json.vim

Patch 8.0.0310

Problem: Not enough testing for GUI functionality.
Solution: Add tests for v:windowid and getwinpos[xy](). (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0311

Problem: Linebreak tests are old style.
Solution: Turn the tests into new style. Share utility functions. (Ozaki Kiichi, closes #1444)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_breakindent.vim, src/testdir/test_listlbr.in, src/testdir/test_listlbr.ok, src/testdir/test_listlbr.vim, src/testdir/test_listlbr_utf8.in, src/testdir/test_listlbr_utf8.ok, src/testdir/test_listlbr_utf8.vim, src/testdir/view_util.vim

Patch 8.0.0312

Problem: When a json message arrives in pieces, the start is dropped and

the decoding fails.
Solution: Do not drop the start when it is still needed. (Kay Zheng) Add a test. Reset the timeout when something is received.
Files: src/channel.c, src/testdir/test_channel.vim, src/structs.h, src/testdir/test_channel_pipe.py

Patch 8.0.0313 (after 8.0.0310)

Problem: Not enough testing for GUI functionality.
Solution: Add tests for the GUI font. (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0314

Problem: getcmdtype(), getcmdpos() and getcmdline() are not tested.
Solution: Add tests. (Yegappan Lakshmanan)
Files: src/testdir/test_cmdline.vim

Patch 8.0.0315

Problem: ":help :[range]" does not work. (Tony Mechelynck)
Solution: Translate to insert a backslash.
Files: src/ex_cmds.c

Patch 8.0.0316

Problem: ":help z?" does not work. (Pavol Juhas)
Solution: Remove exception for z?.
Files: src/ex_cmds.c

Patch 8.0.0317

Problem: No test for setting **'guifont'**.
Solution: Add a test for X11 GUIs. (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0318

Problem: Small mistake in 7x13 font name.
Solution: Use ISO 8859-1 name instead of 10646-1. (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0319

Problem: Insert mode completion does not respect "start" in **'backspace'**.
Solution: Check whether backspace can go before where insert started. (Hirohito Higashi)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0320

Problem: Warning for unused variable with small build.
Solution: Change #ifdef to exclude FEAT_CMDWIN. (Kazunobu Kuriyama)
Files: src/ex_getln.c

Patch 8.0.0321

Problem: When using the tiny version trying to load the matchit plugin gives an error. On MS-Windows some default mappings fail.
Solution: Add a check if the command used is available. (Christian Brabandt)
Files: runtime/mswin.vim, runtime/macros/matchit.vim

Patch 8.0.0322

Problem: Possible overflow with spell file where the tree length is corrupted.
Solution: Check for an invalid length (suggested by shqking)
Files: src/spellfile.c

Patch 8.0.0323

Problem: When running the command line tests there is a one second wait.
Solution: Change an Esc to Ctrl-C. (Yegappan Lakshmanan)
Files: src/testdir/test_cmdline.vim

Patch 8.0.0324

Problem: Illegal memory access with "1;y".
Solution: Call check_cursor() instead of check_cursor_lnum(). (Dominique Pelle, closes #1455)
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.0.0325

Problem: Packadd test does not clean up symlink.
Solution: Delete the link. (Hirohito Higashi)
Files: src/testdir/test_packadd.vim

Patch 8.0.0326 (after 8.0.0325)

Problem: Packadd test uses wrong directory name.
Solution: Use the variable name value. (Hirohito Higashi)
Files: src/testdir/test_packadd.vim

Patch 8.0.0327

Problem: The E11 error message in the command line window is not translated.
Solution: use _(). (Hirohito Higashi)
Files: src/ex_docmd.c

Patch 8.0.0328

Problem: The "zero count" error doesn't have a number. (Hirohito Higashi)
Solution: Give it a number and be more specific about the error.
Files: src/globals.h

Patch 8.0.0329

Problem: Xfontset and guifontwide are not tested.
Solution: Add tests. (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0330

Problem: Illegal memory access after "vapo". (Dominique Pelle)
Solution: Fix the cursor column.
Files: src/search.c, src/testdir/test_visual.vim

Patch 8.0.0331

Problem: Restoring help snapshot accesses freed memory. (Dominique Pelle)
Solution: Don't restore a snapshot when the window closes.
Files: src/window.c, src/Makefile, src/testdir/Make_all.mak, src/testdir/test_help.vim

Patch 8.0.0332

Problem: GUI test fails on some systems.
Solution: Try different language settings. (Kazunobu Kuriyama)
Files: src/testdir/test_gui.vim

Patch 8.0.0333

Problem: Illegal memory access when '**complete**' ends in a backslash.
Solution: Check for trailing backslash. (Dominique Pelle, closes #1478)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0334

Problem: Can't access b:changedtick from a dict reference.
Solution: Make changedtick a member of the b: dict. (inspired by neovim #6112)
Files: src/structs.h, src/buffer.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_docmd.c, src/main.c, src/globals.h, src/fileio.c, src/memline.c, src/misc1.c, src/syntax.c, src/proto/eval.pro, src/testdir/test_changedtick.vim, src/Makefile, src/testdir/test_alot.vim, src/testdir/test91.in, src/testdir/test91.ok, src/testdir/test_functions.vim

Patch 8.0.0335 (after 8.0.0335)

Problem: Functions test fails.
Solution: Use the right buffer number.
Files: src/testdir/test_functions.vim

Patch 8.0.0336

Problem: Flags of :substitute not sufficiently tested.
Solution: Test up to two letter flag combinations. (James McCoy, closes #1479)
Files: src/testdir/test_substitute.vim

Patch 8.0.0337

Problem: Invalid memory access in :recover command.
Solution: Avoid access before directory name. (Dominique Pelle, closes #1488)
Files: src/Makefile, src/memline.c, src/testdir/test_alot.vim, src/testdir/test_recover.vim

Patch 8.0.0338 (after 8.0.0337)

Problem: :recover test fails on MS-Windows.
Solution: Use non-existing directory on MS-Windows.
Files: src/testdir/test_recover.vim

Patch 8.0.0339

Problem: Illegal memory access with vi'
Solution: For quoted text objects bail out if the Visual area spans more than one line.
Files: src/search.c, src/testdir/test_visual.vim

Patch 8.0.0340

Problem: Not checking return value of dict_add(). (Coverity)
Solution: Handle a failure.
Files: src/buffer.c

Patch 8.0.0341

Problem: When using complete() and typing a character undo is saved after the character was inserted. (Shougo)
Solution: Save for undo before inserting the character.
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0342

Problem: Double free when compiled with EXITFREE and setting 'ttytype'.
Solution: Avoid setting P_ALLOCED on 'ttytype'. (Dominique Pelle, closes #1461)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.0.0343

Problem: b:changedtick can be unlocked, even though it has no effect. (Nikolai Pavlov)
Solution: Add a check and error E940. (closes #1496)
Files: src/eval.c, src/testdir/test_changedtick.vim, runtime/doc/eval.txt

Patch 8.0.0344

Problem: Unlet command leaks memory. (Nikolai Pavlov)
Solution: Free the memory on error. (closes #1497)
Files: src/eval.c, src/testdir/test_unlet.vim

Patch 8.0.0345

Problem: islocked('d.changedtick') does not work.
Solution: Make it work.
Files: src/buffer.c, src/eval.c, src/evalfunc.c, src/vim.h, src/testdir/test_changedtick.vim,

Patch 8.0.0346

Problem: Vim relies on limits.h to be included indirectly, but on Solaris 9 it may not be. (Ben Fritz)
Solution: Always include limits.h.
Files: src/os_unixx.h, src/vim.h

Patch 8.0.0347

Problem: When using **CTRL-X CTRL-U** inside a comment, the use of the comment leader may not work. (Klement)
Solution: Save and restore did_ai. (Christian Brabandt, closes #1494)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0348

Problem: When building with a shadow directory on macOS lacks the +clipboard feature.
Solution: Link *.m files, specifically os_macosx.m. (Kazunobu Kuriyama)
Files: src/Makefile

Patch 8.0.0349

Problem: Redrawing errors with GTK 3.
Solution: When updating, first clear all rectangles and then draw them. (Kazunobu Kuriyama, Christian Ludwig, closes #848)
Files: src/gui_gtk_x11.c

Patch 8.0.0350

Problem: Not enough test coverage for Perl.
Solution: Add more Perl tests. (Dominique Pelle, closes #1500)
Files: src/testdir/test_perl.vim

Patch 8.0.0351

Problem: No test for concatenating an empty string that results from out of bounds indexing.
Solution: Add a simple test.
Files: src/testdir/test_expr.vim

Patch 8.0.0352

Problem: The condition for when a typval needs to be cleared is too complicated.
Solution: Init the type to VAR_UNKNOWN and always clear it.
Files: src/eval.c

Patch 8.0.0353

Problem: If [RO] in the status line is translated to a longer string, it is truncated to 4 bytes.
Solution: Skip over the resulting string. (Jente Hidskes, closes #1499)
Files: src/screen.c

Patch 8.0.0354

Problem: Test to check that setting termcap key fails sometimes.
Solution: Check for "t_k1" to exist. (Christian Brabandt, closes #1459)
Files: src/testdir/test_assign.vim

Patch 8.0.0355

Problem: Using uninitialized memory when 'isfname' is empty.
Solution: Don't call getpwnam() without an argument. (Dominique Pelle, closes #1464)
Files: src/misc1.c, src/testdir/test_options.vim

Patch 8.0.0356 (after 8.0.0342)

Problem: Leaking memory when setting 'ttytype'.
Solution: Get free_oldval from the right option entry.
Files: src/option.c

Patch 8.0.0357

Problem: Crash when setting 'guicursor' to weird value.
Solution: Avoid negative size. (Dominique Pelle, closes #1465)
Files: src/misc2.c, src/testdir/test_options.vim

Patch 8.0.0358

Problem: Invalid memory access in C-indent code.
Solution: Don't go over end of empty line. (Dominique Pelle, closes #1492)
Files: src/edit.c, src/testdir/test_options.vim

Patch 8.0.0359

Problem: 'number' and 'relativenumber' are not properly tested.
Solution: Add tests, change old style to new style tests. (Ozaki Kiichi, closes #1447)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test89.in, src/testdir/test89.ok,

src/testdir/test_alot.vim, src/testdir/test_findfile.vim,
src/testdir/test_number.vim

Patch 8.0.0360

Problem: Sometimes VimL is used, which is confusing.
Solution: Consistently use "Vim script". (Hirohito Higashi)
Files: runtime/doc/if_mzsch.txt, runtime/doc/if_pyth.txt,
runtime/doc/syntax.txt, runtime/doc/usr_02.txt,
runtime/doc/version7.txt, src/Makefile, src/eval.c,
src/ex_getln.c, src/if_py_both.h, src/if_xcmdsrv.c,
src/testdir/Make_all.mak, src/testdir/runtest.vim,
src/testdir/test49.vim, src/testdir/test_vimscript.vim,
src/testdir/test_viml.vim

Patch 8.0.0361

Problem: GUI initialisation is not sufficiently tested.
Solution: Add the gui_init test. (Kazunobu Kuriyama)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Makefile,
src/testdir/gui_init.vim, src/testdir/setup_gui.vim,
src/testdir/test_gui.vim, src/testdir/test_gui_init.vim, Filelist

Patch 8.0.0362 (after 8.0.0361)

Problem: Tests fail on MS-Windows.
Solution: Use \$*.vim instead of \$<.
Files: src/testdir/Make_dos.mak

Patch 8.0.0363

Problem: Travis is too slow to keep up with patches.
Solution: Increase git depth to 20
Files: .travis.yml

Patch 8.0.0364

Problem:]s does not move cursor with two spell errors in one line. (Manuel Ortega)
Solution: Don't stop search immediately when wrapped, search the line first. (Ken Takata) Add a test.
Files: src/spell.c, src/Makefile, src/testdir/test_spell.vim,
src/testdir/Make_all.mak

Patch 8.0.0365

Problem: Might free a dict item that wasn't allocated.
Solution: Call dictitem_free(). (Nikolai Pavlov) Use this for b:changedtick.
Files: src/dict.c, src/structs.h, src/buffer.c, src/edit.c,
src/evalfunc.c, src/ex_docmd.c, src/fileio.c, src/main.c,
src/memline.c, src/misc1.c, src/syntax.c

Patch 8.0.0366 (after 8.0.0365)

Problem: Build fails with tiny features.
Solution: Add #ifdef.
Files: src/buffer.c

Patch 8.0.0367

Problem: If configure defines `_LARGE_FILES` some include files are included before it is defined.
Solution: Include `vim.h` first. (Sam Thursfield, closes #1508)
Files: `src/gui_at_sb.c`, `src/gui_athena.c`, `src/gui_motif.c`, `src/gui_x11.c`, `src/gui_xmdlg.c`

Patch 8.0.0368

Problem: Not all options are tested with a range of values.
Solution: Generate a test script from the source code.
Files: `Filelist`, `src/gen_opt_test.vim`, `src/testdir/test_options.vim`, `src/Makefile`

Patch 8.0.0369 (after 8.0.0368)

Problem: The `'balloondelay'`, `'ballooneval'` and `'balloonexpr'` options are not defined without the `+balloon_eval` feature. Testing that an option value fails does not work for unsupported options.
Solution: Make the options defined but not supported. Don't test if setting unsupported options fails.
Files: `src/option.c`, `src/gen_opt_test.vim`

Patch 8.0.0370

Problem: Invalid memory access when setting `wildchar` empty.
Solution: Avoid going over the end of the option value. (Dominique Pelle, closes #1509) Make option test check all number options with empty value.
Files: `src/gen_opt_test.vim`, `src/option.c`, `src/testdir/test_options.vim`

Patch 8.0.0371 (after 8.0.0365)

Problem: Leaking memory when setting `v:completed_item`.
Solution: Or the flags instead of setting them.
Files: `src/eval.c`

Patch 8.0.0372

Problem: More options are not always defined.
Solution: Consistently define all possible options.
Files: `src/option.c`, `src/testdir/test_expand_dllpath.vim`

Patch 8.0.0373

Problem: Build fails without `+folding`.
Solution: Move misplaced `#ifdef`.
Files: `src/option.c`

Patch 8.0.0374

Problem: Invalid memory access when using `:sc` in Ex mode. (Dominique Pelle)
Solution: Avoid the column being negative. Also fix a hang in Ex mode.
Files: `src/ex_getln.c`, `src/ex_cmds.c`, `src/testdir/test_substitute.vim`

Patch 8.0.0375

Problem: The `"+` register is not tested.
Solution: Add a test using another Vim instance to change the `"+` register. (Kazunobu Kuriyama)
Files: `src/testdir/test_gui.vim`

Patch 8.0.0376

Problem: Size computations in spell file reading are not exactly right.
Solution: Make "len" a "long" and check with LONG_MAX.
Files: src/spellfile.c

Patch 8.0.0377

Problem: Possible overflow when reading corrupted undo file.
Solution: Check if allocated size is not too big. (King)
Files: src/undo.c

Patch 8.0.0378

Problem: Another possible overflow when reading corrupted undo file.
Solution: Check if allocated size is not too big. (King)
Files: src/undo.c

Patch 8.0.0379

Problem: **CTRL-Z** and mouse click use **CTRL-O** unnecessary.
Solution: Remove stuffing **CTRL-O**. (James McCoy, closes #1453)
Files: src/edit.c, src/normal.c

Patch 8.0.0380

Problem: With '**linebreak**' set and '**breakat**' includes ">" a double-wide character results in "<<" displayed.
Solution: Check for the character not to be replaced. (Ozaki Kiichi, closes #1456)
Files: src/screen.c, src/testdir/test_listlbr_utf8.vim

Patch 8.0.0381

Problem: Diff mode is not sufficiently tested.
Solution: Add more diff mode tests. (Dominique Pelle, closes #1515)
Files: src/testdir/test_diffmode.vim

Patch 8.0.0382 (after 8.0.0380)

Problem: Warning in tiny build for unused variable. (Tony Mechelynck)
Solution: Add #ifdefs.
Files: src/screen.c

Patch 8.0.0383 (after 8.0.0382)

Problem: Misplaced #ifdef. (Christ van Willegen)
Solution: Split assignment.
Files: src/screen.c

Patch 8.0.0384

Problem: Timer test failed for no apparent reason.
Solution: Mark the test as flaky.
Files: src/testdir/runtest.vim

Patch 8.0.0385

Problem: No tests for arabic.
Solution: Add a first test for arabic. (Dominique Pelle, closes #1518)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/test_arabic.vim

Patch 8.0.0386

Problem: Tiny build has a problem with generating the options test.

Solution: Change the "if" to skip over statements.
Files: src/gen_opt_test.vim

Patch 8.0.0387

Problem: compiler warnings
Solution: Add type casts. (Christian Brabandt)
Files: src/channel.c, src/memline.c

Patch 8.0.0388

Problem: filtering lines through "cat", without changing the line count, changes manual folds.
Solution: Change how marks and folds are adjusted. (Matthew Malcomson, from neovim #6194).
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.0.0389

Problem: Test for arabic does not check what is displayed.
Solution: Improve what is asserted. (Dominique Pelle, closes #1523)
Add a first shaping test.
Files: src/testdir/test_arabic.vim

Patch 8.0.0390

Problem: When the window scrolls horizontally when the popup menu is displayed part of it may not be cleared. (Neovim issue #6184)
Solution: Remove the menu when the windows scrolled. (closes #1524)
Files: src/edit.c

Patch 8.0.0391

Problem: Arabic support is verbose and not well tested.
Solution: Simplify the code. Add more tests.
Files: src/arabic.c, src/testdir/test_arabic.vim

Patch 8.0.0392

Problem: GUI test fails with Athena and Motif.
Solution: Add test_ignore_error(). Use it to ignore the "failed to create input context" error.
Files: src/message.c, src/proto/message.pro, src/evalfunc.c, src/testdir/test_gui.vim, runtime/doc/eval.txt

Patch 8.0.0393 (after 8.0.0190)

Problem: When the same tag appears more than once, the order is unpredictable. (Charles Campbell)
Solution: Besides using a dict for finding duplicates, use a grow array for keeping the tags in sequence.
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.0.0394

Problem: Tabs are not aligned when scrolling horizontally and a Tab doesn't fit. (Axel Bender)
Solution: Handle a Tab as a not fitting character. (Christian Brabandt)
Also fix that ":redraw" does not scroll horizontally to show the cursor. And fix the test that depended on the old behavior.
Files: src/screen.c, src/ex_docmd.c, src/testdir/test_listlbr.vim, src/testdir/test_listlbr_utf8.vim,

src/testdir/test_breakindent.vim

Patch 8.0.0395 (after 8.0.0392)

Problem: Testing the + register fails with Motif.

Solution: Also ignore the "failed to create input context" error in the second gvim. Don't use msg() when it would result in a dialog.

Files: src/message.c, src/testdir/test_gui.vim, src/testdir/setup_gui.vim

Patch 8.0.0396

Problem: **'balloonexpr'** only works synchronously.

Solution: Add balloon_show(). (Jusufadis Bakamovic, closes #1449)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/os_unix.c,
src/os_win32.c

Patch 8.0.0397 (after 8.0.0392)

Problem: Cannot build with the viminfo feature but without the eval feature.

Solution: Adjust #ifdef. (John Marriott)

Files: src/message.c, src/misc2.c

Patch 8.0.0398

Problem: Illegal memory access with "t".

Solution: Use strncmp() instead of memcmp(). (Dominique Pelle, closes #1528)

Files: src/search.c, src/testdir/test_search.vim

Patch 8.0.0399

Problem: Crash when using balloon_show() when not supported. (Hirohito Higashi)

Solution: Check for balloonEval not to be NULL. (Ken Takata)

Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.0.0400

Problem: Some tests have a one second delay.

Solution: Add --not-a-term in RunVim().

Files: src/testdir/shared.vim

Patch 8.0.0401

Problem: Test fails with missing balloon feature.

Solution: Add check for balloon feature.

Files: src/testdir/test_functions.vim

Patch 8.0.0402

Problem: :map completion does not have <special>. (Dominique Pelle)

Solution: Recognize <special> in completion. Add a test.

Files: src/getchar.c, src/testdir/test_cmdline.vim

Patch 8.0.0403

Problem: GUI tests may fail.

Solution: Ignore the E285 error better. (Kazunobu Kuriyama)

Files: src/testdir/test_gui.vim, src/testdir/test_gui_init.vim

Patch 8.0.0404

Problem: Not enough testing for quickfix.

Solution: Add some more tests. (Yegappan Lakshmanan)

Files: src/testdir/test_quickfix.vim

Patch 8.0.0405

Problem: v:progpah may become invalid after ":cd".

Solution: Turn v:progpah into a full path if needed.

Files: src/main.c, src/testdir/test_startup.vim, runtime/doc/eval.txt

Patch 8.0.0406

Problem: The arabic shaping code is verbose.

Solution: Shorten the code without changing the functionality.

Files: src/arabic.c

Patch 8.0.0407 (after 8.0.0388)

Problem: Filtering folds with marker method not tested.

Solution: Also set '**foldmethod**' to "marker".

Files: src/testdir/test_fold.vim

Patch 8.0.0408

Problem: Updating folds does not work properly when inserting a file and a few other situations.

Solution: Adjust the way folds are updated. (Matthew Malcomson)

Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.0.0409

Problem: set_progpah is defined but not always used

Solution: Adjust #ifdef.

Files: src/main.c

Patch 8.0.0410

Problem: Newer gettext/iconv library has extra dll file.

Solution: Add the file to the Makefile and nsis script. (Christian Brabandt)

Files: Makefile, nsis/gvim.nsi

Patch 8.0.0411

Problem: We can't change the case in menu entries, it breaks translations.

Solution: Ignore case when looking up a menu translation.

Files: src/menu.c, src/testdir/test_menu.vim

Patch 8.0.0412 (after 8.0.0411)

Problem: Menu test fails on MS-Windows.

Solution: Use a menu entry with only ASCII characters.

Files: src/testdir/test_menu.vim

Patch 8.0.0413 (after 8.0.0412)

Problem: Menu test fails on MS-Windows using gvim.

Solution: First delete the English menus.

Files: src/testdir/test_menu.vim

Patch 8.0.0414

Problem: Balloon eval is not tested.

Solution: Add a few balloon tests. (Kazunobu Kuriyama)

Files: src/testdir/test_gui.vim

Patch 8.0.0415 (after 8.0.0414)

Problem: Balloon test fails on MS-Windows.
Solution: Test with 0x7fffffff instead of 0xffffffff.
Files: src/testdir/test_gui.vim

Patch 8.0.0416

Problem: Setting v:progbath is not quite right.
Solution: On MS-Windows add the extension. On Unix use the full path for a relative directory. (partly by James McCoy, closes #1531)
Files: src/main.c, src/os_win32.c, src/os_unix.c

Patch 8.0.0417

Problem: Test for the clipboard fails sometimes.
Solution: Add it to the flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0418

Problem: ASAN logs are disabled and don't cause a failure.
Solution: Enable ASAN logs and fail if not empty. (James McCoy, closes #1425)
Files: .travis.yml

Patch 8.0.0419

Problem: Test for v:progbath fails on MS-Windows.
Solution: Expand to full path. Also add ".exe" when the path is an absolute path.
Files: src/os_win32.c, src/main.c

Patch 8.0.0420

Problem: When running :make the output may be in the system encoding, different from **'encoding'**.
Solution: Add the **'makeencoding'** option. (Ken Takata)
Files: runtime/doc/options.txt, runtime/doc/quickfix.txt, runtime/doc/quickref.txt, src/Makefile, src/buffer.c, src/if_cscope.c, src/main.c, src/option.c, src/option.h, src/proto/quickfix.pro, src/quickfix.c, src/structs.h, src/testdir/Make_all.mak, src/testdir/test_makeencoding.py, src/testdir/test_makeencoding.vim

Patch 8.0.0421

Problem: Diff mode is displayed wrong when adding a line at the end of a buffer.
Solution: Adjust marks in diff mode. (James McCoy, closes #1329)
Files: src/misc1.c, src/ops.c, src/testdir/test_diffmode.vim

Patch 8.0.0422

Problem: Python test fails with Python 3.6.
Solution: Convert new exception messages to old ones. (closes #1359)
Files: src/testdir/test87.in

Patch 8.0.0423

Problem: The effect of adding "#" to **'cinoptions'** is not always removed. (David Briscoe)
Solution: Reset b_ind_hash_comment. (Christian Brabandt, closes #1475)
Files: src/misc1.c, src/Makefile, src/testdir/Make_all.mak,

src/testdir/test_cindent.vim, src/testdir/test3.in

Patch 8.0.0424

Problem: Compiler warnings on MS-Windows. (Ajit Thakkar)
Solution: Add type casts.
Files: src/os_win32.c

Patch 8.0.0425

Problem: Build errors when building without folding.
Solution: Add #ifdefs. (John Marriott)
Files: src/diff.c, src/edit.c, src/option.c, src/syntax.c

Patch 8.0.0426

Problem: Insufficient testing for statusline.
Solution: Add several tests. (Dominique Pelle, closes #1534)
Files: src/testdir/test_statusline.vim

Patch 8.0.0427

Problem: **'makeencoding'** missing from the options window.
Solution: Add the entry.
Files: runtime/optwin.vim

Patch 8.0.0428

Problem: Git and hg see new files after running tests. (Manuel Ortega)
Solution: Add the generated file to .hgignore (or .gitignore). Delete the resulting verbose file. (Christian Brabandt) Improve dependency on opt_test.vim. Reset the **'more'** option.
Files: .hgignore, src/gen_opt_test.vim, src/testdir/gen_opt_test.vim, src/Makefile, src/testdir/Make_all.mak, src/testdir/Makefile, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, Filelist

Patch 8.0.0429

Problem: Options test does not always test everything.
Solution: Fix dependency for opt_test.vim. Give a message when opt_test.vim was not found.
Files: src/testdir/test_options.vim, src/testdir/gen_opt_test.vim, src/testdir/Makefile, src/testdir/Make_all.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak

Patch 8.0.0430

Problem: Options test fails or hangs on MS-Windows.
Solution: Run it separately instead of part of test_alot. Use "-S" instead of "-u" to run the script. Fix failures.
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim, src/testdir/Makefile, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/gen_opt_test.vim

Patch 8.0.0431

Problem: **'cinoptions'** cannot set indent for extern block.
Solution: Add the "E" flag in **'cinoptions'**. (Hirohito Higashi)
Files: runtime/doc/indent.txt, src/misc1.c, src/structs.h, src/testdir/test_cindent.vim

Patch 8.0.0432

Problem: "make shadow" creates an invalid link.
Solution: Don't link "*.vim". (Kazunobu Kuriyama)
Files: src/Makefile

Patch 8.0.0433

Problem: Quite a few beeps when running tests.
Solution: Set **'belloff'** for these tests. (Christian Brabandt)
Files: src/testdir/test103.in, src/testdir/test14.in,
src/testdir/test29.in, src/testdir/test30.in,
src/testdir/test32.in, src/testdir/test45.in,
src/testdir/test72.in, src/testdir/test73.in,
src/testdir/test77.in, src/testdir/test78.in,
src/testdir/test85.in, src/testdir/test94.in,
src/testdir/test_alot.vim, src/testdir/test_alot_utf8.vim,
src/testdir/test_close_count.in, src/testdir/test_cmdline.vim,
src/testdir/test_diffmode.vim, src/testdir/test_digraph.vim,
src/testdir/test_erasebackward.in, src/testdir/test_normal.vim,
src/testdir/test_packadd.vim, src/testdir/test_search.vim,
src/testdir/test_textobjects.vim, src/testdir/test_undo.vim,
src/testdir/test_usercommands.vim, src/testdir/test_visual.vim

Patch 8.0.0434

Problem: Clang version not correctly detected.
Solution: Adjust the configure script. (Kazunobu Kuriyama)
Files: src/configure.ac, src/auto/configure

Patch 8.0.0435

Problem: Some functions are not tested.
Solution: Add more tests for functions. (Dominique Pelle, closes #1541)
Files: src/testdir/test_functions.vim

Patch 8.0.0436

Problem: Running the options test sometimes resizes the terminal.
Solution: Clear out t_WS.
Files: src/testdir/gen_opt_test.vim

Patch 8.0.0437

Problem: The packadd test does not create the symlink correctly and does not test the right thing.
Solution: Create the directory and symlink correctly.
Files: src/testdir/test_packadd.vim

Patch 8.0.0438

Problem: The fnamemodify test changes **'shell'** in a way later tests may not be able to use system().
Solution: Save and restore **'shell'**.
Files: src/testdir/test_fnamemodify.vim

Patch 8.0.0439

Problem: Using ":%argdel" while the argument list is already empty gives an error. (Pavol Juhas)
Solution: Don't give an error. (closes #1546)
Files: src/ex_cmds2.c, src/testdir/test_arglist.vim

Patch 8.0.0440

Problem: Not enough test coverage in Insert mode.

Solution: Add lots of tests. Add test_override(). (Christian Brabandt, closes #1521)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/edit.c, src/evalfunc.c, src/globals.h, src/screen.c, src/testdir/Make_all.mak, src/testdir/test_cursor_func.vim, src/testdir/test_edit.vim, src/testdir/test_search.vim, src/testdir/test_assert.vim, src/Makefile, src/testdir/runtest.vim

Patch 8.0.0441

Problem: Dead code in #ifdef.

Solution: Remove the #ifdef and #else part.

Files: src/option.c

Patch 8.0.0442

Problem: Patch shell command uses double quotes around the argument, which allows for \$HOME to be expanded. (Etienne)

Solution: Use single quotes on Unix. (closes #1543)

Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.0.0443

Problem: Terminal width is set to 80 in test3.

Solution: Instead of setting 'columns' set 'wrapmargin' depending on 'columns.

Files: src/testdir/test3.in

Patch 8.0.0444 (after 8.0.0442)

Problem: Diffpatch fails when the file name has a quote.

Solution: Escape the name properly. (zetzei)

Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.0.0445

Problem: Getpgid is not supported on all systems.

Solution: Add a configure check.

Files: src/configure.ac, src/auto/configure, src/config.h.in, src/os_unix.c

Patch 8.0.0446

Problem: The ";" command does not work after characters with a lower byte that is NUL.

Solution: Properly check for not having a previous character. (Hirohito Higashi)

Files: src/Makefile, src/search.c, src/testdir/test_alot_utf8.vim, src/testdir/test_charsearch_utf8.vim

Patch 8.0.0447

Problem: Getting font name does not work on X11.

Solution: Implement gui_mch_get_fontname() for X11. Add more GUI tests. (Kazunobu Kuriyama)

Files: src/gui_x11.c, src/syntax.c, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Makefile, src/testdir/gui_init.vim, src/testdir/gui_preinit.vim,

src/testdir/test_gui.vim, src/testdir/test_gui_init.vim,
Filelist

Patch 8.0.0448

Problem: Some macros are in lower case, which can be confusing.
Solution: Make a few lower case macros upper case.
Files: src/macros.h, src/buffer.c, src/charset.c, src/ops.c, src/diff.c,
src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_getln.c,
src/fileio.c, src/fold.c, src/gui.c, src/gui_beval.c, src/main.c,
src/mark.c, src/misc1.c, src/move.c, src/normal.c,
src/option.c, src/popupmnu.c, src/regexp.c, src/screen.c,
src/search.c, src/spell.c, src/tag.c, src/ui.c, src/undo.c,
src/version.c, src/workshop.c, src/if_perl.xs

Patch 8.0.0449 (after 8.0.0448)

Problem: Part of fold patch accidentally included.
Solution: Revert that part of the patch.
Files: src/ex_cmds.c

Patch 8.0.0450

Problem: v:progpath is not reliably set.
Solution: Read /proc/self/exe if possible. (idea by Michal Grochmal)
Also fixes missing #if.
Files: src/main.c, src/config.h.in

Patch 8.0.0451

Problem: Some macros are in lower case.
Solution: Make a few more macros upper case. Avoid lower case macros use an
argument twice.
Files: src/macros.h, src/charset.c, src/misc2.c, src/proto/misc2.pro,
src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/fold.c,
src/gui.c, src/gui_gtk.c, src/mark.c, src/memline.c, src/mbyte.c,
src/menu.c, src/message.c, src/misc1.c, src/ops.c, src/option.c,
src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c,
src/popupmnu.c, src/regexp.c, src/regexp_nfa.c, src/screen.c,
src/search.c, src/spell.c, src/spellfile.c, src/syntax.c,
src/tag.c, src/ui.c, src/undo.c, src/window.c

Patch 8.0.0452

Problem: Some macros are in lower case.
Solution: Make a few more macros upper case.
Files: src/vim.h, src/macros.h, src/evalfunc.c, src/fold.c,
src/gui_gtk.c, src/gui_gtk_x11.c, src/charset.c, src/diff.c,
src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/getchar.c,
src/gui.c, src/gui_w32.c, src/if_cscope.c, src/mbyte.c,
src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/normal.c,
src/ops.c, src/option.c, src/os_unix.c, src/os_win32.c,
src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/screen.c,
src/search.c, src/spell.c, src/syntax.c, src/tag.c, src/userfunc.c

Patch 8.0.0453

Problem: Adding fold marker creates new comment.

Solution: Use an existing comment if possible. (LemonBoy, closes #1549)
Files: src/ops.c, src/proto/ops.pro, src/fold.c,
src/testdir/test_fold.vim

Patch 8.0.0454

Problem: Compiler warnings for comparing unsigned char with 256 always
being true. (Manuel Ortega)
Solution: Add type cast.
Files: src/screen.c, src/charset.c

Patch 8.0.0455

Problem: The mode test may hang in Test_mode(). (Michael Soyka)
Solution: Set **'complete'** to only search the current buffer (as suggested by
Michael)
Files: src/testdir/test_functions.vim

Patch 8.0.0456

Problem: Typo in MinGW test makefile.
Solution: Change an underscore to a dot. (Michael Soyka)
Files: src/testdir/Make_ming.mak

Patch 8.0.0457

Problem: Using :move messes up manual folds.
Solution: Split adjusting marks and folds. Add foldMoveRange(). (neovim
patch #6221)
Files: src/ex_cmds.c, src/fold.c, src/mark.c, src/proto/fold.pro,
src/proto/mark.pro src/testdir/test_fold.vim

Patch 8.0.0458

Problem: Potential crash if adding list or dict to dict fails.
Solution: Make sure the reference count is correct. (Nikolai Pavlov, closes
#1555)
Files: src/dict.c

Patch 8.0.0459 (after 8.0.0457)

Problem: Old fix for :move messing up folding no longer needed, now that we
have a proper solution.
Solution: Revert patch 7.4.700. (Christian Brabandt)
Files: src/ex_cmds.c

Patch 8.0.0460 (after 8.0.0452)

Problem: Can't build on HPUX.
Solution: Fix argument names in vim_stat(). (John Marriott)
Files: src/misc2.c

Patch 8.0.0461 (after 8.0.0457)

Problem: Test 45 hangs on MS-Windows.
Solution: Reset **'shiftwidth'**. Also remove redundant function.
Files: src/fold.c, src/testdir/test45.in

Patch 8.0.0462

Problem: If an MS-Windows tests succeeds at first and then fails in a way
it does not produce a test.out file it looks like the test
succeeded.

Solution: Delete the previous output file.
Files: src/testdir/Make_dos.mak

Patch 8.0.0463

Problem: Resetting **'compatible'** in defaults.vim has unexpected side effects. (David Fishburn)

Solution: Only reset **'compatible'** if it was set.
Files: runtime/defaults.vim

Patch 8.0.0464

Problem: Can't find executable name on Solaris and FreeBSD.

Solution: Check for `"/proc/self/path/a.out"`. (Danek Duvall) And for `"/proc/curproc/file"`.

Files: src/config.h.in, src/configure.ac, src/main.c,
src/auto/configure

Patch 8.0.0465

Problem: Off-by-one error in using `:move` with folding.

Solution: Correct off-by-one mistakes and add more tests. (Matthew Malcomson)

Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.0.0466

Problem: There are still a few macros that should be all-caps.

Solution: Make a few more macros all-caps.

Files: src/buffer.c, src/edit.c, src/ex_cmds.c, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/farsi.c, src/fileio.c,
src/getchar.c, src/gui_beval.c, src/hardcopy.c, src/if_cscope.c,
src/if_xcmdsrv.c, src/mark.c, src/memline.c, src/menu.c,
src/message.c, src/misc1.c, src/normal.c, src/ops.c, src/option.c,
src/quickfix.c, src/screen.c, src/search.c, src/syntax.c,
src/tag.c, src/term.c, src/term.h, src/ui.c, src/undo.c,
src/userfunc.c, src/version.c, src/vim.h

Patch 8.0.0467

Problem: Using `g<` after `:for` does not show the right output. (Marcin Szamotulski)

Solution: Call `msg_sb_eol()` in `:echomsg`.

Files: src/eval.c

Patch 8.0.0468

Problem: After aborting an Ex command `g<` does not work. (Marcin Szamotulski)

Solution: Postpone clearing scrollback messages to until the command line has been entered. Also fix that the screen isn't redrawn if after `g<` the command line is cancelled.

Files: src/message.c, src/proto/message.pro, src/ex_getln.c, src/misc2.c,
src/gui.c

Patch 8.0.0469

Problem: Compiler warnings on MS-Windows.

Solution: Add type casts. (Christian Brabandt)

Files: src/fold.c

Patch 8.0.0470

Problem: Not enough testing for help commands.
Solution: Add a few more help tests. (Dominique Pelle, closes #1565)
Files: src/testdir/test_help.vim, src/testdir/test_help_tagjump.vim

Patch 8.0.0471

Problem: Exit callback test sometimes fails.
Solution: Add it to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0472

Problem: When a test fails and test.log is created, Test_edit_CTRL_I matches it instead of test1.in.
Solution: Match with runtest.vim instead.
Files: src/testdir/test_edit.vim

Patch 8.0.0473

Problem: No test covering arg_all().
Solution: Add a test expanding ##.
Files: src/testdir/test_arglist.vim

Patch 8.0.0474

Problem: The client-server feature is not tested.
Solution: Add a test.
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/shared.vim, src/testdir/test_clientserver.vim, src/os_mswin.c

Patch 8.0.0475

Problem: Not enough testing for the client-server feature.
Solution: Add more tests. Add the remote_startserver() function. Fix that a locally evaluated expression uses function-local variables.
Files: src/if_xcmdsrv.c, src/evalfunc.c, src/os_mswin.c, src/proto/main.pro, src/testdir/test_clientserver.vim, runtime/doc/eval.txt

Patch 8.0.0476 (after 8.0.0475)

Problem: Missing change to main.c.
Solution: Add new function.
Files: src/main.c

Patch 8.0.0477

Problem: The client-server test may hang when failing.
Solution: Set a timer. Add assert_report()
Files: src/testdir/test_clientserver.vim, src/testdir/runtest.vim, src/eval.c, src/evalfunc.c, src/proto/eval.pro, src/if_xcmdsrv.c, src/os_mswin.c, runtime/doc/eval.txt

Patch 8.0.0478

Problem: Tests use assert_true(0) and assert_false(1) to report errors.
Solution: Use assert_report().
Files: src/testdir/test_cscope.vim, src/testdir/test_expr.vim, src/testdir/test_perl.vim, src/testdir/test_channel.vim, src/testdir/test_cursor_func.vim, src/testdir/test_gui.vim, src/testdir/test_menu.vim, src/testdir/test_popup.vim,

src/testdir/test_viminfo.vim, src/testdir/test_vimscript.vim,
src/testdir/test_assert.vim

Patch 8.0.0479

Problem: remote_peek() is not tested.
Solution: Add a test.
Files: src/testdir/test_clientserver.vim, src/testdir/runtest.vim

Patch 8.0.0480

Problem: The remote_peek() test fails on MS-Windows.
Solution: Check for pending messages. Also report errors in the first run if a flaky test fails twice.
Files: src/os_mswin.c, src/testdir/runtest.vim

Patch 8.0.0481

Problem: Unnecessary if statement.
Solution: Remove the statement. Fix "it's" vs "its" mistakes. (Dominique Pelle, closes #1568)
Files: src/syntax.c

Patch 8.0.0482

Problem: The setbufvar() function may mess up the window layout. (Kay Z.)
Solution: Do not check the window to be valid if it is NULL.
Files: src/window.c, src/testdir/test_functions.vim

Patch 8.0.0483

Problem: Illegal memory access when using :all. (Dominique Pelle)
Solution: Adjust the cursor position right after setting "curwin".
Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.0.0484

Problem: Using :lhelpgrep with an argument that should fail does not produce an error if the previous :helpgrep worked.
Solution: Use another way to detect that autocommands made the quickfix info invalid. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0485

Problem: Not all windows commands are tested.
Solution: Add more tests for windows commands. (Dominique Pelle, closes #1575) Run test_autocmd separately, it interferes with other tests. Fix tests that depended on side effects.
Files: src/testdir/test_window_cmd.vim, src/testdir/test_alot.vim, src/testdir/test_autocmd.vim, src/testdir/test_fnamemodify.vim, src/testdir/test_functions.vim, src/testdir/test_delete.vim, src/testdir/Make_all.mak

Patch 8.0.0486

Problem: Crash and endless loop when closing windows in a SessionLoadPost autocommand.
Solution: Check for valid tabpage. (partly neovim #6308)
Files: src/testdir/test_autocmd.vim, src/fileio.c, src/proto/window.pro, src/window.c

Patch 8.0.0487

Problem: The autocmd test hangs on MS-Windows.
Solution: Skip the hanging tests for now.
Files: src/testdir/test_autocmd.vim

Patch 8.0.0488

Problem: Running tests leaves an "xxx" file behind.
Solution: Delete the **'verbosefile'** after resetting the option.
Files: src/testdir/gen_opt_test.vim

Patch 8.0.0489

Problem: Clipboard and "*" register is not tested.
Solution: Add a test for Mac and X11. (Kazunobu Kuriyama)
Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/test_quotestar.vim, src/testdir/runtest.vim

Patch 8.0.0490

Problem: Splitting a **'winfixwidth'** window vertically makes it one column smaller. (Dominique Pelle)
Solution: Add one to the width for the separator.
Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.0.0491

Problem: The quotestar test fails when a required feature is missing.
Solution: Prepend "Skipped" to the thrown exception.
Files: src/testdir/test_quotestar.vim

Patch 8.0.0492

Problem: A failing client-server request can make Vim hang.
Solution: Add a timeout argument to functions that wait.
Files: src/evalfunc.c, src/if_xcmdsrv.c, src/proto/if_xcmdsrv.pro,
src/main.c, src/os_mswin.c, src/proto/os_mswin.pro,
src/vim.h, runtime/doc/eval.txt, src/testdir/test_clientserver.vim

Patch 8.0.0493

Problem: Crash with cd command with very long argument.
Solution: Check for running out of space. (Dominique Pelle, closes #1576)
Files: src/testdir/test_alot.vim, src/testdir/test_cd.vim, src/Makefile,
src/misc2.c

Patch 8.0.0494

Problem: Build failure with older compiler on MS-Windows.
Solution: Move declaration to start of block.
Files: src/evalfunc.c, src/main.c, src/os_mswin.c

Patch 8.0.0495

Problem: The quotestar test uses a timer instead of a timeout, thus it cannot be rerun like a flaky test.
Solution: Remove the timer and add a timeout. (Kazunobu Kuriyama)
Files: src/testdir/test_quotestar.vim

Patch 8.0.0496

Problem: Insufficient testing for folding.
Solution: Add a couple more fold tests. (Dominique Pelle, closes #1579)

Files: src/testdir/test_fold.vim

Patch 8.0.0497

Problem: Arabic support is not fully tested.

Solution: Add more tests for the untested functions. Comment out unreachable code.

Files: src/arabic.c, src/testdir/test_arabic.vim

Patch 8.0.0498

Problem: Two autocmd tests are skipped on MS-Windows.

Solution: Make the test pass on MS-Windows. Write the messages in a file instead of getting the output of system().

Files: src/testdir/test_autocmd.vim

Patch 8.0.0499

Problem: taglist() does not prioritize tags for a buffer.

Solution: Add an optional buffer argument. (Duncan McDougall, closes #1194)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/tag.pro, src/Makefile, src/tag.c, src/testdir/test_alot.vim, src/testdir/test_taglist.vim

Patch 8.0.0500

Problem: Quotestar test is still a bit flaky.

Solution: Add a slower check for v:version.

Files: src/testdir/test_quotestar.vim

Patch 8.0.0501

Problem: On MS-Windows "!!start" does not work as expected.

Solution: When creating a process fails try passing the argument to ShellExecute(). (Katsuya Hino, closes #1570)

Files: runtime/doc/os_win32.txt, src/os_win32.c

Patch 8.0.0502

Problem: Coverity complains about possible NULL pointer.

Solution: Add an assert(), let's see if this works on all systems.

Files: src/window.c

Patch 8.0.0503

Problem: Endless loop in updating folds with 32 bit ints.

Solution: Subtract from LHS instead of add to the RHS. (Matthew Malcomson)

Files: src/fold.c

Patch 8.0.0504

Problem: Looking up an Ex command is a bit slow.

Solution: Instead of just using the first letter, also use the second letter to skip ahead in the list of commands. Generate the table with a Perl script. (Dominique Pelle, closes #1589)

Files: src/Makefile, src/create_cmdidxs.pl, src/ex_docmd.c, Filelist

Patch 8.0.0505

Problem: Failed window split for :stag not handled. (Coverity CID 99204)

Solution: If the split fails skip to the end. (bstaletic, closes #1577)

Files: src/tag.c

Patch 8.0.0506 (after 8.0.0504)

Problem: Can't build with ANSI C.
Solution: Move declarations to start of block.
Files: src/ex_docmd.c

Patch 8.0.0507

Problem: Client-server tests fail when \$DISPLAY is not set.
Solution: Check for E240 before running the test.
Files: src/testdir/test_quotestar.vim, src/testdir/test_clientserver.vim

Patch 8.0.0508

Problem: Coveralls no longer shows per-file coverage.
Solution: Add coverage from codecov.io. (Christian Brabandt)
Files: .travis.yml

Patch 8.0.0509

Problem: No link to codecov.io results.
Solution: Add a badge to the readme file.
Files: README.md

Patch 8.0.0510 (after 8.0.0509)

Problem: Typo in link to codecov.io results.
Solution: Remove duplicate https:.
Files: README.md

Patch 8.0.0511

Problem: Message for skipping client-server tests is unclear.
Solution: Be more specific about what's missing (Hirohito Higashi, Kazunobu Kuriyama)
Files: src/testdir/test_quotestar.vim, src/testdir/test_clientserver.vim

Patch 8.0.0512

Problem: Check for available characters takes too long.
Solution: Only check did_start_blocking if wtime is negative. (Daisuke Suzuki, closes #1591)
Files: src/os_unix.c

Patch 8.0.0513 (after 8.0.0201)

Problem: Getting name of cleared highlight group is wrong. (Matt Wozniski)
Solution: Only skip over cleared names for completion. (closes #1592)
Also fix that a cleared group causes duplicate completions.
Files: src/syntax.c, src/proto/syntax.pro, src/evalfunc.c,
src/ex_cmds.c, src/testdir/test_syntax.vim,
src/testdir/test_cmdline.vim

Patch 8.0.0514

Problem: Script for creating cmdidxs can be improved.
Solution: Count skipped lines instead of collecting the lines. Add "const".
(Dominique Pelle, closes #1594)
Files: src/create_cmdidxs.pl, src/ex_docmd.c

Patch 8.0.0515

Problem: ml_get errors in silent Ex mode. (Dominique Pelle)
Solution: Clear valid flags when setting the cursor. Set the topline when

not in full screen mode.
Files: src/ex_docmd.c, src/move.c, src/testdir/test_startup.vim

Patch 8.0.0516
Problem: A large count on a normal command causes trouble. (Dominique Pelle)
Solution: Make "opcount" long.
Files: src/globals.h, src/testdir/test_normal.vim

Patch 8.0.0517
Problem: There is no way to remove quickfix lists (for testing).
Solution: Add the 'f' action to setqflist(). Add tests. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0518
Problem: Storing a zero byte from a multibyte character causes fold text to show up wrong.
Solution: Avoid putting zero in ScreenLines. (Christian Brabandt, closes #1567)
Files: src/screen.c, src/testdir/test_display.vim

Patch 8.0.0519
Problem: Character classes are not well tested. They can differ between platforms.
Solution: Add tests. In the documentation make clear which classes depend on what library function. Only use :cntrl: and :graph: for ASCII. (Kazunobu Kuriyama, Dominique Pelle, closes #1560)
Update the documentation.
Files: src/regexp.c, src/regexp_nfa.c, runtime/doc/pattern.txt, src/testdir/test_regexp_utf8.vim

Patch 8.0.0520
Problem: Using a function pointer instead of the actual function, which we know.
Solution: Change mb_ functions to utf_ functions when already checked for Unicode. (Dominique Pelle, closes #1582)
Files: src/message.c, src/misc2.c, src/regexp.c, src/regexp_nfa.c, src/screen.c, src/spell.c

Patch 8.0.0521
Problem: GtkForm handling is outdated.
Solution: Get rid of event filter functions. Get rid of GtkForm.width and .height. Eliminate gtk_widget_size_request() calls. (Kazunobu Kuriyama)
Files: src/gui_gtk_f.c, src/gui_gtk_f.h

Patch 8.0.0522
Problem: MS-Windows: when 'clipboard' is "unnamed" yyp does not work in a :global command.
Solution: When setting the clipboard was postponed, do not clear the register.
Files: src/ops.c, src/proto/ui.pro, src/ui.c, src/globals.h,

src/testdir/test_global.vim, src/Makefile,
src/testdir/test_alot.vim

Patch 8.0.0523

Problem: dv} deletes part of a multibyte character. (Urtica Dioica)
Solution: Include the whole character.
Files: src/search.c, src/testdir/test_normal.vim

Patch 8.0.0524 (after 8.0.0518)

Problem: Folds are messed up when 'encoding' is "utf-8".
Solution: Also set the fold character when it's not multibyte.
Files: src/screen.c, src/testdir/test_display.vim

Patch 8.0.0525

Solution: Completion for user command argument not tested.
Problem: Add a test.
Files: src/testdir/test_cmdline.vim

Patch 8.0.0526

Problem: Coverity complains about possible negative value.
Solution: Check return value of ftell() not to be negative.
Files: src/os_unix.c

Patch 8.0.0527

Problem: RISC OS support was removed long ago, but one file is still
included.
Solution: Delete the file. (Thomas Dzedzic, closes #1603)
Files: Filelist, src/swis.s

Patch 8.0.0528

Problem: When 'wildmenu' is set and 'wildmode' has "longest" then the first
file name is highlighted, even though the text shows the longest
match.
Solution: Do not highlight the first match. (LemonBoy, closes #1602)
Files: src/ex_getln.c

Patch 8.0.0529

Problem: Line in test commented out.
Solution: Uncomment the lines for character classes that were failing before
8.0.0519. (Dominique Pelle, closes #1599)
Files: src/testdir/test_regexp_utf8.vim

Patch 8.0.0530

Problem: Buffer overflow when 'columns' is very big. (Nikolai Pavlov)
Solution: Correctly compute where to truncate. Fix translation.
(closes #1600)
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.0.0531 (after 8.0.0530)

Problem: Test with long directory name fails on non-unix systems.
Solution: Skip the test on non-unix systems.
Files: src/testdir/test_edit.vim

Patch 8.0.0532 (after 8.0.0531)

Problem: Test with long directory name fails on Mac.
Solution: Skip the test on Mac systems.
Files: src/testdir/test_edit.vim

Patch 8.0.0533

Problem: Abbreviation doesn't work after backspacing newline. (Hkonrk)
Solution: Set the insert start column. (closes #1609)
Files: src/testdir/test_mapping.vim, src/edit.c

Patch 8.0.0534

Problem: Defaults.vim does not work well with tiny features. (crd477)
Solution: When the +eval feature is not available always reset **'compatible'**.
Files: runtime/defaults.vim

Patch 8.0.0535

Problem: Memory leak when exiting from within a user function.
Solution: Clear the function call stack on exit.
Files: src/userfunc.c

Patch 8.0.0536

Problem: Quickfix window not updated when freeing quickfix stack.
Solution: Update the quickfix window. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0537

Problem: Illegal memory access with :z and large count.
Solution: Check for number overflow, using long instead of int. (Dominique Pelle, closes #1612)
Files: src/Makefile, src/ex_cmds.c, src/testdir/test_alot.vim, src/testdir/test_ex_z.vim

Patch 8.0.0538

Problem: No test for falling back to default term value.
Solution: Add a test.
Files: src/testdir/test_startup.vim

Patch 8.0.0539 (after 8.0.0538)

Problem: Startup test fails on Mac.
Solution: Use another term name, "unknown" is known. Avoid a 2 second delay.
Files: src/testdir/test_startup.vim, src/main.c, src/proto/main.pro, src/term.c

Patch 8.0.0540 (after 8.0.0540)

Problem: Building unit tests fails.
Solution: Move params outside of #ifdef.
Files: src/main.c, src/message_test.c

Patch 8.0.0541

Problem: Compiler warning on MS-Windows.
Solution: Add a type cast. (Mike Williams)
Files: src/edit.c

Patch 8.0.0542

Problem: getpos() can return a negative line number. (haya14busa)

Solution: Handle a zero topline and botline. (closes #1613)
Files: src/eval.c, runtime/doc/eval.txt

Patch 8.0.0543

Problem: Test_edit causes older xfce4-terminal to close. (Dominique Pelle)
Solution: Reduce number of columns to 2000. Try to restore the window position.
Files: src/testdir/test_edit.vim, src/evalfunc.c, src/term.c, src/proto/term.pro, src/term.h

Patch 8.0.0544

Problem: Cppcheck warnings.
Solution: Use temp variable. Change NUL to NULL. Swap conditions. (Dominique Pelle)
Files: src/channel.c, src/edit.c, src/farsi.c

Patch 8.0.0545

Problem: Edit test may fail on some systems.
Solution: If creating a directory with a very long path fails, bail out.
Files: src/testdir/test_edit.vim

Patch 8.0.0546

Problem: Swap file exists briefly when opening the command window.
Solution: Set the noswapfile command modifier before splitting the window. (James McCoy, closes #1620)
Files: src/ex_getln.c, src/option.c

Patch 8.0.0547

Problem: Extra line break in verbosefile when using ":echomsg". (Ingo Karkat)
Solution: Don't call msg_start(). (closes #1618)
Files: src/eval.c, src/testdir/test_cmdline.vim

Patch 8.0.0548

Problem: Saving the redo buffer only works one time, resulting in the "." command not working well for a function call inside another function call. (Ingo Karkat)
Solution: Save the redo buffer at every user function call. (closes #1619)
Files: src/getchar.c, src/proto/getchar.pro, src/structs.h, src/fileio.c, src/userfunc.c, src/testdir/test_functions.vim

Patch 8.0.0549

Problem: No test for the 8g8 command.
Solution: Add a test. (Dominique Pelle, closes #1615)
Files: src/testdir/test_normal.vim

Patch 8.0.0550

Problem: Some etags format tags file use 0x01, breaking the parsing.
Solution: Use 0x02 for TAG_SEP. (James McCoy, closes #1614)
Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.0.0551

Problem: The typeahead buffer is reallocated too often.
Solution: Re-use the existing buffer if possible.

Files: src/getchar.c

Patch 8.0.0552

Problem: Toupper and tolower don't work properly for Turkish when 'casemap' is empty. (Bjorn Linse)

Solution: Check the 'casemap' options when deciding how to upper/lower case.

Files: src/charset.c, src/testdir/test_normal.vim

Patch 8.0.0553 (after 8.0.0552)

Problem: Toupper/tolower test with Turkish locale fails on Mac.

Solution: Skip the test on Mac.

Files: src/testdir/test_normal.vim

Patch 8.0.0554 (after 8.0.0552)

Problem: Toupper and tolower don't work properly for Turkish when 'casemap' contains "keepascii". (Bjorn Linse)

Solution: When 'casemap' contains "keepascii" use ASCII toupper/tolower.

Files: src/charset.c, src/testdir/test_normal.vim

Patch 8.0.0555 (after 8.0.0552)

Problem: Toupper/tolower test fails on OSX without Darwin.

Solution: Skip that part of the test also for OSX. (Kazunobu Kuriyama)

Files: src/testdir/test_normal.vim

Patch 8.0.0556

Problem: Getting the window position fails if both the GUI and term code is built in.

Solution: Return after getting the GUI window position. (Kazunobu Kuriyama)

Files: src/evalfunc.c

Patch 8.0.0557

Problem: GTK: using static gravities is not useful.

Solution: Remove setting static gravities. (Kazunobu Kuriyama)

Files: src/gui_gtk_f.c

Patch 8.0.0558

Problem: The :ownsyntax command is not tested.

Solution: Add a test. (Dominique Pelle, closes #1622)

Files: src/testdir/test_syntax.vim

Patch 8.0.0559

Problem: Setting 'ttytype' to xxx does not always fail as expected. (Marvin Schmidt)

Solution: Catch both possible errors. (closes #1601)

Files: src/testdir/test_options.vim

Patch 8.0.0560

Problem: :windo allows for ! but it's not supported.

Solution: Disallow passing !. (Hirohito Higashi)

Files: src/ex_cmds.h

Patch 8.0.0561

Problem: Undefined behavior when using backslash after empty line.

Solution: Check for an empty line. (Dominique Pelle, closes #1631)

Files: src/misc2.c, src/testdir/test_vimscript.vim

Patch 8.0.0562

Problem: Not enough test coverage for syntax commands.

Solution: Add a few more tests. (Dominique Pelle, closes #1624)

Files: src/testdir/test_cmdline.vim, src/testdir/test_syntax.vim

Patch 8.0.0563

Problem: Crash when getting the window position in tmux. (Marvin Schmidt)

Solution: Add t_GP to the list of terminal options. (closes #1627)

Files: src/option.c

Patch 8.0.0564

Problem: Cannot detect Bazel BUILD files on some systems.

Solution: Check for BUILD after script checks. (Issue #1340)

Files: runtime/filetype.vim

Patch 8.0.0565

Problem: Using freed memory in :caddbuf after clearing quickfix list.
(Dominique Pelle)

Solution: Set qf_last to NULL.

Files: src/quickfix.c

Patch 8.0.0566

Problem: Setting '**nocompatible**' for the tiny version moves the cursor.

Solution: Use another trick to skip commands when the +eval feature is present. (Christian Brabandt, closes #1630)

Files: runtime/defaults.vim

Patch 8.0.0567

Problem: Call for requesting color and ambiwidth is too early. (Hirohito Higashi)

Solution: Move the call down to below resetting "starting".

Files: src/main.c

Patch 8.0.0568

Problem: "lgd" may hang.

Solution: Don't get stuck in one position. (Christian Brabandt, closes #1643)

Files: src/testdir/test_goto.vim, src/normal.c

Patch 8.0.0569

Problem: Bracketed paste is still enabled when executing a shell command.
(Michael Smith)

Solution: Disable bracketed paste when going into cooked mode. (closes #1638)

Files: src/term.c

Patch 8.0.0570

Problem: Can't run make with several jobs, creating directories has a race condition.

Solution: Use the MKDIR_P autoconf mechanism. (Eric N. Vander Weele, closes #1639)

Files: src/configure.ac, src/auto/configure, src/Makefile, src/config.mk.in, src/install-sh, src/mkinstalldirs, Filelist

Patch 8.0.0571

Problem: The cursor line number becomes negative when using :z^ in an empty buffer. (neovim #6557)
Solution: Correct the line number. Also reset the column.
Files: src/testdir/test_ex_z.vim, src/ex_cmds.c

Patch 8.0.0572

Problem: Building the command table requires Perl.
Solution: Use a Vim script solution. (Dominique Pelle, closes #1641)
Files: src/Makefile, src/create_cmdidxs.pl, src/create_cmdidxs.vim, src/ex_cmdidxs.h, src/ex_docmd.c, Filelist

Patch 8.0.0573

Problem: Running parallel make after distclean fails. (Manuel Ortega)
Solution: Instead of using targets "scratch config myself" use "reconfig".
Files: src/Makefile, src/config.mk.dist

Patch 8.0.0574

Problem: Get only one quickfix list after :caddbuf.
Solution: Reset qf_multiline. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0575

Problem: Using freed memory when resetting 'indentexpr' while evaluating it. (Dominique Pelle)
Solution: Make a copy of 'indentexpr'.
Files: src/misc1.c, src/testdir/test_options.vim

Patch 8.0.0576 (after 8.0.0570 and 8.0.0573)

Problem: Can't build when configure chooses "install-sh". (Daniel Hahler)
Solution: Always use install-sh. Fix remaining use of mkinstalldirs. (closes #1647)
Files: src/installman.sh, src/installml.sh, src/config.mk.in, src/configure.ac, src/auto/configure, src/Makefile

Patch 8.0.0577 (after 8.0.0575)

Problem: Warning for uninitialized variable. (John Marriott)
Solution: Initialize "indent".
Files: src/misc1.c

Patch 8.0.0578

Problem: :simalt on MS-Windows does not work properly.
Solution: Put something in the typeahead buffer. (Christian Brabandt)
Files: src/gui_w32.c

Patch 8.0.0579

Problem: Duplicate test case for quickfix.
Solution: Remove the function. (Yegappan Lakshmanan)
Files: src/testdir/test_quickfix.vim

Patch 8.0.0580

Problem: Cannot set the valid flag with setqflist().
Solution: Add the "valid" argument. (Yegappan Lakshmanan, closes #1642)
Files: runtime/doc/eval.txt, src/quickfix.c,

src/testdir/test_quickfix.vim

Patch 8.0.0581

Problem: Moving folded text is sometimes not correct.
Solution: Bail out when "move_end" is zero. (Matthew Malcomson)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.0.0582

Problem: Illegal memory access with z= command. (Dominique Pelle)
Solution: Avoid case folded text to be longer than the original text. Use MB_PTR2LEN() instead of MB_BYTE2LEN().
Files: src/spell.c, src/testdir/test_spell.vim

Patch 8.0.0583

Problem: Fold test hangs on MS-Windows.
Solution: Avoid overflow in compare.
Files: src/fold.c

Patch 8.0.0584

Problem: Memory leak when executing quickfix tests.
Solution: Free the list reference. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.0585

Problem: Test_options fails when run in the GUI.
Solution: Also check the 'imactivatekey' value when the GUI is not running. Specify test values that work and that fail.
Files: src/option.c, src/testdir/gen_opt_test.vim

Patch 8.0.0586

Problem: No test for mapping timing out.
Solution: Add a test.
Files: src/testdir/test_mapping.vim

Patch 8.0.0587

Problem: Configure check for return value of tgetent is skipped.
Solution: Always perform the check. (Marvin Schmidt, closes #1664)
Files: src/configure.ac, src/auto/configure

Patch 8.0.0588

Problem: job_stop() often assumes the channel will be closed, while the job may not actually be stopped. (Martin Gammelsæter)
Solution: Only assume the job stops on "kill". Don't send a signal if the job has already ended. (closes #1632)
Files: src/channel.c

Patch 8.0.0589 (after 8.0.0578)

Problem: :simalt still does not work.
Solution: Use K_NOP instead of K_IGNORE. (Christian Brabandt)
Files: src/gui_w32.c

Patch 8.0.0590

Problem: Cannot add a context to locations.
Solution: Add the "context" entry in location entries. (Yegappan Lakshmanan,

closes #1012)
Files: src/eval.c, src/proto/quickfix.pro, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.0591

Problem: Changes to eval functionality not documented.
Solution: Include all the changes.
Files: runtime/doc/eval.txt

Patch 8.0.0592

Problem: If a job writes to a buffer and the user is typing a command, the screen isn't updated. When a message is displayed the changed buffer may cause it to be cleared. (Ramel Eshed)
Solution: Update the screen and then the command line if the screen didn't scroll. Avoid inserting screen lines, as it clears any message. Update the status line when the buffer changed.
Files: src/channel.c, src/screen.c, src/ex_getln.c, src/globals.h,
src/vim.h, src/proto/ex_getln.pro, src/proto/screen.pro

Patch 8.0.0593

Problem: Duplication of code for adding a list or dict return value.
Solution: Add rettv_dict_set() and rettv_list_set(). (Yegappan Lakshmanan)
Files: src/dict.c, src/eval.c, src/evalfunc.c, src/if_perl.xs, src/list.c,
src/proto/dict.pro, src/proto/list.pro

Patch 8.0.0594 (after 8.0.0592)

Problem: Build failure when windows feature is missing.
Solution: Add #ifdef.
Files: src/screen.c

Patch 8.0.0595 (after 8.0.0590)

Problem: Coverity warning for not checking return value of dict_add().
Solution: Check the return value for FAIL.
Files: src/quickfix.c

Patch 8.0.0596

Problem: Crash when complete() is called after complete_add() in '**completefunc**'. (Lifepillar)
Solution: Bail out if compl_pattern is NULL. (closes #1668)
Also avoid using freed memory.
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0597

Problem: Off-by-one error in buffer size computation.
Solution: Use ">=" instead of ">". (LemonBoy, closes #1694)
Files: src/quickfix.c

Patch 8.0.0598

Problem: Building with gcc 7.1 yields new warnings.
Solution: Initialize result. (John Marriott)
Files: src/ex_docmd.c

Patch 8.0.0599

Problem: diff mode is insufficiently tested

Solution: Add more test cases. (Dominique Pelle, closes #1685)
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.0.0600

Problem: test_recover fails on some systems.
Solution: Explicitly check if "/" is writable. (Ken Takata)
Files: src/testdir/test_recover.vim

Patch 8.0.0601

Problem: No test coverage for :spellreall.
Solution: Add a test. (Dominique Pelle, closes #1717)
Files: src/testdir/test_spell.vim

Patch 8.0.0602

Problem: When gF fails to edit the file the cursor still moves to the found line number.
Solution: Check the return value of do_ecmd(). (Michael Hwang)
Files: src/normal.c, src/testdir/test_gf.vim

Patch 8.0.0603 (after 8.0.0602)

Problem: gF test fails on MS-Windows.
Solution: Use @ instead of : before the line number
Files: src/testdir/test_gf.vim

Patch 8.0.0604 (after 8.0.0603)

Problem: gF test still fails on MS-Windows.
Solution: Use : before the line number and remove it from 'isfname'.
Files: src/testdir/test_gf.vim

Patch 8.0.0605

Problem: The buffer that quickfix caches for performance may become invalid. (Daniel Hahler)
Solution: Reset qf_last_bufref in qf_init_ext(). (Daniel Hahler, closes #1728, closes #1676)
Files: src/quickfix.c

Patch 8.0.0606

Problem: Cannot set the context for a specified quickfix list.
Solution: Use the list index instead of the current list. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0607

Problem: When creating a bufref, then using :bwipe and :new it might get the same memory and bufref_valid() returns true.
Solution: Add br_fnum to check the buffer number didn't change.
Files: src/structs.h, src/buffer.c, src/globals.h, src/if_py_both.h, src/quickfix.c

Patch 8.0.0608

Problem: Cannot manipulate other than the current quickfix list.
Solution: Pass the list index to quickfix functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.0609

Problem: For some people the hint about quitting is not sufficient.
Solution: Put `<Enter>` separately. Also use `":qa!"` to get out even when there are changes.
Files: `src/normal.c`

Patch 8.0.0610

Problem: The screen is redrawn when `t_BG` is set and used to detect the value for `'background'`.
Solution: Don't redraw when the value of `'background'` didn't change.
Files: `src/term.c`

Patch 8.0.0611

Problem: When `t_u7` is sent a few characters in the second screen line are overwritten and not redrawn later. (Rastislav Barlik)
Solution: Move redrawing the screen to after overwriting the characters.
Files: `src/main.c`, `src/term.c`

Patch 8.0.0612

Problem: Package directories are added to `'runtimepath'` only after loading non-package plugins.
Solution: Split off the code to add package directories to `'runtimepath'`. (Ingo Karkat, closes #1680)
Files: `src/ex_cmds2.c`, `src/globals.h`, `src/main.c`, `src/proto/ex_cmds2.pro`, `src/testdir/test_startup.vim`

Patch 8.0.0613

Problem: The conf filetype detection is done before `ftdetect` scripts from packages that are added later.
Solution: Add the `FALLBACK` argument to `:setfiletype`. (closes #1679, closes #1693)
Files: `src/ex_docmd.c`, `runtime/filetype.vim`, `src/Makefile`, `src/testdir/test_filetype.vim`, `src/testdir/test_alot.vim`

Patch 8.0.0614

Problem: `float2nr()` is not exactly right.
Solution: Make `float2nr()` more accurate. Turn `test65` into a new style test. (Hirohito Higashi, closes #1688)
Files: `src/Makefile`, `src/evalfunc.c`, `src/testdir/Make_all.mak`, `src/testdir/Make_vms.mms`, `src/testdir/test65.in`, `src/testdir/test65.ok`, `src/testdir/test_float_func.vim`, `src/testdir/test_vimscript.vim`, `src/macros.h`

Patch 8.0.0615

Problem: Using `%` with `:hardcopy` wrongly escapes spaces. (Alexey Muranov)
Solution: Expand `%` differently. (Christian Brabandt, closes #1682)
Files: `src/ex_docmd.c`, `src/testdir/test_hardcopy.vim`

Patch 8.0.0616

Problem: When setting the `cterm` background with `":hi Normal"` the value of `'background'` may be set wrongly.
Solution: Check that the color is less than 16. Don't set `'background'` when it was set explicitly. (LemonBoy, closes #1710)

Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.0.0617 (after 8.0.0615)

Problem: Hardcopy test hangs on MS-Windows.

Solution: Check the postscript feature is supported.

Files: src/testdir/test_hardcopy.vim

Patch 8.0.0618

Problem: NFA regex engine handles [0-z] incorrectly.

Solution: Return at the right point. (James McCoy, closes #1703)

Files: src/regexp_nfa.c, src/testdir/test36.in, src/testdir/test36.ok

Patch 8.0.0619

Problem: In the GUI, when a timer uses feedkeys(), it still waits for an event. (Raymond Ko)

Solution: Check tb_change_cnt in one more place.

Files: src/gui.c

Patch 8.0.0620

Problem: Since we only support GTK versions that have it, the check for HAVE_GTK_MULTIHEAD is no longer needed.

Solution: Remove HAVE_GTK_MULTIHEAD. (Kazunobu Kuriyama)

Files: src/config.h.in, src/configure.ac, src/auto/configure, src/gui_beval.c, src/gui_gtk_x11.c, src/mbyte.c

Patch 8.0.0621

Problem: The ":stag" command does not respect 'switchbuf'.

Solution: Check 'switchbuf' for tag commands that may open a new window. (Ingo Karkat, closes #1681) Define macros for the return values of getfile().

Files: src/tag.c, src/testdir/test_tagjump.vim, src/vim.h, src/buffer.c, src/ex_cmds.c, src/search.c,

Patch 8.0.0622

Problem: Using a text object to select quoted text fails when 'selection' is set to "exclusive". (Guraga)

Solution: Swap cursor and visual start position. (Christian Brabandt, closes #1687)

Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.0.0623

Problem: The message "Invalid range" is used for multiple errors.

Solution: Add two more specific error messages. (Itchyny, Ken Hamada)

Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test_regexp_utf8.vim

Patch 8.0.0624 (after 8.0.0623)

Problem: Warning for unused variable in tiny build. (Tony Mechelynck)

Solution: Add an #ifdef.

Files: src/regexp.c

Patch 8.0.0625

Problem: shellescape() always escapes a newline, which does not work with some shells. (Harm te Hennepe)

Solution: Only escape a newline when the "special" argument is non-zero.

(Christian Brabandt, closes #1590)
Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.0.0626

Problem: In the GUI the cursor may flicker.
Solution: Check the cmd_silent flag before updating the cursor shape.
(Hirohito Higashi, closes #1637)
Files: src/getchar.c

Patch 8.0.0627

Problem: When '**wrapsan**' is off "gn" does not select the whole pattern when it's the last one in the text. (KeyboardFire)
Solution: Check if the search fails. (Christian Brabandt, closes #1683)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.0.0628 (after 8.0.0626)

Problem: Cursor disappears after silent mapping. (Ramel Eshed)
Solution: Do restore the cursor when it was changed, but don't change it in the first place for a silent mapping.
Files: src/getchar.c

Patch 8.0.0629 (after 8.0.0611)

Problem: Checking for ambiguous width is not working. (Hirohito Higashi)
Solution: Reset "starting" earlier.
Files: src/main.c

Patch 8.0.0630

Problem: The :global command does not work recursively, which makes it difficult to execute a command on a line where one pattern matches and another does not match. (Miles Cranmer)
Solution: Allow for recursion if it is for only one line. (closes #1760)
Files: src/ex_cmds.c, src/testdir/test_global.vim, runtime/doc/repeat.txt

Patch 8.0.0631

Problem: Perl 5.26 also needs S_TOPMARK and S_POPMARK defined.
Solution: Define the functions when needed. (Jesin, closes #1748)
Files: src/if_perl.xs

Patch 8.0.0632

Problem: The quotestar test is still a bit flaky.
Solution: Kill any existing server to make the retry work. Wait for the register to be filled.
Files: src/testdir/test_quotestar.vim

Patch 8.0.0633

Problem: The client-server test is still a bit flaky.
Solution: Wait a bit for the GUI to start. Check that the version number can be obtained.
Files: src/testdir/test_clientserver.vim

Patch 8.0.0634

Problem: Cannot easily get to the last quickfix list.
Solution: Add "\$" as a value for the "nr" argument of getqflist() and

Files: setqflist(). (Yegappan Lakshmanan)
runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.0635

Problem: When **'ignorecase'** is set script detection is inaccurate.
Solution: Enforce matching case for text. (closes #1753)
Files: runtime/scripts.vim

Patch 8.0.0636

Problem: When reading the undo file fails may use uninitialized data.
Solution: Always clear the buffer on failure.
Files: src/undo.c

Patch 8.0.0637

Problem: Crash when using some version of GTK 3.
Solution: Add #ifdefs around incrementing the menu index. (Kazunobu Kuriyama)
Files: src/gui_gtk.c

Patch 8.0.0638

Problem: Cannot build with new MSVC version VS2017.
Solution: Change the compiler arguments. (Leonardo Valeri Manera, closes #1731, closes #1747)
Files: src/GvimExt/Makefile, src/Make_mvc.mak

Patch 8.0.0639

Problem: The cursor position is set to the last position in a new commit message.
Solution: Don't set the position if the filetype matches "commit". (Christian Brabandt)
Files: runtime/defaults.vim

Patch 8.0.0640

Problem: Mismatch between help and actual message for ":syn conceal".
Solution: Change the message to match the help. (Ken Takata)
Files: src/syntax.c

Patch 8.0.0641

Problem: Cannot set a separate highlighting for the current line in the quickfix window.
Solution: Add QuickFixLine. (anishsane, closes #1755)
Files: src/option.c, src/quickfix.c, src/screen.c, src/syntax.c, src/vim.h, runtime/doc/options.txt, runtime/doc/quickfix.txt

Patch 8.0.0642

Problem: writefile() continues after detecting an error.
Solution: Bail out as soon as an error is detected. (suggestions by Nikolai Pavlov, closes #1476)
Files: src/evalfunc.c, src/testdir/test_writefile.vim

Patch 8.0.0643

Problem: When **'hlsearch'** is set and matching with the last search pattern is very slow, Vim becomes unusable. Cannot quit search by

pressing **CTRL-C**.
Solution: When the search times out set a flag and don't try again. Check for timeout and **CTRL-C** in NFA loop that adds states.
Files: src/screen.c, src/ex_cmds.c, src/quickfix.c, src/regexp.c, src/proto/regexp.pro, src/regexp.h, src/search.c, src/proto/search.pro, src/syntax.c, src/regexp_nfa.c, src/spell.c, src/tag.c, src/gui.c, src/edit.c, src/evalfunc.c, src/ex_docmd.c, src/ex_getln.c, src/normal.c

Patch 8.0.0644

Problem: There is no test for **'hlsearch'** timing out.
Solution: Add a test.
Files: src/testdir/test_hlsearch.vim

Patch 8.0.0645

Problem: The new regexp engine does not give an error for using a back reference where it is not allowed. (Dominique Pelle)
Solution: Check the back reference like the old engine. (closes #1774)
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test_hlsearch.vim, src/testdir/test_statusline.vim, src/testdir/test_regexp_latin1.vim

Patch 8.0.0646

Problem: The hlsearch test fails on fast systems.
Solution: Make the search pattern slower. Fix that the old regexp engine doesn't timeout properly.
Files: src/regexp.c, src/testdir/test_hlsearch.vim

Patch 8.0.0647

Problem: Syntax highlighting can cause a freeze.
Solution: Apply **'redrawtime'** to syntax highlighting, per window.
Files: src/structs.h, src/screen.c, src/syntax.c, src/normal.c, src/regexp.c, src/proto/syntax.pro, src/testdir/test_syntax.vim, runtime/doc/options.txt

Patch 8.0.0648

Problem: Possible use of NULL pointer if buflist_new() returns NULL. (Coverity)
Solution: Check for NULL pointer in set_bufref().
Files: src/buffer.c

Patch 8.0.0649

Problem: When opening a help file the filetype is set several times.
Solution: When setting the filetype to the same value from a modeline, don't trigger FileType autocommands. Don't set the filetype to "help" when it's already set correctly.
Files: src/ex_cmds.c, src/option.c, runtime/filetype.vim

Patch 8.0.0650

Problem: For extra help files the filetype is set more than once.
Solution: In *.txt files check that there is no help file modline.
Files: runtime/filetype.vim

Patch 8.0.0651 (after 8.0.0649)

Problem: Build failure without the auto command feature.
Solution: Add #ifdef. (closes #1782)
Files: src/ex_cmds.c

Patch 8.0.0652

Problem: Unicode information is outdated.
Solution: Update to Unicode 10. (Christian Brabandt)
Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 8.0.0653

Problem: The default highlight for QuickFixLine does not work for several color schemes. (Manas Thakur)
Solution: Make the default use the old color. (closes #1780)
Files: src/syntax.c

Patch 8.0.0654

Problem: Text found after :endfunction is silently ignored.
Solution: Give a warning if **'verbose'** is set. When | or \n are used, execute the text as a command.
Files: src/testdir/test_vimscript.vim, src/userfunc.c, runtime/doc/eval.txt

Patch 8.0.0655

Problem: Not easy to make sure a function does not exist.
Solution: Add ! as an optional argument to :delfunc.
Files: src/userfunc.c, src/ex_cmds.h, src/testdir/test_vimscript.vim

Patch 8.0.0656

Problem: Cannot use ! after some user commands.
Solution: Properly check for existing command. (Hirohito Higashi)
Files: src/ex_docmd.c, src/testdir/test_vimscript.vim

Patch 8.0.0657

Problem: Cannot get and set quickfix list items.
Solution: Add the "items" argument to getqflist() and setqflist(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0658

Problem: Spell test is old style.
Solution: Turn the spell test into a new style test (pschuh, closes #1778)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test58.in, src/testdir/test58.ok, src/testdir/test_spell.vim

Patch 8.0.0659

Problem: No test for conceal mode.
Solution: Add a conceal mode test. (Dominique Pelle, closes #1783)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_syntax.vim

Patch 8.0.0660

Problem: Silent install on MS-Windows does show a dialog.
Solution: Add /SD to the default choice. (allburov, closes #1772)

Files: nsis/gvim.nsi

Patch 8.0.0661

Problem: Recognizing urxvt mouse codes does not work well.
Solution: Recognize "Esc[*M" and "Esc[*m". (Maurice Bos, closes #1486)
Files: src/keymap.h, src/misc2.c, src/os_unix.c, src/term.c

Patch 8.0.0662 (after 8.0.0659)

Problem: Stray FIXME for fixed problem.
Solution: Remove the comment. (Dominique Pelle)
Files: src/testdir/test_syntax.vim

Patch 8.0.0663

Problem: Giving an error message only when **'verbose'** set is unexpected.
Solution: Give a warning message instead.
Files: src/message.c, src/proto/message.pro, src/userfunc.c,
 src/testdir/test_vimscript.vim, runtime/doc/eval.txt

Patch 8.0.0664 (after 8.0.0661)

Problem: Mouse does not work in tmux. (lilydjwg)
Solution: Add flag for SGR release being present.
Files: src/term.c

Patch 8.0.0665 (after 8.0.0661)

Problem: Warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize it.
Files: src/term.c

Patch 8.0.0666

Problem: Dead for loop. (Coverity)
Solution: Remove the for loop.
Files: src/term.c

Patch 8.0.0667

Problem: Memory access error when command follows :endfunction. (Nikolai
 Pavlov)
Solution: Make memory handling in :function straightforward. (closes #1793)
Files: src/userfunc.c, src/testdir/test_vimscript.vim

Patch 8.0.0668 (after 8.0.0660)

Problem: Nsis installer script does not work. (Christian Brabandt)
Solution: Fix the syntax of /SD.
Files: nsis/gvim.nsi

Patch 8.0.0669

Problem: In Insert mode, **CTRL-N** at start of the buffer does not work
 correctly. (zuloloxi)
Solution: Wrap around the start of the buffer. (Christian Brabandt)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.0670

Problem: Can't use input() in a timer callback. (Cosmin Popescu)
Solution: Reset vgetc_busy and set timer_busy. (Ozaki Kiichi, closes #1790,
 closes #1129)

Files: src/evalfunc.c, src/ex_cmds2.c, src/globals.h,
src/testdir/test_timers.vim

Patch 8.0.0671

Problem: When a function invoked from a timer calls confirm() and the user types **CTRL-C** then Vim hangs.

Solution: Reset typebuf_was_filled. (Ozaki Kiichi, closes #1791)

Files: src/getchar.c

Patch 8.0.0672

Problem: Third item of synconcealed() changes too often. (Dominique Pelle)

Solution: Reset the sequence number at the start of each line.

Files: src/syntax.c, src/testdir/test_syntax.vim, runtime/doc/eval.txt

Patch 8.0.0673 (after 8.0.0673)

Problem: Build failure without conceal feature.

Solution: Add #ifdef.

Files: src/syntax.c

Patch 8.0.0674 (after 8.0.0670)

Problem: Cannot build with eval but without timers.

Solution: Add #ifdef (John Marriott)

Files: src/evalfunc.c

Patch 8.0.0675

Problem: **'colorcolumn'** has a higher priority than **'hlsearch'**, it should be the other way around. (Nazri Ramliy)

Solution: Change the priorities. (LemonBoy, closes #1794)

Files: src/screen.c, src/testdir/test_listlbr_utf8.vim

Patch 8.0.0676

Problem: Crash when closing the quickfix window in a FileType autocommand that triggers when the quickfix window is opened.

Solution: Save the new value before triggering the OptionSet autocommand. Add the "starting" flag to test_override() to make the text work.

Files: src/evalfunc.c, src/option.c, runtime/doc/eval.txt

Patch 8.0.0677

Problem: Setting **'filetype'** internally may cause the current buffer and window to change unexpectedly.

Solution: Set curbuf_lock. (closes #1734)

Files: src/quickfix.c, src/ex_cmds.c, src/ex_getln.c,
src/testdir/test_quickfix.vim

Patch 8.0.0678

Problem: When **'equalalways'** is set and closing a window in a separate frame, not all window sizes are adjusted. (Glacambre)

Solution: Resize all windows if the new current window is not in the same frame as the closed window. (closes #1707)

Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.0.0679 (after 8.0.0678)

Problem: Using freed memory.

Solution: Get the parent frame pointer earlier.

Files: src/window.c

Patch 8.0.0680 (after 8.0.0612)

Problem: Plugins in start packages are sourced twice. (mseplowitz)

Solution: Use the unmodified runtime path when loading plugins (test by Ingo Karkat, closes #1801)

Files: src/testdir/test_startup.vim, src/main.c, src/ex_cmds2.c, src/proto/ex_cmds2.pro

Patch 8.0.0681

Problem: Unnamed register only contains the last deleted text when appending deleted text to a register. (Wolfgang Jeltsch)

Solution: Only set y_previous when not using y_append. (Christian Brabandt)

Files: src/ops.c, src/testdir/test_put.vim

Patch 8.0.0682

Problem: No test for synIDtrans().

Solution: Add a test. (Dominique Pelle, closes #1796)

Files: src/testdir/test_syntax.vim

Patch 8.0.0683

Problem: When using a visual bell there is no delay, causing the flash to be very short, possibly unnoticeable. Also, the flash and the beep can lockup the UI when repeated often.

Solution: Do the delay in Vim or flush the output before the delay. Limit the bell to once per half a second. (Ozaki Kiichi, closes #1789)

Files: src/misc1.c, src/proto/term.pro, src/term.c

Patch 8.0.0684

Problem: Old style tests are not nice.

Solution: Turn two tests into new style. (pschuh, closes #1797)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test82.in, src/testdir/test82.ok, src/testdir/test90.in, src/testdir/test90.ok, src/testdir/test_sha256.vim, src/testdir/test_utf8_comparisons.vim

Patch 8.0.0685

Problem: When making backups is disabled and conversion with iconv fails the written file is truncated. (Luo Chen)

Solution: First try converting the file and write the file only when it did not fail. (partly by Christian Brabandt)

Files: src/fileio.c, src/testdir/test_writefile.vim

Patch 8.0.0686

Problem: When typing **CTRL-L** in a window that's not the first one, another redraw will happen later. (Christian Brabandt)

Solution: Reset must_redraw after calling screenclear().

Files: src/screen.c

Patch 8.0.0687

Problem: Minor issues related to quickfix.

Solution: Set the proper return status for all cases in setqflist() and at test cases for this. Move the "adding" flag outside of FEAT_WINDOWS. Minor update to the setqflist() help text. (Yegappan

Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.0688

Problem: Cannot resize the window in a FileType autocommand. (Ingo Karkat)
Solution: Add the CMDWIN flag to :resize. (test by Ingo Karkat,
closes #1804)
Files: src/ex_cmds.h, src/testdir/test_quickfix.vim

Patch 8.0.0689

Problem: The ~ character is not escaped when adding to the search pattern
with **CTRL-L**. (Ramel Eshed)
Solution: Escape the character. (Christian Brabandt)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.0.0690

Problem: Compiler warning on non-Unix system.
Solution: Add #ifdef. (John Marriott)
Files: src/term.c

Patch 8.0.0691

Problem: Compiler warning without the linebreak feature.
Solution: Add #ifdef. (John Marriott)
Files: src/edit.c

Patch 8.0.0692

Problem: Using **CTRL-G** with '**incsearch**' and ? goes in the wrong direction.
(Ramel Eshed)
Solution: Adjust search_start. (Christian Brabandt)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.0.0693

Problem: No terminal emulator support. Cannot properly run commands in the
GUI. Cannot run a job interactively with an ssh connection.
Solution: Very early implementation of the :terminal command. Includes
libvterm converted to ANSI C. Many parts still missing.
Files: src/feature.h, src/Makefile, src/configure.ac, src/auto/configure,
src/config.mk.in, src/config.h.in, src/terminal.c, src/structs.h,
src/ex_cmdidxs.h, src/ex_docmd.c, src/option.c, src/option.h,
src/evalfunc.c, src/proto/terminal.pro, src/proto.h,
runtime/doc/terminal.txt, runtime/doc/Makefile, Filelist,
src/libvterm/.bzrignore, src/libvterm/.gitignore,
src/libvterm/LICENSE, src/libvterm/README, src/libvterm/Makefile,
src/libvterm/tbl2inc_c.pl, src/libvterm/vterm.pc.in,
src/libvterm/bin/unterm.c, src/libvterm/bin/vterm-ctrl.c,
src/libvterm/bin/vterm-dump.c, src/libvterm/doc/URLs,
src/libvterm/doc/seqs.txt, src/libvterm/include/vterm.h,
src/libvterm/include/vterm_keycodes.h,
src/libvterm/src/encoding.c,
src/libvterm/src/encoding/DECdrawing.inc,
src/libvterm/src/encoding/DECdrawing.tbl,
src/libvterm/src/encoding/uk.inc,
src/libvterm/src/encoding/uk.tbl, src/libvterm/src/keyboard.c,

src/libvterm/src/mouse.c, src/libvterm/src/parser.c,
src/libvterm/src/pen.c, src/libvterm/src/rect.h,
src/libvterm/src/screen.c, src/libvterm/src/state.c,
src/libvterm/src/unicode.c, src/libvterm/src/utf8.h,
src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h,
src/libvterm/t/02parser.test, src/libvterm/t/03encoding_utf8.test,
src/libvterm/t/10state_putglyph.test,
src/libvterm/t/11state_movecursor.test,
src/libvterm/t/12state_scroll.test,
src/libvterm/t/13state_edit.test,
src/libvterm/t/14state_encoding.test,
src/libvterm/t/15state_mode.test,
src/libvterm/t/16state_resize.test,
src/libvterm/t/17state_mouse.test,
src/libvterm/t/18state_termprops.test,
src/libvterm/t/20state_wrapping.test,
src/libvterm/t/21state_tabstops.test,
src/libvterm/t/22state_save.test,
src/libvterm/t/25state_input.test,
src/libvterm/t/26state_query.test,
src/libvterm/t/27state_reset.test,
src/libvterm/t/28state_dbl_wh.test,
src/libvterm/t/29state_fallback.test, src/libvterm/t/30pen.test,
src/libvterm/t/40screen_ascii.test,
src/libvterm/t/41screen_unicode.test,
src/libvterm/t/42screen_damage.test,
src/libvterm/t/43screen_resize.test,
src/libvterm/t/44screen_pen.test,
src/libvterm/t/45screen_protect.test,
src/libvterm/t/46screen_extent.test,
src/libvterm/t/47screen_dbl_wh.test,
src/libvterm/t/48screen_termprops.test,
src/libvterm/t/90vttest_01-movement-1.test,
src/libvterm/t/90vttest_01-movement-2.test,
src/libvterm/t/90vttest_01-movement-3.test,
src/libvterm/t/90vttest_01-movement-4.test,
src/libvterm/t/90vttest_02-screen-1.test,
src/libvterm/t/90vttest_02-screen-2.test,
src/libvterm/t/90vttest_02-screen-3.test,
src/libvterm/t/90vttest_02-screen-4.test,
src/libvterm/t/92lp1640917.test, src/libvterm/t/harness.c,
src/libvterm/t/run-test.pl

Patch 8.0.0694

Problem: Building in shadow directory does not work. Running Vim fails.

Solution: Add the new libvterm directory. Add missing change in command list.

Files: src/Makefile, src/ex_cmds.h

Patch 8.0.0695

Problem: Missing dependencies breaks parallel make.

Solution: Add dependencies for terminal.o.

Files: src/Makefile

Patch 8.0.0696

Problem: The .inc files are missing in git. (Nazri Ramliy)
Solution: Remove the .inc line from .gitignore.
Files: src/libvterm/.gitignore

Patch 8.0.0697

Problem: Recorded key sequences may become invalid.
Solution: Add back KE_SNIFF removed in 7.4.1433. Use fixed numbers for the key_extra enum.
Files: src/keymap.h

Patch 8.0.0698

Problem: When a timer uses ":pyeval" or another Python command and it happens to be triggered while exiting a Crash may happen. (Ricky Zhou)
Solution: Avoid running a Python command after python_end() was called. Do not trigger timers while exiting. (closes #1824)
Files: src/if_python.c, src/if_python3.c, src/ex_cmds2.c

Patch 8.0.0699

Problem: Checksum tests are not actually run.
Solution: Add the tests to the list. (Dominique Pelle, closes #1819)
Files: src/testdir/test_alot.vim, src/testdir/test_alot_utf8.vim

Patch 8.0.0700

Problem: Segfault with QuitPre autocommand closes the window. (Marek)
Solution: Check that the window pointer is still valid. (Christian Brabandt, closes #1817)
Files: src/testdir/test_tabpage.vim, src/ex_docmd.c

Patch 8.0.0701

Problem: System test failing when using X11 forwarding.
Solution: Set \$XAUTHORITY before changing \$HOME. (closes #1812)
Also use a better check for the exit value.
Files: src/testdir/setup.vim, src/testdir/test_system.vim

Patch 8.0.0702

Problem: An error in a timer can make Vim unusable.
Solution: Don't set the error flag or exception from a timer. Stop a timer if it causes an error 3 out of 3 times. Discard an exception caused inside a timer.
Files: src/ex_cmds2.c, src/structs.h, src/testdir/test_timers.vim, runtime/doc/eval.txt

Patch 8.0.0703

Problem: Illegal memory access with empty :doau command.
Solution: Check the event for being out of range. (James McCoy)
Files: src/testdir/test_autocmd.vim, src/fileio.c

Patch 8.0.0704

Problem: Problems with autocommands when opening help.
Solution: Avoid using invalid "varp" value. Allow using :wincmd if buffer is locked. (closes #1806, closes #1804)
Files: src/option.c, src/ex_cmds.h

Patch 8.0.0705 (after 8.0.0702)

Problem: Crash when there is an error in a timer callback. (Aron Griffis, Ozaki Kiichi)

Solution: Check did_throw before discarding an exception. NULLify current_exception when no longer valid.

Files: src/ex_eval.c, src/ex_cmds2.c

Patch 8.0.0706

Problem: Crash when cancelling the cmdline window in Ex mode. (James McCoy)

Solution: Do not set cmdbuff to NULL, make it empty.

Files: src/ex_getln.c

Patch 8.0.0707

Problem: Freeing wrong memory when manipulating buffers in autocommands. (James McCoy)

Solution: Also set the w_s pointer if w_buffer was NULL.

Files: src/ex_cmds.c

Patch 8.0.0708

Problem: Some tests are old style.

Solution: Change a few tests from old style to new style. (pschuh, closes #1813)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, src/testdir/main.aap, src/testdir/test23.in, src/testdir/test23.ok, src/testdir/test24.in, src/testdir/test24.ok, src/testdir/test26.in, src/testdir/test26.ok, src/testdir/test67.in, src/testdir/test67.ok, src/testdir/test75.in, src/testdir/test75.ok, src/testdir/test97.in, src/testdir/test97.ok, src/testdir/test_comparators.in, src/testdir/test_comparators.ok, src/testdir/test_comparators.vim, src/testdir/test_escaped_glob.vim, src/testdir/test_exec_while_if.vim, src/testdir/test_exists_autocmd.vim, src/testdir/test_getcwd.in, src/testdir/test_getcwd.ok, src/testdir/test_getcwd.vim, src/testdir/test_maparg.vim, src/testdir/test_plus_arg_edit.vim, src/testdir/test_regex_char_classes.vim

Patch 8.0.0709

Problem: Libvterm cannot use vsnprintf(), it does not exist in C90.

Solution: Use vim_vsnprintf() instead.

Files: src/message.c, src/Makefile, src/proto.h, src/evalfunc.c, src/netbeans.c, src/libvterm/src/vterm.c

Patch 8.0.0710

Problem: A job that writes to a buffer clears command line completion. (Ramel Eshed)

Solution: Do not redraw while showing the completion menu.

Files: src/screen.c

Patch 8.0.0711 (after 8.0.0710)

Problem: Cannot build without the wildmenu feature.

Solution: Add #ifdef
Files: src/screen.c

Patch 8.0.0712

Problem: The terminal implementation is incomplete.
Solution: Add the **'termkey'** option.
Files: src/option.c, src/option.h, src/structs.h

Patch 8.0.0713 (after 8.0.0712)

Problem: **'termkey'** option not fully implemented.
Solution: Add initialisation.
Files: src/option.c

Patch 8.0.0714

Problem: When a timer causes a command line redraw the " that is displayed for **CTRL-R** goes missing.
Solution: Remember an extra character to display.
Files: src/ex_getln.c

Patch 8.0.0715

Problem: Writing to the wrong buffer if the buffer that a channel writes to was closed.
Solution: Do not write to a buffer that was unloaded.
Files: src/channel.c, src/testdir/test_channel.vim, src/testdir/test_channel_write.py

Patch 8.0.0716

Problem: Not easy to start Vim cleanly without changing the viminfo file. Not possible to know whether the -i command line flag was used.
Solution: Add the --clean command line argument. Add the **'viminfofile'** option. Add "-u DEFAULTS".
Files: src/main.c, runtime/doc/starting.txt, src/option.c, src/option.h, src/ex_cmds.c, src/globals.h, runtime/doc/options.txt

Patch 8.0.0717

Problem: Terminal feature not included in :version output.
Solution: Add +terminal or -terminal.
Files: src/version.c, src/terminal.c

Patch 8.0.0718

Problem: Output of job in terminal is not displayed.
Solution: Connect the job output to the terminal.
Files: src/channel.c, src/proto/channel.pro, src/terminal.c, src/proto/terminal.pro, src/channel.c, src/proto/channel.pro, src/evalfunc.c, src/screen.c, src/proto/screen.pro

Patch 8.0.0719

Problem: Build failure without +terminal feature.
Solution: Add #ifdefs.
Files: src/screen.c, src/channel.c

Patch 8.0.0720

Problem: Unfinished mapping not displayed when running timer.
Solution: Also use the extra_char while waiting for a mapping and digraph.

(closes #1844)
Files: src/ex_getln.c

Patch 8.0.0721

Problem: :argedit can only have one argument.
Solution: Allow for multiple arguments. (Christian Brabandt)
Files: runtime/doc/editing.txt, src/ex_cmds.h, src/ex_cmds2.c,
src/testdir/test_arglist.vim

Patch 8.0.0722

Problem: Screen is messed by timer up at inputlist() prompt.
Solution: Set state to ASKMORE. (closes #1843)
Files: src/misc1.c

Patch 8.0.0723 (after 8.0.0721)

Problem: Arglist test fails if file name case is ignored.
Solution: Wipe existing buffers, check for fname_case property.
Files: src/testdir/test_arglist.vim

Patch 8.0.0724

Problem: The message for yanking doesn't indicate the register.
Solution: Show the register name in the "N lines yanked" message. (LemonBoy,
closes #1803, closes #1809)
Files: src/ops.c, src/Makefile, src/testdir/test_registers.vim,
src/testdir/Make_all.mak

Patch 8.0.0725

Problem: A terminal window does not handle keyboard input.
Solution: Add terminal_loop(). ":term bash -i" sort of works now.
Files: src/main.c, src/terminal.c, src/proto/terminal.pro, src/normal.c

Patch 8.0.0726

Problem: Translations cleanup script is too conservative.
Solution: Also delete untranslated messages.
Files: src/po/cleanup.vim

Patch 8.0.0727

Problem: Message about what register to yank into is not translated.
(LemonBoy)
Solution: Add _().
Files: src/ops.c

Patch 8.0.0728

Problem: The terminal structure is never freed.
Solution: Free the structure and unreference what it contains.
Files: src/terminal.c, src/buffer.c, src/proto/terminal.pro,
src/channel.c, src/proto/channel.pro, src/evalfunc.c

Patch 8.0.0729

Problem: The help for the terminal configure option is wrong.
Solution: Change "Disable" to "Enable". (E Kawashima, closes #1849)
Improve alignment.
Files: src/configure.ac, src/auto/configure

Patch 8.0.0730

Problem: Terminal feature only supports Unix-like systems.
Solution: Prepare for adding an MS-Windows implementation.
Files: src/terminal.c

Patch 8.0.0731

Problem: Cannot build the terminal feature on MS-Windows.
Solution: Add the Makefile changes. (Yasuhiro Matsumoto, closes #1851)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.0.0732

Problem: When updating a buffer for a callback the modeless selection is lost.
Solution: Do not insert or delete screen lines when redrawing for a callback and there is a modeless selection.
Files: src/screen.c

Patch 8.0.0733

Problem: Can only add entries to one list in the quickfix stack.
Solution: Move state variables from qf_list_T to qf_list_T. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.0734

Problem: The script to check translations can be improved.
Solution: Restore the view when no errors are found. Check for matching line break at the end of the message. (Christian Brabandt)
Files: src/po/check.vim

Patch 8.0.0735

Problem: There is no way to notice that the quickfix window contents has changed.
Solution: Increment b:changedtick when updating the quickfix window. (Yegappan Lakshmanan)
Files: runtime/doc/quickfix.txt, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.0736

Problem: The OptionSet autocommand event is not triggered when entering diff mode.
Solution: use set_option_value() instead of setting the option directly. Change the tests from old to new style. (Christian Brabandt)
Files: src/diff.c, src/testdir/Make_all.mak, src/Makefile, src/testdir/test_autocmd.vim, src/testdir/test_autocmd_option.in, src/testdir/test_autocmd_option.ok

Patch 8.0.0737

Problem: Crash when X11 selection is very big.
Solution: Use static items instead of allocating them. Add callbacks. (Ozaki Kiichi)
Files: src/testdir/shared.vim, src/testdir/test_quotestar.vim, src/ui.c

Patch 8.0.0738

Problem: Cannot use the mouse to resize window while the focus is in a terminal window.
Solution: Recognize nice mouse events in the terminal window. A few more fixes for the terminal window.
Files: src/terminal.c

Patch 8.0.0739

Problem: Terminal resizing doesn't work well.
Solution: Resize the terminal to the Vim window and the other way around. Avoid mapping typed keys. Set the environment properly.
Files: src/terminal.c, src/os_unix.c, src/structs.h

Patch 8.0.0740

Problem: Cannot resize a terminal window by the command running in it.
Solution: Add support for the window size escape sequence. Make BS work.
Files: src/terminal.c, src/libvterm/src/state.c

Patch 8.0.0741

Problem: Cannot build with HPUX.
Solution: Rename envbuf_TERM to envbuf_Term. (John Marriott)
Files: src/os_unix.c

Patch 8.0.0742

Problem: Terminal feature does not work on MS-Windows.
Solution: Use libvterm and libwinpty on MS-Windows. (Yasuhiro Matsumoto)
Files: src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/channel.c, src/proto/channel.pro, src/terminal.c

Patch 8.0.0743

Problem: The **'termsize'** option can be set to an invalid value.
Solution: Check the **'termsize'** option to be valid.
Files: src/option.c, src/testdir/gen_opt_test.vim

Patch 8.0.0744

Problem: A terminal window uses pipes instead of a pty.
Solution: Add pty support.
Files: src/structs.h, src/os_unix.c, src/terminal.c, src/channel.c, src/proto/os_unix.pro, src/os_win32.c, src/proto/os_win32.pro

Patch 8.0.0745

Problem: multibyte characters in a terminal window are not displayed properly.
Solution: Set the unused screen characters. (Yasuhiro Matsumoto, closes #1857)
Files: src/terminal.c

Patch 8.0.0746

Problem: When :term fails the job is not properly cleaned up.
Solution: Free the terminal. Handle a job that failed to start. (closes #1858)
Files: src/os_unix.c, src/channel.c, src/terminal.c

Patch 8.0.0747

Problem: :terminal without an argument doesn't work.

Solution: Use the **'shell'** option. (Yasuhiro Matsumoto, closes #1860)
Files: src/terminal.c

Patch 8.0.0748

Problem: When running Vim in a terminal window it does not detect the right number of colors available.

Solution: Detect the version string that libvterm returns. Pass the number of colors in \$COLORS.

Files: src/term.c, src/os_unix.c

Patch 8.0.0749

Problem: Some unicode digraphs are hard to remember.

Solution: Add alternatives with a backtick. (Chris Harding, closes #1861)

Files: src/digraph.c

Patch 8.0.0750

Problem: OpenPTY missing in non-GUI build.

Solution: Always include pty.c, add an #ifdef to skip over the contents.

Files: src/pty.c, src/Makefile

Patch 8.0.0751 (after 8.0.0750)

Problem: OpenPTY missing with some combination of features. (Kazunobu Kuriyama)

Solution: Adjust #ifdef. Also include pty.pro when needed.

Files: src/pty.c, src/misc2.c, src/proto.h

Patch 8.0.0752

Problem: Build fails on MS-Windows.

Solution: Change #ifdef for set_color_count().

Files: src/term.c

Patch 8.0.0753

Problem: A job running in a terminal does not get notified of changes in the terminal size.

Solution: Use ioctl() and SIGWINCH to report the terminal size.

Files: src/terminal.c, src/os_unix.c, src/proto/os_unix.pro

Patch 8.0.0754

Problem: Terminal window does not support colors.

Solution: Lookup the color attribute.

Files: src/terminal.c, src/syntax.c, src/proto/syntax.pro

Patch 8.0.0755

Problem: Terminal window does not have colors in the GUI.

Solution: Lookup the GUI color.

Files: src/terminal.c, src/syntax.c, src/proto/syntax.pro, src/term.c, src/proto/term.pro, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/gui_x11.c, src/proto/gui_x11.pro, src/gui_mac.c, src/proto/gui_mac.pro, src/gui_photon.c, src/proto/gui_photon.pro, src/gui_w32.c, src/proto/gui_w32.pro,

Patch 8.0.0756

Problem: Cannot build libvterm with MSVC.

Solution: Add an MSVC Makefile to libvterm. (Yasuhiro Matsumoto, closes

#1865)
Files: src/INSTALLpc.txt, src/Make_mvc.mak, src/libvterm/Makefile.msc

Patch 8.0.0757

Problem: Libvterm MSVC Makefile not included in the distribution.
Solution: Add the file to the list.
Files: Filelist

Patch 8.0.0758

Problem: Possible crash when using a terminal window.
Solution: Check for NULL pointers. (Yasuhiro Matsumoto, closes #1864)
Files: src/terminal.c

Patch 8.0.0759

Problem: MS-Windows: terminal does not adjust size to the Vim window size.
Solution: Add a call to winpty_set_size(). (Yasuhiro Matsumoto, closes #1863)
Files: src/terminal.c

Patch 8.0.0760

Problem: Terminal window colors wrong with **'termguicolors'**.
Solution: Add **'termguicolors'** support.
Files: src/terminal.c, src/syntax.c, src/proto/syntax.pro

Patch 8.0.0761

Problem: Options of a buffer for a terminal window are not set properly.
Solution: Add "terminal" value for **'buftype'**. Make **'buftype'** and **'bufhidden'** not depend on the quickfix feature.
Also set the buffer name and show "running" or "finished" in the window title.
Files: src/option.c, src/terminal.c, src/proto/terminal.pro, runtime/doc/options.txt, src/quickfix.c, src/proto/quickfix.pro, src/structs.h, src/buffer.c, src/ex_docmd.c, src/fileio.c, src/channel.c

Patch 8.0.0762

Problem: ml_get error with :psearch in buffer without a name. (Dominique Pelle)
Solution: Use the buffer number instead of the file name. Check the cursor position.
Files: src/search.c, src/testdir/test_preview.vim, src/Makefile, src/testdir/Make_all.mak

Patch 8.0.0763

Problem: Libvterm can be improved.
Solution: Various small improvements, more comments.
Files: src/libvterm/README, src/libvterm/include/vterm.h, src/libvterm/include/vterm_keycodes.h, src/libvterm/src/keyboard.c, src/libvterm/src/parser.c, src/libvterm/src/screen.c, src/libvterm/src/state.c

Patch 8.0.0764

Problem: **'termkey'** does not work yet.
Solution: Implement **'termkey'**.
Files: src/terminal.c, src/option.c, src/proto/option.pro

Patch 8.0.0765

Problem: Build fails with tiny features.
Solution: Adjust #ifdef. (John Marriott)
Files: src/option.c, src/option.h

Patch 8.0.0766

Problem: Option test fails with +terminal feature.
Solution: Fix using the right option when checking the value.
Files: src/option.c

Patch 8.0.0767

Problem: Build failure with Athena and Motif.
Solution: Move local variable declarations. (Kazunobu Kuriyama)
Files: src/gui_x11.c

Patch 8.0.0768

Problem: Terminal window status shows "[Scratch]".
Solution: Show "[Terminal]" when no title was set. (Yasuhiro Matsumoto)
Store the terminal title that vterm sends and use it. Update the special buffer name. (closes #1869)
Files: src/terminal.c, src/proto/terminal.pro, src/buffer.c

Patch 8.0.0769

Problem: Build problems with terminal on MS-Windows using MSVC.
Solution: Remove stdbool.h dependency. Only use ScreenLinesUC when it was allocated. Fix typos. (Ken Takata)
Files: src/libvterm/bin/vterm-ctrl.c, runtime/doc/terminal.txt, src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/libvterm/Makefile.msc, src/terminal.c

Patch 8.0.0770

Problem: Compiler warning for missing field initializer.
Solution: Add two more values. (Yegappan Lakshmanan)
Files: src/libvterm/src/encoding.c

Patch 8.0.0771

Problem: Cursor in a terminal window not always updated in the GUI.
Solution: Call gui_update_cursor(). (Yasuhiro Matsumoto, closes #1868)
Files: src/terminal.c

Patch 8.0.0772

Problem: Other stdbool.h dependencies in libvterm.
Solution: Remove the dependency and use TRUE/FALSE/int. (Ken Takata)
Files: src/libvterm/include/vterm.h, src/libvterm/src/mouse.c, src/libvterm/src/pen.c, src/libvterm/t/harness.c, src/libvterm/bin/unterm.c

Patch 8.0.0773

Problem: Mixing 32 and 64 bit libvterm builds fails.
Solution: Use OUTDIR. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/libvterm/Makefile.msc

Patch 8.0.0774

Problem: Build failure without the multibyte feature on HPUX.
Solution: Move #ifdefs. (John Marriott)
Files: src/term.c

Patch 8.0.0775

Problem: In a terminal the cursor is updated too often.
Solution: Only flush when needed. (Yasuhiro Matsumoto). Remember whether the cursor is visible. (closes #1873)
Files: src/terminal.c

Patch 8.0.0776

Problem: Function prototypes missing without the quickfix feature. (Tony Mechelynck)
Solution: Move non-quickfix functions to buffer.c.
Files: src/buffer.c, src/proto/buffer.pro, src/quickfix.c, src/proto/quickfix.pro

Patch 8.0.0777

Problem: Compiler warnings with 64 bit compiler.
Solution: Add type casts. (Mike Williams)
Files: src/libvterm/src/pen.c, src/libvterm/src/state.c, src/terminal.c

Patch 8.0.0778

Problem: In a terminal the cursor may be hidden and screen updating lags behind. (Nazri Ramliy)
Solution: Switch the cursor on and flush output when needed. (Ozaki Kiichi)
Files: src/terminal.c

Patch 8.0.0779

Problem: :term without an argument uses empty buffer name but runs the shell.
Solution: Change the command to the shell earlier.
Files: src/terminal.c

Patch 8.0.0780

Problem: Build failure on Travis.
Solution: Set distribution explicitly. Use Lua and Ruby dev. (Ken Takata, closes #1884)
Files: .travis.yml

Patch 8.0.0781

Problem: MS-Windows: Memory leak when using :terminal.
Solution: Handle failures properly. (Ken Takata)
Files: src/terminal.c

Patch 8.0.0782

Problem: Using freed memory in quickfix code. (Dominique Pelle)
Solution: Handle a help window differently. (Yegappan Lakshmanan)
Files: src/buffer.c, src/proto/buffer.pro, src/quickfix.c, src/testdir/test_quickfix.vim, src/ex_cmds.c, src/window.c

Patch 8.0.0783

Problem: Job of terminal may be freed too early.
Solution: Increment job refcount. (Yasuhiro Matsumoto)

Files: src/terminal.c

Patch 8.0.0784

Problem: Job of terminal may be garbage collected.

Solution: Set copyID on job in terminal. (Ozaki Kiichi)

Files: src/terminal.c, src/eval.c, src/proto/terminal.pro

Patch 8.0.0785

Problem: Wildcards are not expanded for :terminal.

Solution: Add FILES to the command flags. (Yasuhiro Matsumoto, closes #1883)
Also complete commands.

Files: src/ex_cmds.h, src/ex_docmd.c

Patch 8.0.0786

Problem: Build failures on Travis.

Solution: Go back to precise temporarily. Disable coverage with clang.

Files: .travis.yml

Patch 8.0.0787

Problem: Cannot send **CTRL-W** command to terminal job.

Solution: Make **CTRL-W** . a prefix for sending a key to the job.

Files: src/terminal.c, runtime/doc/terminal.txt, src/option.c

Patch 8.0.0788

Problem: MS-Windows: cannot build with terminal feature.

Solution: Move set_ref_in_term(). (Ozaki Kiichi)

Files: src/terminal.c

Patch 8.0.0789

Problem: When splitting a terminal window where the terminal follows the size of the window doesn't work.

Solution: Use the size of the smallest window. (Yasuhiro Matsumoto, closes #1885)

Files: src/terminal.c

Patch 8.0.0790

Problem: MSVC compiler warning for strncpy in libvterm.

Solution: Add a define to stop the warnings. (Mike Williams)

Files: src/Make_mvc.mak

Patch 8.0.0791

Problem: Terminal colors depend on the system.

Solution: Use the highlight color lookup tables.

Files: src/syntax.c, src/proto/syntax.pro, src/terminal.c

Patch 8.0.0792

Problem: Spell test leaves files behind.

Solution: Delete the files.

Files: src/testdir/test_spell.vim

Patch 8.0.0793

Problem: Using wrong terminal name for terminal window.

Solution: When '**term**' starts with "xterm" use it for \$TERM in a terminal window.

Files: src/os_unix.c

Patch 8.0.0794

Problem: The script to check translations fails if there is more than one NL in one line.

Solution: Count the number of NL characters. Make count() accept a string.

Files: src/po/check.vim, src/evalfunc.c, runtime/doc/eval.txt, src/testdir/test_functions.vim

Patch 8.0.0795

Problem: Terminal feature does not build with older MSVC.

Solution: Do not use stdint.h.

Files: src/libvterm/include/vterm.h

Patch 8.0.0796

Problem: No coverage on Travis with clang.

Solution: Use a specific coveralls version. (Ozaki Kiichi, closes #1888)

Files: .travis.yml

Patch 8.0.0797

Problem: Finished job in terminal window is not handled.

Solution: Add the scrollbar buffer. Use it to fill the buffer when the job has ended.

Files: src/terminal.c, src/screen.c, src/proto/terminal.pro, src/channel.c, src/os_unix.c, src/buffer.c

Patch 8.0.0798

Problem: No highlighting in a terminal window with a finished job.

Solution: Highlight the text.

Files: src/terminal.c, src/proto/terminal.pro, src/screen.c, src/undo.c

Patch 8.0.0799

Problem: Missing semicolon.

Solution: Add it.

Files: src/terminal.c

Patch 8.0.0800

Problem: Terminal window scrollbar contents is wrong.

Solution: Fix handling of multibyte characters (Yasuhiro Matsumoto) Handle empty lines correctly. (closes #1891)

Files: src/terminal.c

Patch 8.0.0801

Problem: The terminal window title sometimes still says "running" even though the job has finished.

Solution: Also consider the job finished when the channel has been closed.

Files: src/terminal.c

Patch 8.0.0802

Problem: After a job exits the last line in the terminal window does not get color attributes.

Solution: Fix off-by-one error.

Files: src/terminal.c

Patch 8.0.0803

Problem: Terminal window functions not yet implemented.
Solution: Implement several functions. Add a first test. (Yasuhiro Matsumoto, closes #1871)
Files: runtime/doc/eval.txt, src/Makefile, src/evalfunc.c, src/proto/evalfunc.pro, src/proto/terminal.pro, src/terminal.c, src/testdir/Make_all.mak, src/testdir/test_terminal.vim

Patch 8.0.0804

Problem: Running tests fails when stdin is /dev/null. (James McCoy)
Solution: Do not bail out from getting input if the --not-a-term argument was given. (closes #1460)
Files: src/eval.c, src/evalfunc.c

Patch 8.0.0805

Problem: GUI test fails with gnome2.
Solution: Set \$HOME to an existing directory.
Files: src/testdir/setup.vim, src/testdir/runtest.vim

Patch 8.0.0806

Problem: Tests may try to create XfakeHOME twice.
Solution: Avoid loading setup.vim twice.
Files: src/testdir/setup.vim

Patch 8.0.0807

Problem: Terminal window can't handle mouse buttons. (Hirohito Higashi)
Solution: Implement mouse buttons and many other keys. Ignore the ones that are not implemented.
Files: src/terminal.c

Patch 8.0.0808

Problem: Cannot build with terminal feature and DEBUG defined. (Christian Brabandt)
Solution: Use DEBUG_LOG3().
Files: src/libvterm/src/pen.c

Patch 8.0.0809

Problem: MS-Windows: tests hang.
Solution: Delete the XfakeHOME directory.
Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak

Patch 8.0.0810

Problem: MS-Windows: tests still hang.
Solution: Only create the XfakeHOME directory if it does not exist yet.
Files: src/testdir/setup.vim

Patch 8.0.0811

Problem: MS-Windows: test_expand_dllpath fails.
Solution: Change backslashes to forward slashes
Files: src/testdir/test_expand_dllpath.vim

Patch 8.0.0812

Problem: Terminal window colors shift when 'number' is set. (Nazri Ramliy)
Solution: Use vcol instead of col.

Files: src/screen.c

Patch 8.0.0813

Problem: Cannot use Vim commands in a terminal window while the job is running.

Solution: Implement Terminal Normal mode.

Files: src/terminal.c, src/proto/terminal.pro, src/main.c, src/screen.c, src/normal.c, src/option.c, runtime/doc/terminal.txt

Patch 8.0.0814 (after 8.0.0757)

Problem: File in Filelist does not exist.

Solution: Remove the line.

Files: Filelist

Patch 8.0.0815

Problem: Terminal window not correctly updated when **'statusline'** invokes `":sleep"`. (Nikolay Pavlov)

Solution: Clear `got_int`. Repeat redrawing when needed.

Files: src/terminal.c

Patch 8.0.0816

Problem: Crash when using invalid buffer number.

Solution: Check for NULL buffer. (Yasuhiro Matsumoto, closes #1899)

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0817

Problem: Cannot get the line of a terminal window at the cursor.

Solution: Make the row argument optional. (Yasuhiro Matsumoto, closes #1898)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/terminal.c

Patch 8.0.0818

Problem: Cannot get the cursor position of a terminal.

Solution: Add `term_getcursor()`.

Files: runtime/doc/eval.txt, src/evalfunc.c, src/terminal.c, src/proto/terminal.pro

Patch 8.0.0819

Problem: After changing current window the cursor position in the terminal window is not updated.

Solution: Set `w_wrow`, `w_wcol` and `w_valid`.

Files: src/terminal.c

Patch 8.0.0820

Problem: GUI: cursor in terminal window lags behind.

Solution: call `gui_update_cursor()` under different conditions. (Ozaki Kiichi, closes #1893)

Files: src/terminal.c

Patch 8.0.0821

Problem: Cannot get the title and status of a terminal window.

Solution: Implement `term_gettitle()` and `term_getstatus()`.

Files: src/evalfunc.c, src/terminal.c, src/proto/terminal.pro, runtime/doc/eval.txt

Patch 8.0.0822

Problem: Test_with_partial_callback is a tiny bit flaky.
Solution: Add it to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.0823

Problem: Cannot paste text into a terminal window.
Solution: Make **CTRL-W** " work.
Files: src/terminal.c

Patch 8.0.0824

Problem: In Terminal mode the cursor and screen gets redrawn when the job produces output.
Solution: Check for tl_terminal_mode. (partly by Yasuhiro Matsumoto, closes #1904)
Files: src/terminal.c

Patch 8.0.0825

Problem: Not easy to see that a window is a terminal window.
Solution: Add StatusLineTerm highlighting.
Files: src/option.c, src/vim.h, src/screen.c, src/syntax.c

Patch 8.0.0826

Problem: Cannot use text objects in Terminal mode.
Solution: Check for pending operator and Visual mode first. (Yasuhiro Matsumoto, closes #1906)
Files: src/normal.c

Patch 8.0.0827

Problem: Coverity: could leak pty file descriptor, theoretically.
Solution: If channel is NULL, free the file descriptors.
Files: src/os_unix.c

Patch 8.0.0828

Problem: Coverity: may dereference NULL pointer.
Solution: Bail out if calloc_state() returns NULL.
Files: src/regexp_nfa.c

Patch 8.0.0829

Problem: A job running in a terminal window cannot easily communicate with the Vim it is running in.
Solution: Pass v:servername in an environment variable. (closes #1908)
Files: src/os_unix.c

Patch 8.0.0830

Problem: Translating messages is not ideal.
Solution: Add a remark about obsolete messages. Use msgfmt in the check script. (Christian Brabandt)
Files: src/po/README.txt, src/po/check.vim

Patch 8.0.0831 (after 8.0.0791)

Problem: With 8 colors the bold attribute is not set properly.
Solution: Move setting HL_TABLE() out of lookup_color. (closes #1901)
Files: src/syntax.c, src/proto/syntax.pro, src/terminal.c

Patch 8.0.0832

Problem: Terminal function arguments are not consistent.
Solution: Use one-based instead of zero-based rows and cols. Use "." for the current row.
Files: src/terminal.c, runtime/doc/eval.txt

Patch 8.0.0833

Problem: Terminal test fails.
Solution: Update the row argument to one based.
Files: src/testdir/test_terminal.vim

Patch 8.0.0834

Problem: Can't build without the client-server feature.
Solution: Add #ifdef.
Files: src/os_unix.c

Patch 8.0.0835

Problem: Translations check with msgfmt does not work.
Solution: Add a space before the file name.
Files: src/po/check.vim

Patch 8.0.0836

Problem: When a terminal buffer is changed it can still be accidentally abandoned.
Solution: When making a change reset the **'buftype'** option.
Files: src/terminal.c, src/testdir/test_terminal.vim, src/option.c

Patch 8.0.0837

Problem: Signs can be drawn on top of console messages.
Solution: don't redraw at a prompt or when scrolled up. (Christian Brabandt, closes #1907)
Files: src/screen.c

Patch 8.0.0838

Problem: Buffer hangs around when terminal window is closed.
Solution: When the job has ended wipe out a terminal buffer when the window is closed.
Files: src/buffer.c, src/terminal.c, src/proto/terminal.pro, src/testdir/test_terminal.vim

Patch 8.0.0839

Problem: Cannot kill a job in a terminal with **CTRL-C**.
Solution: Set the controlling tty and send SIGINT. (closes #1910)
Files: src/os_unix.c, src/terminal.c, src/proto/os_unix.pro

Patch 8.0.0840

Problem: MS-Windows: fopen() and open() prototypes do not match the ones in the system header file. Can't build without FEAT_MBYTE.
Solution: Add "const". Move macro to after including protoo.h.
Files: src/os_win32.c, src/proto/os_win32.pro, src/macros.h, src/vim.h

Patch 8.0.0841

Problem: term_getline() may cause a crash.

Solution: Check that the row is valid. (Hirohito Higashi)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0842

Problem: Using slave pty after closing it.
Solution: Do the ioctl() before dup'ing it.
Files: src/os_unix.c

Patch 8.0.0843

Problem: MS-Windows: compiler warning for signed/unsigned.
Solution: Add type cast. (Yasuhiro Matsumoto, closes #1912)
Files: src/terminal.c

Patch 8.0.0844

Problem: Wrong function prototype because of missing static.
Solution: Add "static".
Files: src/os_win32.c, src/proto/os_win32.pro

Patch 8.0.0845

Problem: MS-Windows: missing semicolon in terminal code.
Solution: Add it. (Naruhiko Nishino, closes #1923)
Files: src/terminal.c

Patch 8.0.0846

Problem: Cannot get the name of the pty of a job.
Solution: Add the "tty" entry to the job info. (Ozaki Kiichi, closes #1920)
Add the term_gettty() function.
Files: runtime/doc/eval.txt, src/channel.c, src/os_unix.c, src/structs.h,
src/terminal.c, src/proto/terminal.pro, src/evalfunc.c,
src/testdir/test_terminal.vim

Patch 8.0.0847

Problem: :argadd without argument can't handle space in file name. (Harm te Hennepe)
Solution: Escape the space. (Yasuhiro Matsumoto, closes #1917)
Files: src/ex_cmds2.c, src/proto/ex_cmds2.pro,
src/testdir/test_arglist.vim

Patch 8.0.0848

Problem: Using multiple ch_log functions is clumsy.
Solution: Use variable arguments. (Ozaki Kiichi, closes #1919)
Files: src/channel.c, src/message.c, src/proto/channel.pro,
src/terminal.c

Patch 8.0.0849

Problem: Crash when job exit callback wipes the terminal.
Solution: Check for b_term to be NULL. (Yasuhiro Matsumoto, closes #1922)
Implement options for term_start() to be able to test.
Make term_wait() more reliable.
Files: src/terminal.c, src/testdir/test_terminal.vim, src/channel.c

Patch 8.0.0850

Problem: MS-Windows: Depending on the console encoding, an error message that is given during startup may be broken.

Solution: Convert the message to the console codepage. (Yasuhiro Matsumoto, closes #1927)
Files: src/message.c

Patch 8.0.0851

Problem: **'smartindent'** is used even when **'indentexpr'** is set.
Solution: Ignore **'smartindent'** when **'indentexpr'** is set. (Hirohito Higashi)
Files: src/misc1.c, src/testdir/test_smartindent.vim

Patch 8.0.0852 (after 8.0.0850)

Problem: MS-Windows: possible crash when giving a message on startup.
Solution: Initialize length. (Yasuhiro Matsumoto, closes #1931)
Files: src/message.c

Patch 8.0.0853

Problem: Crash when running terminal with unknown command.
Solution: Check "term" not to be NULL. (Yasuhiro Matsumoto, closes #1932)
Files: src/terminal.c

Patch 8.0.0854

Problem: No redraw after terminal was closed.
Solution: Set typebuf_was_filled. (Yasuhiro Matsumoto, closes #1925, closes #1924) Add function to check for messages even when input is available.
Files: src/terminal.c, src/os_unix.c, src/proto/os_unix.pro, src/os_win32.c, src/proto/os_win32.pro, src/os_mswin.c

Patch 8.0.0855

Problem: MS-Windows: can't get tty name of terminal.
Solution: Use the winpty process number. (Yasuhiro Matsumoto, closes #1929)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0856

Problem: MS-Windows: terminal job doesn't take options.
Solution: Call job_set_options(). (Yasuhiro Matsumoto)
Files: src/terminal.c

Patch 8.0.0857

Problem: Terminal test fails on MS-Windows.
Solution: Sleep a fraction of a second.
Files: src/testdir/test_terminal.vim

Patch 8.0.0858

Problem: Can exit while a terminal is still running a job.
Solution: Consider a buffer with a running job like a changed file.
Files: src/undo.c, src/terminal.c, src/option.h, src/buffer.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/normal.c, src/window.c, src/testdir/test_terminal.vim

Patch 8.0.0859

Problem: NULL pointer access when term_free_vterm called twice.
Solution: Return when tl_vterm is NULL. (Yasuhiro Matsumoto, closes #1934)
Files: src/terminal.c

Patch 8.0.0860

Problem: There may be side effects when a channel appends to a buffer that is not the current buffer.

Solution: Properly switch to another buffer before appending. (Yasuhiro Matsumoto, closes #1926, closes #1937)

Files: src/channel.c, src/buffer.c, src/proto/buffer.pro, src/if_py_both.h

Patch 8.0.0861

Problem: Still many old style tests.

Solution: Convert several tests to new style. (Yegappan Lakshmanan)

Files: src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/main.aap, src/testdir/test104.in, src/testdir/test104.ok, src/testdir/test22.in, src/testdir/test22.ok, src/testdir/test77.in, src/testdir/test77.ok, src/testdir/test84.in, src/testdir/test84.ok, src/testdir/test9.in, src/testdir/test9.ok, src/testdir/test98.in, src/testdir/test98.ok, src/testdir/test_autocmd.vim, src/testdir/test_curswant.vim, src/testdir/test_file_size.vim, src/testdir/test_let.vim, src/testdir/test_lineending.vim, src/testdir/test_scrollbind.vim, src/Makefile

Patch 8.0.0862 (after 8.0.0862)

Problem: File size test fails on MS-Windows.

Solution: Set fileformat after opening new buffer. Strip CR.

Files: src/testdir/test_file_size.vim

Patch 8.0.0863

Problem: A remote command starting with **CTRL-\ CTRL-N** does not work in the terminal window. (Christian J. Robinson)

Solution: Use **CTRL-\ CTRL-N** as a prefix or a Normal mode command.

Files: src/terminal.c, runtime/doc/terminal.txt

Patch 8.0.0864

Problem: Cannot specify the name of a terminal.

Solution: Add the "term_name" option. (Yasuhiro Matsumoto, closes #1936)

Files: src/channel.c, src/structs.h, src/terminal.c, runtime/doc/eval.txt

Patch 8.0.0865

Problem: Cannot build with channel but without terminal feature.

Solution: Add #ifdef

Files: src/channel.c

Patch 8.0.0866

Problem: Solaris also doesn't have MIN and MAX.

Solution: Define MIN and MAX whenever they are not defined. (Ozaki Kiichi, closes #1939)

Files: src/terminal.c

Patch 8.0.0867

Problem: When using a job or channel value as a dict value, when turning it into a string the quotes are missing.

Solution: Add quotes to the job and channel values. (Yasuhiro Matsumoto,

closes #1930)
Files: src/list.c, src/eval.c, src/testdir/test_terminal.vim

Patch 8.0.0868

Problem: Cannot specify the terminal size on the command line.
Solution: Use the address range for the terminal size. (Yasuhiro Matsumoto, closes #1941)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0869

Problem: Job output is sometimes not displayed in a terminal.
Solution: Flush output before closing the channel.
Files: src/channel.c, src/terminal.c

Patch 8.0.0870

Problem: Mouse escape codes sent to terminal unintentionally.
Solution: Fix libvterm to send mouse codes only when enabled.
Files: src/terminal.c, src/libvterm/src/mouse.c

Patch 8.0.0871

Problem: The status line for a terminal window always has "[+]".
Solution: Do make the status line include "[+]" for a terminal window.
Files: src/screen.c

Patch 8.0.0872

Problem: Using mouse scroll while a terminal window has focus and the mouse pointer is on another window does not work. Same for focus in a non-terminal window and the mouse pointer is over a terminal window.
Solution: Send the scroll action to the right window.
Files: src/terminal.c, src/normal.c, src/proto/terminal.pro

Patch 8.0.0873

Problem: In a terminal window cannot use **CTRL-\ CTRL-N** to start Visual mode.
Solution: After **CTRL-\ CTRL-N** enter Terminal-Normal mode for one command.
Files: src/main.c, src/terminal.c, src/proto/terminal.pro

Patch 8.0.0874 (after 8.0.0873)

Problem: Can't build with terminal feature.
Solution: Include change to term_use_loop(). (Dominique Pelle)
Files: src/normal.c

Patch 8.0.0875

Problem: Crash with weird command sequence. (Dominique Pelle)
Solution: Use vim_snprintf() instead of STRCPY().
Files: src/misc1.c

Patch 8.0.0876

Problem: MS-Windows: Backslashes and wildcards in backticks don't work.
Solution: Do not handle backslashes inside backticks in the wrong place. (Yasuhiro Matsumoto, closes #1942)
Files: src/os_mswin.c, src/os_win32.c

Patch 8.0.0877

Problem: Using CTRL-\ **CTRL-N** in terminal is inconsistent.
Solution: Stay in Normal mode.
Files: src/terminal.c, src/proto/terminal.pro, src/main.c, src/normal.c,
src/option.c

Patch 8.0.0878

Problem: No completion for :mapclear.
Solution: Add completion (Nobuhiro Takasaki et al. closes #1943)
Files: runtime/doc/eval.txt, runtime/doc/map.txt, src/ex_docmd.c,
src/ex_getln.c, src/proto/ex_docmd.pro,
src/testdir/test_cmdline.vim, src/vim.h

Patch 8.0.0879

Problem: Crash when shifting with huge number.
Solution: Check for overflow. (Dominique Pelle, closes #1945)
Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.0.0880

Problem: Travis uses an old Ubuntu version.
Solution: Switch from precise to trusty. (Ken Takata, closes #1897)
Files: .travis.yml, Filelist, src/testdir/if_ver-1.vim,
src/testdir/if_ver-2.vim, src/testdir/lsan-suppress.txt

Patch 8.0.0881

Problem: win32.mak no longer included in Windows SDK.
Solution: Do not include win32.mak. (Ken Takata)
Files: src/GvimExt/Makefile, src/Make_mvc.mak

Patch 8.0.0882

Problem: term_scrape() and term_getline() require two arguments but it is not enforced.
Solution: Correct minimal number of arguments. (Hirohito Higashi) Update documentation. (Ken Takata)
Files: src/evalfunc.c, runtime/doc/eval.txt

Patch 8.0.0883

Problem: Invalid memory access with nonsensical script.
Solution: Check "dstlen" being positive. (Dominique Pelle)
Files: src/misc1.c

Patch 8.0.0884

Problem: Can't specify the wait time for term_wait().
Solution: Add an optional second argument.
Files: src/evalfunc.c, src/terminal.c, runtime/doc/eval.txt

Patch 8.0.0885

Problem: Terminal window scrollbar is stored inefficiently.
Solution: Store the text in the Vim buffer.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0886

Problem: Crash when using ":term ls".
Solution: Fix line number computation. Add a test for this.

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0887

Problem: Can create a logfile in the sandbox.

Solution: Disable ch_logfile() in the sandbox. (Yasuhiro Matsumoto)

Files: src/evalfunc.c

Patch 8.0.0888

Problem: Compiler warnings with 64 bit build.

Solution: Add type cast of change the type. (Mike Williams)

Files: src/message.c, src/os_mswin.c, src/os_win32.c

Patch 8.0.0889

Problem: Gcc gives warnings for uninitialized variables. (Tony Mechelynck)

Solution: Initialize variables even though they are not used.

Files: src/terminal.c

Patch 8.0.0890

Problem: Still many old style tests.

Solution: Convert several tests to new style. (Yegappan Lakshmanan)

Files: src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test103.in, src/testdir/test103.ok,
src/testdir/test107.in, src/testdir/test107.ok,
src/testdir/test51.in, src/testdir/test51.ok,
src/testdir/test91.in, src/testdir/test91.ok,
src/testdir/test_getvar.vim, src/testdir/test_highlight.vim,
src/testdir/test_visual.vim, src/testdir/test_window_cmd.vim,
src/Makefile

Patch 8.0.0891

Problem: Uninitialized memory use with empty line in terminal.

Solution: Initialize growarray earlier. (Yasuhiro Matsumoto, closes #1949)

Files: src/terminal.c

Patch 8.0.0892

Problem: When opening a terminal the pty size doesn't always match.

Solution: Update the pty size after opening the terminal. (Ken Takata)

Files: src/terminal.c

Patch 8.0.0893

Problem: Cannot get the scroll count of a terminal window.

Solution: Add term_getscrolled().

Files: src/terminal.c, src/proto/terminal.pro, src/evalfunc.c,
runtime/doc/eval.txt, src/testdir/test_terminal.vim

Patch 8.0.0894

Problem: There is no test for runtime filetype detection.

Solution: Test a list of filetypes from patterns.

Files: src/testdir/test_filetype.vim, runtime/filetype.vim

Patch 8.0.0895 (after 8.0.0894)

Problem: Filetype test fails on MS-Windows.

Solution: Fix file names.

Files: src/testdir/test_filetype.vim

Patch 8.0.0896

Problem: Cannot automatically close a terminal window when the job ends.
Solution: Add the ++close argument to :term. Add the term_finish option to term_start(). (Yasuhiro Matsumoto, closes #1950) Also add ++open.
Files: runtime/doc/eval.txt, runtime/doc/terminal.txt, src/channel.c, src/structs.h, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0897 (after 8.0.0896)

Problem: Wrong error message for invalid term_finish value
Solution: Pass the right argument to emsg().
Files: src/channel.c

Patch 8.0.0898

Problem: Can't use the alternate screen in a terminal window.
Solution: Initialize the alternate screen. (Yasuhiro Matsumoto, closes #1957) Add term_getaltscreen().
Files: src/libvterm/include/vterm.h, src/terminal.c, src/proto/terminal.pro, src/evalfunc.c, runtime/doc/eval.txt

Patch 8.0.0899

Problem: Function name mch_stop_job() is confusing.
Solution: Rename to mch_signal_job().
Files: src/channel.c, src/os_unix.c, src/proto/os_unix.pro, src/os_win32.c, src/proto/os_win32.pro, src/terminal.c

Patch 8.0.0900

Problem: :tab options doesn't open a new tab page. (Aviany)
Solution: Support the :tab modifier. (closes #1960)
Files: src/ex_cmds2.c, runtime/optwin.vim

Patch 8.0.0901

Problem: Asan suppress file missing from distribution.
Solution: Add the file.
Files: Filelist

Patch 8.0.0902

Problem: Cannot specify directory or environment for a job.
Solution: Add the "cwd" and "env" arguments to job options. (Yasuhiro Matsumoto, closes #1160)
Files: runtime/doc/channel.txt, src/channel.c, src/terminal.c, src/os_unix.c, src/os_win32.c, src/structs.h, src/testdir/test_channel.vim, src/testdir/test_terminal.vim

Patch 8.0.0903 (after 8.0.0902)

Problem: Early return from test function.
Solution: Remove the return.
Files: src/testdir/test_terminal.vim

Patch 8.0.0904

Problem: Cannot set a location list from text.
Solution: Add the "text" argument to setqflist(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c,

src/testdir/test_quickfix.vim

Patch 8.0.0905

Problem: MS-Windows: broken multibyte characters in the console.
Solution: Restore all regions of the console buffer. (Ken Takata)
Files: src/os_win32.c

Patch 8.0.0906

Problem: Don't recognize Couchbase files.
Solution: Add filetype detection. (Eugene Ciurana, closes #1951)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.0.0907

Problem: With cp932 font names might be misinterpreted.
Solution: Do not see "_" as a space when it is the second byte of a double
byte character. (Ken Takata)
Files: src/os_win32.c

Patch 8.0.0908

Problem: Cannot set terminal size with options.
Solution: Add "term_rows", "term_cols" and "vertical".
Files: src/terminal.c, runtime/doc/eval.txt, src/channel.c,
src/proto/channel.pro, src/structs.h, src/evalfunc.c,
src/testdir/test_terminal.vim

Patch 8.0.0909

Problem: Channel test fails.
Solution: Allow for "cwd" and "env" arguments.
Files: src/channel.c

Patch 8.0.0910

Problem: Cannot create a terminal in the current window.
Solution: Add option "curwin" and ++curwin.
Files: src/terminal.c, runtime/doc/eval.txt, src/channel.c,
src/structs.h, src/ex_cmds.h, src/testdir/test_terminal.vim

Patch 8.0.0911

Problem: Terminal test takes too long.
Solution: Instead of "sleep 1" use a Python program to briefly sleep.
Files: src/testdir/test_terminal.vim, src/testdir/test_short_sleep.py

Patch 8.0.0912

Problem: Cannot run a job in a hidden terminal.
Solution: Add option "hidden" and ++hidden.
Files: src/terminal.c, src/structs.h, src/channel.c, src/fileio.c,
runtime/doc/terminal.txt, src/testdir/test_terminal.vim

Patch 8.0.0913

Problem: MS-Windows: **CTRL-C** kills shell in terminal window instead of the
command running in the shell.
Solution: Make **CTRL-C** only send a CTRL_C_EVENT and have **CTRL-BREAK** kill the
job. (partly by Yasuhiro Matsumoto, closes #1962)
Files: src/os_win32.c, src/gui_w32.c, src/terminal.c, src/globals.h

Patch 8.0.0914

Problem: Highlight attributes are always combined.
Solution: Add the **'nocombine'** value to replace attributes instead of combining them. (scauligi, closes #1963)
Files: runtime/doc/syntax.txt, src/syntax.c, src/vim.h

Patch 8.0.0915

Problem: Wrong initialisation of global.
Solution: Use INIT().
Files: src/globals.h

Patch 8.0.0916

Problem: Cannot specify properties of window for when opening a window for a finished terminal job.
Solution: Add "term_opencmd".
Files: src/channel.c, src/structs.h, src/terminal.c, runtime/doc/eval.txt, src/testdir/test_terminal.vim

Patch 8.0.0917

Problem: MS-Windows:CTRL-C handling in terminal window is wrong
Solution: Pass **CTRL-C** as a key. Turn **CTRL-BREAK** into a key stroke. (Yasuhiro Matsumoto, closes #1965)
Files: src/os_win32.c, src/terminal.c

Patch 8.0.0918

Problem: Cannot get terminal window cursor shape or attributes.
Solution: Support cursor shape, attributes and color.
Files: src/terminal.c, runtime/doc/eval.txt, src/libvterm/include/vterm.h, src/libvterm/src/state.c, src/libvterm/src/vterm.c, src/feature.h, src/ui.c, src/proto/ui.pro, src/term.c, src/proto/term.pro, src/option.c, src/term.h

Patch 8.0.0919

Problem: Cursor color isn't set on startup.
Solution: Initialize showing_mode to invalid value.
Files: src/term.c

Patch 8.0.0920

Problem: The cursor shape is wrong after switch back from an alternate screen in a terminal window. (Marius Gedminas)
Solution: Change bitfield to unsigned. Set flag that cursor shape was set.
Files: src/terminal.c, src/libvterm/src/vterm_internal.h

Patch 8.0.0921

Problem: Terminal window cursor shape not supported in the GUI.
Solution: Use the terminal window cursor shape in the GUI.
Files: src/terminal.c, src/proto/terminal.pro, src/gui.c, src/syntax.c, src/proto/syntax.pro

Patch 8.0.0922

Problem: Quickfix list always added after current one.
Solution: Make it possible to add a quickfix list after the last one. (Yegappan Lakshmanan)

Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.0923

Problem: Crash in GUI when terminal job exits. (Kazunobu Kuriyama)
Solution: reset in_terminal_loop when a terminal is freed.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.0924

Problem: Terminal window not updated after using term_sendkeys().
Solution: Call redraw_after_callback().
Files: src/terminal.c

Patch 8.0.0925

Problem: MS-Windows GUI: channel I/O not handled right away.
Solution: Don't call process_message() unless a message is available.
(Yasuhiro Matsumoto, closes #1969)
Files: src/gui_w32.c

Patch 8.0.0926

Problem: When job in terminal window ends topline may be wrong.
Solution: When the job ends adjust topline so that the active part of the
terminal is displayed.
Files: src/terminal.c

Patch 8.0.0927

Problem: If a terminal job sends a blank title "running" is not shown.
Solution: When the title is blank make it empty.
Files: src/terminal.c

Patch 8.0.0928

Problem: MS-Windows: passing arglist to job has escaping problems.
Solution: Improve escaping. (Yasuhiro Matsumoto, closes #1954)
Files: src/testdir/test_channel.vim, src/testdir/test_terminal.vim,
src/channel.c, src/proto/channel.pro, src/terminal.c

Patch 8.0.0929

Problem: :term without argument does not work.
Solution: Use shell for empty command. (Yasuhiro Matsumoto, closes #1970)
Files: src/terminal.c

Patch 8.0.0930

Problem: Terminal buffers are stored in the viminfo file while they can't
be useful.
Solution: Skip terminal buffers for file marks and buffer list
Files: src/buffer.c, src/mark.c

Patch 8.0.0931

Problem: getwininfo() does not indicate a terminal window.
Solution: Add "terminal" to the dictionary.
Files: runtime/doc/eval.txt, src/evalfunc.c

Patch 8.0.0932

Problem: Terminal may not use right characters for BS and Enter.

Solution: Get the characters from the tty.
Files: src/os_unix.c, src/proto/os_unix.pro, src/terminal.c

Patch 8.0.0933

Problem: Terminal test tries to start GUI when it's not possible.
Solution: Check if the GUI can run. (James McCoy, closes #1971)
Files: src/testdir/shared.vim, src/testdir/test_terminal.vim,
src/testdir/test_gui.vim, src/testdir/test_gui_init.vim

Patch 8.0.0934 (after 8.0.0932)

Problem: Change to struts.h missing in patch.
Solution: Include adding ttyinfo_T.
Files: src/structs.h

Patch 8.0.0935

Problem: Cannot recognize a terminal buffer in :ls output.
Solution: Use R for a running job and F for a finished job.
Files: src/buffer.c

Patch 8.0.0936

Problem: mode() returns wrong value for a terminal window.
Solution: Return 't' when typed keys go to a job.
Files: src/evalfunc.c, src/testdir/test_terminal.vim

Patch 8.0.0937

Problem: User highlight groups are not adjusted for StatusLineTerm.
Solution: Combine attributes like for StatusLineNC.
Files: src/syntax.c, src/globals.h, src/screen.c

Patch 8.0.0938

Problem: Scrolling in terminal window is inefficient.
Solution: Use win_del_lines().
Files: src/terminal.c

Patch 8.0.0939

Problem: Test_terminal_env is flaky. (James McCoy)
Solution: Use WaitFor() instead of term_wait().
Files: src/testdir/test_terminal.vim

Patch 8.0.0940

Problem: Test_terminal_scrape_multibyte is flaky. (James McCoy)
Solution: Use WaitFor() instead of term_wait().
Files: src/testdir/test_terminal.vim

Patch 8.0.0941

Problem: Existing color schemes don't work well with StatusLineTerm.
Solution: Don't use "reverse", use fg and bg colors. Also add
StatusLineTermNC.
Files: src/syntax.c, src/vim.h, src/screen.c, src/globals.h, src/option.c

Patch 8.0.0942

Problem: Using freed memory with ":terminal" if an autocommand changes
'shell' when splitting the window. (Marius Gedminas)
Solution: Make a copy of 'shell'. (closes #1974)

Files: src/terminal.c

Patch 8.0.0943

Problem: Test_terminal_scrape_multibyte fails if the codepage is not utf-8.
Solution: Start "cmd" with the utf-8 codepage. (micbou, closes #1975)
Files: src/testdir/test_terminal.vim

Patch 8.0.0944

Problem: Test_profile is a little bit flaky.
Solution: Accept a match when self and total time are the same. (James McCoy, closes #1972)
Files: src/testdir/test_profile.vim

Patch 8.0.0945

Problem: 64-bit compiler warnings.
Solution: Use "size_t" instead of "int". (Mike Williams)
Files: src/os_win32.c

Patch 8.0.0946

Problem: Using PATH_MAX does not work well on some systems.
Solution: use MAXPATHL instead. (James McCoy, closes #1973)
Files: src/main.c

Patch 8.0.0947

Problem: When in Insert mode and using **CTRL-O CTRL-W CTRL-W** to move to a terminal window, get in a weird Insert mode.
Solution: Don't go to Insert mode in a terminal window. (closes #1977)
Files: src/normal.c

Patch 8.0.0948

Problem: Crash if timer closes window while dragging status line.
Solution: Check if the window still exists. (Yasuhiro Matsumoto, closes #1979)
Files: src/edit.c, src/evalfunc.c, src/gui.c, src/normal.c, src/ui.c

Patch 8.0.0949

Problem: winpty.dll name is fixed.
Solution: Add the **'winptydll'** option. Make the default name depend on whether it is a 32-bit or 64-bit build. (idea by Yasuhiro Matsumoto, closes #1978)
Files: src/option.c, src/option.h, src/terminal.c, runtime/doc/options.txt

Patch 8.0.0950

Problem: MS-Windows: wrong #ifdef, compiler warnings for signed/unsigned.
Solution: Change variable type. Change TERMINAL to FEAT_TERMINAL.
Files: src/os_win32.c, src/option.h

Patch 8.0.0951

Problem: Another wrong #ifdef.
Solution: Change TERMINAL to FEAT_TERMINAL. (closes #1981)
Files: src/option.c

Patch 8.0.0952

Problem: MS-Windows: has('terminal') does not check existence of dll file.
Solution: Check if the winpty dll file can be loaded. (Ken Takata)
Files: src/evalfunc.c, src/proto/terminal.pro, src/terminal.c

Patch 8.0.0953

Problem: Get "no write since last change" error in terminal window.
Solution: Use another message when closing a terminal window. Make ":quit!" also end the job.
Files: src/globals.h, src/buffer.c, src/proto/buffer.pro, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/quickfix.c, src/terminal.c

Patch 8.0.0954

Problem: /proc/self/exe might be a relative path.
Solution: Make the path a full path. (James McCoy, closes #1983)
Files: src/main.c

Patch 8.0.0955

Problem: Test_existent_file() fails on some file systems.
Solution: Run the test again with a sleep when the test fails without a sleep. (James McCoy, closes #1984)
Files: src/testdir/test_stat.vim

Patch 8.0.0956

Problem: Scrolling in a terminal hwindow as flicker when the Normal background differs from the terminal window background.
Solution: Set the attribute to clear with.
Files: src/terminal.c, src/screen.c, src/proto/screen.pro, src/message.c, src/move.c

Patch 8.0.0957

Problem: When term_sendkeys() sends many keys it may get stuck in writing to the job.
Solution: Make the write non-blocking, buffer keys to be sent.
Files: src/terminal.c, src/channel.c, src/proto/channel.pro, src/structs.h src/testdir/test_terminal.vim

Patch 8.0.0958

Problem: The terminal test fails on MS-Windows when compiled with the terminal feature but the winpty DLL is missing.
Solution: Check if the terminal feature works. (Ken Takata)
Files: src/testdir/test_terminal.vim

Patch 8.0.0959

Problem: Build failure on MS-Windows.
Solution: Use ioctlsocket() instead of fcntl().
Files: src/channel.c

Patch 8.0.0960

Problem: Job in terminal does not get **CTRL-C**, we send a SIGINT instead.
Solution: Don't call may_send_sigint() on **CTRL-C**. Make **CTRL-W CTRL-C** end the job.
Files: src/terminal.c, runtime/doc/terminal.txt

Patch 8.0.0961

Problem: The script to build the installer does not include winpty.
Solution: Add winpty32.dll and winpty-agent.exe like diff.exe
Files: nsis/gvim.nsi

Patch 8.0.0962

Problem: Crash with virtualedit and joining lines. (Joshua T Corbin, Neovim #6726)
Solution: When using a mark check that coladd is valid.
Files: src/normal.c, src/misc2.c, src/Makefile, src/testdir/test_virtualedit.vim, src/testdir/test_alot.vim

Patch 8.0.0963

Problem: Terminal test fails on macOS. (chdiza)
Solution: Wait for the shell to echo the characters. (closes #1991)
Files: src/testdir/test_terminal.vim

Patch 8.0.0964

Problem: Channel write buffer does not work with poll().
Solution: Use the same mechanism as with select().
Files: src/channel.c

Patch 8.0.0965

Problem: The cursor shape is not reset after it was changed in a terminal.
Solution: Request the original cursor shape and restore it. Add t_RS.
Do not add t_SH for now, it does not work properly.
Files: src/term.c, src/term.h, src/option.c, src/terminal.c

Patch 8.0.0966 (after 8.0.0965)

Problem: Build failure without terminal feature.
Solution: Move #endif.
Files: src/term.c

Patch 8.0.0967

Problem: Using a terminal may cause the cursor to blink.
Solution: Do not set t_vs, since we cannot restore the old blink state.
Files: src/term.c

Patch 8.0.0968

Problem: Crash when switching terminal modes. (Nikolai Pavlov)
Solution: Check that there are scrollbar lines.
Files: src/terminal.c

Patch 8.0.0969

Problem: Coverity warning for unused return value.
Solution: Add (void) to avoid the warning.
Files: src/channel.c

Patch 8.0.0970

Problem: if there is no StatusLine highlighting and there is StatusLineNC or StatusLineTermNC highlighting then an invalid highlight id is passed to combine_stl_hlt(). (Coverity)
Solution: Check id_S to be -1 instead of zero.
Files: src/syntax.c

Patch 8.0.0971

Problem: **'winptydll'** missing from :options.
Solution: Add the entry.
Files: runtime/optwin.vim

Patch 8.0.0972

Problem: Compiler warnings for unused variables. (Tony Mechelynck)
Solution: Add #ifdefs.
Files: src/term.c

Patch 8.0.0973

Problem: initial info about blinking cursor is wrong
Solution: Invert the blink flag. Add t_VS to stop a blinking cursor.
Files: src/term.c, src/proto/term.pro, src/term.h, src/option.c, src/terminal.c

Patch 8.0.0974

Problem: Resetting a string option does not trigger OptionSet. (Rick Howe)
Solution: Set the origval.
Files: src/option.c, src/testdir/test_autocmd.vim

Patch 8.0.0975

Problem: Using freed memory when setting **'backspace'**.
Solution: When changing oldval also change origval.
Files: src/option.c

Patch 8.0.0976

Problem: Cannot send lines to a terminal job.
Solution: Make **[range]**terminal send selected lines to the job.
Use ++rows and ++cols for the terminal size.
Files: src/ex_cmds.h, src/terminal.c, src/os_unix.c, src/testdir/test_terminal.vim

Patch 8.0.0977

Problem: Cannot send lines to a terminal job on MS-Windows.
Solution: Set jv_in_buf. Command doesn't get EOF yet though.
Files: src/terminal.c

Patch 8.0.0978

Problem: Writing to terminal job is not tested.
Solution: Add a test.
Files: src/testdir/test_terminal.vim

Patch 8.0.0979

Problem: Terminal noblock test fails on MS-Windows. (Christian Brabandt)
Solution: Ignore empty line below "done".
Files: src/testdir/test_terminal.vim

Patch 8.0.0980

Problem: Coverity warning for failing to open /dev/null.
Solution: When /dev/null can't be opened exit the child.
Files: src/os_unix.c

Patch 8.0.0981

Problem: Cursor in terminal window blinks by default, while in a real xterm it does not blink, unless the `-bc` argument is used.
Solution: Do not use a blinking cursor by default.
Files: `src/terminal.c`

Patch 8.0.0982

Problem: When **'encoding'** is set to a multibyte encoding other than utf-8 the characters from their terminal are messed up.
Solution: Convert displayed text from utf-8 to **'encoding'** for MS-Windows. (Yasuhiro Matsumoto, close #2000)
Files: `src/terminal.c`

Patch 8.0.0983

Problem: Unnecessary check for NULL pointer.
Solution: Remove the NULL check in `dialog_changed()`, it already happens in `dialog_msg()`. (Ken Takata)
Files: `src/ex_cmds2.c`

Patch 8.0.0984

Problem: Terminal blinking cursor not correct in the GUI.
Solution: Set `blinkoff` correctly. Also make the cursor blink on MS-Windows by default. (Ken Takata)
Files: `src/terminal.c`

Patch 8.0.0985

Problem: Libvterm has its own idea of character width.
Solution: Use the Vim functions for character width and composing to avoid a mismatch. (idea by Yasuhiro Matsumoto)
Files: `src/Makefile`, `src/libvterm/src/unicode.c`, `src/mbyte.c`, `src/proto/mbyte.pro`, `src/Make_cyg_ming.mak`, `src/Make_mvc.mak`

Patch 8.0.0986

Problem: Terminal feature always requires multibyte feature.
Solution: Remove `#ifdef FEAT_MBYTE`, disable terminal without multibyte.
Files: `src/terminal.c`, `src/feature.h`

Patch 8.0.0987

Problem: `terminal`: second byte of double-byte char wrong
Solution: Set the second byte to NUL only for utf-8 and non-multibyte.
Files: `src/terminal.c`

Patch 8.0.0988

Problem: Warning from Covscan about using NULL pointer.
Solution: Add extra check for NULL. (zdohnal)
Files: `src/fileio.c`, `src/undo.c`

Patch 8.0.0989

Problem: ActiveTcl dll name has changed in 8.6.6.
Solution: Adjust the makefile. (Ken Takata)
Files: `src/INSTALLpc.txt`, `src/Make_cyg_ming.mak`, `src/Make_mvc.mak`

Patch 8.0.0990

Problem: When **'encoding'** is a double-byte encoding, pasting a register into a terminal ends up with the wrong characters.

Solution: Convert from '**encoding**' to utf-8. (Yasuhiro Matsumoto, closes #2007)
Files: src/terminal.c

Patch 8.0.0991

Problem: Using wrong character conversion for DBCS.
Solution: Use utf_char2bytes instead of mb_char2bytes. (Yasuhiro Matsumoto, closes #2012)
Files: src/terminal.c

Patch 8.0.0992

Problem: Terminal title is wrong when '**encoding**' is DBCS.
Solution: Convert the title from DBCS to utf-8. (Yasuhiro Matsumoto, closes #2009)
Files: src/terminal.c

Patch 8.0.0993

Problem: Sometimes an xterm sends an extra **CTRL-X** after the response for the background color. Related to t_RS.
Solution: Check for the **CTRL-X** after the terminating 0x7.
Files: src/term.c

Patch 8.0.0994

Problem: MS-Windows: cursor in terminal blinks even though the blinking cursor was disabled on the system.
Solution: Use GetCaretBlinkTime(). (Ken Takata)
Files: src/terminal.c

Patch 8.0.0995

Problem: Terminal tests fail on Mac.
Solution: Add workaround: sleep a moment in between sending keys.
Files: src/testdir/test_terminal.vim

Patch 8.0.0996

Problem: Mac: t_RS is echoed on the screen in Terminal.app. Even though \$TERM is set to "xterm-256colors" it cannot handle this xterm escape sequence.
Solution: Recognize Terminal.app from the termresponse and skip sending t_RS if it looks like Terminal.app.
Files: src/term.c

Patch 8.0.0997 (after 8.0.0996)

Problem: Libvterm and Terminal.app not recognized from termresponse.
Solution: Adjust string compare.
Files: src/term.c

Patch 8.0.0998

Problem: Strange error when using K while only spaces are selected. (Christian J. Robinson)
Solution: Check for blank argument.
Files: src/normal.c, src/testdir/test_help.vim

Patch 8.0.0999

Problem: Indenting raw C++ strings is wrong.

Solution: Add special handling of raw strings. (Christian Brabandt)
Files: src/misc1.c, src/testdir/test_cindent.vim

Patch 8.0.1000

Problem: Cannot open a terminal without running a job in it.
Solution: Make ":terminal NONE" open a terminal with a pty.
Files: src/terminal.c, src/os_unix.c, src/proto/os_unix.pro,
src/channel.c, src/proto/channel.pro, src/structs.h,
src/testdir/test_terminal.c, src/misc2.c, src/gui_gtk_x11.c

Patch 8.0.1001

Problem: Setting '**encoding**' makes '**printhead**' invalid.
Solution: Do not translate the default value of '**printhead**'. (Yasuhiro Matsumoto, closes #2026)
Files: src/option.c

Patch 8.0.1002

Problem: Unnecessarily updating screen after timer callback.
Solution: Check if calling the timer sets must_redraw.
Files: src/ex_cmds2.c, src/channel.c, src/screen.c, src/proto/screen.pro,
src/terminal.c

Patch 8.0.1003

Problem: 64 bit compiler warning
Solution: Add type cast. (Mike Williams)
Files: src/channel.c

Patch 8.0.1004

Problem: matchstrpos() without a match returns too many items.
Solution: Also remove the second item when the position is beyond the end of the string. (Hirohito Higashi) Use an enum for the type.
Files: src/evalfunc.c, src/testdir/test_match.vim

Patch 8.0.1005

Problem: Terminal without job updates slowly in GUI.
Solution: Poll for input when a channel has the keep_open flag.
Files: src/channel.c, src/proto/channel.pro, src/gui_gtk_x11.c

Patch 8.0.1006

Problem: Cannot parse text with '**errorformat**' without changing a quickfix list.
Solution: Add the "text" argument to getqflist(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/quickfix.pro,
src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1007

Problem: No test for filetype detection for scripts.
Solution: Add a first test file script filetype detection.
Files: src/testdir/test_filetype.vim, runtime/scripts.vim

Patch 8.0.1008

Problem: Slow updating of terminal window in Motif.
Solution: Add a timeout to the wait-for-character loop.
Files: src/gui_x11.c

Patch 8.0.1009

Problem: Xterm cursor blinking status may be inverted.

Solution: Use another request to get the blink status and compare with the cursor style report

Files: src/term.c, src/proto/term.pro, src/term.h, src/option.c, src/terminal.c

Patch 8.0.1010 (after 8.0.1009)

Problem: Build failure without termresponse feature.

Solution: Add #ifdef.

Files: src/term.c

Patch 8.0.1011

Problem: Terminal test fails with Athena and Motif.

Solution: Ignore the error for the input context. (Kazunobu Kuriyama)

Files: src/testdir/test_terminal.vim

Patch 8.0.1012

Problem: MS-Windows: Problem with \$HOME when it was set internally.

Solution: Only use the \$HOME default internally. (Yasuhiro Matsumoto, closes #2013)

Files: src/misc1.c, src/testdir/Make_all.mak, src/Makefile, src/testdir/test_windows_home.vim

Patch 8.0.1013

Problem: A terminal window with a running job behaves different from a window containing a changed buffer.

Solution: Do not set **'bufhidden'** to "hide". Fix that a buffer where a terminal used to run is listed as "[Scratch]".

Files: src/terminal.c, runtime/doc/terminal.txt, src/buffer.c

Patch 8.0.1014

Problem: Old compiler doesn't know uint32_t. Warning for using NULL instead of NUL.

Solution: Use UINT32_T. Use NUL instead of NULL.

Files: src/mbyte.c, src/proto/mbyte.pro, src/misc1.c

Patch 8.0.1015 (after 8.0.1013)

Problem: Missing update to terminal test.

Solution: Add the changes to the test.

Files: src/testdir/test_terminal.vim

Patch 8.0.1016

Problem: Gnome terminal echoes t_RC.

Solution: Detect Gnome terminal by the version string. Add v: variables for all the term responses.

Files: src/term.c, src/eval.c, src/vim.h, runtime/doc/eval.txt

Patch 8.0.1017

Problem: Test for MS-Windows \$HOME always passes.

Solution: Rename the test function. Make the test pass.

Files: src/testdir/test_windows_home.vim

Patch 8.0.1018

Problem: Warnings from 64-bit compiler. (Christian Brabandt)
Solution: Add type casts.
Files: src/terminal.c

Patch 8.0.1019

Problem: Pasting in virtual edit happens in the wrong place.
Solution: Do not adjust coladd when after the end of the line (closes #2015)
Files: src/testdir/test_virtualedit.vim, src/misc2.c

Patch 8.0.1020

Problem: When a timer calls getchar(1) input is overwritten.
Solution: Increment tb_change_cnt in inchar(). (closes #1940)
Files: src/getchar.c

Patch 8.0.1021

Problem: Older Gnome terminal still echoes t_RC. (François Ingelrest)
Solution: Check for version > 3000 instead of 4000.
Files: src/term.c

Patch 8.0.1022

Problem: Test 80 is old style.
Solution: Turn it into a new style test. (Yegappan Lakshmanan)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test80.in, src/testdir/test80.ok,
src/testdir/test_substitute.vim

Patch 8.0.1023

Problem: It is not easy to identify a quickfix list.
Solution: Add the "id" field. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1024

Problem: Manual folds are lost when a session file has the same buffer in
two windows. (Jeansen)
Solution: Use ":edit" only once. (Christian Brabandt, closes #1958)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.0.1025

Problem: Stray copy command in test.
Solution: Remove the copy command.
Files: src/testdir/test_mksession.vim

Patch 8.0.1026

Problem: GTK on-the-spot input has problems. (Gerd Wachsmuth)
Solution: Support over-the-spot. (Yukihiro Nakadaira, Ken Takata, closes
#1215)
Files: runtime/doc/mbyte.txt, runtime/doc/options.txt, src/edit.c,
src/ex_getln.c, src/mbyte.c, src/misc1.c, src/option.c,
src/option.h, src/screen.c, src/undo.c,
src/testdir/gen_opt_test.vim

Patch 8.0.1027

Problem: More terminals can't handle requesting cursor mode.
Solution: Recognize Putty. (Hirohito Higashi) Also include Xfce in the version check. (Dominique Pelle) Recognize Konsole.
Files: src/term.c

Patch 8.0.1028

Problem: MS-Windows: viminfo uses \$VIM/_viminfo if \$HOME not set. (Yongwei Wu)
Solution: Use vim_getenv() but check it's returning the default "C:/".
Files: src/ex_cmds.c

Patch 8.0.1029

Problem: Return value of getqflist() is inconsistent. (Lcd47)
Solution: Always return an "items" entry.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1030

Problem: MS-Windows: wrong size computation in is_cygpty().
Solution: Compute the size properly. (Ken Takata)
Files: src/iscygpty.c, src/iscygpty.h

Patch 8.0.1031

Problem: "text" argument for getqflist() is confusing. (Lcd47)
Solution: Use "lines" instead. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1032

Problem: "make tags" doesn't work well on MS-Windows.
Solution: Add or fix tags target. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.0.1033

Problem: Detecting background color does not work in screen, even when it is working like an xterm.
Solution: Make "screen.xterm" use termcap entries like an xterm. (Lubomir Rintel, closes #2048) When termresponse version is huge also recognize as not being an xterm.
Files: src/os_unix.c, src/term.c

Patch 8.0.1034

Problem: Sending buffer lines to terminal doesn't work on MS-Windows.
Solution: Send **CTRL-D** to mark the end of the text. (Yasuhiro Matsumoto, closes #2043) Add the "eof_chars" option.
Files: src/channel.c, src/proto/terminal.pro, src/terminal.c, src/testdir/test_terminal.vim, src/structs.h

Patch 8.0.1035

Problem: Sending buffer lines to terminal doesn't work on MS-Windows.
Solution: Use CR instead of NL after every line. Make the EOF text work properly. Add the ++eof argument to :terminal.
Files: src/structs.h, src/channel.c, src/terminal.c, runtime/doc/terminal.txt, runtime/doc/eval.txt

Patch 8.0.1036

Problem: ++eof argument for terminal only available on MS-Windows.
Solution: Also support ++eof on Unix. Add a test.
Files: src/channel.c, src/terminal.c, src/structs.h,
src/testdir/test_terminal.vim

Patch 8.0.1037

Problem: "icase" of '**diffopt**' is not used for highlighting differences.
Solution: Also use "icase". (Rick Howe)
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.0.1038

Problem: Strike-through text not supported.
Solution: Add support for the "strikethrough" attribute. (Christian Brabandt, Ken Takata)
Files: runtime/doc/eval.txt, runtime/doc/options.txt,
runtime/doc/syntax.txt, runtime/doc/term.txt, src/evalfunc.c,
src/gui.c, src/gui.h, src/gui_gtk_x11.c, src/gui_mac.c,
src/gui_w32.c, src/gui_x11.c, src/option.c, src/screen.c,
src/syntax.c, src/term.c, src/term.h, src/terminal.c, src/vim.h

Patch 8.0.1039

Problem: Cannot change a line in a buffer other than the current one.
Solution: Add setbuflineline(). (Yasuhiro Matsumoto, Ozaki Kiichi, closes #1953)
Files: src/evalfunc.c, runtime/doc/eval.txt, src/Makefile,
src/testdir/test_buflineline.vim, src/testdir/test_alot.vim

Patch 8.0.1040

Problem: Cannot use another error format in getqflist().
Solution: Add the "efm" argument to getqflist(). (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1041

Problem: Bogus characters appear when indenting kicks in while doing a visual-block append.
Solution: Recompute when indenting is done. (Christian Brabandt)
Files: runtime/doc/visual.txt, src/charset.c, src/edit.c, src/misc1.c,
src/ops.c, src/proto/charset.pro, src/proto/misc1.pro,
src/screen.c, src/spell.c, src/testdir/test_cindent.vim

Patch 8.0.1042 (after 8.0.1038)

Problem: Without the syntax feature highlighting doesn't work.
Solution: Always use unsigned short to store attributes.
Files: src/vim.h

Patch 8.0.1043

Problem: Warning for uninitialized variable. (John Marriott)
Solution: Move code to check indent inside "if".
Files: src/ops.c

Patch 8.0.1044

Problem: Warning for uninitialized variable. (John Marriott)

Solution: Initialize ind_pre.
Files: src/ops.c

Patch 8.0.1045

Problem: Running tests may pollute shell history. (Manuel Ortega)
Solution: Make \$HISTFILE empty.
Files: src/testdir/setup.vim

Patch 8.0.1046

Problem: Code duplication in diff mode.
Solution: Use diff_equal_char() also in diff_cmp(). (Rick Howe)
Files: src/diff.c

Patch 8.0.1047

Problem: Buffer overflow in Ruby.
Solution: Allocate one more byte. (Dominique Pelle)
Files: src/if_ruby.c

Patch 8.0.1048

Problem: No test for what 8.0.1020 fixes.
Solution: Add test_feedinput(). Add a test. (Ozaki Kiichi, closes #2046)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_timers.vim, src/ui.c

Patch 8.0.1049

Problem: Shell on Mac can't handle long text, making terminal test fail.
Solution: Only write 1000 characters instead of 5000.
Files: src/testdir/test_terminal.vim

Patch 8.0.1050

Problem: Terminal window feature not included by default.
Solution: Include the terminal feature for the "huge" build.
Files: src/configure.ac, src/auto/configure

Patch 8.0.1051

Problem: Cannot run terminal with spaces in argument.
Solution: Accept backslash to escape space and other characters. (closes #1999)
Files: src/os_unix.c, src/testdir/test_terminal.vim

Patch 8.0.1052

Problem: term_start() does not allow in_io, out_io and err_io options.
Solution: Add JO_OUT_IO to get_job_options().
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1053

Problem: setline() does not work on startup. (Manuel Ortega)
Solution: Do not check for ml_mfp to be set for the current buffer. (Christian Brabandt)
Files: src/testdir/shared.vim, src/testdir/test_alot.vim, src/testdir/test_buflines.vim, src/testdir/test_timers.vim, src/evalfunc.c

Patch 8.0.1054

Problem: Terminal test fails on MS-Windows.
Solution: Disable the redirection test for now. Improve scrape test to make it less flaky.
Files: src/testdir/test_terminal.vim

Patch 8.0.1055

Problem: Buflin test hangs on MS-Windows.
Solution: Avoid message for writing file. Source shared.vim when running test individually.
Files: src/testdir/test_buflin.vim, src/testdir/test_timers.vim

Patch 8.0.1056

Problem: Cannot build with the diff feature but without the multibyte feature.
Solution: Remove #ifdefs. (John Marriott)
Files: src/diff.c

Patch 8.0.1057

Problem: Terminal scrape test waits too long, it checks for one instead of three.
Solution: Check there are three characters. (micbou)
Files: src/testdir/test_terminal.vim

Patch 8.0.1058

Problem: Terminal redirection test is flaky.
Solution: Wait for job to finish.
Files: src/testdir/test_terminal.vim

Patch 8.0.1059

Problem: older Gnome terminal returns smaller version number. (antarestreue)
Solution: Lower version limit from 2800 to 2500. (#2032)
Files: src/term.c

Patch 8.0.1060

Problem: When imstyle is zero, mapping <Left> breaks preediting.
Solution: Pass though preediting key-events. (Yasuhiro Matsumoto, closes #2064, closes #2063)
Files: src/getchar.c, src/mbyte.c

Patch 8.0.1061

Problem: Coverity: no check for NULL command.
Solution: Check for NULL list item.
Files: src/terminal.c

Patch 8.0.1062

Problem: Coverity warnings in libvterm.
Solution: Add (void) to avoid warning for not checking return value. Add "break" before "case".
Files: src/libvterm/src/screen.c, src/libvterm/src/state.c

Patch 8.0.1063

Problem: Coverity warns for NULL check and using variable pointer as an array.
Solution: Remove the NULL check. Make "argvar" an array.

Files: src/terminal.c

Patch 8.0.1064

Problem: Coverity warns for leaking resource.

Solution: Free pty_master_fd on failure.

Files: src/os_unix.c

Patch 8.0.1065

Problem: Not all macro examples are included in the self-installing executable. (lkintact)

Solution: Add the directories to the NSIS script. (closes #2065)

Files: nsis/gvim.nsi

Patch 8.0.1066

Problem: Some terminals can't handle requesting cursor mode. (Steven Hartland)

Solution: Recognize vandyke SecureCRT. (closes #2008)

Files: src/term.c

Patch 8.0.1067

Problem: Using try/catch in timer does not prevent it from being stopped.

Solution: Reset the exception context and use did_emsg instead of called_emsg.

Files: src/ex_cmds2.c, src/testdir/test_timers.vim, src/globals.h, src/message.c

Patch 8.0.1068 (after 8.0.1066)

Problem: Vandyke SecureCRT terminal can't handle cursor mode request. (Steven Hartland)

Solution: Fix pointer computation. (closes #2008)

Files: src/term.c

Patch 8.0.1069

Problem: Still get **CTRL-X** sometimes for t_RS request.

Solution: Also skip 0x18 after a key code response.

Files: src/term.c

Patch 8.0.1070

Problem: Terminal test is flaky on Mac.

Solution: Add Test_terminal_noblock() to list of flaky tests.

Files: src/testdir/runtest.vim

Patch 8.0.1071

Problem: \$TERM names starting with "putty" and "cygwin" are likely to have a dark background, but are not recognized.

Solution: Only check the first few characters of \$TERM to match "putty" or "cygwin". (Christian Brabandt)

Files: src/option.c

Patch 8.0.1072

Problem: The :highlight command causes a redraw even when nothing changed.

Solution: Only set "need_highlight_changed" when an attribute changed.

Files: src/syntax.c

Patch 8.0.1073

Problem: May get an endless loop if **'statusline'** changes a highlight.
Solution: Do not let evaluating **'statusline'** trigger a redraw.
Files: src/buffer.c

Patch 8.0.1074

Problem: `":term NONE"` does not work on MS-Windows.
Solution: Make it work. Split `"pty"` into `"pty_in"` and `"pty_out"`. (Yasuhiro Matsumoto, closes #2058, closes #2045)
Files: runtime/doc/eval.txt,
runtime/pack/dist/opt/termdebug/plugin/termdebug.vim,
src/channel.c, src/evalfunc.c, src/os_unix.c, src/structs.h,
src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1075

Problem: MS-Windows: mouse does not work in terminal.
Solution: Force the winpty mouse on. (Yasuhiro Matsumoto, closes #2072)
Files: src/terminal.c

Patch 8.0.1076

Problem: `term_start()` does not take callbacks. When using two terminals without a job only one is read from. A terminal without a window returns the wrong pty.
Solution: Support `"callback"`, `"out_cb"` and `"err_cb"`. Fix terminal without a window. Fix reading from multiple channels.
Files: src/terminal.c, src/proto/terminal.pro, src/channel.c

Patch 8.0.1077

Problem: No debugger making use of the terminal window.
Solution: Add the term debugger plugin. So far only displays the current line when stopped.
Files: Filelist, runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1078

Problem: Using freed memory with `":hi Normal"`.
Solution: Get `"item"` again after updating the table.
Files: src/syntax.c

Patch 8.0.1079

Problem: Memory leak when `remote_foreground()` fails.
Solution: Free the error message.
Files: src/evalfunc.c, src/if_xcmdsrv.c

Patch 8.0.1080

Problem: Memory leak for `eof_chars` terminal option and buffer name.
Solution: Free job options. Free the buffer name
Files: src/terminal.c

Patch 8.0.1081

Problem: Memory leak for the channel write queue.
Solution: Free the write queue when clearing a channel.
Files: src/channel.c

Patch 8.0.1082

Problem: Tests fail when run under valgrind.
Solution: Increase waiting times.
Files: src/testdir/test_clientserver.vim, src/testdir/test_terminal.vim

Patch 8.0.1083

Problem: Leaking memory in input part of channel.
Solution: Clear the input part of channel. Free the entry. Move failing command test to a separate file to avoid bogus leak reports clouding tests that should not leak.
Files: src/channel.c, src/testdir/test_terminal.vim, src/Makefile, src/testdir/test_terminal_fail.vim, src/testdir/Make_all.mak

Patch 8.0.1084

Problem: GTK build has compiler warnings. (Christian Brabandt)
Solution: Get screen size with a different function. (Ken Takata, Yasuhiro Matsumoto)
Files: src/mbyte.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/gui_beval.c

Patch 8.0.1085

Problem: The terminal debugger can't set breakpoints.
Solution: Add :Break and :Delete commands. Also commands for stepping through code.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/terminal.txt

Patch 8.0.1086 (after 8.0.1084)

Problem: Can't build with GTK 3.
Solution: Rename function argument. (Kazunobu Kuriyama)
Files: src/gui_gtk_x11.c

Patch 8.0.1087

Problem: Test_terminal_cwd is flaky. MS-Windows: term_start() "cwd" argument does not work.
Solution: Wait for the condition to be true instead of using a sleep. Pass the directory to winpty.
Files: src/testdir/test_terminal.vim, src/terminal.c

Patch 8.0.1088

Problem: Occasional memory use after free.
Solution: Use the highlight table directly, don't keep a pointer.
Files: src/syntax.c

Patch 8.0.1089

Problem: Cannot get range count in user command.
Solution: Add <range> argument.
Files: src/ex_docmd.c, runtime/doc/map.txt

Patch 8.0.1090

Problem: cannot get the text under the cursor like v:beval_text
Solution: Add <cexpr>.
Files: src/ex_docmd.c, src/testdir/test_normal.vim, runtime/doc/cmdline.txt

Patch 8.0.1091 (after 8.0.1090)

Problem: Test for <cexpr> fails without +balloon_eval feature.

Solution: Remove #ifdefs.

Files: src/normal.c

Patch 8.0.1092

Problem: Terminal debugger can't evaluate expressions.

Solution: Add :Evaluate and K. Various other improvements.

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim,
runtime/doc/terminal.txt

Patch 8.0.1093

Problem: Various small quickfix issues.

Solution: Remove ":" prefix from title set by a user. Add the qf_id2nr().
function. Add a couple more tests. Update documentation.
(Yegappan Lakshmanan)

Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/evalfunc.c,
src/proto/quickfix.pro, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1094

Problem: Using ssh from Terminal.app runs into xterm incompatibility.

Solution: Also detect Terminal.app on non-Mac systems.

Files: src/term.c

Patch 8.0.1095

Problem: Terminal multibyte scrape test is flaky.

Solution: Add another condition to wait for.

Files: src/testdir/test_terminal.vim

Patch 8.0.1096

Problem: Terminal window in Normal mode has wrong background.

Solution: Store the default background and use it for clearing until the
end of the line. Not for below the last line, since there is no
text there.

Files: src/screen.c, src/terminal.c

Patch 8.0.1097 (after 8.0.1096)

Problem: Background color wrong if job changes background color.

Solution: Get the background color from vterm.

Files: src/terminal.c, src/screen.c

Patch 8.0.1098

Problem: Build failure if libvterm installed on the system. (Oleh
Hushchenkov)

Solution: Change the CCCTERM argument order. (Ken Takata, closes #2080)

Files: src/Makefile

Patch 8.0.1099

Problem: Warnings for GDK calls.

Solution: Use other calls for GTK 3 and fix a few problems. (Kazunobu
Kuriyama)

Files: src/mbyte.c

Patch 8.0.1100

Problem: Stuck in redraw loop when **'lazyredraw'** is set.
Solution: Don't loop on update_screen() when not redrawing. (Yasuhiro Matsumoto, closes #2082)
Files: src/terminal.c, src/screen.c, src/proto/screen.pro

Patch 8.0.1101

Problem: Channel write fails if writing to log fails.
Solution: Ignore return value of fwrite(). (Ozaki Kiichi, closes #2081)
Files: src/channel.c

Patch 8.0.1102

Problem: Terminal window does not use Normal colors.
Solution: For the GUI and when **'termguicolors'** is enabled, use the actual foreground and background colors for the terminal. (Yasuhiro Matsumoto, closes #2067)
Use the "Terminal" highlight group if defined.
Files: src/terminal.c, src/syntax.c, src/proto/syntax.pro

Patch 8.0.1103 (after 8.0.1102)

Problem: Converting cterm color fails for grey ramp.
Solution: Use index instead of number.
Files: src/terminal.c

Patch 8.0.1104

Problem: The qf_jump() function is too long.
Solution: Split of parts to separate functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1105

Problem: match() and matchend() are not tested.
Solution: Add tests. (Ozaki Kiichi, closes #2088)
Files: src/testdir/test_functions.vim, src/testdir/test_match.vim

Patch 8.0.1106

Problem: Terminal colors on an MS-Windows console are not matching the normal colors.
Solution: Use the normal colors for the terminal. (Yasuhiro Matsumoto, closes #2087)
Files: src/terminal.c

Patch 8.0.1107

Problem: Terminal debugger jumps to non-existing file.
Solution: Check that the file exists. Add an option to make the Vim width wide. Fix removing highlight groups.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/terminal.txt

Patch 8.0.1108

Problem: Cannot specify mappings for the terminal window.
Solution: Add the :tmap command and associated code. (Jacob Askeland, closes #2073)
Files: runtime/doc/map.txt, runtime/doc/terminal.txt, src/ex_cmdidxs.h, src/ex_cmds.h, src/ex_doccmd.c, src/getchar.c, src/gui.c,

src/terminal.c, src/testdir/test_terminal.vim, src/vim.h,
src/proto/terminal.pro, src/main.c, src/evalfunc.c

Patch 8.0.1109

Problem: Timer causes error on exit from Ex mode. (xtal8)
Solution: save and restore the ex_pressedreturn flag. (Christian Brabandt,
closes #2079)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_cmds2.c,
src/testdir/test_timers.vim

Patch 8.0.1110

Problem: FORTIFY_SOURCE from Perl causes problems. (Scott Baker)
Solution: Filter out the flag. (Christian Brabandt, closes #2068)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1111

Problem: Syntax error in configure when using Perl.
Solution: Add missing quote
Files: src/configure.ac, src/auto/configure

Patch 8.0.1112

Problem: Can't get size or current index from quickfix list.
Solution: Add "idx" and "size" options. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1113

Problem: Can go to Insert mode from Terminal-Normal mode.
Solution: Prevent :startinsert and "VA" to enter Insert mode. (Yasuhiro
Matsumoto, closes #2092)
Files: src/normal.c

Patch 8.0.1114

Problem: Default for 'iminsert' is annoying.
Solution: Make the default always zero. (Yasuhiro Matsumoto, closes #2071)
Files: src/option.c, runtime/doc/options.txt

Patch 8.0.1115

Problem: Crash when using foldtextresult() recursively.
Solution: Avoid recursive calls. (Yasuhiro Matsumoto, closes #2098)
Files: src/evalfunc.c, src/testdir/test_fold.vim

Patch 8.0.1116

Problem: Terminal test fails on MS-Windows.
Solution: Wait for the text to appear. (micbou, closes #2097)
Files: src/testdir/test_terminal.vim

Patch 8.0.1117

Problem: Test_terminal_no_cmd hangs on MS-Windows with GUI. (Christian
Brabandt)
Solution: Run the command with "start" and wait for the text to appear.
(micbou, closes #2096)
Files: src/testdir/test_terminal.vim

Patch 8.0.1118

Problem: FEAT_WINDOWS adds a lot of #ifdefs while it is nearly always enabled and only adds 7% to the binary size of the tiny build.

Solution: Graduate FEAT_WINDOWS.

Files: src/feature.h, src/window.c, src/vim.h, src/structs.h, src/globals.h, src/gui.h, src/if_py_both.h, src/option.h, src/term.h, src/buffer.c, src/charset.c, src/digraph.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/fold.c, src/getchar.c, src/gui.c, src/gui_athena.c, src/gui_beval.c, src/gui_gtk.c, src/gui_motif.c, src/gui_w32.c, src/if_cscope.c, src/if_lua.c, src/if_mzsch.c, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/main.c, src/mark.c, src/memline.c, src/misc1.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/option.c, src/popupmnu.c, src/quickfix.c, src/screen.c, src/search.c, src/spell.c, src/syntax.c, src/tag.c, src/term.c, src/ui.c, src/version.c, src/workshop.c, src/if_perl.xs, src/testdir/test_normal.vim

Patch 8.0.1119

Problem: Quitting a split terminal window kills the job. (Yasuhiro Matsumoto)

Solution: Only stop terminal job if it is the last window.

Files: src/buffer.c, src/testdir/test_terminal.vim

Patch 8.0.1120 (after 8.0.1108)

Problem: :tm means :tmap instead of :tmenu. (Taro Muraoka)

Solution: Move the new entry below the old entry. (closes #2102)

Files: src/ex_cmds.h, runtime/doc/map.txt

Patch 8.0.1121

Problem: Can uncheck executables in MS-Windows installer.

Solution: Make the choice read-only. (Ken Takata, closes #2106)

Files: nsis/gvim.nsi

Patch 8.0.1122

Problem: vimtutor.bat doesn't work well with vim.bat.

Solution: Use "call vim". (Ken Takata, closes #2105)

Files: vimtutor.bat

Patch 8.0.1123

Problem: Cannot define a toolbar for a window.

Solution: Add a window-local toolbar.

Files: src/syntax.c, src/proto/syntax.pro, src/structs.h, src/menu.c, src/proto/menu.pro, src/testdir/test_winbar.vim, src/Makefile, src/normal.c, src/testdir/Make_all.mak, src/if_perl.xs, src/eval.c, src/evalfunc.c, src/window.c, src/ui.c, src/terminal.c, src/screen.c, runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/gui.txt, runtime/doc/terminal.txt

Patch 8.0.1124

Problem: Use of MZSCHEME_VER is unclear.

Solution: Add a comment. (Ken Takata)

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.0.1125

Problem: Wrong window height when splitting window with window toolbar.
Solution: Add or subtract the window toolbar height.
Files: src/window.c

Patch 8.0.1126

Problem: Endless resize when terminal showing in two buffers. (Hirohito Higashi)
Solution: Set a flag to prevent resizing the window.
Files: src/terminal.c

Patch 8.0.1127

Problem: Test_peek_and_get_char fails on 32 bit system. (Elimar Riesebieter)
Solution: Avoid an integer overflow. (James McCoy, closes #2116)
Files: src/ex_cmds2.c

Patch 8.0.1128

Problem: Old xterm sends **CTRL-X** in response to t_RS.
Solution: Only send t_RS for xterm 279 and later. Remove the workaround to ignore **CTRL-X**.
Files: src/term.c

Patch 8.0.1129

Problem: Window toolbar missing a part of the patch.
Solution: Add change in vim.h.
Files: src/vim.h

Patch 8.0.1130

Problem: The qf_jump() function is still too long.
Solution: Split of parts to separate functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1131

Problem: It is not easy to trigger an autocommand for new terminal window. (Marco Restelli)
Solution: Trigger BufWinEnter after setting **'buftype'**.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1132

Problem: #if condition is not portable.
Solution: Add defined(). (Zuloloxi, closes #2136)
Files: src/libvterm/src/vterm.c

Patch 8.0.1133

Problem: Syntax timeout not used correctly.
Solution: Do not pass the timeout to syntax_start() but set it explicitly. (Yasuhiro Matsumoto, closes #2139)
Files: src/proto/syntax.pro, src/screen.c, src/syntax.c

Patch 8.0.1134

Problem: Superfluous call to syn_get_final_id().

Solution: Remove it. (Ken Takata)
Files: src/syntax.c

Patch 8.0.1135

Problem: W_WINCOL() is always the same.
Solution: Expand the macro.
Files: src/edit.c, src/ex_docmd.c, src/gui_gtk.c, src/gui_w32.c, src/netbeans.c, src/popupmnu.c, src/screen.c, src/term.c, src/terminal.c, src/ui.c, src/window.c, src/if_py_both.h, src/structs.h, src/vim.h

Patch 8.0.1136

Problem: W_WIDTH() is always the same.
Solution: Expand the macro.
Files: src/charset.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/getchar.c, src/gui.c, src/gui_beval.c, src/gui_mac.c, src/if_lua.c, src/if_mzsch.c, src/if_py_both.h, src/if_ruby.c, src/misc1.c, src/misc2.c, src/move.c, src/normal.c, src/popupmnu.c, src/quickfix.c, src/screen.c, src/search.c, src/structs.h, src/ui.c, src/vim.h, src/window.c

Patch 8.0.1137 (after 8.0.1136)

Problem: Cannot build with Ruby.
Solution: Fix misplaced brace.
Files: src/if_ruby.c

Patch 8.0.1138

Problem: Click in window toolbar starts Visual mode.
Solution: Add the MOUSE_WINBAR flag.
Files: src/ui.c, src/vim.h, src/normal.c

Patch 8.0.1139

Problem: Using window toolbar changes state.
Solution: Always execute window toolbar actions in Normal mode.
Files: runtime/doc/gui.txt, src/structs.h, src/ex_docmd.c, src/proto/ex_docmd.pro, src/menu.c

Patch 8.0.1140

Problem: Still old style tests.
Solution: Convert two tests to new style. (Yegappan Lakshmanan)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test56.in, src/testdir/test56.ok, src/testdir/test57.in, src/testdir/test57.ok, src/testdir/test_sort.vim, src/testdir/test_vimscript.vim

Patch 8.0.1141

Problem: MS-Windows build dependencies are incomplete.
Solution: Fix the dependencies. (Ken Takata)
Files: src/Make_cyg.mak, src/Make_cyg_ming.mak, src/Make_ming.mak, src/Make_mvc.mak

Patch 8.0.1142

Problem: Window toolbar menu gets a tear-off item.
Solution: Recognize the window toolbar.

Files: src/menu.c

Patch 8.0.1143

Problem: Macros always expand to the same thing.

Solution: Remove W_VSEP_WIDTH() and W_STATUS_HEIGHT().

Files: src/vim.h, src/structs.h, src/gui.c, src/ex_getln.c, src/screen.c

Patch 8.0.1144

Problem: Using wrong #ifdef for computing length.

Solution: use BACKSLASH_IN_FILENAME instead of COLON_IN_FILENAME. (Yasuhiro Matsumoto, closes #2153)

Files: src/quickfix.c

Patch 8.0.1145

Problem: Warning when compiling with Perl.

Solution: Remove unused variable. (Ken Takata)

Files: src/if_perl.xs

Patch 8.0.1146

Problem: Redraw when highlight is set with same names. (Ozaki Kiichi)

Solution: Only free and save a name when it changed. (closes #2120)

Files: src/syntax.c

Patch 8.0.1147

Problem: Fail to build with tiny features. (Tony Mechelynck)

Solution: Move #ifdefs.

Files: src/syntax.c

Patch 8.0.1148

Problem: "gN" doesn't work on last match with **'wrapscan'** off. (fcpq)

Solution: Adjust for searching backward. (Christian Brabandt)

Files: src/search.c, src/testdir/test_gn.vim

Patch 8.0.1149

Problem: libvterm colors differ from xterm.

Solution: Use the xterm colors for libvterm.

Files: src/terminal.c, src/libvterm/src/pen.c, src/testdir/xterm_ramp.vim, Filelist

Patch 8.0.1150

Problem: MS-Windows GUI: dialog font size is incorrect.

Solution: Pass flag to indicate **'encoding'** or active codepage. (Yasuhiro Matsumoto, closes #2160)

Files: src/gui_w32.c

Patch 8.0.1151

Problem: "vim -c startinsert!" doesn't append.

Solution: Correct line number on startup. (Christian Brabandt, closes #2117)

Files: src/ex_docmd.c, src/testdir/test_startup.vim

Patch 8.0.1152

Problem: Encoding of error message wrong in Cygwin terminal.

Solution: Get locale from environment variables. (Ken Takata)

Files: src/main.c, src/mbyte.c, src/proto/mbyte.pro

Patch 8.0.1153

Problem: No tests for diff_hlID() and diff_filler().
Solution: Add tests. (Dominique Pelle, closes #2156)
Files: src/testdir/test_diffmode.vim

Patch 8.0.1154

Problem: **'indentkeys'** does not work properly. (Gary Johnson)
Solution: Get the cursor line again. (Christian Brabandt, closes #2151)
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.0.1155

Problem: Ruby command triggers a warning when RUBYOPT is set to "-w".
Solution: use "-e_=0" instead of "-e0". (Masataka Pocke Kuwabara, closes #2143)
Files: src/if_ruby.c

Patch 8.0.1156

Problem: Removing one -W argument from Perl CFLAGS may cause trouble.
Solution: Remove all -W flags. (Christian Brabandt)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1157

Problem: Compiler warning on MS-Windows.
Solution: Add type cast. (Yasuhiro Matsumoto)
Files: src/main.c

Patch 8.0.1158

Problem: Still old style tests.
Solution: Convert several tests to new style. (Yegappan Lakshmanan)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/main.aap, src/testdir/test33.in,
src/testdir/test33.ok, src/testdir/test41.in,
src/testdir/test41.ok, src/testdir/test43.in,
src/testdir/test43.ok, src/testdir/test53.in,
src/testdir/test53.ok, src/testdir/test_file_size.vim,
src/testdir/test_lispwords.vim, src/testdir/test_search.vim,
src/testdir/test_textobjects.vim

Patch 8.0.1159

Problem: Typo in #ifdef.
Solution: Change "PROT" to "PROTO". (Nobuhiro Takasaki, closes #2165)
Files: src/syntax.c

Patch 8.0.1160

Problem: Getting tab-local variable fails after closing window.
Solution: set tp_firstwin and tp_lastwin. (Jason Franklin, closes #2170)
Files: src/window.c, src/evalfunc.c, src/testdir/test_getvar.vim

Patch 8.0.1161

Problem: Popup menu drawing problem when resizing terminal.
Solution: Redraw after resizing also when a popup menu is visible. (Ozaki Kiichi, closes #2110)
Files: src/popupmnu.c, src/term.c, src/testdir/shared.vim,

src/testdir/test_popup.vim

Patch 8.0.1162

Problem: Shared script for tests cannot be included twice.
Solution: Include it where needed, it will "finish" if loaded again.
Files: src/testdir/test_alot.vim, src/testdir/test_buflines.vim,
src/testdir/test_timers.vim

Patch 8.0.1163

Problem: Popup test is flaky.
Solution: Add a WaitFor() and fix another.
Files: src/testdir/test_popup.vim

Patch 8.0.1164

Problem: Changing StatusLine highlight while evaluating '**statusline**' may not change the status line color.
Solution: When changing highlighting while redrawing don't cause another redraw. (suggested by Ozaki Kiichi, closes #2171, closes #2120)
Files: src/buffer.c, src/syntax.c

Patch 8.0.1165

Problem: Popup test is still flaky.
Solution: Add a term_wait() call. (Ozaki Kiichi)
Files: src/testdir/test_popup.vim

Patch 8.0.1166

Problem: :terminal doesn't work on Mac High Sierra.
Solution: Change #ifdef for OpenPTY(). (Ozaki Kiichi, Kazunobu Kuriyama, closes #2162)
Files: src/pty.c

Patch 8.0.1167

Problem: Motif: typing in terminal window is slow.
Solution: Do not redraw the whole terminal window but only what was changed.
Files: src/terminal.c

Patch 8.0.1168

Problem: wrong highlighting with combination of match and '**cursorline**'.
Solution: Use "line_attr" when appropriate. (Ozaki Kiichi, closes #2111)
But don't highlight more than one character.
Files: src/screen.c, src/testdir/test_highlight.vim,
src/testdir/view_util.vim

Patch 8.0.1169

Problem: Highlighting one char too many with '**list**' and '**cul**'.
Solution: Check for '**list**' being active. (Ozaki Kiichi, closes #2177)
Files: src/screen.c, src/testdir/test_highlight.vim

Patch 8.0.1170

Problem: Using termdebug results in 100% CPU time. (tomleb)
Solution: Use polling instead of select().
Files: src/os_unix.c, src/channel.c, src/proto/channel.pro

Patch 8.0.1171

Problem: Popup test is still a bit flaky.
Solution: Change term_wait() calls. (Ozaki Kiichi)
Files: src/testdir/test_popup.vim

Patch 8.0.1172

Problem: When E734 is given option is still set.
Solution: Assign NULL to "s". (Christian Brabandt)
Files: src/eval.c, src/testdir/test_assign.vim

Patch 8.0.1173

Problem: Terminal window is not redrawn after **CTRL-L**. (Marcin Szamotulski)
Solution: Redraw the whole terminal when w_redr_type is NOT_VALID.
Files: src/terminal.c

Patch 8.0.1174

Problem: Mac Terminal.app has wrong color for white.
Solution: Use white from the color cube.
Files: src/globals.h, src/term.c, src/syntax.c

Patch 8.0.1175 (after 8.0.1174)

Problem: Build failure without +termresponse.
Solution: Add #ifdef.
Files: src/syntax.c

Patch 8.0.1176

Problem: Job_start() does not handle quote and backslash correctly.
Solution: Remove quotes, recognize and remove backslashes.
Files: src/testdir/test_channel.vim, src/os_unix.c

Patch 8.0.1177

Problem: In a terminal window the popup menu is not cleared. (Gerry Agbobada)
Solution: Redraw when SOME_VALID is used instead of NOT_VALID. (closes #2194)
Files: src/terminal.c

Patch 8.0.1178

Problem: Using old compiler on MS-Windows.
Solution: Switch default build on MS-Windows to use MSVC 2015. (Ken Takata)
Files: src/msvc2015.bat, src/INSTALLpc.txt, src/GvimExt/Makefile, src/Make_mvc.mak, src/tee/Make_mvc.mak, src/xxd/Make_mvc.mak

Patch 8.0.1179

Problem: Test_popup_and_window_resize() does not always pass.
Solution: Do not use \$VIMPROG, pass the Vim executable in the vimcmd file. (Ozaki Kiichi, closes #2186)
Files: src/testdir/Makefile, src/testdir/shared.vim, src/testdir/test_popup.vim

Patch 8.0.1180

Problem: MS-Windows testclean target deletes the color script.
Solution: Rename the script file.
Files: src/testdir/xterm_ramp.vim, src/testdir/color_ramp.vim

Patch 8.0.1181

Problem: Tests using Vim command fail on MS-Windows.
Solution: Do not add quotes around the Vim command.
Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak

Patch 8.0.1182

Problem: Cannot see or change mzscheme dll name.
Solution: Add **'mzschemedll'** and **'mzschemegcdll'**.
Files: src/if_mzsch.c, src/option.h, src/option.c,
runtime/doc/if_mzsch.txt

Patch 8.0.1183

Problem: MS-Windows build instructions are outdated.
Solution: Update instructions for MSVC 2015. Update the build script.
Files: Filelist, Makefile, src/INSTALLpc.txt, src/bigvim.bat

Patch 8.0.1184

Problem: The :marks command is not tested.
Solution: Add a test. (Dominique Pelle, closes #2197)
Files: src/testdir/test_marks.vim

Patch 8.0.1185

Problem: Ruby library includes minor version number.
Solution: Only use the API version number. (Ben Boeckel, closes #2199)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1186

Problem: Still quite a few old style tests.
Solution: Convert old to new style tests. (Yegappan Lakshmanan)
Avoid ringing the bell while running tests.
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_ming.mak,
src/testdir/Make_vms.mms, src/testdir/main.aap,
src/testdir/test31.in, src/testdir/test31.ok,
src/testdir/test4.in, src/testdir/test4.ok, src/testdir/test5.in,
src/testdir/test5.ok, src/testdir/test60.in,
src/testdir/test60.ok, src/testdir/test60.vim,
src/testdir/test7.in, src/testdir/test7.ok, src/testdir/test78.in,
src/testdir/test78.ok, src/testdir/test_autocmd.vim,
src/testdir/test_exists.vim, src/testdir/test_recover.vim,
src/testdir/test_winbuf_close.vim, src/testdir/runtest.vim

Patch 8.0.1187

Problem: Building with lua fails for OSX on Travis.
Solution: Separate brew-update and brew-install. (Ozaki Kiichi, closes #2203)
Files: .travis.yml

Patch 8.0.1188

Problem: Autocmd test fails on MS-Windows.
Solution: Give the buffer a name and find the buffer to be wiped out by
name.
Files: src/testdir/test_autocmd.vim

Patch 8.0.1189

Problem: E172 is not actually useful, it's only on Unix anyway.

Solution: Remove the check and the error.
Files: src/ex_docmd.c, runtime/doc/message.txt

Patch 8.0.1190

Problem: Vim becomes unusable after opening new window in BufWritePre event.

Solution: Call not_exiting(). (Martin Tournoij, closes #2205)
Also for "2q" when a help window is open. Add a test.

Files: src/ex_docmd.c, src/testdir/test_writefile.vim

Patch 8.0.1191

Problem: MS-Windows: missing 32 and 64 bit files in installer.

Solution: Include both 32 and 64 bit GvimExt and related dll files. Remove old Windows code from the installer. (Ken Takata, closes #2144)

Files: nsis/README.txt, nsis/gvim.nsi, src/GvimExt/gvimext.cpp, src/dosinst.c, src/dosinst.h, src/uninstal.c, Makefile

Patch 8.0.1192

Problem: MS-Windows: terminal feature not enabled by default.

Solution: Enable it. (Ken Takata)

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.0.1193

Problem: Crash when wiping out a buffer after using getbufinfo().
(Yegappan Lakshmanan)

Solution: Remove b:changedtick from the buffer variables.

Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 8.0.1194

Problem: Actual fg and bg colors of terminal are unknown.

Solution: Add t_RF. Store response to t_RB and t_RF, use for terminal.

Files: src/term.c, src/term.h, src/proto/term.pro, src/terminal.c, src/vim.h, src/eval.c, runtime/doc/eval.txt

Patch 8.0.1195 (after 8.0.1194)

Problem: Can't build on MS-Windows.

Solution: Adjust #ifdef and add #ifdefs.

Files: src/term.c, src/terminal.c

Patch 8.0.1196 (after 8.0.1194)

Problem: Crash when t_RF is not set. (Brian Pina)

Solution: Add t_RF to the list of terminal options. (Hirohito Higashi)

Files: src/option.c

Patch 8.0.1197

Problem: MS-Windows build instructions are not up to date.

Solution: Adjust the instructions. Fix the nsis script.

Files: Makefile, nsis/gvim.nsi

Patch 8.0.1198

Problem: Older compilers don't know uint8_t.

Solution: Use char_u instead.

Files: src/term.c, src/proto/term.pro

Patch 8.0.1199

Problem: When **'clipboard'** is "autoselectplus" the star register is also set. (Gilles Moris)
Solution: Don't set the star register in this situation.
Files: src/ops.c

Patch 8.0.1200

Problem: Tests switch the bell off twice.
Solution: Don't set **'belloff'** in individual tests. (Christian Brabandt)
Files: src/testdir/test_alot.vim, src/testdir/test_alot_utf8.vim, src/testdir/test_autocmd.vim, src/testdir/test_cmdline.vim, src/testdir/test_diffmode.vim, src/testdir/test_digraph.vim, src/testdir/test_edit.vim, src/testdir/test_file_size.vim, src/testdir/test_gn.vim, src/testdir/test_normal.vim, src/testdir/test_packadd.vim, src/testdir/test_popup.vim, src/testdir/test_recover.vim, src/testdir/test_search.vim, src/testdir/test_textobjects.vim, src/testdir/test_undo.vim, src/testdir/test_usercommands.vim, src/testdir/test_visual.vim

Patch 8.0.1201

Problem: "yL" is affected by **'scrolloff'**. (Eli the Bearded)
Solution: Don't use **'scrolloff'** when an operator is pending.
Files: src/normal.c, runtime/doc/motion.txt

Patch 8.0.1202

Problem: :wall gives an error for a terminal window. (Marius Gedminas)
Solution: Don't try writing a buffer that can't be written. (Yasuhiro Matsumoto, closes #2190)
Files: src/ex_cmds.c, src/testdir/test_terminal.vim

Patch 8.0.1203

Problem: Terminal window mistreats composing characters.
Solution: Count composing characters with the base character. (Ozaki Kiichi, closes #2195)
Files: src/mbyte.c, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1204

Problem: A QuitPre autocommand may get the wrong file name.
Solution: Pass the buffer being closed to apply_autocmds(). (Rich Howe)
Files: src/ex_docmd.c, src/testdir/test_autocmd.vim

Patch 8.0.1205

Problem: Using "lq" it is possible to unload a changed buffer. (Rick Howe)
Solution: Check the right window for changes.
Files: src/testdir/test_edit.vim, src/ex_docmd.c

Patch 8.0.1206

Problem: No autocmd for entering or leaving the command line.
Solution: Add CmdlineEnter and CmdlineLeave.
Files: runtime/doc/autocmd.txt, src/ex_getln.c, src/fileio.c, src/vim.h, src/testdir/test_autocmd.vim

Patch 8.0.1207

Problem: Profiling skips the first and last script line.

Solution: Check for BOM after setting script ID. (LemonBoy, closes #2103, closes #2112) Add a test. List the trailing script lines.
Files: src/testdir/test_profile.vim, src/ex_cmds2.c

Patch 8.0.1208

Problem: **'statusline'** drops empty group with highlight change.
Solution: Do not drop an empty group if it changes highlighting. (Marius Gedminas, closes #2228)
Files: src/buffer.c, src/testdir/test_statusline.vim

Patch 8.0.1209

Problem: Still too many old style tests.
Solution: Convert a few more tests to new style. (Yegappan Lakshmanan, closes #2230)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_ming.mak, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Makefile, src/testdir/Make_vms.mms, src/testdir/main.aap, src/testdir/test34.in, src/testdir/test34.ok, src/testdir/test54.in, src/testdir/test54.ok, src/testdir/test8.in, src/testdir/test8.ok, src/testdir/test_autocmd.vim, src/testdir/test_autoformat_join.in, src/testdir/test_autoformat_join.ok, src/testdir/test_join.vim, src/testdir/test_user_func.vim

Patch 8.0.1210

Problem: When typing a search pattern **CTRL-G** and **CTRL-T** are ignored when there is typeahead.
Solution: Don't pass SEARCH_PEEK and don't call char_avail(). (haya14busa, closes #2233)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.0.1211

Problem: Cannot reorder tab pages with drag & drop.
Solution: Support drag & drop for GTK and MS-Windows. (Ken Takata, Masamichi Abe)
Files: src/gui_gtk_x11.c, src/gui_w32.c

Patch 8.0.1212

Problem: MS-Windows: tear-off menu does not work on 64 bit. (shaggyaxe)
Solution: Change how the menu handle is looked up. (Ken Takata, closes #1205)
Files: src/gui_w32.c

Patch 8.0.1213

Problem: Setting **'mzschemedll'** has no effect.
Solution: Move loading .vimrc to before call to mzscheme_main().
Files: src/main.c

Patch 8.0.1214

Problem: Accessing freed memory when EXITFREE is set and there is more than one tab and window. (Dominique Pelle)
Solution: Free options later. Skip redraw when exiting.
Files: src/screen.c, src/misc2.c

Patch 8.0.1215

Problem: Newer gcc warns for implicit fallthrough.
Solution: Consistently use a FALLTHROUGH comment. (Christian Brabandt)
Files: src/buffer.c, src/edit.c, src/eval.c, src/ex_docmd.c,
src/ex_getln.c, src/main.c, src/message.c, src/normal.c,
src/regexp.c, src/regexp_nfa.c, src/spell.c, src/window.c,
src/if_perl.xs

Patch 8.0.1216

Problem: Tabline is not always updated for :file command. (Norio Takagi)
Solution: Set redraw_tabline. (Hirohito Higashi)
Files: src/ex_cmds.c

Patch 8.0.1217

Problem: Can't use remote eval to inspect vars in debug mode.
Solution: Don't discard the call stack in debug mode. (closes #2237, #2247)
Files: src/globals.h, src/ex_cmds2.c, src/main.c

Patch 8.0.1218

Problem: Writing to freed memory in autocmd.
Solution: Make a copy of the tag line. (Dominique Pelle, closes #2245)
Files: src/tag.c, src/testdir/test_autocmd.vim

Patch 8.0.1219

Problem: Terminal test is flaky.
Solution: Add test function to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.1220

Problem: Skipping empty statusline groups is not correct.
Solution: Also set group_end_userhl. (itchyny)
Files: src/buffer.c, src/testdir/test_statusline.vim

Patch 8.0.1221

Problem: Still too many old style tests.
Solution: Convert a few more tests to new style. (Yegappan Lakshmanan, closes #2256)
Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_vms.mms,
src/testdir/main.aap, src/testdir/test19.in,
src/testdir/test19.ok, src/testdir/test20.in,
src/testdir/test20.ok, src/testdir/test25.in,
src/testdir/test25.ok, src/testdir/test28.in,
src/testdir/test28.ok, src/testdir/test32.in,
src/testdir/test32.ok, src/testdir/test38.in,
src/testdir/test38.ok, src/testdir/test66.in,
src/testdir/test66.ok, src/testdir/test79.in,
src/testdir/test79.ok, src/testdir/test_ins_complete.vim,
src/testdir/test_source_utf8.vim, src/testdir/test_substitute.vim,
src/testdir/test_tab.vim, src/testdir/test_tagjump.vim,
src/testdir/test_undo.vim, src/testdir/test_visual.vim,
src/testdir/test79.ok, src/testdir/test79.in,
src/testdir/test28.in

Patch 8.0.1222

Problem: Test functions interfere with each other.
Solution: Cleanup tab pages, windows and buffers. Reset option.
Files: src/testdir/runtest.vim, src/testdir/test_filetype.vim,
src/testdir/test_tabpage.vim, src/testdir/test_lispwords.vim

Patch 8.0.1223

Problem: Crash when using autocompleate and tab pages.
Solution: Check if the current tab changed. (Christian Brabandt, closes #2239)
Files: src/popupmnu.c, src/testdir/test_popup.vim, src/misc1.c,

Patch 8.0.1224

Problem: Still interference between test functions.
Solution: Clear autocommands. Wipe all buffers. Fix tests that depend on a specific start context.
Files: src/testdir/runtest.vim, src/testdir/test_autocmd.vim,
src/testdir/test_arglist.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_command_count.vim, src/testdir/test_quickfix.vim,
src/testdir/test_hardcopy.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_packadd.vim, src/testdir/test_signs.vim,
src/testdir/test_autochdir.vim

Patch 8.0.1225

Problem: No check for spell region being zero. (geeknik)
Solution: Check for zero. (closes #2252)
Files: src/spellfile.c, src/testdir/test_spell.vim

Patch 8.0.1226

Problem: Edit and popup tests failing.
Solution: Make the tests pass.
Files: src/testdir/test_edit.vim, src/testdir/test_popup.vim

Patch 8.0.1227

Problem: Undefined left shift in readfile(). (Brian 'geeknik' Carpenter)
Solution: Add cast to unsigned. (Dominique Pelle, closes #2253)
Files: src/fileio.c

Patch 8.0.1228

Problem: Invalid memory access in GUI test.
Solution: Check that the row is not outside of the screen.
Files: src/screen.c

Patch 8.0.1229

Problem: Condition in vim_str2nr() is always true. (Nikolai Pavlov)
Solution: Remove the condition. (Closes #2259)
Files: src/charset.c

Patch 8.0.1230

Problem: **CTRL-A** in Visual mode uses character after selection. (Nikolai Pavlov)
Solution: Check the length before using a character.
Files: src/charset.c

Patch 8.0.1231

Problem: Expanding file name drops dash. (stucki)
Solution: Use the right position. (Christian Brabandt, closes #2184)
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.0.1232

Problem: MS-Windows users are confused about default mappings.
Solution: Don't map keys in the console where they don't work. Add a choice in the installer to use MS-Windows key bindings or not. (Christian Brabandt, Ken Takata, closes #2093)
Files: Filelist, nsis/gvim.nsi, nsis/vimrc.ini, src/dosinst.c, runtime/mswin.vim

Patch 8.0.1233

Problem: Typo in dos installer.
Solution: Remove comma.
Files: src/dosinst.c

Patch 8.0.1234

Problem: MS-Windows: composing characters are not shown properly.
Solution: Pass base character and composing characters to the renderer at once. (Ken Takata, closes #2206)
Files: src/gui.c, src/gui_w32.c

Patch 8.0.1235

Problem: Cannot disable the terminal feature in a huge build. (lindhobe)
Solution: Adjust the autoconf check. (Kazunobu Kuriyama, closes #2242)
Files: src/configure.ac, src/auto/configure, src/Makefile

Patch 8.0.1236

Problem: Mac features are confusing.
Solution: Make feature names more consistent, add "osxdarwin". Rename feature flags, cleanup Mac code. (Kazunobu Kuriyama, closes #2178) Also includes a fix for when Ruby throws an exception inside :rubyfile. (ujihisa)
Files: runtime/doc/eval.txt, runtime/doc/os_mac.txt, src/auto/configure, src/config.h.in, src/configure.ac, src/digraph.c, src/edit.c, src/evalfunc.c, src/feature.h, src/fileio.c, src/getchar.c, src/globals.h, src/gui.c, src/gui_mac.c, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/keymap.h, src/macros.h, src/main.c, src/mbyte.c, src/message.c, src/misc1.c, src/misc2.c, src/option.c, src/os_mac.h, src/os_macosx.m, src/os_unix.c, src/proto.h, src/pty.c, src/structs.h, src/term.c, src/termlib.c, src/ui.c, src/undo.c, src/version.c, src/vim.h, src/window.c

Patch 8.0.1237

Problem: ":set scroll&" often gives an error.
Solution: Don't use a fixed default value, use half the window height. Add a test. (Ozaki Kiichi, closes #2104)
Files: src/Makefile, src/option.c, src/testdir/test_alot.vim, src/testdir/test_scroll_opt.vim

Patch 8.0.1238

Problem: Incremental search only shows one match.
Solution: When `'incsearch'` and `'hlsearch'` are both set highlight all matches. (haya14busa, itchy, closes #2198)
Files: runtime/doc/options.txt, src/ex_getln.c, src/proto/search.pro, src/search.c, src/testdir/test_search.vim

Patch 8.0.1239

Problem: Cannot use a lambda for the skip argument to searchpair().
Solution: Evaluate a partial, funcref and lambda. (LemonBoy, closes #1454, closes #2265)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/evalfunc.pro, src/eval.c, src/proto/eval.pro, src/search.c, src/testdir/test_search.vim

Patch 8.0.1240

Problem: MS-Windows: term_start() does not support environment.
Solution: Implement the environment argument. (Yasuhiro Matsumoto, closes #2264)
Files: src/os_win32.c, src/proto/os_win32.pro, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1241

Problem: Popup test is flaky. (James McCoy)
Solution: Increase the wait time. (Dominique Pelle)
Files: src/testdir/test_popup.vim

Patch 8.0.1242

Problem: Function argument with only dash is seen as number zero. (Wang Shidong)
Solution: See a dash as a string. (Christian Brabandt)
Files: src/testdir/test_ins_complete.vim, src/Makefile, src/eval.c

Patch 8.0.1243

Problem: No test for what 8.0.1227 fixes.
Solution: Add a test that triggers the problem. (Christian Brabandt)
Files: src/testdir/test_normal.vim, src/testdir/test_search.vim

Patch 8.0.1244

Problem: Search test does not work correctly on MS-Windows.
Solution: Put text in a file instead of sending it to the terminal. (Christian Brabandt)
Files: src/testdir/test_search.vim

Patch 8.0.1245

Problem: When WaitFor() has a wrong expression it just waits a second, which goes unnoticed. (James McCoy)
Solution: When WaitFor() times out throw an exception. Fix places where the expression was wrong.
Files: src/testdir/shared.vim, src/testdir/test_channel.vim, src/testdir/test_netbeans.vim, src/testdir/test_terminal.vim

Patch 8.0.1246

Problem: Popup test has an arbitrary delay.
Solution: Wait for the ruler to show. (James McCoy)

Files: src/testdir/test_popup.vim

Patch 8.0.1247

Problem: Not easy to find Debian build info.

Solution: Add a badge in the README file. (Dominique Pelle)

Files: README.md

Patch 8.0.1248 (after 8.0.1247)

Problem: Stray + in README file.

Solution: Remove the +. Add a line break.

Files: README.md

Patch 8.0.1249

Problem: No error when WaitFor() gets an invalid wrong expression.

Solution: Do not ignore errors in evaluation of the expression. Fix places where the expression was wrong.

Files: src/testdir/shared.vim, src/testdir/test_netbeans.vim

Patch 8.0.1250

Problem: **'hlsearch'** highlighting not removed after incsearch (lacygoill)

Solution: Redraw all windows. Start search at the end of the match. Improve how **CTRL-G** works with incremental search. Add tests. (Christian Brabandt, Hirohito Higashi, haya14busa, closes #2267)

Files: runtime/doc/options.txt, src/ex_getln.c, src/testdir/test_search.vim

Patch 8.0.1251 (after 8.0.1249)

Problem: Invalid expression passed to WaitFor().

Solution: Check if the variable exists.

Files: src/testdir/test_clientserver.vim

Patch 8.0.1252

Problem: Incomplete translations makefile for MinGW/Cygwin.

Solution: Add missing source files. Make it work with msys2's bash. (Ken Takata)

Files: src/po/Make_cyg.mak, src/po/Make_ming.mak, src/po/Make_mvc.mak

Patch 8.0.1253

Problem: Still too many old style tests.

Solution: Convert a few more tests to new style. (Yegappan Lakshmanan, closes #2272)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, src/testdir/main.aap, src/testdir/test12.in, src/testdir/test12.ok, src/testdir/test40.in, src/testdir/test40.ok, src/testdir/test45.in, src/testdir/test45.ok, src/testdir/test83.in, src/testdir/test83.ok, src/testdir/test_autocmd.vim, src/testdir/test_fold.vim, src/testdir/test_swap.vim, src/testdir/test_tagjump.vim

Patch 8.0.1254

Problem: Undefined left shift in gethexchr(). (geeknik)

Solution: Use unsigned long. (idea by Christian Brabandt, closes #2255)
Files: src/regexp.c, src/regexp_nfa.c

Patch 8.0.1255 (after 8.0.1248)
Problem: duplicate badge README file.
Solution: Remove one. (Dominique Pelle)
Files: README.md

Patch 8.0.1256
Problem: Typo in configure variable vim_cv_tgent. (Matthieu Guillard)
Solution: Rename the variable. (closes #2281)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1257 (after 8.0.1254)
Problem: No test for fix of undefined behavior.
Solution: Add a test. (closes #2255)
Files: src/testdir/test_search.vim

Patch 8.0.1258
Problem: **'tymouse'** is set to "sgr" even though it's not supported. (Gary Johnson)
Solution: Adjust #ifdef
Files: src/term.c

Patch 8.0.1259
Problem: Search test can be flaky.
Solution: Use WaitFor() instead of a delay. Make it possible to pass a funcref to WaitFor() to avoid the need for global variables. (James McCoy, closes #2282)
Files: src/testdir/shared.vim, src/testdir/test_search.vim

Patch 8.0.1260 (after 8.0.1259)
Problem: Using global variables for WaitFor().
Solution: Use a lambda function instead. Don't check a condition if WaitFor() already checked it.
Files: src/testdir/test_popup.vim, src/testdir/test_terminal.vim, src/testdir/test_channel.vim, src/testdir/test_clientserver.vim, src/testdir/test_job_fails.vim, src/testdir/test_quotestar.vim

Patch 8.0.1261
Problem: Program in terminal window gets NL instead of CR. (Lifepillar)
Solution: Check the tty setup more often. (closes #1998)
Files: src/terminal.c

Patch 8.0.1262
Problem: Terminal redir test is flaky.
Solution: Add it to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.1263
Problem: Others can read the swap file if a user is careless with his primary group.
Solution: If the group permission allows for reading but the world

permissions doesn't, make sure the group is right.
Files: src/fileio.c, src/testdir/test_swap.vim, src/Makefile

Patch 8.0.1264

Problem: Terminal debugger gets stuck in small window.
Solution: Add "-quiet" to the gdb command. (Christian Brabandt, closes #2154)
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1265 (after 8.0.1263)

Problem: Swap test not skipped when there is one group.
Solution: Convert list to string for the message.
Files: src/testdir/test_swap.vim

Patch 8.0.1266 (after 8.0.1263)

Problem: Test_swap_directory was accidentally commented out.
Solution: Uncomment the test.
Files: src/testdir/test_swap.vim

Patch 8.0.1267 (after 8.0.1263)

Problem: Test_swap_group may leave file behind.
Solution: Add a try/finally.
Files: src/testdir/test_swap.vim, src/testdir/test_undo.vim

Patch 8.0.1268

Problem: PC install instructions are incomplete.
Solution: Update the instructions. (Ken Takata)
Files: src/INSTALLpc.txt

Patch 8.0.1269

Problem: Effect of autocommands on marks is not tested.
Solution: Add a couple of tests. (James McCoy, closes #2271)
Files: src/testdir/test_autocmd.vim

Patch 8.0.1270

Problem: Mismatching file name with Filelist.
Solution: Rename color_ramp.vim to xterm_ramp.vim
Files: src/testdir/color_ramp.vim, src/testdir/xterm_ramp.vim

Patch 8.0.1271

Problem: Still too many old style tests.
Solution: Convert a few more tests to new style. (Yegappan Lakshmanan, closes #2290)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/sautest/autoload/footest.vim, src/testdir/test55.in, src/testdir/test55.ok, src/testdir/test_changelist.in, src/testdir/test_changelist.ok, src/testdir/test_fold.vim, src/testdir/test_ins_complete.vim, src/testdir/test_insertcount.in, src/testdir/test_insertcount.ok, src/testdir/test_listdict.vim, src/testdir/test_normal.vim, src/testdir/test_search.vim, src/testdir/test_search_mbyte.in

Patch 8.0.1272

Problem: Warnings for unused variables in tiny build.
Solution: Add #ifdef. (Dominique Pelle, closes #2288)

Files: src/term.c

Patch 8.0.1273 (after 8.0.1271)

Problem: Old test file remaining.

Solution: Delete it.

Files: src/testdir/test_search_mbyte.ok

Patch 8.0.1274

Problem: setbuflines() fails when using folding.

Solution: Set "curwin" if needed. (Ozaki Kiichi, closes #2293)

Files: src/evalfunc.c, src/testdir/test_buflines.vim

Patch 8.0.1275

Problem: CmdlineLeave autocmd prevents fold from opening. (Waivek)

Solution: Save and restore KeyTyped. (closes #2305)

Files: src/fileio.c

Patch 8.0.1276

Problem: Typed key is lost when the terminal window is closed in exit callback. (Gabriel Barta)

Solution: When the current window changes bail out of the wait loop. (closes #2302)

Files: src/misc2.c, src/terminal.c

Patch 8.0.1277

Problem: Terminal window CR-NL conversions may cause problems.

Solution: Avoid most conversions, only fetch the current backspace key value from the tty. (mostly by Ozaki Kiichi, closes #2278)

Files: src/terminal.c

Patch 8.0.1278

Problem: GUI window always resizes when adding/removing a scrollbar, toolbar, etc.

Solution: Add the 'k' flag in **'guioptions'** to keep the GUI window size and change the number of lines/columns instead. (Ychin, closes #703)

Files: runtime/doc/options.txt, src/gui.c, src/gui_gtk_x11.c, src/gui_w32.c, src/option.h

Patch 8.0.1279

Problem: Initializing menus can be slow, especially when there are many keymaps, color schemes, etc.

Solution: Do the globbing for runtime files lazily. (Ken Takata)

Files: runtime/doc/gui.txt, runtime/menu.vim

Patch 8.0.1280

Problem: Python None cannot be converted to a Vim type.

Solution: Convert it to v:none. (Ken Takata)

Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok, runtime/doc/if_pyth.txt

Patch 8.0.1281

Problem: Loading file type detection slows down startup.

Solution: Move functions to an autoloader script.

Files: runtime/filetype.vim, runtime/autoload/filetype.vim,

runtime/scripts.vim

Patch 8.0.1282 (after 8.0.1281)

Problem: script-local variable defined in the wrong script
Solution: Move variable to autoload/filetype.vim.
Files: runtime/filetype.vim, runtime/autoload/filetype.vim

Patch 8.0.1283

Problem: Test 86 fails under ASAN.
Solution: Fix that an item was added to a dictionary twice.
Files: src/if_py_both.h

Patch 8.0.1284

Problem: Loading file type detection slows down startup.
Solution: Store the last pattern of an autocommand event to make appending quicker.
Files: src/fileio.c

Patch 8.0.1285

Problem: Distributed autoload files may clash with user files. (Andy Wokula)
Solution: Use the "autoload/dist" directory.
Files: runtime/filetype.vim, runtime/autoload/filetype.vim, runtime/autoload/dist/ft.vim, runtime/scripts.vim, Filelist, src/Makefile, nsis/gvim.nsi

Patch 8.0.1286

Problem: Occasional crash when using a channel. (Marek)
Solution: Decrement reference count later. (closes #2315)
Files: src/channel.c

Patch 8.0.1287

Problem: The temp file used when updating the viminfo file may have the wrong permissions if setting the group fails.
Solution: Check if the group matches and reduce permissions if not.
Files: src/ex_cmds.c

Patch 8.0.1288

Problem: GUI: cannot drag the statusline of a terminal window.
Solution: Handle the TERMINAL state. (Hirohito Higashi)
Files: src/gui.c

Patch 8.0.1289

Problem: Mkview always includes the local directory.
Solution: Add the "curdir" value in '**viewoptions**'. (Eric Roberts, closes #2316)
Files: runtime/doc/options.txt, runtime/doc/starting.txt, src/ex_docmd.c, src/option.c

Patch 8.0.1290

Problem: seq_cur of undotree() wrong after undo.
Solution: Get the actual sequence number instead of decrementing the current one. (Ozaki Kiichi, closes #2319)
Files: src/undo.c, src/testdir/test_undo.vim

Patch 8.0.1291

Problem: C indent wrong when * immediately follows comment. (John Bowler)
Solution: Do not see "/*" after "*" as a comment start. (closes #2321)
Files: src/search.c, src/testdir/test3.in, src/testdir/test3.ok

Patch 8.0.1292

Problem: Quick clicks in the WinBar start Visual mode.
Solution: Use a double click in the WinBar like a normal click.
Files: src/ui.c

Patch 8.0.1293

Problem: Setting a breakpoint in the terminal debugger sometimes fails.
Solution: Interrupt the program if needed. Set the interface to async.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim,
runtime/doc/terminal.txt

Patch 8.0.1294

Problem: GUI: get stuck when splitting a terminal window.
Solution: Stop blinking when values become zero. (Hirohito Higashi)
Files: src/gui.c

Patch 8.0.1295

Problem: Cannot automatically get a server name in a terminal.
Solution: Add the --enable-autoservername flag to configure. (Cimbali,
closes #2317)
Files: runtime/doc/eval.txt, runtime/doc/various.txt, src/config.h.in,
src/configure.ac, src/auto/configure, src/evalfunc.c,
src/feature.h, src/main.c, src/version.c, src/Makefile

Patch 8.0.1296 (after 8.0.1294)

Problem: Checking the same condition twice. (John Marriott)
Solution: Check blinkwait.
Files: src/gui.c

Patch 8.0.1297

Problem: +autoservername does not show enabled on MS-Windows.
Solution: Always define the flag on MS-Windows. (Ken Takata)
Files: src/feature.h

Patch 8.0.1298

Problem: Missing test file.
Solution: Add samples/test000. (Christian Brabandt)
Files: src/testdir/samples/test000, Filelist

Patch 8.0.1299

Problem: Bracketed paste does not work well in terminal window.
Solution: Send translated string to job right away. (Ozaki Kiichi, closes
#2341)
Files: src/terminal.c

Patch 8.0.1300

Problem: File permissions may end up wrong when writing.
Solution: Use fchmod() instead of chmod() when possible. Don't truncate

until we know we can change the file.
Files: src/os_unix.c, src/proto/os_unix.pro, src/configure.ac,
src/auto/configure, src/config.h.in, src/fileio.c

Patch 8.0.1301

Problem: Generated license file for NSIS has a modeline.
Solution: Adjust the pattern for sed. (Ken Takata)
Files: runtime/doc/Makefile

Patch 8.0.1302

Problem: Still too many old style tests.
Solution: Convert a few more tests to new style. (Yegappan Lakshmanan,
closes #2326)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_ming.mak,
src/testdir/Make_vms.mms, src/testdir/runtest.vim,
src/testdir/test68.in, src/testdir/test68.ok,
src/testdir/test73.in, src/testdir/test73.ok,
src/testdir/test_close_count.in, src/testdir/test_close_count.ok,
src/testdir/test_close_count.vim,
src/testdir/test_erasebackward.in,
src/testdir/test_erasebackward.ok,
src/testdir/test_erasebackward.vim,
src/testdir/test_find_complete.vim, src/testdir/test_fixeol.in,
src/testdir/test_fixeol.ok, src/testdir/test_fixeol.vim,
src/testdir/test_listchars.in, src/testdir/test_listchars.ok,
src/testdir/test_listchars.vim, src/testdir/test_textformat.vim

Patch 8.0.1303

Problem: **'ttermouse'** is not set to "sgr" for Terminal.app and Iterm2.
Solution: Recognize Iterm2 by the termresponse.
Files: src/term.c

Patch 8.0.1304

Problem: **CTRL-G/CTRL-T** don't work with incsearch and empty pattern.
Solution: Use the last search pattern. (Christian Brabandt, closes #2292)
Files: src/ex_getln.c, src/proto/search.pro, src/search.c,
src/testdir/test_search.vim

Patch 8.0.1305

Problem: writefile() never calls fsync().
Solution: Follow the **'fsync'** option with override to enable or disable.
Files: src/fileio.c, src/evalfunc.c, runtime/doc/eval.txt, src/globals.h,
src/testdir/test_writefile.vim

Patch 8.0.1306

Problem: ASAN error stack trace is not useful.
Solution: Add "asan_symbolize". (James McCoy, closes #2344)
Files: .travis.yml

Patch 8.0.1307 (after 8.0.1300)

Problem: Compiler warning for ignoring return value of ftruncate(). (Tony
Mechelynck)
Solution: Assign returned value to "ignore".
Files: src/fileio.c

Patch 8.0.1308

Problem: The "Reading from stdin" message may be undesired and there is no easy way to skip it.
Solution: Don't show the message with --not-a-term was used.
Files: src/fileio.c

Patch 8.0.1309

Problem: Cannot use 'balloonexpr' in a terminal.
Solution: Add 'balloonevalterm' and add code to handle mouse movements in a terminal. Initial implementation for Unix with GUI.
Files: src/option.c, src/option.h, src/os_unix.c, src/proto/os_unix.pro, src/feature.h, src/misc2.c, src/keymap.h, src/edit.c, src/ex_getln.c, src/message.c, src/misc1.c, src/normal.c, src/terminal.c, src/getchar.c, src/ex_cmds2.c, src/gui_beval.c, src/proto/gui_beval.pro, src/evalfunc.c, src/popupmnu.c, src/proto/popupmnu.pro, src/version.c, src/globals.h, src/gui.c, runtime/doc/options.txt, src/term.c, runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1310

Problem: Cproto generates errors because of missing type.
Solution: Define _Float128 when generating prototypes.
Files: src/vim.h

Patch 8.0.1311

Problem: No test for strpart().
Solution: Add a test. (Dominique Pelle, closes #2347)
Files: src/testdir/test_functions.vim

Patch 8.0.1312 (after 8.0.1309)

Problem: balloon_show() only works in terminal when compiled with the GUI.
Solution: Add FEAT_BEVAL_GUI and refactor to move common code out of the GUI specific file.
Files: src/feature.h, src/evalfunc.c, src/gui.c, src/gui_athena.c, src/gui_beval.c, src/proto/gui_beval.pro, src/beval.c, src/proto/beval.pro, src/gui_motif.c, src/gui_w32.c, src/gui_x11.c, src/integration.c, src/workshop.c, src/menu.c, src/netbeans.c, src/option.c, src/os_unix.c, src/os_win32.c, src/syntax.c, src/version.c, src/gui.h, src/gui_beval.h, src/vim.h, src/beval.h, src/option.h, src/ex_cmds2.c, src/ui.c, src/getchar.c, src/normal.c, src/popupmnu.c, src/globals.h, src/Makefile, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, Filelist

Patch 8.0.1313 (after 8.0.1312)

Problem: Missing dependencies cause parallel make to fail.
Solution: Update dependencies.
Files: src/Makefile

Patch 8.0.1314 (after 8.0.1312)

Problem: Build fails on Mac. (chdiza)
Solution: Add #ifdef around GUI fields.
Files: src/beval.h

Patch 8.0.1315 (after 8.0.1312)

Problem: Build still fails on Mac. (chdiza)

Solution: Remove bogus typedef.

Files: src/os_macosx.m

Patch 8.0.1316 (after 8.0.1312)

Problem: Build still still fails on Mac. (chdiza)

Solution: Remove another bogus typedef.

Files: src/os_mac_conv.c

Patch 8.0.1317

Problem: Accessing freed memory in term_wait(). (Dominique Pelle)

Solution: Check that the buffer still exists.

Files: src/terminal.c

Patch 8.0.1318

Problem: Terminal balloon only shows one line.

Solution: Split into several lines in a clever way. Add balloon_split().
Make balloon_show() accept a list in the terminal.

Files: src/popupmnu.c, src/proto/popupmnu.pro, src/evalfunc.c,
src/beval.c, src/proto/beval.pro, src/testdir/test_popup.vim,
runtime/doc/eval.txt,
runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1319

Problem: Can't build GUI on MS-Windows.

Solution: Don't define the balloon_split() function in a GUI-only build.

Files: src/evalfunc.c, runtime/doc/eval.txt

Patch 8.0.1320

Problem: Popup test fails on GUI-only build.

Solution: Don't test balloon_split() when it's not available.

Files: src/testdir/test_popup.vim

Patch 8.0.1321

Problem: Can't build huge version with Athena. (Mark Kelly)

Solution: Move including beval.h to before structs.h. Include beval.pro like
other proto files.

Files: src/vim.h, src/beval.h, src/proto.h

Patch 8.0.1322

Problem: Textformat test isn't run. (Yegappan Lakshmanan)

Solution: Add target to the list of tests.

Files: src/testdir/Make_all.mak

Patch 8.0.1323

Problem: Mouse events in a terminal window may cause endless loop.

Solution: Adjust position computation. Don't stuff a mouse event when
coming from normal_cmd().

Files: src/normal.c, src/terminal.c

Patch 8.0.1324

Problem: Some xterm sends different mouse move codes.

Solution: Also accept 0x80 as a move event.
Files: src/term.c

Patch 8.0.1325

Problem: More tests are not run.
Solution: Add targets to the list of tests. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak

Patch 8.0.1326

Problem: Largefile test fails on CI, glob test on MS-Windows.
Solution: Remove largefile test from list of all tests. Don't run Test_glob() on non-unix systems. More cleanup. (Yegappan Lakshmanan, closes #2354)
Files: src/testdir/Make_all.mak, src/testdir/test_escaped_glob.vim, src/testdir/test_plus_arg_edit.vim

Patch 8.0.1327

Problem: New proto file missing from distribution.
Solution: Add it. (closes #2355)
Files: Filelist

Patch 8.0.1328

Problem: Trouble when using ":term ++close" with autocmd. (Gabriel Barta)
Solution: Use aucmd_prepbuff() and aucmd_restbuff() instead of setting curbuf. (closes #2339)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1329

Problem: When a flaky test fails it also often fails the second time.
Solution: Sleep a couple of seconds before the second try.
Files: src/testdir/runtest.vim

Patch 8.0.1330

Problem: MS-Windows: job in terminal can't get back to Vim.
Solution: set VIM_SERVERNAME in the environment. (Yasuhiro Matsumoto, closes #2360)
Files: runtime/doc/terminal.txt, src/os_win32.c, src/proto/os_win32.pro, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1331

Problem: Possible crash when window can be zero lines high. (Joseph Dornisch)
Solution: Only set w_fraction if the window is at least two lines high.
Files: src/window.c

Patch 8.0.1332

Problem: Highlighting in quickfix window could be better. (Axel Bender)
Solution: Use the qfSeparator highlight item. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1333

Problem: Some tests are run twice.
Solution: Invoked most utf8 tests only from test_alot_utf8. (Yegappan Lakshmanan, closes #2369)

Files: src/testdir/Make_all.mak, src/testdir/test_alot_utf8.vim,
src/testdir/test_mksession_utf8.vim

Patch 8.0.1334

Problem: Splitting a window with a WinBar damages window layout.
(Lifepillar)

Solution: Take the winbar into account when computing the new window
position. Add WINBAR_HEIGHT().

Files: src/vim.h, src/window.c

Patch 8.0.1335

Problem: writefile() using fsync() may give an error for a device.
(Yasuhiro Matsumoto)

Solution: Ignore fsync() failing. (closes #2373)

Files: src/evalfunc.c

Patch 8.0.1336

Problem: Cannot use imactivatefunc() unless compiled with +xim.

Solution: Allow using imactivatefunc() when not compiled with +xim.
(Yasuhiro Matsumoto, closes #2349)

Files: runtime/doc/options.txt, runtime/doc/mbyte.txt, src/mbyte.c,
src/option.c, src/option.h, src/structs.h,
src/testdir/test_iminsert.vim, src/Makefile,
src/testdir/Make_all.mak, src/vim.h

Patch 8.0.1337 (after 8.0.1336)

Problem: Typo in #ifdef.

Solution: Fix the #if line.

Files: src/mbyte.c

Patch 8.0.1338 (after 8.0.1337)

Problem: USE_IM_CONTROL is confusing and incomplete.

Solution: Just use FEAT_MBYTE. Call **'imactivatefunc'** also without GUI.

Files: src/vim.h, src/edit.c, src/ex_getln.c, src/getchar.c, src/gui.c,
src/gui_mac.c, src/gui_w32.c, src/mbyte.c, src/normal.c,
src/option.c, src/ui.c, src/globals.h, src/option.h

Patch 8.0.1339

Problem: No test for what 8.0.1335 fixes.

Solution: Add a test. (Yasuhiro Matsumoto, closes #2373)

Files: src/testdir/test_writefile.vim

Patch 8.0.1340

Problem: MS-Windows: cannot build GUI without IME.

Solution: Define im_get_status() and im_set_active() when IME is not used.

Files: src/mbyte.c

Patch 8.0.1341

Problem: **'imactivatefunc'** test fails on MS-Windows.

Solution: Skip the text.

Files: src/testdir/test_iminsert.vim, runtime/doc/options.txt

Patch 8.0.1342

Problem: Cannot build with Motif and multibyte. (Mohamed Boughaba)

Solution: Use the right input method status flag. (closes #2374)
Files: src/mbyte.c

Patch 8.0.1343

Problem: MS-Windows: does not show colored emojis.
Solution: Implement colored emojis. Improve drawing speed. Make **'taamode'** work. (Taro Muraoka, Yasuhiro Matsumoto, Ken Takata, close #2375)
Files: appveyor.yml, runtime/doc/options.txt, src/gui_dwrite.cpp, src/gui_dwrite.h, src/gui_w32.c, src/proto/gui_w32.pro

Patch 8.0.1344

Problem: Using **'imactivatefunc'** in the GUI does not work.
Solution: Do not use **'imactivatefunc'** and **'imstatusfunc'** in the GUI.
Files: runtime/doc/options.txt, src/mbyte.c, src/testdir/test_iminsert.vim

Patch 8.0.1345

Problem: Race condition between stat() and open() for the viminfo temp file. (Simon Ruderich)
Solution: use open() with O_EXCL to atomically check if the file exists. Don't try using a temp file, renaming it will fail anyway.
Files: src/ex_cmds.c

Patch 8.0.1346

Problem: Crash when passing 50 char string to balloon_split().
Solution: Fix off-by-one error.
Files: src/testdir/test_popup.vim, src/popupmnu.c

Patch 8.0.1347

Problem: MS-Windows: build broken by misplaced curly.
Solution: Move curly after #endif.
Files: src/ex_cmds.c

Patch 8.0.1348

Problem: Make testclean deletes script file on MS-Windows.
Solution: Rename file to avoid it starting with an "x".
Files: src/testdir/xterm_ramp.vim, src/testdir/color_ramp.vim, Filelist

Patch 8.0.1349

Problem: Options test fails when using Motif or GTK GUI.
Solution: Use "fixed" instead of "fixedsys" for Unix. Don't try "xxx" for guifonteset. Don't set **'termencoding'** to anything but "utf-8" for GTK. Give an error if **'termencoding'** can't be converted.
Files: src/testdir/gen_opt_test.vim, src/option.c

Patch 8.0.1350

Problem: Cannot build with +eval and -multi_byte.
Solution: Adjust #ifdefs. (John Marriott) Always include the multi_byte feature when an input method feature is enabled.
Files: src/mbyte.c, src/feature.h

Patch 8.0.1351

Problem: Warning for unused variables building with MinGW.
Solution: Change a few #ifdefs (suggested by John Marriott). Remove

Files: superfluous checks of FEAT_MBYTE.
 src/gui_w32.c

Patch 8.0.1352
Problem: Dead URLs in the help go unnoticed.
Solution: Add a script to check URLs in the help files. (Christian Brabandt)
Files: runtime/doc/Makefile, runtime/doc/test_urls.vim, Filelist

Patch 8.0.1353
Problem: QuickFixCmdPost is not used consistently.
Solution: Invoke QuickFixCmdPost consistently after QuickFixCmdPre.
 (Yegappan Lakshmanan, closes #2377)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1354
Problem: Shift-Insert doesn't always work in MS-Windows console.
Solution: Handle K_NUL differently. (Yasuhiro Matsumoto, closes #2381)
Files: src/os_win32.c

Patch 8.0.1355 (after 8.0.1354)
Problem: Cursor keys don't work in MS-Windows console.
Solution: Revert the previous patch. Also delete dead code.
Files: src/os_win32.c

Patch 8.0.1356
Problem: Using simalt in a GUIEnter autocommand inserts strange characters.
 (Chih-Long Chang)
Solution: Ignore K_NOP in Insert mode. (closes #2379)
Files: src/edit.c, src/ex_getln.c

Patch 8.0.1357
Problem: Startup test fails on OpenBSD. (Edd Barrett)
Solution: Check for "BSD" instead of "FreeBSD" being defined. (James McCoy,
 closes #2376, closes #2378)
Files: src/vim.h

Patch 8.0.1358
Problem: Undercurl is not used in the terminal. (Kovid Goyal)
Solution: Only fall back to underline when undercurl highlighting is not
 defined. (closes #1306)
Files: src/screen.c

Patch 8.0.1359
Problem: Libvterm ANSI colors can not always be recognized from the RGB
 values. The default color is wrong when t_RB is empty.
Solution: Add the ANSI color index to VTermColor.
Files: src/libvterm/include/vterm.h, src/libvterm/src/pen.c,
 src/terminal.c

Patch 8.0.1360
Problem: The Terminal highlighting doesn't work in a terminal. (Ozaki
 Kiichi)
Solution: Use the Terminal highlighting when the cterm index is zero.
Files: src/terminal.c

Patch 8.0.1361

Problem: Some users don't want to diff with hidden buffers.
Solution: Add the "hiddenoff" item to 'diffopt'. (Alisue, closes #2394)
Files: runtime/doc/options.txt, src/buffer.c, src/diff.c,
src/proto/diff.pro, src/testdir/test_diffmode.vim

Patch 8.0.1362

Problem: Terminal window colors wrong when using Terminal highlighting.
Solution: Set ansi_index when setting the default color. Also cache the color index for Terminal. (Ozaki Kiichi, closes #2393)
Files: src/libvterm/src/pen.c, src/proto/terminal.pro, src/syntax.c,
src/terminal.c

Patch 8.0.1363

Problem: Recovering does not work when swap file ends in .stz.
Solution: Check for all possible swap file names. (Elfling, closes #2395,
closes #2396)
Files: src/memline.c

Patch 8.0.1364

Problem: There is no easy way to get the window position.
Solution: Add win_screenpos().
Files: src/evalfunc.c, src/testdir/test_window_cmd.vim,
runtime/doc/eval.txt

Patch 8.0.1365

Problem: When one channel test fails others fail as well.
Solution: Stop the job after a failure. Also add a couple of tests to the list of flaky tests.
Files: src/testdir/test_channel.vim, src/testdir/runtest.vim

Patch 8.0.1366

Problem: Balloon shows when cursor is in WinBar.
Solution: Don't show the balloon when row is negative.
Files: src/beval.c

Patch 8.0.1367

Problem: terminal test hangs, executing abcde. (Stucki)
Solution: Rename abcde to abxde.
Files: src/testdir/test_terminal.vim

Patch 8.0.1368

Problem: Cannot drag status line or vertical separator of new terminal window. (UncleBill)
Solution: Adjust mouse row and column computation. (Yasuhiro Matsumoto, closes #2410)
Files: src/terminal.c

Patch 8.0.1369

Problem: MS-Windows: drawing underline, curl and strikethrough is slow, mFallbackDC not properly updated.
Solution: Several performance improvements. (Ken Takata, Taro Muraoka, Yasuhiro Matsumoto, closes #2401)

Files: runtime/doc/options.txt, src/gui_dwrite.cpp, src/gui_dwrite.h,
src/gui_w32.c

Patch 8.0.1370

Problem: Channel test for callback is flaky.
Solution: Add the test to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.1371

Problem: Shift-Insert doesn't always work in MS-Windows console.
Solution: Handle K_NUL differently if the second character is more than one
byte. (Yasuhiro Matsumoto, closes #2381)
Files: src/os_win32.c

Patch 8.0.1372

Problem: Profile log may be truncated halfway a character.
Solution: Find the start of the character. (Ozaki Kiichi, closes #2385)
Files: src/ex_cmds2.c, src/testdir/test_profile.vim

Patch 8.0.1373

Problem: No error when setting '**renderoptions**' to an invalid value before
starting the GUI.
Solution: Always check the value. (Ken Takata, closes #2413)
Files: src/gui_w32.c, src/option.c

Patch 8.0.1374

Problem: **CTRL-A** does not work with an empty line. (Alex)
Solution: Decrement the end only once. (Hirohito Higashi, closes #2387)
Files: src/ops.c, src/testdir/test_increment.vim

Patch 8.0.1375

Problem: Window size wrong after maximizing with WinBar. (Lifepillar)
Solution: Fix height computations. Redraw window when it is zero height but
has a WinBar. (closes #2356)
Files: src/window.c, src/screen.c, src/vim.h

Patch 8.0.1376

Problem: Cursor in terminal not always updated.
Solution: Call gui_mch_flush(). (Ken Takata)
Files: src/terminal.c

Patch 8.0.1377

Problem: Cannot call a dict function in autoloading dict.
Solution: Call get_lval() passing the read-only flag.
Files: src/userfunc.c, src/eval.c, src/testdir/sautest/autoload/foo.vim,
src/testdir/sautest/autoload/globone.vim,
src/testdir/sautest/autoload/globtwo.vim,
src/testdir/test_escaped_glob.vim, src/Makefile,
src/testdir/test_autoload.vim, src/Makefile,
src/testdir/Make_all.mak

Patch 8.0.1378

Problem: Autoload script sources itself when defining function.
Solution: Pass TFN_NO_AUTOLOAD to trans_function_name(). (Yasuhiro)

Matsumoto, closes #2423)
Files: src/userfunc.c, src/testdir/test_autoload.vim,
src/testdir/sautest/autoload/sourced.vim

Patch 8.0.1379

Problem: Configure check for selinux does not check for header file.
Solution: Add an AC_CHECK_HEADER(). (Benny Siegert)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1380

Problem: When recovering a file with "vim -r swapfile" the hit-enter prompt
is at the top of the window.
Solution: Invalidate the cursor position.
Files: src/term.c

Patch 8.0.1381

Problem: ch_readraw() waits for NL if channel mode is NL.
Solution: Pass a "raw" flag to channel_read_block(). (Yasuhiro Matsumoto)
Files: src/channel.c, src/proto/channel.pro,
src/testdir/test_channel.vim, src/testdir/test_channel_pipe.py

Patch 8.0.1382

Problem: Get "no write since last change" message if a terminal is open.
(Fritz mehner)
Solution: Don't consider a buffer changed if it's a terminal window.
Files: src/ex_cmds.c, src/undo.c, src/proto/undo.pro

Patch 8.0.1383

Problem: Local additions in help skips some files. (joshklod)
Solution: Check the base file name length equals.
Files: src/ex_cmds.c, src/testdir/test_help.vim

Patch 8.0.1384

Problem: Not enough quickfix help; confusing winid.
Solution: Add more examples in the help. When the quickfix window is not
present, return zero for getqflist() with 'winid'. Add more tests
for jumping to quickfix list entries. (Yegappan Lakshmanan, closes
#2427)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1385

Problem: Python 3.5 is getting old.
Solution: Make Python 3.6 the default. (Ken Takata, closes #2429)
Files: runtime/doc/if_pyth.txt, src/INSTALLpc.txt, src/Make_cyg_ming.mak,
src/Make_mvc.mak, src/bigvim.bat

Patch 8.0.1386

Problem: Cannot select modified buffers with getbufinfo().
Solution: Add the "bufmodified" flag. (Yegappan Lakshmanan, closes #2431)
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_bufwintabinfo.vim

Patch 8.0.1387

Problem: Wordcount test is old style.
Solution: Change into a new style test. (Yegappan Lakshmanan, closes #2434)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_ming.mak,
src/testdir/Make_vms.mms, src/testdir/test_wordcount.in,
src/testdir/test_wordcount.ok, src/testdir/test_wordcount.vim

Patch 8.0.1388

Problem: Char not overwritten with ambiguous width char, if the ambiguous
char is single width but we reserve double-width space.
Solution: First clear the screen cells. (Ozaki Kiichi, closes #2436)
Files: src/screen.c

Patch 8.0.1389

Problem: getqflist() items are missing if not set, that makes it more
difficult to handle the values.
Solution: When a value is not available return zero or another invalid
value. (Yegappan Lakshmanan, closes #2430)
Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1390

Problem: DirectX scrolling can be slow, vertical positioning is off.
Solution: Make scroll slightly faster when using "scrlines:1". Fix y
position of displayed text. Fix DirectX with non-utf8 encoding.
(Ken Takata, closes #2440)
Files: src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak,
src/gui_dwrite.cpp, src/gui_w32.c

Patch 8.0.1391

Problem: Encoding empty string to JSON sometimes gives "null".
Solution: Handle NULL string as empty string. (closes #2446)
Files: src/testdir/test_json.vim, src/json.c

Patch 8.0.1392

Problem: Build fails with --with-features=huge --disable-channel.
Solution: Don't enable the terminal feature when the channel feature is
missing. (Dominique Pelle, closes #2453)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1393

Problem: Too much highlighting with '**hlsearch**' and '**incsearch**' set.
Solution: Do not highlight matches when the pattern matches everything.
Files: src/ex_getln.c

Patch 8.0.1394

Problem: Cannot intercept a yank command.
Solution: Add the TextYankPost autocommand event. (Philippe Vaucher et al.,
closes #2333)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, src/dict.c,
src/eval.c, src/fileio.c, src/ops.c, src/proto/dict.pro,
src/proto/eval.pro, src/proto/fileio.pro,
src/testdir/test_autocmd.vim, src/vim.h

Patch 8.0.1395

Problem: It is not easy to see if a colorscheme is well written.
Solution: Add a script that checks for common mistakes. (Christian Brabandt)
Files: runtime/colors/check_colors.vim, runtime/colors/README.txt

Patch 8.0.1396

Problem: Memory leak when **CTRL-G** in search command line fails.
Solution: Move restore_last_search_pattern to after "if".
Files: src/ex_getln.c

Patch 8.0.1397

Problem: Pattern with \& following nothing gives an error.
Solution: Emit an empty node when needed.
Files: src/regexp_nfa.c, src/testdir/test_search.vim

Patch 8.0.1398

Problem: :packadd does not load packages from the "start" directory.
(Alejandro Hernandez)
Solution: Make :packadd look in the "start" directory if those packages were not loaded on startup.
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 8.0.1399

Problem: Warnings and errors when building tiny version. (Tony Mechelynck)
Solution: Add #ifdefs.
Files: src/ex_getln.c, src/ops.c

Patch 8.0.1400

Problem: Color scheme check script shows up as color scheme.
Solution: Move it to the "tools" subdirectory. (closes #2457)
Files: Filelist, runtime/colors/check_colors.vim,
runtime/colors/tools/check_colors.vim, runtime/colors/README.txt

Patch 8.0.1401

Problem: Cannot build with GTK but without XIM. (Guido)
Solution: Adjust #ifdef. (closes #2461)
Files: src/gui.c

Patch 8.0.1402

Problem: Crash with nasty autocommand. (gy741, Dominique Pelle)
Solution: Check that the new current buffer isn't wiped out. (closes #2447)
Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 8.0.1403

Problem: Using freed buffer in grep command. (gy741, Dominique Pelle)
Solution: Lock the dummy buffer to avoid autocommands wiping it out.
Files: src/quickfix.c, src/testdir/test_autocmd.vim

Patch 8.0.1404

Problem: Invalid memory access on exit when autocommands wipe out a buffer.
(gy741, Dominique Pelle)
Solution: Check if the buffer is still valid. (closes #2449)
Files: src/main.c

Patch 8.0.1405

Problem: Duplicated code for getting a typed character. CursorHold is called too often in the GUI. (lilydjwg)
Solution: Refactor code to move code up from mch_inchar(). Don't fire CursorHold if feedkeys() was used. (closes #2451)
Files: src/gui.c, src/proto/gui.pro, src/main.c, src/ui.c, src/proto/ui.pro, src/os_unix.c

Patch 8.0.1406

Problem: Difficult to track changes to a quickfix list.
Solution: Add a "changedtick" value. (Yegappan Lakshmanan, closes #2460)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1407

Problem: GUI: CursorHold may trigger before **'updatetime'** when using timers.
Solution: Check that **'updatetime'** has passed.
Files: src/gui.c

Patch 8.0.1408

Problem: Crash in setqflist().
Solution: Check for string to be NULL. (Dominique Pelle, closes #2464)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1409

Problem: Buffer overflow in :tags command.
Solution: Use vim_snprintf(). (Dominique Pelle, closes #2471, closes #2475)
Add a test.
Files: src/testdir/test_taglist.vim, src/tag.c

Patch 8.0.1410

Problem: Hang when using count() with an empty string.
Solution: Return zero for an empty string. (Dominique Pelle, closes #2465)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.0.1411

Problem: Reading invalid memory with **CTRL-W** :.
Solution: Correct the command characters. (closes #2469)
Files: src/normal.c, src/testdir/test_window_cmd.vim, src/ops.c

Patch 8.0.1412

Problem: Using free memory using setloclist(). (Dominique Pelle)
Solution: Mark location list context as still in use when needed. (Yegappan Lakshmanan, closes #2462)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1413

Problem: Accessing freed memory in :cbuffer.
Solution: Get quickfix list after executing autocmds. (closes #2470)
Files: src/quickfix.c, src/testdir/test_autocmd.vim

Patch 8.0.1414

Problem: Accessing freed memory in :lfile.
Solution: Get the current window after executing autocommands. (Yegappan

Lakshmanan, closes #2473)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1415

Problem: Warning for unused function without timers feature.
Solution: Add #ifdef. (John Marriott)
Files: src/gui.c

Patch 8.0.1416

Problem: Crash when searching for a sentence.
Solution: Return NUL when getting character at MAXCOL. (closes #2468)
Files: src/misc1.c, src/misc2.c, src/testdir/test_search.vim,
src/ex_docmd.c

Patch 8.0.1417

Problem: Test doesn't search for a sentence. Still fails when searching for
start of sentence. (Dominique Pelle)
Solution: Add paren. Check for MAXCOL in dec().
Files: src/testdir/test_search.vim, src/misc2.c

Patch 8.0.1418

Problem: No test for expanding backticks.
Solution: Add a test. (Dominique Pelle, closes #2479)
Files: src/testdir/test_normal.vim

Patch 8.0.1419

Problem: Cursor column is not updated after]s. (Gary Johnson)
Solution: Set the curswant flag.
Files: src/testdir/test_spell.vim, src/normal.c, src/evalfunc.c

Patch 8.0.1420

Problem: Accessing freed memory in vimgrep.
Solution: Check that the quickfix list is still valid. (Yegappan Lakshmanan,
closes #2474)
Files: src/quickfix.c, src/testdir/test_autocmd.vim,
src/testdir/test_quickfix.vim

Patch 8.0.1421

Problem: Accessing invalid memory with overlong byte sequence.
Solution: Check for NUL character. (test by Dominique Pelle, closes #2485)
Files: src/misc2.c, src/testdir/test_functions.vim

Patch 8.0.1422

Problem: No fallback to underline when undercurl is not set. (Ben Jackson)
Solution: Check for the value to be empty instead of NULL. (closes #2424)
Files: src/screen.c

Patch 8.0.1423

Problem: Error in return not caught by try/catch.
Solution: Call update_force_abort(). (Yasuhiro Matsumoto, closes #2483)
Files: src/testdir/test_eval.in, src/testdir/test_eval_stuff.vim,
src/Makefile, src/testdir/Make_all.mak, src/userfunc.c

Patch 8.0.1424

Problem: The timer_pause test is flaky on Travis.
Solution: Accept a longer sleep time on Mac.
Files: src/testdir/test_timers.vim

Patch 8.0.1425

Problem: execute() does not work in completion of user command. (thinca)
Solution: Switch off redir_off and restore it. (Ozaki Kiichi, closes #2492)
Files: src/evalfunc.c, src/testdir/test_usercommands.vim

Patch 8.0.1426

Problem: "gf" and <cf> don't accept ? and & in URL. (Dmitrii Tcyganok)
Solution: Check for a URL and allow for extra characters. (closes #2493)
Files: src/window.c, src/testdir/test_gf.vim

Patch 8.0.1427

Problem: The :leftabove modifier doesn't work for :copen.
Solution: Respect the split modifier. (Yegappan Lakshmanan, closes #2496)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1428

Problem: Compiler warning on 64 bit MS-Windows system.
Solution: Change type from "int" to "size_t". (Mike Williams)
Files: src/ex_getln.c

Patch 8.0.1429

Problem: Crash when calling term_start() with empty argument.
Solution: Check for invalid argument. (Yasuhiro Matsumoto, closes #2503)
Fix memory leak.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1430 (after 8.0.1429)

Problem: Crash when term_start() fails.
Solution: Initialize winpty_err.
Files: src/terminal.c

Patch 8.0.1431

Problem: MS-Windows: vimtutor fails if %TMP% has special chars.
Solution: Add quotes. (Tamce, closes #2561)
Files: vimtutor.bat

Patch 8.0.1432

Problem: After ":copen" can't get the window-ID of the quickfix window.
(FalacerSelene)
Solution: Make it work without a quickfix list. Add a test. (Yegappan
Lakshmanan, closes #2541)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1433

Problem: Illegal memory access after undo. (Dominique Pelle)
Solution: Avoid the column becomes negative. (Christian Brabant, closes #2533)
Files: src/mbyte.c, src/testdir/test_undo.vim

Patch 8.0.1434

Problem: GTK: :promptfind does not put focus on text input. (Adam Novak)
Solution: When re-opening the dialog put focus on the text input. (Kazunobu Kuriyama, closes #2563)
Files: src/gui_gtk.c

Patch 8.0.1435

Problem: Memory leak in test_arabic.
Solution: Free the from and to parts. (Christian Brabandt, closes #2569)
Files: src/buffer.c, src/digraph.c, src/proto/digraph.pro

Patch 8.0.1436

Problem: Not enough information about what Python version may work.
Solution: Add "python_compiled", "python3_compiled", "python_dynamic" and "python3_dynamic" values for has().
Files: src/evalfunc.c, runtime/doc/eval.txt

Patch 8.0.1437

Problem: Pkg-config doesn't work with cross compiling.
Solution: Use AC_PATH_TOOL() instead of AC_PATH_PROG(). (James McCoy, closes #2513)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1438

Problem: Filetype detection test not updated for change.
Solution: Update the test.
Files: src/testdir/test_filetype.vim

Patch 8.0.1439

Problem: If cscope fails a search Vim may hang.
Solution: Bail out when a search error is encountered. (Safouane Baroudi, closes #2598)
Files: src/if_cscope.c

Patch 8.0.1440

Problem: Terminal window: some vterm responses are delayed.
Solution: After writing input. check if there is output to read. (Ozaki Kiichi, closes #2594)
Files: src/terminal.c, src/testdir/test_search.vim, src/testdir/test_terminal.vim

Patch 8.0.1441

Problem: Using ":undo 0" leaves undo in wrong state.
Solution: Instead of searching for state 1 and go above, just use the start. (Ozaki Kiichi, closes #2595)
Files: src/undo.c, src/testdir/test_undo.vim

Patch 8.0.1442 (after 8.0.1439)

Problem: Using pointer before it is set.
Solution: Search in whole buffer instead of next token.
Files: src/if_cscope.c

Patch 8.0.1443 (after 8.0.1441)

Problem: Compiler complains about uninitialized variable. (Tony Mechelynck)
Solution: Assign a value to the variable.

Files: src/undo.c

Patch 8.0.1444

Problem: Missing `-D_FILE_OFFSET_BITS=64` may cause problems if a library is compiled with it.

Solution: Include `-D_FILE_OFFSET_BITS` if some CFLAGS has it. (James McCoy, closes #2600)

Files: src/configure.ac, src/auto/configure

Patch 8.0.1445

Problem: Cannot act on edits in the command line.

Solution: Add the `CmdlineChanged` autocommand event. (xtal8, closes #2603, closes #2524)

Files: runtime/doc/autocmd.txt, src/ex_getln.c, src/fileio.c, src/testdir/test_autocmd.vim, src/vim.h

Patch 8.0.1446

Problem: Accessing freed memory after window command in auto command. (gy741)

Solution: Adjust the pointer in the parent frame. (Christian Brabandt, closes #2467)

Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.0.1447

Problem: Still too many old style tests.

Solution: Turn a few tests into new style. (Yegappan Lakshmanan, closes #2509)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/main.aap, src/testdir/test15.in, src/testdir/test15.ok, src/testdir/test36.in, src/testdir/test36.ok, src/testdir/test50.in, src/testdir/test50.ok, src/testdir/test_regex_char_classes.vim, src/testdir/test_shortpathname.vim, src/testdir/test_textformat.vim

Patch 8.0.1448

Problem: Segmentation fault when Ruby throws an exception inside `:rubyfile` command.

Solution: Use `rb_protect()` instead of `rb_load_protect()`. (ujihisa, closes #2147, greywolf, closes #2512, #2511)

Files: src/if_ruby.c, src/testdir/test_ruby.vim

Patch 8.0.1449

Problem: Slow redrawing with DirectX.

Solution: Avoid calling `gui_mch_flush()` unnecessarily, especially when updating the cursor. (Ken Takata, closes #2560)

Files: runtime/doc/options.txt, src/channel.c, src/edit.c, src/getchar.c, src/gui.c, src/gui_dwrite.cpp, src/gui_dwrite.h, src/gui_w32.c, src/macros.h, src/main.c, src/message.c, src/netbeans.c, src/proto/gui.pro, src/proto/term.pro, src/screen.c, src/search.c, src/term.c, src/ui.c

Patch 8.0.1450

Problem: Endless loop when `gui_mch_stop_blink()` is called while `blink_state`

is BLINK_OFF. (zdohnal)
Solution: Avoid calling gui_update_cursor() recursively.
Files: src/gui.c, src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro,
src/gui_mac.c, src/proto/gui_mac.pro, src/gui_photon.c,
src/proto/gui_photon.pro, src/gui_w32.c, src/proto/gui_w32.pro,
src/gui_x11.c, src/proto/gui_x11.pro

Patch 8.0.1451

Problem: It is difficult to set the python home directory properly for Python 2.7 and 3.5 since both use \$PYTHONHOME.
Solution: Add the **'pythonhome'** and **'pythonthreehome'** options. (Kazuki Sakamoto, closes #1266)
Files: runtime/doc/options.txt, runtime/doc/quickref.txt,
runtime/optwin.vim, src/if_python.c, src/if_python3.c,
src/option.c, src/option.h

Patch 8.0.1452

Problem: Terminal test fails on some systems. (jonathonf)
Solution: Use "cat" instead of Python to produce the input. Add a delay. (closes #2607)
Files: src/testdir/test_terminal.vim

Patch 8.0.1453

Problem: Terminal test fails on some slow terminals.
Solution: Increase timeout to 10 seconds.
Files: src/testdir/test_terminal.vim

Patch 8.0.1454

Problem: When in silent mode too much output is buffered.
Solution: Use line buffering instead of fully buffered. (Brian M. Carlson, closes #2537)
Files: src/main.c

Patch 8.0.1455

Problem: If \$SHELL contains a space then the default value of **'shell'** is incorrect. (Matthew Horan)
Solution: Escape spaces in \$SHELL. (Christian Brabandt, closes #459)
Files: src/option.c, runtime/doc/options.txt,
src/testdir/test_startup.vim

Patch 8.0.1456

Problem: Timer test on travis Mac is still flaky.
Solution: Increase time range a bit more.
Files: src/testdir/test_timers.vim

Patch 8.0.1457

Problem: Clojure now supports a shebang line.
Solution: Detect clojure script from the shebang line. (David Burgin, closes #2570)
Files: runtime/scripts.vim

Patch 8.0.1458

Problem: Filetype detection test does not check all scripts.
Solution: Add most scripts to the test

Files: src/testdir/test_filetype.vim

Patch 8.0.1459

Problem: Cannot handle change of directory.

Solution: Add the DirChanged autocommand event. (Andy Massimino, closes #888) Avoid changing directory for '**autochdir**' too often.

Files: runtime/doc/autocmd.txt, src/buffer.c, src/ex_docmd.c, src/fileio.c, src/main.c, src/vim.h, src/proto/misc2.pro, src/gui_mac.c, src/netbeans.c, src/os_win32.c, src/testdir/test_autocmd.vim

Patch 8.0.1460 (after 8.0.1459)

Problem: Missing file in patch.

Solution: Add changes to missing file.

Files: src/misc2.c

Patch 8.0.1461 (after 8.0.1459)

Problem: Missing another file in patch.

Solution: Add changes to missing file.

Files: src/ex_cmds.c

Patch 8.0.1462 (after 8.0.1459)

Problem: Missing yet another file in patch.

Solution: Add changes to missing file.

Files: src/gui.c

Patch 8.0.1463

Problem: Test fails without '**autochdir**' option.

Solution: Skip test if '**autochdir**' is not supported.

Files: src/testdir/test_autocmd.vim

Patch 8.0.1464

Problem: Completing directory after :find does not add slash.

Solution: Adjust the flags for globpath(). (Genki Sky)

Files: src/misc1.c, src/testdir/test_find_complete.vim

Patch 8.0.1465

Problem: Python2 and python3 detection not tested. (Matej Cepl)

Solution: Add test for detecting python2 and python3. Also detect a script using "js" as javascript.

Files: runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.0.1466

Problem: Older GTK versions don't have gtk_entry_get_text_length().

Solution: Add a function with #ifdefs to take care of GTK version differences. (Kazunobu Kuriyama, closes #2605)

Files: src/gui_gtk.c

Patch 8.0.1467

Problem: Libvterm doesn't handle illegal byte sequence correctly.

Solution: After the invalid code check if there is space to store another character. Allocate one more character. (zhykzhykzhyk, closes #2614, closes #2613)

Files: src/libvterm/src/encoding.c, src/libvterm/src/state.c

Patch 8.0.1468

Problem: Illegal memory access in del_bytes().
Solution: Check for negative byte count. (Christian Brabandt, closes #2466)
Files: src/message.c, src/misc1.c

Patch 8.0.1469

Problem: When package path is a symlink adding it to 'runtimepath' happens at the end.
Solution: Do not resolve symlinks before locating the position in 'runtimepath'. (Ozaki Kiichi, closes #2604)
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 8.0.1470

Problem: Integer overflow when using regexp pattern. (geeknik)
Solution: Use a long instead of int. (Christian Brabandt, closes #2251)
Files: src/regexp_nfa.c

Patch 8.0.1471 (after 8.0.1401)

Problem: On MS-Windows CursorIM highlighting no longer works.
Solution: Adjust #if statements. (Ken Takata)
Files: src/gui.c

Patch 8.0.1472

Problem: MS-Windows: nsis installer is a bit slow.
Solution: Use ReserveFile for vimrc.ini. (Ken Takata, closes #2522)
Files: nsis/gvim.nsi

Patch 8.0.1473

Problem: MS-Windows: D&D fails between 32 and 64 bit apps.
Solution: Add the /HIGHENTROPYVA:NO linker option. (Ken Takata, closes #2504)
Files: src/Make_mvc.mak

Patch 8.0.1474

Problem: Visual C 2017 has multiple MSVCVER numbers.
Solution: Assume the 2017 version if MSVCVER >= 1910. (Leonardo Valeri Manera, closes #2619)
Files: src/Make_mvc.mak

Patch 8.0.1475

Problem: Invalid memory access in read_redo(). (gy741)
Solution: Convert the replacement character back from a negative number to CR or NL. (hint by Dominique Pelle, closes #2616)
Files: src/testdir/test_undo.vim, src/normal.c, src/vim.h, src/ops.c

Patch 8.0.1476

Problem: Screen isn't always updated right away.
Solution: Adjust #ifdef: Call out_flush() when not running the GUI.
Files: src/screen.c

Patch 8.0.1477

Problem: Redraw flicker when moving the mouse outside of terminal window.
Solution: Instead of updating the cursor color and shape every time leaving and entering a terminal window, only update when different from

the previously used cursor.
Files: src/terminal.c

Patch 8.0.1478

Problem: Unnecessary condition for "len" being zero.
Solution: Remove the condition. (Dominique Pelle)
Files: src/regexp_nfa.c

Patch 8.0.1479

Problem: Insert mode completion state is confusing.
Solution: Move ctrl_x_mode into edit.c. Add CTRL_X_NORMAL for zero.
Files: src/edit.c, src/globals.h, src/proto/edit.pro, src/search.c,
src/getchar.c

Patch 8.0.1480 (after 8.0.1479)

Problem: Patch missing change.
Solution: Add missing change.
Files: src/evalfunc.c

Patch 8.0.1481

Problem: Clearing a pointer takes two lines.
Solution: Add vim_clear() to free and clear the pointer.
Files: src/misc2.c, src/proto/misc2.pro, src/edit.c

Patch 8.0.1482

Problem: Using feedkeys() does not work to test Insert mode completion.
(Lifepillar)
Solution: Do not check for typed keys when executing :normal or feedkeys().
Fix thesaurus completion not working when 'complete' is empty.
Files: src/edit.c, src/testdir/test_ins_complete.vim,
src/testdir/test_popup.vim, src/testdir/test_edit.vim

Patch 8.0.1483

Problem: searchpair() might return an invalid value on timeout.
Solution: When the second search times out, do not accept a match from the
first search. (Daniel Hahler, closes #2552)
Files: src/search.c

Patch 8.0.1484

Problem: Redundant conditions.
Solution: Remove them. (Dominique Pelle)
Files: src/terminal.c

Patch 8.0.1485

Problem: Weird autocmd may cause arglist to be changed recursively.
Solution: Prevent recursively changing the argument list. (Christian
Brabandt, closes #2472)
Files: src/ex_docmd.c, src/globals.h

Patch 8.0.1486

Problem: Accessing invalid memory with "it". (Dominique Pelle)
Solution: Avoid going over the end of the line. (Christian Brabandt,
closes #2532)
Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.0.1487 (after 8.0.1486)
Problem: Test 14 fails.
Solution: Fix of-by-one error.
Files: src/search.c

Patch 8.0.1488 (after 8.0.1218)
Problem: Emacs tags no longer work. (zdohnal)
Solution: Do not skip over end of line.
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.0.1489
Problem: There is no easy way to get the global directory, esp. if some windows have a local directory.
Solution: Make getcwd(-1) return the global directory. (Andy Massimino, closes #2606)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_getcwd.vim

Patch 8.0.1490
Problem: Number of spell regions is spread out through the code.
Solution: Define MAXREGIONS.
Files: src/spell.h, src/spellfile.c

Patch 8.0.1491
Problem: The minimum width of the popup menu is hard coded.
Solution: Add the 'pumwidth' option. (Christian Brabandt, James McCoy, closes #2314)
Files: runtime/doc/options.txt, src/option.c, src/option.h, src/popupmnu.c

Patch 8.0.1492
Problem: Memory leak in balloon_split().
Solution: Free the balloon lines. Free the balloon when exiting.
Files: src/misc2.c, src/evalfunc.c

Patch 8.0.1493
Problem: Completion items cannot be annotated.
Solution: Add a "user_data" entry to the completion item. (Ben Jackson, closes #2608, closes #2508)
Files: runtime/doc/insert.txt, src/edit.c, src/structs.h, src/testdir/test_ins_complete.vim

Patch 8.0.1494
Problem: No autocmd triggered in Insert mode with visible popup menu.
Solution: Add TextChangedP. (Prabir Shrestha, Christian Brabandt, closes #2372, closes #1691)
Fix that the TextChanged autocommands are not always triggered when sourcing a script.
Files: runtime/doc/autocmd.txt, src/edit.c, src/globals.h, src/structs.h, src/fileio.c, src/proto/fileio.pro, src/vim.h, src/main.c, src/testdir/test_autocmd.vim

Patch 8.0.1495
Problem: Having 'pumwidth' default to zero has no merit.

Solution: Make the default 15, as the actual default value.
Files: src/popupmnu.c, src/option.c

Patch 8.0.1496

Problem: Clearing a pointer takes two lines.
Solution: Add VIM_CLEAR() and replace vim_clear(). (Hirohito Higashi, closes #2629)
Files: src/buffer.c, src/channel.c, src/encrypt.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/gui_gtk_x11.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/hardcopy.c, src/if_cscope.c, src/macros.h, src/main.c, src/mark.c, src/mbyte.c, src/memfile.c, src/memline.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_amiga.c, src/os_mac_conv.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/popupmnu.c, src/proto/misc2.pro, src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/screen.c, src/search.c, src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c, src/term.c, src/terminal.c, src/ui.c, src/undo.c, src/userfunc.c, src/window.c

Patch 8.0.1497

Problem: Getting the jump list requires parsing the output of :jumps.
Solution: Add getjumplist(). (Yegappan Lakshmanan, closes #2609)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/Makefile, src/evalfunc.c, src/list.c, src/proto/list.pro, src/testdir/Make_all.mak, src/testdir/test_jumplist.vim

Patch 8.0.1498 (after 8.0.1497)

Problem: getjumplist() returns duplicate entries. (lacygoill)
Solution: Call cleanup_jumplist(). (Yegappan Lakshmanan)
Files: src/evalfunc.c, src/mark.c, src/proto/mark.pro, src/testdir/test_jumplist.vim

Patch 8.0.1499

Problem: Out-of-memory situation not correctly handled. (Coverity)
Solution: Check for NULL value.
Files: src/terminal.c

Patch 8.0.1500

Problem: Possible NULL pointer dereference. (Coverity)
Solution: Check for the pointer not being NULL.
Files: src/quickfix.c

Patch 8.0.1501

Problem: Out-of-memory situation not correctly handled. (Coverity)
Solution: Check for NULL value.
Files: src/ops.c

Patch 8.0.1502

Problem: In out-of-memory situation character is not restored. (Coverity)
Solution: Restore the character in all situations.
Files: src/ex_getln.c

Patch 8.0.1503

Problem: Access memory beyond end of string. (Coverity)
Solution: Keep allocated memory in separate pointer. Avoid outputting the NUL character.
Files: src/hardcopy.c

Patch 8.0.1504

Problem: Win32: the screen may be cleared on startup.
Solution: Only call shell_resized() when the size actually changed. (Ken Takata, closes #2527)
Files: src/os_win32.c

Patch 8.0.1505

Problem: Debugger can't break on a condition. (Charles Campbell)
Solution: Add ":breakadd expr". (Christian Brabandt, closes #859)
Files: runtime/doc/repeat.txt, src/eval.c, src/evalfunc.c, src/userfunc.c, src/ex_cmds2.c, src/ex_docmd.c, src/proto/eval.pro, src/proto/ex_cmds2.pro, src/structs.h

Patch 8.0.1506

Problem: New version of HP NonStop (Tandem) doesn't like the default header for setenv().
Solution: Put a #ifdef around the setenv() entry. (Joachim Schmitz)
Files: src/osdef2.h.in

Patch 8.0.1507

Problem: Timer test is a bit flaky.
Solution: Add it to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.0.1508

Problem: The :drop command is not always available.
Solution: Include :drop in all builds. (Yasuhiro Matsumoto, closes #2639)
Files: runtime/doc/windows.txt, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/testdir/test_normal.vim, src/testdir/test_tabpage.vim

Patch 8.0.1509 (after 8.0.1508)

Problem: Test for failing drag-n-drop command no longer fails.
Solution: Check for the "dnd" feature.
Files: src/testdir/test_normal.vim

Patch 8.0.1510

Problem: Cannot test if a command causes a beep.
Solution: Add assert_beeps().
Files: runtime/doc/eval.txt, src/evalfunc.c, src/eval.c, src/proto/eval.pro, src/misc1.c, src/globals.h, src/testdir/test_normal.vim, src/testdir/test_assert.vim

Patch 8.0.1511 (after 8.0.1505)

Problem: Some code for the debugger watch expression is clumsy.
Solution: Clean up the code.
Files: src/ex_cmds2.c, src/eval.c, src/proto/eval.pro

Patch 8.0.1512

Problem: Warning for possibly using NULL pointer. (Coverity)
Solution: Skip using the pointer if it's NULL.
Files: src/ex_cmds.c

Patch 8.0.1513

Problem: The jumplist is not always properly cleaned up.
Solution: Call fname2fnum() before cleanup_jumplist(). (Yegappan Lakshmanan)
Files: src/evalfunc.c, src/mark.c, src/proto/mark.pro

Patch 8.0.1514

Problem: Getting the list of changes is not easy.
Solution: Add the getchangelist() function. (Yegappan Lakshmanan, closes #2634)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/testdir/Make_all.mak, src/testdir/test_changelist.vim, src/Makefile

Patch 8.0.1515

Problem: BufWinEnter event fired when opening hidden terminal.
Solution: Do not fire BufWinEnter when the terminal is hidden and does not open a window. (Kenta Sato, closes #2636)
Files: src/terminal.c

Patch 8.0.1516

Problem: Errors for job options are not very specific.
Solution: Add more specific error messages.
Files: src/channel.c, src/globals.h

Patch 8.0.1517

Problem: Invalid memory access with pattern using look-behind match. (Dominique Pelle)
Solution: Get a pointer to the right line.
Files: src/regexp.c

Patch 8.0.1518

Problem: Error messages suppressed after ":silent! try". (Ben Reilly)
Solution: Restore emsg_silent before executing :try. (closes #2531)
Files: src/ex_docmd.c, src/testdir/test_eval_stuff.vim

Patch 8.0.1519

Problem: getchangelist() does not use argument as bufname().
Solution: Use get_buf_tv(). (Yegappan Lakshmanan, closes #2641)
Files: src/evalfunc.c, src/testdir/test_changelist.vim

Patch 8.0.1520

Problem: Cursor is in the wrong line when using a WinBar in a Terminal window.
Solution: Adjust the row number. (Christian Brabandt, closes #2362)
Files: src/screen.c, src/terminal.c

Patch 8.0.1521

Problem: Shift-Tab does not work in a terminal window.
Solution: Recognize Shift-Tab key press. (Jsees Luehrs, closes #2644)

Files: src/terminal.c

Patch 8.0.1522 (after 8.0.1491)

Problem: Popup menu is positioned in the wrong place. (Davit Samvelyan, Boris Staletic)

Solution: Correct computation of the column and the conditions for that. (Hirohito Higashi, closes #2640)

Files: src/popupmnu.c

Patch 8.0.1523

Problem: Cannot write and read terminal screendumps.

Solution: Add term_dumpwrite(), term_dumpread() and term_dumpdiff(). Also add assert_equalfile().

Files: src/terminal.c, src/proto/terminal.pro, src/evalfunc.c, src/normal.c, src/eval.c, src/proto/eval.pro, runtime/doc/eval.txt, src/testdir/test_assert.vim

Patch 8.0.1524 (after 8.0.1523)

Problem: Compiler warnings for uninitialized variables. (Tony Mechelynck)

Solution: Initialize variables.

Files: src/terminal.c

Patch 8.0.1525

Problem: Using :wqa exits even if a job runs in a terminal window. (Jason Felice)

Solution: Check if a terminal has a running job. (closes #2654)

Files: src/ex_cmds2.c, src/buffer.c, src/proto/buffer.pro, src/ex_cmds.c, src/testdir/test_terminal.vim

Patch 8.0.1526

Problem: No test using a screen dump yet.

Solution: Add a test for C syntax highlighting. Add helper functions.

Files: src/terminal.c, src/testdir/test_syntax.vim, src/testdir/shared.vim, src/testdir/screendump.vim, src/testdir/dumps/Test_syntax_c_01.dump, runtime/doc/terminal.txt, src/testdir/README.txt

Patch 8.0.1527 (after 8.0.1526)

Problem: Screen dump test fails on MS-Windows.

Solution: Skip dump test on MS-Windows for now.

Files: src/testdir/test_syntax.vim

Patch 8.0.1528

Problem: Dead code found.

Solution: Remove the useless lines. (CodeAi, closes #2656)

Files: src/screen.c, src/spell.c, src/syntax.c, src/window.c

Patch 8.0.1529

Problem: Assert_equalfile() does not close file descriptors. (Coverity)

Solution: Close the file descriptors.

Files: src/eval.c

Patch 8.0.1530

Problem: Dump test fails when using a shadow directory.

Solution: Add the directory to the list of symlinks to make (Elimar Riesebieter)
Files: src/Makefile

Patch 8.0.1531

Problem: Cannot use 24 bit colors in MS-Windows console.
Solution: Add support for vcon. (Nobuhiro Takasaki, Ken Takata, fixes #1270, fixes #2060)
Files: runtime/doc/options.txt, src/misc1.c, src/option.c, src/evalfunc.c, src/os_win32.c, src/proto/os_win32.pro, src/feature.h, src/proto/term.pro, src/screen.c, src/syntax.c, src/term.c, src/testdir/gen_opt_test.vim, src/version.c

Patch 8.0.1532

Problem: Compiler warnings without termguicolors feature.
Solution: Add #ifdef. (John Marriott) Cleanup the code a bit.
Files: src/term.c

Patch 8.0.1533

Problem: Libterm doesn't support requesting fg and bg color.
Solution: Implement t_RF and t_RB.
Files: src/libvterm/src/vterm_internal.h, src/libvterm/src/state.c, src/libvterm/src/vterm.c

Patch 8.0.1534

Problem: C syntax test fails when using gvim
Solution: Force running in a terminal. Check that **'background'** is correct even when \$COLORFGBG is set.
Files: src/testdir/test_syntax.vim, src/testdir/screendump.vim

Patch 8.0.1535 (after 8.0.1534)

Problem: C syntax test still fails when using gvim.
Solution: Clear Normal cterm highlighting instead of setting it.
Files: src/testdir/test_syntax.vim, src/testdir/screendump.vim, src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.0.1536

Problem: Quotestar test is flaky when using the GUI.
Solution: Add check that the star register arrived at the server. Increase timeouts.
Files: src/testdir/test_quotestar.vim

Patch 8.0.1537

Problem: Xxd does not skip NUL lines when using ebcdic.
Solution: Check for a NUL before converting a character for ebcdic. (Tim Sell, closes #2668)
Files: src/xxd/xxd.c

Patch 8.0.1538

Problem: Popuption is too far left when completion is long. (Linwei)
Solution: Adjust column computations. (Hirohito Higashi, closes #2661)
Files: src/popuption.c

Patch 8.0.1539

Problem: No test for the popup menu positioning.
Solution: Add a screendump test for the popup menu.
Files: src/terminal.c, src/testdir/test_syntax.vim,
src/testdir/screendump.vim,
src/testdir/test_popup.vim,
src/testdir/dumps/Test_popup_position_01.dump,
src/testdir/dumps/Test_popup_position_02.dump,
src/testdir/dumps/Test_popup_position_03.dump,
runtime/doc/eval.txt

Patch 8.0.1540

Problem: Popup menu positioning fails with longer string.
Solution: Only align with right side of window when width is less than
'pumwidth' (closes #2661)
Files: src/popupmnu.c, src/testdir/screendump.vim,
src/testdir/test_popup.vim,
src/testdir/dumps/Test_popup_position_04.dump

Patch 8.0.1541

Problem: synpat_T is taking too much memory.
Solution: Reorder members to reduce padding. (Dominique Pelle, closes #2671)
Files: src/syntax.c

Patch 8.0.1542

Problem: Terminal screen dump does not include cursor position.
Solution: Mark the cursor position in the dump.
Files: src/terminal.c,
src/testdir/dumps/Test_popup_position_01.dump,
src/testdir/dumps/Test_popup_position_02.dump,
src/testdir/dumps/Test_popup_position_03.dump,
src/testdir/dumps/Test_popup_position_04.dump,
src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.0.1543

Problem: With 'termguicolors' Normal color doesn't work correctly.
Solution: Set cterm_normal_bg_gui_color and cterm_normal_fg_color always.
(Kazunobu Kuriyama, closes #981, closes #2332)
Files: src/syntax.c

Patch 8.0.1544

Problem: When using 'termguicolors' SpellBad doesn't show.
Solution: When the GUI colors are not set fall back to the cterm colors.
Files: src/syntax.c, src/screen.c, src/gui.h, src/structs.h

Patch 8.0.1545

Problem: Screen dumps not included in distribution.
Solution: Add dumps to the list of distributed files.
Files: Filelist

Patch 8.0.1546

Problem: Using feedkeys() in a terminal window may trigger mappings.
(Charles Sheridan)
Solution: Avoid triggering a mapping when peeking for a key.
Files: src/getchar.c, src/terminal.c

Patch 8.0.1547

Problem: Undo in the options window makes it empty.
Solution: Set **'undolevels'** while filling the buffer. (Yasuhiro Matsumoto, closes #2645)
Files: runtime/optwin.vim

Patch 8.0.1548

Problem: Screen dump test script not included in distribution.
Solution: Add the script to the list of distributed files.
Files: Filelist

Patch 8.0.1549

Problem: Various small problems in test files.
Solution: Include small changes.
Files: src/testdir/test_channel.py, src/testdir/shared.vim, src/testdir/test_gui.vim, src/testdir/test_gui_init.vim

Patch 8.0.1550

Problem: Various small problems in source files.
Solution: Fix the problems.
Files: src/README.txt, src/beval.c, src/json_test.c, src/mbyte.c, src/libvterm/include/vterm_keycodes.h, src/Makefile, src/gui_gtk.c, src/if_xcmdsrv.c, src/pty.c, src/if_python.c, src/if_py_both.h, uninstal.txt, src/dosinst.c, src/iscygpty.c, src/vimrun.c, src/os_vms.c

Patch 8.0.1551

Problem: On Mac **'maxmemtot'** is set to a weird value.
Solution: For Mac use total memory and subtract system memory. For other systems accept both a 32 bit and 64 bit result. (Ozaki Kiichi, closes #2646)
Files: src/os_unix.c

Patch 8.0.1552

Problem: May leak file descriptors when executing job.
Solution: Close more file descriptors. (Ozaki Kiichi, closes #2651)
Files: src/os_unix.c, src/testdir/test_channel.vim

Patch 8.0.1553

Problem: Cannot see what digraph is used to insert a character.
Solution: Show the digraph with the "ga" command. (Christian Brabandt)
Files: runtime/doc/various.txt, src/digraph.c, src/ex_cmds.c, src/proto/digraph.pro, src/testdir/shared.vim, src/testdir/test_matchadd_conceal.vim, src/testdir/test_digraph.vim, src/testdir/test_ga.vim, src/testdir/test_arabic.vim

Patch 8.0.1554

Problem: Custom plugins loaded with --clean.
Solution: Do not include the home directory in **'runtimepath'**.
Files: src/option.c, src/main.c, src/proto/option.pro, src/structs.h, src/os_unix.h, src/os_amiga.h, src/os_dos.h, src/os_mac.h, runtime/doc/starting.txt

Patch 8.0.1555

Problem: Build error for some combination of features.
Solution: Declare variable in more situations.
Files: src/main.c

Patch 8.0.1556

Problem: May not parse the t_RS response correctly, resulting in wrong characters in the input stream.
Solution: When the t_RS response is partly received wait for more characters.
Files: src/term.c

Patch 8.0.1557

Problem: printf() does not work with only one argument. (Daniel Hahler)
Solution: Allow using just the format. (Ken Takata, closes #2687)
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 8.0.1558

Problem: No right-click menu in a terminal.
Solution: Implement the right click menu for the terminal.
Files: src/popupmnu.c, src/proto/popupmnu.pro, src/normal.c, src/menu.c, src/proto/menu.pro, src/feature.h

Patch 8.0.1559

Problem: Build failure without GUI.
Solution: Adjust #ifdef for get_fpos_of_mouse().
Files: src/ui.c

Patch 8.0.1560

Problem: Build failure without GUI on MS-Windows.
Solution: Adjust #ifdef for vcol2col().
Files: src/ui.c

Patch 8.0.1561

Problem: Crash with rust syntax highlighting. (Edd Barrett)
Solution: Avoid going past the end of an empty line.
Files: src/syntax.c

Patch 8.0.1562

Problem: The terminal debugger can't set a breakpoint with the mouse.
Solution: Add popup menu entries.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/terminal.txt

Patch 8.0.1563

Problem: Timeout of getwinposx() can be too short. (lilydjwg)
Solution: Add getwinpos(). (closes #2689)
Files: src/evalfunc.c, src/term.c, src/proto/term.pro, runtime/doc/eval.txt

Patch 8.0.1564

Problem: Too many #ifdefs.
Solution: Graduate the +autocmd feature. Takes away 450 #ifdefs and increases code size of tiny Vim by only 40 Kbyte.

Files: src/buffer.c, src/diff.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/globals.h, src/gui.c, src/if_cscope.c, src/if_xcmdsrv.c, src/main.c, src/mbyte.c, src/memline.c, src/menu.c, src/misc1.c, src/gui_mac.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/option.h, src/feature.h, src/vim.h, src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/quickfix.c, src/screen.c, src/search.c, src/spell.c, src/structs.h, src/syntax.c, src/tag.c, src/term.c, src/terminal.c, src/ui.c, src/undo.c, src/userfunc.c, src/version.c, src/window.c

Patch 8.0.1565

Problem: Can't build Mac version without GUI.
Solution: Adjust when IME_WITHOUT_XIM is defined.
Files: src/vim.h

Patch 8.0.1566

Problem: Too many #ifdefs.
Solution: Graduate FEAT_SCROLLBIND and FEAT_CURSORBIND.
Files: src/buffer.c, src/diff.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/gui.c, src/main.c, src/move.c, src/normal.c, src/option.c, src/term.c, src/version.c, src/window.c, src/globals.h, src/macros.h, src/option.h, src/structs.h

Patch 8.0.1567

Problem: Cannot build Win32 GUI without IME. (John Marriott)
Solution: Adjust when IME_WITHOUT_XIM and HAVE_INPUT_METHOD are defined and use it in a few more places.
Files: src/vim.h, src/gui.c

Patch 8.0.1568

Problem: Can't build on older Mac, header file is missing.
Solution: Remove the header file. (Ozaki Kiichi, closes #2691)
Files: src/os_unix.c

Patch 8.0.1569

Problem: Warning for uninitialized variable from gcc.
Solution: Initialize the variable.
Files: src/quickfix.c

Patch 8.0.1570

Problem: Can't use :popup for a menu in the terminal. (Wei Zhang)
Solution: Make :popup work in the terminal. Also fix that entries were included that don't work in the current state.
Files: src/ex_docmd.c, src/popupmnu.c, src/proto/popupmnu.pro, src/menu.c, src/proto/menu.pro

Patch 8.0.1571 (after 8.0.1571)

Problem: Can't build without GUI.
Solution: Adjust #ifdef for gui_find_menu().
Files: src/menu.c

Patch 8.0.1572

Problem: Mac: getting memory size doesn't work everywhere.
Solution: Use MACOS_X instead of MACOS_X_DARWIN. (Kazunobu Kuriyama)
Files: src/os_unix.c

Patch 8.0.1573

Problem: getwinpos(1) may cause response to be handled as command.
Solution: Handle any cursor position report once one was requested. (partly by Hirohito Higashi)
Files: src/term.c

Patch 8.0.1574

Problem: Show cursor in wrong place when using popup menu. (Wei Zhang)
Solution: Force updating the cursor position. Fix skipping over unused entries.
Files: src/screen.c, src/proto/screen.pro, src/popupmnu.c

Patch 8.0.1575

Problem: Crash when using virtual replace.
Solution: Adjust orig_line_count. Add more tests. (Christian Brabandt)
Files: src/edit.c, src/testdir/test_visual.vim

Patch 8.0.1576

Problem: Perl VIM::Buffers() does not find every buffer.
Solution: Also find unlisted buffer by number or name. (Chris Weyl, closes #2692)
Files: src/if_perl.xs

Patch 8.0.1577

Problem: Virtual replace test fails on MS-Windows.
Solution: Make adding a termcap entry work for a builtin terminal. Restore terminal keys in a better way.
Files: src/term.c, src/testdir/test_visual.vim

Patch 8.0.1578

Problem: No test for :popup in terminal.
Solution: Add a screen dump test.
Files: src/testdir/test_popup.vim,
src/testdir/dumps/Test_popup_command_01.dump,
src/testdir/dumps/Test_popup_command_02.dump,
src/testdir/dumps/Test_popup_command_03.dump

Patch 8.0.1579

Problem: Virtual replace test fails in GUI.
Solution: Don't save key options if they were not set.
Files: src/testdir/test_visual.vim

Patch 8.0.1580

Problem: FEAT_CURSORBIND and FEAT_SCROLLBIND are unused.
Solution: Delete them.
Files: src/feature.h

Patch 8.0.1581

Problem: Cannot build Win32 GUI without +eval.
Solution: Define HAVE_INPUT_METHOD without +eval. (Ken Takata)
Files: src/vim.h

Patch 8.0.1582

Problem: In the MS-Windows console mouse movement is not used.
Solution: Pass mouse movement events when useful.
Files: src/os_win32.c, src/proto/os_win32.pro, src/feature.h

Patch 8.0.1583

Problem: Using C99 comment.
Solution: Use old style comment. (Kazunobu Kuriyama)
Files: src/quickfix.c

Patch 8.0.1584

Problem: Using C99 in Mac file gives compiler warning messages.
Solution: Add #pragmas to avoid the warnings. (Kazunobu Kuriyama)
Files: src/os_macosx.m

Patch 8.0.1585

Problem: Enabling beval_term feature in Win32 GUI.
Solution: Only enable beval_term in Win32 console.
Files: src/feature.h

Patch 8.0.1586

Problem: Imactivatefunc does not work on non-GUI Mac.
Solution: Fix logic in #ifdef.
Files: src/vim.h

Patch 8.0.1587

Problem: inserting from the clipboard doesn't work literally
Solution: When pasting from the * or + register always assume literally.
Files: src/ops.c, src/proto/ops.pro, src/testdir/test_paste.vim

Patch 8.0.1588

Problem: Popup menu hangs after typing **CTRL-C**.
Solution: Make **CTRL-C** exit the loop. (Ozaki Kiichi, closes #2697)
Files: src/popupmnu.c

Patch 8.0.1589

Problem: Error for setting 'modifiable' when resetting it.
Solution: Check if 'modifiable' was actually set.
Files: src/option.c

Patch 8.0.1590

Problem: Padding in list type wastes memory.
Solution: Reorder struct members to optimize padding. (Dominique Pelle, closes #2704)
Files: src/structs.h

Patch 8.0.1591

Problem: MS-Windows: when reparsing the arguments 'wildignore' matters.
Solution: Save and reset 'wildignore'. (Yasuhiro Matsumoto, closes #2702)
Files: src/os_win32.c

Patch 8.0.1592

Problem: Terminal windows in a session are not properly restored.
Solution: Add "terminal" in '**sessionoptions**'. When possible restore the command running in a terminal.
Files: src/option.c, src/option.h, src/ex_docmd.c, src/terminal.c, src/proto/terminal.pro, src/evalfunc.c, src/structs.h, src/channel.c, src/testdir/test_terminal.vim, src/testdir/shared.vim, src/testdir/test_mksession.vim

Patch 8.0.1593

Problem: :qall never exits with an active terminal window.
Solution: Add a way to kill a job in a terminal window.
Files: src/ex_cmds2.c, src/terminal.c, src/proto/terminal.pro, src/structs.h, src/channel.c, src/evalfunc.c, src/testdir/test_terminal.vim, runtime/doc/terminal.txt, runtime/doc/eval.txt

Patch 8.0.1594

Problem: :confirm qall not tested with active terminal window.
Solution: Add a test.
Files: src/testdir/test_terminal.vim

Patch 8.0.1595

Problem: No autocommand triggered before exiting.
Solution: Add the ExitPre autocommand event.
Files: src/ex_docmd.c, src/fileio.c, src/vim.h, src/testdir/test_exit.vim, src/Makefile, src/testdir/Make_all.mak, runtime/doc/autocmd.txt

Patch 8.0.1596

Problem: No autocommand specifically for opening a terminal window.
Solution: Add TerminalOpen. (Yasuhiro Matsumoto, closes #2484)
Files: runtime/doc/autocmd.txt, src/fileio.c, src/terminal.c, src/testdir/test_terminal.vim, src/vim.h

Patch 8.0.1597

Problem: Autocommand events are not sorted.
Solution: Sort the autocommand events.
Files: src/vim.h

Patch 8.0.1598

Problem: Cannot select text in a terminal with the mouse.
Solution: When a job in a terminal is not consuming mouse events, use them for modeless selection. Also stop Insert mode when clicking in a terminal window.
Files: src/libvterm/include/vterm.h, src/libvterm/src/state.c, src/libvterm/src/vterm_internal.h, src/terminal.c, src/proto/terminal.pro, src/ui.c

Patch 8.0.1599

Problem: No error message when gdb does not support the terminal debugger.
Solution: Check for the response to open the Machine Interface.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1600

Problem: Crash when setting t_Co to zero when **'termguicolors'** is set.
Solution: Use IS_CTERM instead of checking the number of colors.
(closes #2710)
Files: src/screen.c, src/testdir/test_highlight.vim

Patch 8.0.1601

Problem: Highlight test fails on Win32.
Solution: Check for vtp and vcon support.
Files: src/evalfunc.c, src/testdir/test_highlight.vim

Patch 8.0.1602

Problem: Crash in parsing JSON.
Solution: Fail when using array or dict as dict key. (Damien)
Files: src/json.c, src/testdir/test_json.vim

Patch 8.0.1603

Problem: Cannot build with +terminal but without +menu.
Solution: Add #ifdef. (Damien)
Files: src/terminal.c

Patch 8.0.1604

Problem: Paste test may fail if \$DISPLAY is not set.
Solution: Add WorkingClipboard() and use it in the paste test.
Files: src/testdir/shared.vim, src/testdir/test_paste.vim

Patch 8.0.1605

Problem: Terminal test is a bit flaky.
Solution: Check for the shell prompt. Use more lambda functions.
Files: src/testdir/test_terminal.vim

Patch 8.0.1606

Problem: Singular/plural variants not translated.
Solution: Add NGETTEXT argument to xgettext. (Sergey Alyoshin)
Files: src/po/Make_cyg.mak, src/po/Make_ming.mak, src/po/Make_mvc.mak,
src/po/Makefile

Patch 8.0.1607

Problem: --clean loads user settings from .gvimrc.
Solution: Behave like "-U NONE" was used. (Ken Takata)
Files: src/main.c, runtime/doc/starting.txt

Patch 8.0.1608

Problem: Win32: directx not enabled by default.
Solution: Change Makefile to enable directx by default. (Ken Takata)
Files: runtime/doc/various.txt, src/Make_cyg_ming.mak,
src/Make_mvc.mak

Patch 8.0.1609

Problem: Shell commands in the GUI use a dumb terminal.
Solution: Add the "!" flag to **'guioptions'** to execute system commands in a
special terminal window. Only for Unix now.
Files: src/os_unix.c, src/option.h, src/evalfunc.c, src/terminal.c,

src/proto/terminal.pro, src/channel.c, src/proto/channel.pro,
src/vim.h, runtime/doc/options.txt

Patch 8.0.1610 (after 8.0.1609)

Problem: Cannot build without GUI.
Solution: Add #ifdef.
Files: src/terminal.c

Patch 8.0.1611

Problem: **CTRL-W** in system terminal does not go to job.
Solution: Do not use **CTRL-W** as a terminal command in a system terminal.
Files: src/terminal.c

Patch 8.0.1612

Problem: Need to close terminal after shell stopped.
Solution: Make :terminal without argument close the window by default.
Files: src/terminal.c, src/testdir/test_terminal.vim,
runtime/doc/terminal.txt

Patch 8.0.1613

Problem: Warning for unused variable in tiny build. (Tony Mechelynck)
Solution: Move declaration to inner block.
Files: src/os_unix.c

Patch 8.0.1614

Problem: "make tags" doesn't include libvterm.
Solution: Add the libvterm sources to the tags command.
Files: src/Makefile

Patch 8.0.1615

Problem: term_dumpload() does not use the right colors.
Solution: Initialize colors when not using create_vterm().
Files: src/terminal.c

Patch 8.0.1616

Problem: Win32: shell commands in the GUI open a new console.
Solution: Use a terminal window for interactive use when **'guioptions'**
contains "!".
Files: src/os_win32.c

Patch 8.0.1617 (after 8.0.1616)

Problem: Win32: :shell command in the GUI crashes.
Solution: Handle the situation that "cmd" is NULL. (Yasuhiro Matsumoto,
closes #2721)
Files: src/os_win32.c

Patch 8.0.1618

Problem: Color Grey50, used for ToolbarLine, is missing in the compiled-in
table.
Solution: Add the color to the list. (Kazunobu Kuriyama)
Files: src/term.c

Patch 8.0.1619

Problem: Win32 GUI: crash when winpty is not installed and trying to use

:shell in a terminal window.
Solution: Check for NULL return form term_start(). (Yasuhiro Matsumoto,
 closes #2727)
Files: src/os_win32.c

Patch 8.0.1620

Problem: Reading spell file has no good EOF detection.
Solution: Check for EOF at every character read for a length field.
Files: src/misc2.c

Patch 8.0.1621

Problem: Using invalid default value for highlight attribute.
Solution: Use zero instead of -1.
Files: src/syntax.c

Patch 8.0.1622

Problem: Possible NULL pointer dereference. (Coverity)
Solution: Reverse the check for a NULL pointer.
Files: src/quickfix.c

Patch 8.0.1623

Problem: Terminal kill tests are flaky.
Solution: Instead of running Vim in a terminal, run it as a normal command.
Files: src/testdir/test_terminal.vim

Patch 8.0.1624

Problem: Options for term_dumpdiff() and term_dumpload() not implemented
 yet.
Solution: Implement the relevant options.
Files: src/terminal.c, runtime/doc/eval.txt

Patch 8.0.1625

Problem: Test_quotestar is flaky when run in GTK GUI.
Solution: Do not call lose_selection when invoked from
 selection_clear_event().
Files: src/gui_gtk_x11.c

Patch 8.0.1626

Problem: Compiler warning for possible loss of data.
Solution: Use size_t instead of int. (Christian Brabandt)
Files: src/terminal.c

Patch 8.0.1627

Problem: Compiler warning for visibility attribute not supported on MinGW
 builds.
Solution: Don't add the attribute when we don't expect it to work.
 (Christian Brabandt)
Files: src/libvterm/src/vterm_internal.h

Patch 8.0.1628

Problem: Channel log doesn't mention exiting.
Solution: Add a ch_log() call in getout().
Files: src/main.c

Patch 8.0.1629

Problem: Mac: getpagesize() is deprecated.
Solution: Use sysconf() instead. (Ozaki Kiichi, closes #2741)
Files: src/os_unix.c

Patch 8.0.1630

Problem: Trimming white space is not that easy.
Solution: Add the trim() function. (Bukn, Yasuhiro Matsumoto, closes #1280)
Files: src/evalfunc.c, runtime/doc/eval.txt,
src/testdir/test_functions.vim

Patch 8.0.1631

Problem: Testing with Vim running in terminal is a bit flaky.
Solution: Delete any .swp file so that later tests don't fail.
Files: src/testdir/screendump.vim

Patch 8.0.1632

Problem: In a terminal dump NUL and space considered are different,
although they are displayed the same.
Solution: When encountering NUL handle it like space.
Files: src/terminal.c

Patch 8.0.1633

Problem: A TextChanged autocmd triggers when it is defined after creating a
buffer.
Solution: Set b_last_changedtick when opening a buffer. (Hirohito Higashi,
closes #2742)
Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 8.0.1634

Problem: The ex_vimgrep() function is too long.
Solution: Split it in smaller functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1635

Problem: Undefined _POSIX_THREADS causes problems with Python 3. (Micah
Bucy, closes #2748)
Solution: Remove the lines.
Files: src/if_python3.c

Patch 8.0.1636

Problem: No test for term_dumpload() and term_dumpdiff().
Solution: Add tests.
Files: src/testdir/test_terminal.vim

Patch 8.0.1637

Problem: No test for term_dumpdiff() options argument.
Solution: Add a test.
Files: src/testdir/test_terminal.vim

Patch 8.0.1638

Problem: Popup test fails depending on environment variable.
Solution: Reset \$COLORFGBG when running Vim in a terminal. (closes #2693)
Files: src/testdir/screendump.vim

Patch 8.0.1639

Problem: Libvterm code lags behind master.

Solution: Sync to head, solve merge problems.

Files: src/libvterm/README, src/libvterm/bin/unterm.c,
src/libvterm/bin/vterm-ctrl.c, src/libvterm/bin/vterm-dump.c,
src/libvterm/doc/URLs, src/libvterm/doc/seqs.txt,
src/libvterm/include/vterm.h,
src/libvterm/include/vterm_keycodes.h, src/libvterm/src/mouse.c,
src/libvterm/src/parser.c, src/libvterm/src/pen.c,
src/libvterm/src/screen.c, src/libvterm/src/state.c,
src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h,
src/libvterm/t/10state_putglyph.test,
src/libvterm/t/25state_input.test, src/libvterm/t/harness.c,
src/libvterm/t/26state_query.test

Patch 8.0.1640

Problem: Test_cwd() is flaky.

Solution: Add to list of flaky tests.

Files: src/testdir/runtest.vim

Patch 8.0.1641

Problem: Job in terminal can't communicate with Vim.

Solution: Add the terminal API.

Files: src/terminal.c, src/buffer.c, src/testdir/test_terminal.vim,
src/testdir/screendump.vim, runtime/doc/terminal.txt

Patch 8.0.1642

Problem: Running Vim in terminal fails with two windows.

Solution: Pass the number of rows to RunVimInTerminal().

Files: src/testdir/screendump.vim, src/testdir/test_terminal.vim

Patch 8.0.1643

Problem: Terminal API tests fail.

Solution: Explicitly set **'title'**.

Files: src/testdir/test_terminal.vim

Patch 8.0.1644

Problem: Terminal API tests still fail.

Solution: Explicitly set **'title'** in the terminal job. (Ozaki Kiichi,
closes #2750)

Files: src/testdir/test_terminal.vim, src/testdir/screendump.vim

Patch 8.0.1645

Problem: Test for terminal response to escape sequence fails for some
people. (toothpik)

Solution: Run "cat" and let it echo the characters.

Files: src/testdir/test_terminal.vim

Patch 8.0.1646

Problem: MS-Windows: executable contains unreferenced functions and data.

Solution: Add /opt:ref to the compiler command. (Ken Takata)

Files: src/Make_mvc.mak

Patch 8.0.1647

Problem: Terminal API may call a function not meant to be called by this API.

Solution: Require the function to start with Tapi_.

Files: runtime/doc/terminal.txt, src/terminal.c,
src/testdir/test_terminal.vim

Patch 8.0.1648

Problem: Resource fork tool doesn't work on Python 3.

Solution: Use "print()" instead of "print". (Marius Gedminas)

Files: src/dehqx.py

Patch 8.0.1649

Problem: No completion for argument list commands.

Solution: Add arglist completion. (Yegappan Lakshmanan, closes #2706)

Files: runtime/doc/eval.txt, runtime/doc/map.txt, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/proto/ex_cmds2.pro,
src/testdir/test_cmdline.vim, src/vim.h

Patch 8.0.1650

Problem: Too many #ifdefs.

Solution: Graduate FEAT_LISTCMDS, no reason to leave out buffer commands.

Files: runtime/doc/various.txt, src/buffer.c, src/charset.c,
src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c,
src/version.c, src/feature.h

Patch 8.0.1651

Problem: Cannot filter :ls output for terminal buffers.

Solution: Add flags for terminal buffers. (Marcin Szamotulski, closes #2751)

Files: runtime/doc/windows.txt, src/buffer.c,
src/testdir/test_terminal.vim

Patch 8.0.1652

Problem: term_dumpwrite() does not output composing characters.

Solution: Use the cell index.

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1653

Problem: Screen dump is made too soon.

Solution: Wait until the ruler is displayed. (Ozaki Kiichi, closes #2755)

Files: src/testdir/dumps/Test_popup_command_01.dump,
src/testdir/dumps/Test_popup_command_02.dump,
src/testdir/screendump.vim, src/testdir/test_autocmd.vim,
src/testdir/test_terminal.vim

Patch 8.0.1654

Problem: Warnings for conversion of void to function pointer.

Solution: Use a temp variable that is a function pointer.

Files: src/if_python.c, src/if_python3.c

Patch 8.0.1655

Problem: Outdated gdb message in terminal debugger unclear.

Solution: Specifically mention the required gdb version. Avoid getting stuck on pagination.

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.0.1656

Problem: No option to have xxd produce upper case variable names.

Solution: Add the -C argument. (Matt Panaro, closes #2772)

Files: src/xxd/xxd.c

Patch 8.0.1657

Problem: Crash when reading a channel.

Solution: Clear the write flag before writing. (idea by Shinya Ohyanagi, closes #2769).

Files: src/channel.c

Patch 8.0.1658

Problem: Capitalize argument not available in long form.

Solution: Recognize -capitalize. Update man page.

Files: src/xxd/xxd.c, runtime/doc/xxd.1, runtime/doc/xxd.man

Patch 8.0.1659

Problem: Scroll events not recognized for some xterm emulators.

Solution: Recognize mouse codes 0x40 and 0x41 as scroll events.

Files: src/term.c

Patch 8.0.1660

Problem: The terminal API "drop" command doesn't support options.

Solution: Implement the options.

Files: src/terminal.c, src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_cmds.h, src/eval.c, src/misc2.c, src/fileio.c, src/testdir/test_terminal.vim, runtime/doc/terminal.txt

Patch 8.0.1661

Problem: Warnings from 64 bit compiler.

Solution: Add type casts. (Mike Williams)

Files: src/terminal.c

Patch 8.0.1662

Problem: Showing dump diff doesn't mention both file names.

Solution: Add the file name in the separator line.

Files: src/terminal.c

Patch 8.0.1663 (after 8.0.1660)

Problem: Cannot build without multibyte feature.

Solution: Add #ifdef.

Files: src/ex_docmd.c

Patch 8.0.1664

Problem: Test failure because of not allocating enough space.

Solution: Allocate more bytes.

Files: src/terminal.c

Patch 8.0.1665

Problem: When running a terminal from the GUI '**term**' is not useful.

Solution: Use \$TERM in the GUI if it starts with "xterm". (closes #2776)

Files: src/os_unix.c, runtime/doc/terminal.txt

Patch 8.0.1666

Problem: % argument in ch_log() causes trouble.
Solution: Use string as third argument in internal ch_log(). (Dominique Pelle, closes #2784)
Files: src/evalfunc.c, src/testdir/test_channel.vim

Patch 8.0.1667

Problem: Terminal window tests are flaky.
Solution: Increase the waiting time for Vim to start.
Files: src/testdir/screendump.vim

Patch 8.0.1668

Problem: Terminal debugger: can't re-open source code window.
Solution: Add the :Source command. Also create the window if needed when gdb stops at a source line.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/terminal.txt

Patch 8.0.1669

Problem: :vimgrep may add entries to the wrong quickfix list.
Solution: Use the list identifier. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1670

Problem: Terminal window tests are still a bit flaky.
Solution: Increase the waiting time for the buffer to be created.
Files: src/testdir/test_terminal.vim

Patch 8.0.1671

Problem: Crash when passing non-dict argument as env to job_start().
Solution: Check for valid argument. (Ozaki Kiichi, closes #2765)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.0.1672

Problem: Error during completion causes command to be cancelled.
Solution: Reset did_emsg before waiting for another character. (Tom M.)
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.0.1673

Problem: Terminal window tests are still a bit flaky.
Solution: Increase the waiting time even more. (Elimar Riesebieter)
Files: src/testdir/test_terminal.vim

Patch 8.0.1674

Problem: Libvterm can't handle a long OSC string that is split.
Solution: When an incomplete OSC string is received copy it to the parser buffer. Increase the size of the parser buffer to be able to handle longer strings.
Files: src/libvterm/src/parser.c, src/libvterm/src/vterm.c

Patch 8.0.1675

Problem: Unused macro argument in libvterm. (Randall W. Morris)
Solution: Remove the argument.

Files: src/libvterm/src/parser.c

Patch 8.0.1676

Problem: No compiler warning for wrong printf format.

Solution: Add a printf attribute for gcc. Fix reported problems. (Dominique Pelle, closes #2789)

Files: src/channel.c, src/vim.h, src/proto/channel.pro

Patch 8.0.1677

Problem: No compiler warning for wrong format in vim_snprintf().

Solution: Add printf attribute for gcc. Fix reported problems.

Files: src/vim.h, src/proto.h, src/eval.c, src/fileio.c, src/mbyte.c, src/ops.c, src/spellfile.c, src/undo.c, src/json.c

Patch 8.0.1678

Problem: Errorformat "%r" implies "%>". (Jan Gosmann)

Solution: Jump to before setting fmt_ptr. (Yegappan Lakshmanan, closes #2785)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1679

Problem: Compiler warning for printf format. (Chdiza)

Solution: Change type to "long long". (closes #2791)

Files: src/ops.c

Patch 8.0.1680

Problem: Memory allocated by libvterm does not show up in profile.

Solution: Pass allocator functions to vterm_new().

Files: src/terminal.c

Patch 8.0.1681

Problem: The format attribute fails with MinGW. (John Marriott)

Solution: Don't use the format attribute with MinGW.

Files: src/vim.h, src/proto.h, src/channel.c

Patch 8.0.1682

Problem: Auto indenting breaks inserting a block.

Solution: Do not check for cursor movement if indent was changed. (Christian Brabandt, closes #2778)

Files: src/testdir/test_blockedit.vim, src/testdir/Make_all.mak, src/Makefile, src/ops.c

Patch 8.0.1683

Problem: Python upgrade breaks Vim when defining PYTHON_HOME.

Solution: Do not define PYTHON_HOME and PYTHON3_HOME in configure. (Naoki Inada, closes #2787)

Files: src/configure.ac, src/auto/configure

Patch 8.0.1684

Problem: ml_get errors when using terminal window for shell command. (Blay263)

Solution: Do not change the size of the current window.

Files: src/terminal.c

Patch 8.0.1685

Problem: Can't set ANSI colors of a terminal window.

Solution: Add term_setansicolors(), term_getansicolors() and g:term_ansi_colors. (Andy Massimino, closes #2747)

Files: runtime/doc/eval.txt, runtime/doc/terminal.txt, src/channel.c, src/evalfunc.c, src/proto/terminal.pro, src/structs.h, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1686 (after 8.0.1683)

Problem: Python does not work when configuring with specific dir. (Rajdeep)

Solution: Do define PYTHON_HOME and PYTHON3_HOME in configure if the Python config dir was specified.

Files: src/configure.ac, src/auto/configure

Patch 8.0.1687

Problem: 64 bit compiler warnings.

Solution: change type, add type cast. (Mike Williams)

Files: src/terminal.c

Patch 8.0.1688

Problem: Some macros are used without a semicolon, causing auto-indent to be wrong.

Solution: Use the do-while(0) trick. (Ozaki Kiichi, closes #2729)

Files: src/buffer.c, src/dosinst.c, src/ex_cmds.c, src/gui_at_sb.c, src/macros.h, src/main.c, src/memline.c, src/option.c, src/os_vms.c, src/screen.c, src/window.c

Patch 8.0.1689

Problem: No tests for xxd.

Solution: Add a test. (Christian Brabandt)

Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Makefile, src/testdir/test_xxd.vim, src/testdir/runtest.vim

Patch 8.0.1690

Problem: Not easy to run one test with gvim instead of vim.

Solution: Add VIMTESTTARGET in Makefile.

Files: src/Makefile

Patch 8.0.1691

Problem: Xxd test sometimes fails.

Solution: Wipe out the XXDfile buffer.

Files: src/testdir/test_xxd.vim

Patch 8.0.1692 (after 8.0.1686)

Problem: Python may not work when using statically linked library.

Solution: Do not define PYTHON_HOME and PYTHON3_HOME in configure if the Python library is linked statically.

Files: src/configure.ac, src/auto/configure

Patch 8.0.1693

Problem: Xxd is excluded from coverage statistics.

Solution: Don't skip the xxd directory. (Christian Brabandt)

Files: .travis.yml

Patch 8.0.1694

Problem: Terminal API test is a bit flaky.
Solution: Wait longer for Vim to stop.
Files: src/testdir/screendump.vim

Patch 8.0.1695

Problem: Xxd test not run on MS-Windows.
Solution: Use xxd.exe if it exists.
Files: src/testdir/test_xxd.vim

Patch 8.0.1696

Problem: Coverage statistics don't work.
Solution: Include the xxd directory. (Christian Brabandt)
Files: .travis.yml

Patch 8.0.1697

Problem: Various tests are still a bit flaky.
Solution: Increase the default wait time to five seconds.
Files: src/testdir/shared.vim, src/testdir/screendump.vim,
src/testdir/test_channel.vim, src/testdir/test_clientserver.vim,
src/testdir/test_quotestar.vim, src/testdir/test_terminal.vim

Patch 8.0.1698

Problem: Coverage statistics don't work on coveralls.
Solution: Use curly braces for \$SRCDIR.
Files: .travis.yml

Patch 8.0.1699

Problem: Leftover stuff for Python 1.4.
Solution: Remove outdated Python 1.4 stuff. (Naoki Inada, closes #2794)
Files: src/Makefile, src/config.aap.in, src/config.mk.in,
src/configure.ac, src/auto/configure

Patch 8.0.1700

Problem: Coverage statistics still don't work on coveralls.
Solution: Exclude the xxd directory again.
Files: .travis.yml

Patch 8.0.1701

Problem: Can disable COLOR_EMOJI with MSVC but not MinGW.
Solution: Add COLOR_EMOJI flag. Also add some empty lines for readability.
Files: src/Make_cyg_ming.mak

Patch 8.0.1702

Problem: Leaking memory when autocommands make a quickfix list invalid.
Solution: Call FreeWild(). (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1703

Problem: In the tutor **'showcmd'** is not set.
Solution: Set **'showcmd'** in the vimtutor script. (Ken Takata, closes #2792)
Files: src/vimtutor

Patch 8.0.1704

Problem: **'backupskip'** default doesn't work for Mac.
Solution: Use `"/private/tmp"`. (Rainer Müller, closes #2793)
Files: `src/option.c`, `src/testdir/test_options.vim`,
`runtime/doc/options.txt`

Patch 8.0.1705

Problem: When making a vertical split the mode message isn't always updated, "VISUAL" remains. (Alexei Averchenko)
Solution: Only reset `clear_cmdline` when filling all columns of the last screen line. (Tom M. closes #2611)
Files: `src/screen.c`, `src/testdir/test_window_cmd.vim`

Patch 8.0.1706

Problem: Cannot send `CTRL-\` to a terminal window.
Solution: Make **CTRL-W** `CTRL-\` send `CTRL-\` to a terminal window.
Files: `src/terminal.c`, `runtime/doc/terminal.txt`

Patch 8.0.1707

Problem: When **'wfh'** is set `":bel 10new"` scrolls window. (Andrew Pyatkov)
Solution: Set the fraction before changing the window height. (closes #2798)
Files: `src/window.c`

Patch 8.0.1708

Problem: `Mkdir` with `'p'` flag fails on existing directory, which is different from the `mkdir` shell command.
Solution: Don't fail if the directory already exists. (James McCoy, closes #2775)
Files: `src/evalfunc.c`, `src/testdir/test_eval_stuff.vim`,
`runtime/doc/eval.txt`

Patch 8.0.1709

Problem: Some non-C89 code may slip through.
Solution: Enforce C89 in `configure`. Fix detected problems. (James McCoy, closes #2735)
Files: `src/channel.c`, `src/configure.ac`, `src/auto/configure`,
`src/gui_gtk_x11.c`, `src/if_python3.c`

Patch 8.0.1710

Problem: Building with Ruby fails.
Solution: Don't add `-ansi` when building with Ruby.
Files: `src/configure.ac`, `src/auto/configure`

Patch 8.0.1711

Problem: `Term_setsize()` is not implemented yet.
Solution: Implement it.
Files: `src/evalfunc.c`, `src/terminal.c`, `src/proto/terminal.pro`,
`src/testdir/test_terminal.vim`, `runtime/doc/eval.txt`

Patch 8.0.1712

Problem: Terminal scrollbar is not limited.
Solution: Add the **'terminalscroll'** option.
Files: `src/terminal.c`, `src/option.h`, `src/option.c`,
`runtime/doc/options.txt`, `runtime/doc/terminal.txt`

Patch 8.0.1713

Problem: Terminal debugger doesn't handle arguments.

Solution: Use `<f-args>` and pass all the arguments to gdb, e.g. the core file or process number. (suggested by Christian Brabandt) Disallow starting the debugger twice.

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim,
runtime/doc/terminal.txt

Patch 8.0.1714

Problem: Term_setsize() does not give an error in a normal buffer.

Solution: Add an error message.

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1715

Problem: Terminal buffer can be 1 more than `'terminalscroll'` lines.

Solution: Change `>` to `>=`.

Files: src/terminal.c

Patch 8.0.1716

Problem: Test for term_setsize() does not give a good error message.

Solution: use `assert_inrange()`.

Files: src/testdir/test_terminal.vim

Patch 8.0.1717

Problem: C89 check causes too much trouble.

Solution: Remove enforcing C89 for now.

Files: src/configure.ac, src/auto/configure

Patch 8.0.1718

Problem: Terminal scrollbar test fails on MS-Windows.

Solution: Check for the last line of output anticipating there might be an empty line below it.

Files: src/testdir/test_terminal.vim

Patch 8.0.1719

Problem: Cannot specify which Python executable configure should use.

Solution: Add `--with-python-command` and `--with-python3-command`.

Files: src/configure.ac, src/auto/configure

Patch 8.0.1720

Problem: When a timer is running a terminal window may not close after a shell has exited.

Solution: Call `job_status()` more often.

Files: src/terminal.c

Patch 8.0.1721

Problem: No test for using the `'termsize'` option.

Solution: Add a test.

Files: src/testdir/screendump.vim, src/testdir/test_terminal.vim

Patch 8.0.1722

Problem: Cannot specify a minimal size for a terminal window.

Solution: Support the `"rows*cols"` format for `'winsize'`.

Files: src/terminal.c, src/testdir/test_terminal.vim, src/option.c,

runtime/doc/options.txt

Patch 8.0.1723

Problem: Using one item array size declaration is misleading.
Solution: Instead of using "[1]" and actually using a larger array, use "[]". This is to verify that this C99 feature works for all compilers.
Files: src/structs.h, src/getchar.c

Patch 8.0.1724

Problem: Declarations cannot be halfway a block.
Solution: Move one declaration to check if this works for all compilers.
Files: src/main.c

Patch 8.0.1725

Problem: Terminal debugger doesn't handle command arguments.
Solution: Add the :TermdebugCommand command. Use a ! to execute right away. (Christian Brabandt)
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, runtime/doc/terminal.txt

Patch 8.0.1726 (after 8.0.1724)

Problem: Older MSVC doesn't support declarations halfway a block.
Solution: Move the declaration back to the start of the block.
Files: src/main.c

Patch 8.0.1727

Problem: qf_get_properties() function is too long.
Solution: Refactor the code. (Yegappan Lakshmanan, closes #2807)
Files: src/quickfix.c

Patch 8.0.1728

Problem: Condition always false, useless code.
Solution: Remove the code. (Nikolai Pavlov, closes #2808)
Files: src/message.c

Patch 8.0.1729

Problem: No comma after last enum item.
Solution: Add a few commas to check if this works for all compilers. Also add a few // comments.
Files: src/structs.h

Patch 8.0.1730

Problem: No configure check for the used C99 features.
Solution: Add a compilation check. Tentatively document C99 features.
Files: src/configure.ac, src/auto/configure, runtime/doc/develop.txt

Patch 8.0.1731

Problem: Characters deleted on completion. (Adrià Farrés)
Solution: Also check the last item for the ORIGINAL_TEXT flag. (Christian Brabandt, closes #1645)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.1732

Problem: Crash when terminal API call deletes the buffer.
Solution: Lock the buffer while calling a function. (closes #2813)
Files: src/buffer.c, src/terminal.c, src/testdir/test_terminal.vim,
src/testdir/test_autocmd.vim

Patch 8.0.1733

Problem: Incomplete testing for completion fix. (Lifepillar)
Solution: Add a test with **CTRL-P**.
Files: src/testdir/test_popup.vim

Patch 8.0.1734

Problem: Package directory not added to **'rtp'** if prefix matches.
Solution: Check the match is a full match. (Ozaki Kiichi, closes #2817)
Also handle different ways of spelling a path.
Files: src/testdir/test_packadd.vim, src/ex_cmds2.c

Patch 8.0.1735 (after 8.0.1723 and 8.0.1730)

Problem: Flexible array member feature not supported by HP-UX. (John
Marriott)
Solution: Do not use the flexible array member feature of C99.
Files: src/configure.ac, src/auto/configure, src/structs.h,
src/getchar.c, runtime/doc/develop.txt

Patch 8.0.1736

Problem: Check for C99 features is incomplete.
Solution: Use AC_PROG_CC_C99 and when C99 isn't fully supported check the
features we need. (James McCoy, closes #2820)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1737

Problem: fchown() used when it is not supported.
Solution: Add #ifdef.
Files: src/fileio.c

Patch 8.0.1738

Problem: ":args" output is hard to read.
Solution: Make columns with the names if the output is more than one line.
Files: src/ex_cmds2.c, src/version.c, src/proto/version.pro,
src/testdir/test_arglist.vim

Patch 8.0.1739

Problem: MS-Windows with msys2 cannot build Ruby statically.
Solution: Define RUBY_VERSION. (Gray Wolf, closes #2826)
Files: src/Make_cyg_ming.mak

Patch 8.0.1740

Problem: Warning for signed-unsigned incompatibility.
Solution: Change type from "char *" to "char_u *". (John Marriott)
Files: src/ex_cmds2.c

Patch 8.0.1741

Problem: MS-Windows with msys2 cannot build Ruby statically.
Solution: Add RUBY_VERSION to CFLAGS later. (Gray Wolf, closes #2833)
Files: src/Make_cyg_ming.mak

Patch 8.0.1742

Problem: Cannot get a list of all the jobs. Cannot get the command of the job.
Solution: When job_info() is called without an argument return a list of jobs. Otherwise, include the command that the job is running. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/channel.c, src/evalfunc.c, src/proto/channel.pro, src/structs.h, src/testdir/test_channel.vim

Patch 8.0.1743

Problem: Terminal window options are named inconsistently.
Solution: prefix terminal window options with "termwin". Keep the old names for now as an alias.
Files: src/option.c, src/option.h, src/structs.h, src/terminal.c, src/testdir/test_terminal.vim, src/testdir/gen_opt_test.vim, runtime/doc/options.txt, runtime/doc/quickref.txt, runtime/doc/terminal.txt, runtime/optwin.vim

Patch 8.0.1744

Problem: On some systems /dev/stdout isn't writable.
Solution: Skip test if writing is not possible. (James McCoy, closes #2830)
Files: src/testdir/test_writefile.vim

Patch 8.0.1745

Problem: Build failure on MS-Windows.
Solution: Build job arguments for MS-Windows. Fix allocating job twice.
Files: src/structs.h, src/channel.c, src/os_unix.c, src/misc2.c, src/terminal.c, src/proto/misc2.pro

Patch 8.0.1746

Problem: MS-Windows: channel tests fail.
Solution: Make a copy of the command before splitting it.
Files: src/channel.c

Patch 8.0.1747

Problem: MS-Windows: term_start() does not set job_info() cmd.
Solution: Share the code from job_start() to set jv_argv.
Files: src/testdir/test_terminal.vim, src/channel.c, src/misc2.c, src/proto/misc2.pro, src/terminal.c

Patch 8.0.1748

Problem: CmdlineEnter command uses backslash instead of slash.
Solution: Don't treat the character as a file name. (closes #2837)
Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 8.0.1749

Problem: VMS: 100% CPU use, redefining mch_open() and mch_fopen() fails.
Solution: Do not wait indefinitely in RealWaitForChar(). (Neil Rieck)
Do not redefine mch_open() and mch_fopen() on VMS. (Zoltan Arpadffy)
Files: src/os_vms.c, src/vim.h

Patch 8.0.1750

Problem: Crash when clearing location list in autocommand.
Solution: Check if "qi" equals "ql_info". (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1751

Problem: #ifdef causes bad highlighting.
Solution: Move code around. (Ozaki Kiichi, closes #2731)
Files: src/ui.c

Patch 8.0.1752

Problem: qf_set_properties() is too long.
Solution: Refactor the function. Define INVALID_QFIDX. (Yegappan Lakshmanan, closes #2812)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1753

Problem: Various warnings from a static analyser
Solution: Add type casts, remove unneeded conditions. (Christian Brabandt, closes #2770)
Files: src/evalfunc.c, src/ex_cmds2.c, src/fileio.c, src/getchar.c, src/normal.c, src/os_unix.c, src/search.c, src/term.c

Patch 8.0.1754

Problem: ex_helpgrep() is too long.
Solution: Refactor the function. (Yegappan Lakshmanan, closes #2766)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1755

Problem: MS-Windows GUI: high unicode char received as two utf-16 words.
Solution: Keep the first word until the second word is received. (Chris Morgan, closes #2800)
Files: src/gui_w32.c

Patch 8.0.1756

Problem: GUI: after prompting for a number the mouse shape is sometimes wrong.
Solution: Call setmouse() after setting "State". (Hirohito Higashi, closes #2709)
Files: src/misc1.c

Patch 8.0.1757

Problem: Unnecessary changes in libvterm.
Solution: Bring back // comments and trailing comma in enums.
Files: src/libvterm/bin/unterm.c, src/libvterm/bin/vterm-ctrl.c, src/libvterm/bin/vterm-dump.c, src/libvterm/include/vterm.h, src/libvterm/include/vterm_keycodes.h, src/libvterm/src/encoding.c, src/libvterm/src/keyboard.c, src/libvterm/src/parser.c, src/libvterm/src/pen.c, src/libvterm/src/screen.c, src/libvterm/src/state.c, src/libvterm/src/unicode.c, src/libvterm/src/utf8.h, src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h

Patch 8.0.1758

Problem: open_line() returns TRUE/FALSE for success/failure.

Solution: Return OK or FAIL.
Files: src/misc1.c, src/normal.c, src/edit.c

Patch 8.0.1759

Problem: Memory leak from duplicate options. (Yegappan Lakshmanan)
Solution: Don't set the default value twice.
Files: src/option.c

Patch 8.0.1760

Problem: Wrong number of arguments to vms_read().
Solution: Drop the first argument. (Ozaki Kiichi)
Files: src/ui.c

Patch 8.0.1761

Problem: Job in terminal window with no output channel is killed.
Solution: Keep the job running when the input is a tty. (Ozaki Kiichi, closes #2734)
Files: src/channel.c, src/os_unix.c, src/testdir/test_channel.vim

Patch 8.0.1762

Problem: Terminal debug logging is a bit complicated.
Solution: Make log_tr() use variable arguments (Ozaki Kiichi, closes #2730)
Files: src/term.c

Patch 8.0.1763

Problem: :argedit does not reuse an empty unnamed buffer.
Solution: Add the BLN_CURBUF flag and fix all the side effects. (Christian Brabandt, closes #2713)
Files: src/buffer.c, src/ex_cmds2.c, src/proto/buffer.pro, src/testdir/test_arglist.vim, src/testdir/test_command_count.vim

Patch 8.0.1764

Problem: Lgtm considers tutor.es to be EcmaScript.
Solution: Add a config file for lgtm. (Bas van Schaik, closes #2844)
Files: .lgtm.yml, Filelist

Patch 8.0.1765

Problem: **CTRL-G** j in Insert mode is incorrect when '**virtualedit**' is set.
Solution: Take coladd into account. (Christian Brabandt, closes #2743)
Files: src/charset.c, src/testdir/test_virtualedit.vim

Patch 8.0.1766 (after 8.0.1758)

Problem: Expanding abbreviation doesn't work. (Tooth Pik)
Solution: Return OK instead of FALSE and FAIL instead of TRUE. (Christian Brabandt)
Files: src/edit.c, src/testdir/test_mapping.vim

Patch 8.0.1767

Problem: With '**incsearch**' text may jump up and down. ()
Solution: Besides w_botline also save and restore w_empty_rows. (closes #2530)
Files: src/ex_getln.c, src/testdir/test_search.vim, src/testdir/dumps/Test_incsearch_scrolling_01.dump

Patch 8.0.1768

Problem: SET_NO_HLSEARCH() used in a wrong way.
Solution: Make it a function. (suggested by Dominique Pelle, closes #2850)
Files: src/vim.h, src/ex_docmd.c, src/proto/ex_docmd.pro, src/search.c, src/ex_getln.c, src/option.c, src/screen.c, src/tag.c

Patch 8.0.1769

Problem: Repeated saving and restoring viewstate for 'incsearch'.
Solution: Use a structure.
Files: src/ex_getln.c

Patch 8.0.1770

Problem: Assert functions don't return anything.
Solution: Return non-zero when the assertion fails.
Files: src/evalfunc.c, src/eval.c, src/proto/eval.pro, src/testdir/test_assert.vim, runtime/doc/eval.txt

Patch 8.0.1771

Problem: In tests, when WaitFor() fails it doesn't say why. (James McCoy)
Solution: Add WaitForAssert(), which produces an assert error when it fails.
Files: src/testdir/shared.vim, src/testdir/test_terminal.vim, src/testdir/screendump.vim, src/testdir/test_autocmd.vim, src/testdir/test_channel.vim, src/testdir/test_clientserver.vim, src/testdir/test_job_fails.vim

Patch 8.0.1772

Problem: Quickfix: mixup of FALSE and FAIL, returning -1.
Solution: Use FAIL and INVALID_QFIDX. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.0.1773

Problem: Dialog messages are not translated.
Solution: Add N_() and _() where needed. (Sergey Alyoshin)
Files: src/diff.c, src/ex_cmds2.c, src/ex_docmd.c, src/message.c, src/po/Make_cyg.mak, src/po/Make_ming.mak, src/po/Make_mvc.mak, src/po/Makefile, src/quickfix.c, src/vim.h

Patch 8.0.1774

Problem: Reading very long lines can be slow.
Solution: Read up to 1 Mbyte at a time to avoid a lot of copying. Add a check for going over the column limit.
Files: src/fileio.c

Patch 8.0.1775

Problem: MS-Windows: warning for unused variable.
Solution: Move declaration inside #ifdef. (Mike Williams)
Files: src/channel.c

Patch 8.0.1776

Problem: In tests, when WaitFor() fails it doesn't say why.
Solution: Turn a few more WaitFor() into WaitForAssert().
Files: src/testdir/test_popup.vim, src/testdir/test_quotestar.vim, src/testdir/test_search.vim, src/testdir/test_terminal.vim,

src/testdir/test_timers.vim

Patch 8.0.1777

Problem: Cannot cleanup before loading another colorscheme.
Solution: Add the ColorSchemePre autocommand event.
Files: src/fileio.c, src/syntax.c, src/vim.h, src/testdir/test_gui.vim,
runtime/colors/README.txt

Patch 8.0.1778

Problem: Script to check translations does not always work.
Solution: Go to first line before searching for MIME.
Files: src/po/check.vim

Patch 8.0.1779

Problem: Deleting in a block selection causes problems.
Solution: Check the length of the line before adding bd.textcol and
bd.textlen. (Christian Brabandt, closes #2825)
Files: src/ops.c, src/testdir/test_blockedit.vim

Patch 8.0.1780

Problem: Test fails because Vim in a terminal uses wrong **'encoding'**.
Solution: Set encoding in the test where it matters. (James McCoy,
closes #2847)
Files: src/testdir/test_terminal.vim

Patch 8.0.1781

Problem: File names in quickfix window are not always shortened.
Solution: Shorten the file name when opening the quickfix window. (Yegappan
Lakshmanan, closes #2851, closes #2846)
Files: src/testdir/test_quickfix.vim, src/fileio.c, src/proto/fileio.pro,
src/quickfix.c

Patch 8.0.1782

Problem: No simple way to label quickfix entries.
Solution: Add the "module" item, to be used instead of the file name for
display purposes. (Marcin Szamotulski, closes #1757)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/alloc.h,
src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1783

Problem: Cannot use 256 colors in a MS-Windows console.
Solution: Add 256 color support. (Nobuhiro Takasaki, closes #2821)
Files: src/misc1.c, src/option.c, src/os_win32.c, src/proto/os_win32.pro,
src/term.c, src/proto/term.pro, src/terminal.c

Patch 8.0.1784 (after 8.0.1782)

Problem: Gvim test gets stuck in dialog.
Solution: Rename the file used.
Files: src/testdir/test_quickfix.vim

Patch 8.0.1785 (after 8.0.1783)

Problem: Missing symbol in Win32 small build.
Solution: Define VTERM_ANSI_INDEX_NONE without the terminal feature. Also
fix unused function with #ifdef.

Files: src/term.c, src/os_win32.c

Patch 8.0.1786

Problem: No test for **'termwinkey'**.

Solution: Add a test. Make feedkeys() handle terminal_loop() returning before characters are consumed.

Files: src/testdir/test_terminal.vim, src/terminal.c, src/evalfunc.c, src/ex_docmd.c, src/getchar.c, src/keymap.h

Patch 8.0.1787

Problem: Cannot insert the whole cursor line.

Solution: Make **CTRL-R CTRL-L** work. (Andy Massimino, closes #2857)

Files: runtime/doc/cmdline.txt, src/ex_getln.c, src/ops.c, src/testdir/test_cmdline.vim

Patch 8.0.1788

Problem: Tool to check a color scheme is not installed.

Solution: Update the install rule. (Christian Brabandt)

Files: src/Makefile

Patch 8.0.1789

Problem: BufWinEnter does not work well for a terminal window.

Solution: Do not trigger BufWinEnter when opening a terminal window.

Files: src/terminal.c, runtime/doc/autocmd.txt, src/testdir/test_terminal.vim

Patch 8.0.1790

Problem: **'winfixwidth'** is not always respected by :close.

Solution: Prefer a frame without **'winfixwidth'** or **'winfixheight'**. (Jason Franklin)

Files: src/window.c, src/testdir/test_winbuf_close.vim

Patch 8.0.1791

Problem: Using uint8_t does not work everywhere.

Solution: Use char_u instead.

Files: src/term.c, src/proto/term.pro, src/os_win32.c

Patch 8.0.1792

Problem: MS-Windows users expect -? to work like --help.

Solution: Add -?. (Christian Brabandt, closes #2867)

Files: src/main.c

Patch 8.0.1793

Problem: No test for "vim -g".

Solution: Add a test for "-g" and "-y".

Files: src/testdir/shared.vim, src/testdir/test_gui.vim

Patch 8.0.1794

Problem: Duplicate term options after renaming.

Solution: Remove the old names **'termkey'**, **'termsize'** and **'terminalscroll'**.

Files: src/option.c, src/terminal.c, src/option.h, src/testdir/gen_opt_test.vim, src/testdir/screendump.vim

Patch 8.0.1795

Problem: Lose contact with jobs when :gui forks.
Solution: Don't fork when there is a running job. Make log message for a died job clearer. Also close the terminal when stderr and stdout are the same FD.
Files: src/gui.h, src/gui.c, src/channel.c, src/proto/channel.pro, src/os_unix.c, src/terminal.c

Patch 8.0.1796

Problem: GUI: click on tab fails when the focus is in a terminal window.
Solution: Handle K_TABLINE.
Files: src/terminal.c

Patch 8.0.1797

Problem: Terminal window is redrawn too often and scrolling is repeated.
Solution: Don't scroll immediately but only when redrawing. Avoid redrawing the whole terminal window on every change.
Files: src/terminal.c, src/screen.c, src/proto/terminal.pro

Patch 8.0.1798

Problem: MS-Windows: file considered read-only when another program has opened it.
Solution: Pass file sharing flag to CreateFile(). (Linwei, closes #2860)
Files: src/os_win32.c

Patch 8.0.1799

Problem: No test for :registers command.
Solution: Add a test. (Dominique Pelle, closes #2880)
Files: src/testdir/test_registers.vim

Patch 8.0.1800

Problem: X11: getting color is slow.
Solution: Avoid using sprintf() and XParseColor(), put the RGB values in XColor directly.
Files: src/gui_x11.c

Patch 8.0.1801

Problem: MS-Windows: redirecting terminal output does not work.
Solution: Intercept the text written to the terminal and write it to the file.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1802 (after 8.0.1802)

Problem: MS-Windows: terminal test fails.
Solution: Close redirected output file earlier.
Files: src/terminal.c

Patch 8.0.1803

Problem: Warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize it.
Files: src/terminal.c

Patch 8.0.1804

Problem: Using :normal in terminal window causes problems. (Dominique Pelle)

Solution: Don't call terminal_loop() for :normal. (closes #2886)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/evalfunc.c

Patch 8.0.1805

Problem: qf_parse_line() is too long.
Solution: Split it in parts. Properly handle vim_realloc() failing.
(Yegappan Lakshmanan, closes #2881)
Files: src/quickfix.c

Patch 8.0.1806

Problem: InsertCharPre causes problems for autocompile. (Lifepillar)
Solution: Check for InsertCharPre before calling vpeekc(). (Christian
Brabandt, closes #2876)
Files: src/edit.c, src/testdir/test_popup.vim

Patch 8.0.1807

Problem: Function to set terminal name is too long.
Solution: Refactor the function. Fix typo in test.
Files: src/term.c, src/testdir/test_options.vim

Patch 8.0.1808 (after 8.0.1807)

Problem: Can't build without TGETENT.
Solution: Add #ifdef
Files: src/term.c

Patch 8.0.1809

Problem: Various typos.
Solution: Correct the mistakes, change "cursur" to "cursor". (closes #2887)
Files: src/edit.c, src/normal.c, src/screen.c, src/proto/screen.pro,
src/ui.c

Patch 8.0.1810

Problem: Buffer of a terminal only updated in Terminal-Normal mode.
Solution: Copy the terminal window content to the buffer when in
Terminal-Job mode.
Files: src/terminal.c, src/proto/terminal.pro, src/ex_cmds2.c,
src/proto/ex_cmds2.pro

Patch 8.0.1811

Problem: No test for winrestcmd().
Solution: Add a test. (Dominique Pelle, closes #2894)
Files: src/testdir/test_window_cmd.vim

Patch 8.0.1812

Problem: The qf_jump_to_usable_window() function is too long.
Solution: Split it in parts. (Yegappan Lakshmanan, closes #2891)
Files: src/quickfix.c

Patch 8.0.1813

Problem: Windows installer doesn't install terminal debugger.
Solution: Add the package to the list of files to install.
Files: nsis/gvim.nsi

Patch 8.0.1814

Problem: Crash with terminal window and with **'lazyredraw'** set. (Antoine)
Solution: Check the terminal still exists after update_screen().
Files: src/terminal.c

Patch 8.0.1815 (after 8.0.1814)

Problem: Still a crash with terminal window and with **'lazyredraw'** set. (Antoine)
Solution: Do not wipe out the buffer when updating the screen.
Files: src/terminal.c, src/proto/terminal.pro, src/screen.c, src/proto/screen.pro, src/ui.c

Patch 8.0.1816

Problem: No test for setcmdpos().
Solution: Add a test. (Dominique Pelle, closes #2901)
Files: src/testdir/test_cmdline.vim

Patch 8.0.1817

Problem: A timer may change v:count unexpectedly.
Solution: Save and restore v:count and similar variables when a timer callback is invoked. (closes #2897)
Files: src/eval.c, src/proto/eval.pro, src/ex_cmds2.c, src/structs.h, src/testdir/test_timers.vim

Patch 8.0.1818 (after 8.0.1810)

Problem: Lines remove from wrong buffer when using terminal window.
Solution: Make sure to use tl_buffer.
Files: src/terminal.c

Patch 8.0.1819

Problem: Swap file warning for a file in a non-existing directory, if there is another with the same file name. (Juergen Weigert)
Solution: When expanding the file name fails compare the file names.
Files: src/testdir/test_swap.vim, src/memline.c

Patch 8.0.1820

Problem: Terminal window redirecting stdout does not show stderr. (Matéo Zanibelli)
Solution: When stdout is not connected to pty_master_fd then use it for stderr. (closes #2903)
Files: src/os_unix.c, src/testdir/test_terminal.vim

Patch 8.0.1821

Problem: Cursor in terminal window moves when pressing **CTRL-W**. (Dominique Pelle)
Solution: Do not move the cursor or redraw when not in Terminal-Normal mode. (closes #2904)
Files: src/terminal.c

Patch 8.0.1822

Problem: Make uninstall does not remove colors/tools.
Solution: Add a line to delete the tools directory. (Kazunobu Kuriyama)
Files: src/Makefile

Patch 8.0.1823

Problem: Test for terminal stdout redirection is flaky.
Solution: Wait for the job to finish.
Files: src/testdir/test_terminal.vim

Patch 8.0.1824

Problem: Coverity warns for variable that may be uninitialized.
Solution: Initialize the variable.
Files: src/terminal.c

Patch 8.0.1825

Problem: Might use NULL pointer when out of memory. (Coverity)
Solution: Handle NULL pointer better.
Files: src/getchar.c

Patch 8.0.1826

Problem: Configure uses old compiler flag.
Solution: Remove _DARWIN_C_SOURCE. (Kazunobu Kuriyama)
Files: src/configure.ac, src/auto/configure

Patch 8.0.1827

Problem: Compiler warning for signed/unsigned char pointers. (Cesar Romani)
Solution: Change the type of jv_argv.
Files: src/channel.c, src/structs.h

Patch 8.0.1828

Problem: Get no clue why :gui does not fork.
Solution: Add a channel log message.
Files: src/channel.c

Patch 8.0.1829

Problem: MS-Windows: script for vimdiff can't handle ! chars.
Solution: Escape the ! chars. (Hans Ginzl, closes #2896)
Files: src/dosinst.c

Patch 8.0.1830

Problem: Switching to Terminal-Normal mode does not redraw. (Dominique Pelle)
Solution: Also redraw when not updating the snapshot. (closes #2904)
Files: src/terminal.c

Patch 8.0.1831

Problem: Sometimes the quickfix title is incorrectly prefixed with ':'.
Solution: Prepend the colon in another way. (Yegappan Lakshmanan, closes #2905)
Files: src/evalfunc.c, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.0.1832

Problem: Cannot use :unlet for an environment variable.
Solution: Make it work. Use unsetenv() if available. (Yasuhiro Matsumoto, closes #2855)
Files: runtime/doc/eval.txt, src/config.h.in, src/configure.ac, src/auto/configure, src/eval.c, src/misc1.c, src/proto/misc1.pro, src/testdir/test_unlet.vim

Patch 8.0.1833

Problem: X11: ":echo 3.14" gives E806.
Solution: set LC_NUMERIC to "C". (Dominique Pelle, closes #2368)
Files: src/gui_x11.c

Patch 8.0.1834

Problem: GUI: find/replace dialog does not handle some chars properly.
Solution: Escape '?' when needed. Always escape backslash. (closes #2418, closes #2435)
Files: src/gui.c

Patch 8.0.1835

Problem: Print document name does not support multibyte.
Solution: Use StartDocW() if needed. (Yasuhiro Matsumoto, closes #2478)
Files: src/os_mswin.c

Patch 8.0.1836

Problem: Buffer-local window options may not be recent if the buffer is still open in another window.
Solution: Copy the options from the window instead of the outdated window options. (Bjorn Linse, closes #2336)
Files: src/buffer.c, src/testdir/test_options.vim

Patch 8.0.1837

Problem: One character cmdline abbreviation not triggered after '<,>'.
Solution: Skip over the special range. (Christian Brabandt, closes #2320)
Files: src/ex_getln.c, src/testdir/test_mapping.vim

Patch 8.0.1838

Problem: Cursor in wrong position when switching to Terminal-Normal mode. (Dominique Pelle)
Solution: Move to the end of the line if coladvance() fails. Do not take a snapshot a second time.
Files: src/terminal.c

Patch 8.0.1839

Problem: Script to check .po file doesn't check for plural header.
Solution: Add a check that the plural header is present when needed.
Files: src/po/check.vim

Patch 8.0.1840

Problem: getwinpos() is not tested.
Solution: Add a test. (Dominique Pelle, closes #2911)
Files: src/testdir/test_gui.vim

Patch 8.0.1841

Problem: HP-UX does not have setenv().
Solution: Use vim_setenv(). (John Marriott)
Files: src/misc1.c

Patch 8.0.1842

Problem: Popup menu inside terminal window isn't cleared.
Solution: Use NOT_VALID in pum_undisplay(). (suggested by Christian Brabandt, closes #2908)

Files: src/popupmnu.c

Patch 8.0.1843

Problem: Entry for **'wrap'** in options window is wrong. (John Little)

Solution: Make the change apply locally.

Files: runtime/optwin.vim

Patch 8.0.1844

Problem: Superfluous quickfix code, missing examples.

Solution: Remove unneeded code. Add a few examples. Add a bit more testing. (Yegappan Lakshmanan, closes #2916)

Files: runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.0.1845

Problem: Various comment updates needed, missing white space.

Solution: Update comments, add white space.

Files: src/getchar.c, src/testdir/test_cscope.vim, src/gui_mac.c

Patch 8.0.1846

Problem: Python interface is incompatible with lldb.

Solution: For OutputType set the base to be PyFile_Type. (Boxu Zhang)
Partly disabled to avoid a crash.

Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 8.0.1847

Problem: Some build options don't have an example.

Solution: Add a couple more examples and compiler flags.

Files: src/Makefile

Patch 8.0.1848

Problem: **'termwincscroll'** does not work properly. (Dominique Pelle)

Solution: Subtract removed scrollbar from the scrollbar count. Add a test for **'termwincscroll'**. (closes #2909)

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.0.1849

Problem: Compiler warning for unused arguments and missing prototype.

Solution: Add UNUSED. Add static.

Files: src/mbyte.c, src/if_ruby.c

Patch 8.0.1850

Problem: Todo items in source code not visible for users.

Solution: Move the todo items to the help file.

Files: src/terminal.c

VERSION 8.2

version-8.2 version8.2 vim-8.2

This section is about improvements made between version 8.1 and 8.2.

This release has hundreds of bug fixes, there are several new features and there are many minor improvements.

Popup windows

[new-popup-window](#)

Popup windows can be used to display text on top of other windows. This can be for a simple message such as "Build finished successfully", showing a function prototype while editing a function call, a flexible popup menu and many other purposes. See [popup-window](#) .

Popup windows are very flexible: they can be positioned relative to text, an absolute position or just in the middle of the screen. The size can be fixed or adjusts to fit the text. A "zindex" value specifies what popup window goes on top of others.

The new **'wincolor'** option allows for setting the color for the whole popup window. This also works for normal windows.

Text properties

[new-text-properties](#)

Text properties give a plugin author flexibility about what to highlight. This can be used with an external asynchronous parser to do syntax highlighting. Or to highlight text in a popup window. The text properties stick with the text when characters are deleted or inserted, which makes them also useful as text markers. See [text-properties](#) .

The listener functions have been added to report text changes to a server so that it can dynamically update highlighting, mark syntax errors and the like. See [listener_add\(\)](#) .

Vim script improvements

[new-vimscript-8.2](#)

Functions can now be called in a chain, using `"->":`

```
mylist->filter(filterexpr)->map(mapexpr)->sort()->join()
```

The new ``:eval`` command can be used if the chain has no result.

Function arguments can be made optional by giving them a default value

```
optional-function-argument :  
function Something(key, value = 10)
```

The ``:scriptversion`` command was added to allow for changes that are not backwards compatible. E.g. to only use `".."` for string concatenation, so that `"."` can be used to access a dictionary member consistently.

``:const`` was added to allow for declaring a variable that cannot change:

```
const TIMER_DELAY = 400
```

A heredoc-style assignment was added to easily assign a list of lines to a variable without quoting or line continuation:

```
let lines =<< trim END
```

```
    line one
    line two
END
```

The `Blob` type was added. This makes it easy to deal with binary data.

The `/=` and `%=` assignment operators were added.

A Dictionary can be defined with literal keys using `#{}`. This avoids having to use a lot of quotes:

```
    let options = #{width: 30, height: 24}
```

Other improvements

new-other-8.2

- When `'incsearch'` is set it also applies to `':substitute'`.
- `modifyOtherKeys` was added to allow mapping more key combinations.
- ConPTY support was added for Windows 10, supports full color in the terminal.
- The MS-Windows installer supports translations, silent install and looks much better.

Changed

changed-8.2

The `xdiff` library was included to avoid the need for an external diff program and to make updating diffs much faster.

The code is using a few more modern C features, such as `//` comments.

Support for old compilers has been dropped: Borland C++, MSVC 2008.

Hangul input support was removed, it actually didn't work anymore.

Makefiles for old Amiga compilers were removed: Dice, Manx and SAS.

If a swap file is found without any changes it is automatically deleted.

The `FEAT_TAG_OLDSTATIC` code was removed, it slowed down tag searches.

The `FEAT_TAG_ANYWHITE` code was removed, it was not enabled in any build.

The `UNICODE16` code was removed, it was not useful.

Workshop support was removed, nobody was using it.

The Aap build files were removed, they were outdated.

Farsi support was removed, it was outdated and unused.

VIMDLL was re-implemented, this shares the common parts between vim and gvim to reduce the total install size.

The following features are now included in all versions: `+multi_byte` , `+virtualedit` , `+vreplace` , `+localmap` , `+cmdline_hist` , `+cmdline_compl` , `+insert_expand` , `+modify_fname` , `+comments`

Added

added-8.2

Added functions:

All the popup_ functions.
All the prop_ functions.
All the sign_ functions.
All the sound_ functions.

```
appendbufline()
balloon_gettext()
bufadd()
bufload()
ch_readblob()
chdir()
debugbreak()
deletebufline()
environ()
expandcmd()
getenv()
getimstatus()
getmousepos()
gettagstack()
interrupt()
isinf()
list2str()
listener_add()
listener_flush()
listener_remove()
prompt_setcallback()
prompt_setinterrupt()
prompt_setprompt()
pum_getpos()
rand()
readdir()
reg_executing()
reg_recording()
rubyeval()
screenchars()
screenpos()
screenstring()
setenv()
settagstack()
srand()
state()
str2list()
strptime()
swapinfo()
swapname()
term_setapi()
test_getvalue()
test_null_blob()
test_refcount()
test_scrollbar() (later replaced with test_gui_event() )
```

```
test_setmouse()
win_execute()
win_splitmove()
winlayout()
```

Added autocommands:

```
CompleteChanged
DiffUpdated
SafeState
SafeStateAgain
SourcePost
TerminalWinOpen
```

Added commands:

Jumping to errors relative to the cursor position:

```
`:cabove`
`:cafter`
`:cbefore`
`:cbelow`
`:labove`
`:lbefore`
`:lbelow`
`:lafter`
```

Tab-local directory:

```
`:tcd`
`:tchdir`
```

Others:

```
`:const`
`:eval`
`:redrawtabline`
`:scriptversion`
`:spellrare`
`:tmenu`
`:tlnoremenu`
`:tlunmenu`
`:xrestore`
```

Added options:

```
'completepopup'
'completeslash'
'cursorlineopt'
'modelineexpr'
'previewpopup'
'scrollfocus'
'tagfunc'
'termwintype'
'varsofttabstop'
'vartabstop'
'wincolor'
```

Patches

patches-8.2

These patches were applied after the 8.1 release and are included in the 8.2 release.

Patch 8.1.0001

Problem: The netrw plugin does not work.
Solution: Make it accept version 8.x.
Files: runtime/autoload/netrw.vim

Patch 8.1.0002

Problem: :stopinsert changes the message position.
Solution: Save and restore msg_col and msg_row in clearmode(). (Jason Franklin)
Files: src/screen.c, src/testdir/test_messages.vim

Patch 8.1.0003

Problem: The :compiler command is not tested.
Solution: Add a test. (Dominique Pelle, closes #2930)
Files: src/Makefile, src/testdir/test_alot.vim, src/testdir/test_compiler.vim

Patch 8.1.0004

Problem: Test for :compiler command sometimes fails.
Solution: Be less strict about the error message. (Dominique Pelle)
Files: src/testdir/test_compiler.vim

Patch 8.1.0005

Problem: Test for :compiler command fails on MS-Windows.
Solution: Ignore difference in path.
Files: src/testdir/test_compiler.vim

Patch 8.1.0006

Problem: syn_id2cterm_bg() may be undefined. (Axel Bender)
Solution: Adjust #ifdef.
Files: src/syntax.c

Patch 8.1.0007

Problem: No test for "o" and "O" in Visual block mode.
Solution: Add a test. (Dominique Pelle, closes #2932)
Files: src/testdir/test_visual.vim

Patch 8.1.0008

Problem: No test for strwidth().
Solution: Add a test. (Dominique Pelle, closes #2931)
Files: src/testdir/test_functions.vim

Patch 8.1.0009

Problem: Tabpages insufficiently tested.
Solution: Add more test coverage. (Dominique Pelle, closes #2934)
Files: src/testdir/test_tabpage.vim

Patch 8.1.0010

Problem: efm_to_regpat() is too long.
Solution: Split off three functions. (Yegappan Lakshmanan, closes #2924)
Files: src/quickfix.c

Patch 8.1.0011

Problem: maparg() and mapcheck() confuse empty and non-existing.
Solution: Return <Nop> for an existing non-empty mapping. (closes #2940)
Files: src/evalfunc.c, src/testdir/test_maparg.vim

Patch 8.1.0012

Problem: Misplaced #endif.
Solution: Move the #endif to after the expression. (David Binderman)
Files: src/fileio.c

Patch 8.1.0013

Problem: Using freed memory when changing terminal cursor color.
Solution: Make a copy of the color. (Dominique Pelle, closes #2938, closes #2941)
Files: src/terminal.c

Patch 8.1.0014

Problem: qf_init_ext() is too long.
Solution: Split it into multiple functions. (Yegappan Lakshmanan, closes #2939)
Files: src/quickfix.c

Patch 8.1.0015

Problem: Cursor color wrong when closing a terminal window, ending up in another terminal window. (Dominique Pelle)
Solution: Bail out of terminal_loop() when the buffer changes. (closes #2942)
Files: src/terminal.c

Patch 8.1.0016

Problem: Possible crash in term_wait(). (Dominique Pelle)
Solution: Check for a valid buffer after ui_delay(). (closes #2944)
Files: src/terminal.c

Patch 8.1.0017

Problem: Shell command completion has duplicates. (Yegappan Lakshmanan)
Solution: Use a hash table to avoid duplicates. (Ozaki Kiichi, closes #539, closes #2733)
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.1.0018

Problem: Using "gn" may select wrong text when wrapping.
Solution: Avoid wrapping when searching forward. (Christian Brabandt)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.1.0019

Problem: Error when defining a Lambda with index of a function result.
Solution: When not evaluating an expression and skipping a function call, set the return value to VAR_UNKNOWN.
Files: src/userfunc.c, src/testdir/test_lambda.vim

Patch 8.1.0020

Problem: Cannot tell whether a register is being used for executing or

recording.
Solution: Add `reg_executing()` and `reg_recording()`. (Hirohito Higashi, closes #2745) Rename the global variables for consistency. Store the register name in `reg_executing`.
Files: `runtime/doc/eval.txt`, `runtime/doc/usr_41.txt`, `src/evalfunc.c`, `src/testdir/test_functions.vim`, `src/getchar.c`, `src/normal.c`, `src/ops.c`, `src/globals.h`, `src/edit.c`, `src/fileio.c`, `src/message.c`, `src/screen.c`

Patch 8.1.0021

Problem: Clang warns for undefined behavior.
Solution: Move `#ifdef` outside of `sprintf()` call. (suggestion by Michael Jarvis, closes #2946)
Files: `src/term.c`

Patch 8.1.0022

Problem: Repeating `put` from expression register fails.
Solution: Re-evaluate the expression register. (Andy Massimino, closes #2945)
Files: `src/getchar.c`, `src/testdir/test_put.vim`

Patch 8.1.0023

Problem: gcc 8.1 warns for use of `strncpy()`. (John Marriott)
Solution: Use `mch_memmove()` instead of `STRNCPY()`.
Files: `src/memline.c`

Patch 8.1.0024

Problem: `%` command not tested on `#ifdef` and comment.
Solution: Add tests. (Dominique Pelle, closes #2956)
Files: `src/testdir/test_goto.vim`

Patch 8.1.0025

Problem: No test for the `undofile()` function.
Solution: Add test. (Dominique Pelle, closes #2958)
Files: `src/testdir/test_undo.vim`

Patch 8.1.0026

Problem: Terminal test fails with very tall terminal. (Tom)
Solution: Fix the terminal window size in the test.
Files: `src/testdir/test_terminal.vim`

Patch 8.1.0027

Problem: Difficult to make a plugin that feeds a line to a job.
Solution: Add the initial code for the "prompt" buftype.
Files: `runtime/doc/channel.txt`, `runtime/doc/eval.txt`, `runtime/doc/options.txt`, `runtime/doc/tags`, `runtime/doc/todo.txt`, `src/Makefile`, `src/buffer.c`, `src/channel.c`, `src/diff.c`, `src/edit.c`, `src/evalfunc.c`, `src/normal.c`, `src/ops.c`, `src/option.c`, `src/proto/buffer.pro`, `src/proto/channel.pro`, `src/proto/edit.pro`, `src/proto/ops.pro`, `src/structs.h`, `src/testdir/Make_all.mak`, `src/testdir/screendump.vim`, `src/testdir/test_prompt_buffer.vim`

Patch 8.1.0028 (after 8.1.0027)

Problem: Prompt buffer test fails on MS-Windows.

Solution: Disable the test for now. Remove stray assert.
Files: src/testdir/test_prompt_buffer.vim

Patch 8.1.0029

Problem: Terminal test fails on MS-Windows when "wc" exists.
Solution: Skip test with redirection on MS-Windows.
Files: src/testdir/test_terminal.vim

Patch 8.1.0030

Problem: Stopping Vim running in a terminal may not work.
Solution: Instead of sending <Esc> send **CTRL-O**.
Files: src/testdir/screendump.vim, src/testdir/test_prompt_buffer.vim

Patch 8.1.0031

Problem: Terminal test aucmd_on_close is flaky.
Solution: Wait a bit longer.
Files: src/testdir/test_terminal.vim

Patch 8.1.0032

Problem: BS in prompt buffer starts new line.
Solution: Do not allow BS over the prompt. Make term_sendkeys() handle special keys. Add a test.
Files: src/option.c, src/terminal.c, src/testdir/test_prompt_buffer.vim

Patch 8.1.0033

Problem: Keys to stop Vim in terminal are wrong. (Marius Gedminas)
Solution: Move ":" to before **CTRL-U**.
Files: src/testdir/screendump.vim

Patch 8.1.0034

Problem: Cursor not restored with ":edit #".
Solution: Don't assume autocommands moved the cursor when it was moved to the first non-blank.
Files: src/ex_cmds.c, src/testdir/test_edit.vim

Patch 8.1.0035

Problem: Not easy to switch between prompt buffer and other windows.
Solution: Accept **CTRL-W** commands in Insert mode. Start and stop Insert mode as one would expect.
Files: src/edit.c, src/ex_docmd.c, src/structs.h, src/window.c

Patch 8.1.0036

Problem: Not restoring Insert mode if leaving a prompt buffer by using a mouse click.
Solution: Set b_prompt_insert appropriately. Also correct cursor position when moving cursor to last line.
Files: src/buffer.c, src/edit.c, src/window.c

Patch 8.1.0037

Problem: Cannot easily append lines to another buffer.
Solution: Add appendbuflineline().
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_buflineline.vim, src/testdir/test_edit.vim

Patch 8.1.0038

Problem: Popup test causes Vim to exit.
Solution: Disable the broken part of the test for now.
Files: src/testdir/test_popup.vim

Patch 8.1.0039

Problem: Cannot easily delete lines in another buffer.
Solution: Add deletebufline().
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_bufline.vim

Patch 8.1.0040

Problem: Warnings from 64-bit compiler.
Solution: Add type casts. (Mike Williams)
Files: src/edit.c

Patch 8.1.0041

Problem: Attribute "width" missing from python window attribute list.
Solution: Add the item. (Ken Takata) Order the list like the items are used in the WindowAttr() function.
Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok

Patch 8.1.0042

Problem: If omni completion opens a window Insert mode is stopped. (Hirohito Higashi)
Solution: Only set stop_insert_mode in a prompt buffer window.
Files: src/window.c

Patch 8.1.0043

Problem: ++bad argument of :edit does not work properly.
Solution: Return FAIL from get_bad_opt() only when there is no valid argument. (Dominique Pelle, Christian Brabandt, closes #2966, closes #2947)
Files: src/ex_docmd.c, src/testdir/test_plus_arg_edit.vim

Patch 8.1.0044

Problem: If a test function exits Vim this may go unnoticed.
Solution: Check for a test function quitting Vim. Fix tests that did exit Vim.
Files: src/testdir/runtest.vim, src/testdir/test_assert.vim

Patch 8.1.0045 (after 8.1.0038)

Problem: Popup test isn't run completely.
Solution: Remove "finish". Clean up function definitions.
Files: src/testdir/test_popup.vim

Patch 8.1.0046

Problem: Loading a session file fails if 'winheight' is a big number.
Solution: Set 'minwinheight' to zero at first. Don't give an error when setting 'minwinheight' while 'winheight' is a big number. Fix using vertical splits. Fix setting 'minwinwidth'. (closes #2970)
Files: src/testdir/test_mksession.vim, src/option.c, src/window.c, src/proto/window.pro

Patch 8.1.0047

Problem: No completion for :unlet \$VAR.
Solution: Add completion. (Jason Franklin)
Files: src/ex_docmd.c, src/testdir/test_unlet.vim

Patch 8.1.0048

Problem: vim_str2nr() does not handle numbers close to the maximum.
Solution: Check for overflow more precisely. (Ken Takata, closes #2746)
Files: src/charset.c

Patch 8.1.0049

Problem: Shell cannot tell running in a terminal window.
Solution: Add the VIM_TERMINAL environment variable. (Christian Brabandt)
Files: runtime/doc/terminal.txt, src/os_unix.c, src/os_win32.c,
src/testdir/test_terminal.vim

Patch 8.1.0050 (after 8.1.0049)

Problem: \$VIM_TERMINAL is also set when not in a terminal window.
Solution: Pass a flag to indicate whether the job runs in a terminal.
Files: src/channel.c, src/proto/channel.pro, src/evalfunc.c,
src/terminal.c, src/os_unix.c, src/proto/os_unix.pro,
src/os_win32.c

Patch 8.1.0051 (after 8.1.0050)

Problem: MS-Windows: missing #endif.
Solution: Add the #endif.
Files: src/os_win32.c

Patch 8.1.0052

Problem: When a mapping to <Nop> times out the next mapping is skipped.
Solution: Reset "timedout" when waiting for a character. (Christian Brabandt, closes #2921)
Files: src/getchar.c

Patch 8.1.0053

Problem: The first argument given to 'completefunc' can be Number or String, depending on the value.
Solution: Avoid guessing the type of an argument, use typval_T in the callers of call_vim_function(). (Ozaki Kiichi, closes #2993)
Files: src/edit.c, src/eval.c, src/ex_getln.c, src/mbyte.c, src/normal.c,
src/proto/eval.pro, src/testdir/test_ins_complete.vim

Patch 8.1.0054

Problem: Compiler warning for using %ld for "long long".
Solution: Add a type cast. (closes #3002)
Files: src/os_unix.c

Patch 8.1.0055 (after 8.1.0053)

Problem: Complete test has wrong order of arguments. Wrong type for sentinel variable.
Solution: Swap arguments, use VAR_UNKNOWN. (Ozaki Kiichi)
Files: src/mbyte.c, src/testdir/test_ins_complete.vim

Patch 8.1.0056

Problem: Crash when using :hardcopy with illegal byte.
Solution: Check for string_convert() returning NULL. (Dominique Pelle)
Files: src/hardcopy.c, src/testdir/test_hardcopy.vim

Patch 8.1.0057

Problem: Popup menu displayed wrong when using autocmd.
Solution: Use aucmd_prebuf(). Force updating status line if the popup menu is going to be redrawn anyway. (Christian Brabandt, closes #3009)
Files: src/edit.c, src/screen.c, src/proto/screen.pro

Patch 8.1.0058

Problem: Display problem with margins and scrolling.
Solution: Place the cursor in the right column. (Kouichi Iwamoto, closes #3016)
Files: src/screen.c

Patch 8.1.0059

Problem: Displayed digraph for "ga" wrong with 'encoding' "cp1251".
Solution: Convert from 'encoding' to "utf-8" if needed. (closes #3015)
Files: src/digraph.c, src/testdir/test_digraph.vim

Patch 8.1.0060

Problem: Crash when autocommands delete the current buffer. (Dominique Pelle)
Solution: Check that autocommands don't change the buffer.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0061

Problem: Window title is wrong after resetting and setting 'title'.
Solution: Move resetting the title into maketitle(). (Jason Franklin)
Files: src/option.c, src/buffer.c

Patch 8.1.0062

Problem: Popup menu broken if a callback changes the window layout. (Qiming Zhao)
Solution: Recompute the popup menu position if needed. Redraw the ruler even when the popup menu is displayed.
Files: src/popupmnu.c, src/proto/popupmnu.pro, src/screen.c

Patch 8.1.0063

Problem: Mac: NSStringPboardType is deprecated.
Solution: Use NSPasteboardTypeString. (Akshay Hegde, closes #3022)
Files: src/os_macosx.m

Patch 8.1.0064

Problem: Typing **CTRL-W** in a prompt buffer shows mode "-- --".
Solution: Set restart_edit to 'A' and check for it.
Files: src/edit.c, src/window.c, src/screen.c

Patch 8.1.0065 (after 8.1.0062)

Problem: Balloon displayed at the wrong position.
Solution: Do not reposition the popup menu at the cursor position.
Files: src/popupmnu.c

Patch 8.1.0066

Problem: Nasty autocommand causes using freed memory. (Dominique Pelle)
Solution: Do not force executing autocommands if the value of **'syntax'** or **'filetype'** did not change.
Files: src/option.c

Patch 8.1.0067

Problem: Syntax highlighting not working when re-entering a buffer.
Solution: Do force executing autocommands when not called recursively.
Files: src/option.c

Patch 8.1.0068

Problem: Nasty autocommands can still cause using freed memory.
Solution: Disallow using setloclist() and setqflist() recursively.
Files: src/evalfunc.c

Patch 8.1.0069

Problem: Cannot handle pressing **CTRL-C** in a prompt buffer.
Solution: Add prompt_setinterrupt().
Files: runtime/doc/eval.txt, src/edit.c, src/evalfunc.c, src/channel.c, src/proto/channel.pro

Patch 8.1.0070

Problem: Missing part of the changes for prompt_setinterrupt().
Solution: Add the missing changes.
Files: src/structs.h

Patch 8.1.0071

Problem: Terminal debugger only works with the terminal feature.
Solution: Make it also work with a prompt buffer. Makes it possible to use on MS-Windows. Various other improvements. (closes #3012)
Files: runtime/doc/terminal.txt, runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0072

Problem: Use of **'termwinkey'** is inconsistent.
Solution: Change the documentation and the behavior. (Ken Takata)
Files: src/terminal.c, runtime/doc/terminal.txt

Patch 8.1.0073

Problem: Crash when autocommands call setloclist(). (Dominique Pelle)
Solution: If the quickfix list changes then don't jump to the error.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0074 (after 8.1.0073)

Problem: Crash when running quickfix tests.
Solution: Do not alloc a new location list when checking for the reference to be still valid.
Files: src/quickfix.c

Patch 8.1.0075

Problem: No Vim logo in README file.
Solution: Add one. (Árni Dagur, closes #3024)
Files: README.md

Patch 8.1.0076

Problem: Command getting cleared with **CTRL-W** : in a terminal window. (Jason Franklin)
Solution: Call `redraw_after_callback()` when editing the command line.
Files: `src/terminal.c`

Patch 8.1.0077

Problem: Header of README file is not nice.
Solution: Move text to the bottom.
Files: `README.md`

Patch 8.1.0078

Problem: "... " used inconsistently in messages.
Solution: Drop the space before " ...".
Files: `src/spellfile.c`, `src/regexp_nfa.c`

Patch 8.1.0079

Problem: Superfluous space in messages.
Solution: Remove the spaces. (closes #3030)
Files: `src/gui_w32.c`

Patch 8.1.0080

Problem: Can't see the breakpoint number in the terminal debugger.
Solution: Use the breakpoint number for the sign. (Christian Brabandt)
Files: `runtime/doc/terminal.txt`,
`runtime/pack/dist/opt/termdebug/plugin/termdebug.vim`

Patch 8.1.0081

Problem: The terminal debugger doesn't adjust to changed **'background'**.
Solution: Add an OptionSet autocommand. (Christian Brabandt)
Files: `runtime/pack/dist/opt/termdebug/plugin/termdebug.vim`

Patch 8.1.0082

Problem: In terminal window, typing `:` at more prompt, inserts `:` instead of starting another Ex command.
Solution: Add `skip_term_loop` and set it when putting `:` in the typeahead buffer.
Files: `src/globals.h`, `src/main.c`, `src/message.c`

Patch 8.1.0083

Problem: "is" and "as" have trouble with quoted punctuation.
Solution: Check for punctuation before a quote. (Jason Franklin)
Files: `src/search.c`, `src/testdir/test_textobjects.vim`

Patch 8.1.0084

Problem: User name completion does not work on MS-Windows.
Solution: Use `NetUserEnum()` to get user names. (Yasuhiro Matsumoto)
Files: `src/Make_ivc.mak`, `src/Make_cyg_ming.mak`, `src/Make_mvc.mak`,
`src/misc1.c`

Patch 8.1.0085

Problem: No test for completing user name and language.
Solution: Add tests. (Dominique Pelle, closes #2978)

Files: src/testdir/test_cmdline.vim

Patch 8.1.0086

Problem: No tests for libcall() and libcallnr().
Solution: Add tests. (Dominique Pelle, closes #2982)
Files: src/testdir/test_functions.vim

Patch 8.1.0087

Problem: v:shell_error is always zero when using terminal for "!cmd".
Solution: Use "exitval" of terminal-job. (Ozaki Kiichi, closes #2994)
Files: src/os_unix.c, src/os_win32.c, src/proto/terminal.pro,
src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0088

Problem: Terminal test for stdout and stderr is a bit flaky.
Solution: Wait for both stdout and stderr to have been processed. (Ozaki
Kiichi, closes #2991)
Files: src/testdir/test_terminal.vim

Patch 8.1.0089

Problem: error when ending the terminal debugger
Solution: Fix deleting defined signs for breakpoints. Make the debugger
work better on MS-Windows.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0090

Problem: "... " used inconsistently in a message.
Solution: Define the message with " ..." once. (hint by Ken Takata)
Files: src/regexp_nfa.c

Patch 8.1.0091

Problem: MS-Windows: Cannot interrupt gdb when program is running.
Solution: Add debugbreak() and use it in the terminal debugger.
Respect 'modified' in a prompt buffer.
Files: src/evalfunc.c, runtime/doc/eval.txt, src/undo.c,
runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0092 (after 8.1.0091)

Problem: Prompt buffer test fails.
Solution: Set 'nomodified' before closing the window. (Ozaki Kiichi,
closes #3051)
Files: src/testdir/test_prompt_buffer.vim

Patch 8.1.0093

Problem: non-MS-Windows: Cannot interrupt gdb when program is running.
Solution: Only use debugbreak() on MS-Windows.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0094

Problem: Help text "usage:" is not capitalized.
Solution: Make it "Usage:". (closes #3044)
Files: src/main.c

Patch 8.1.0095

Problem: Dialog for ":browse tabnew" says "new window".
Solution: Use "new tab page". (closes #3053)
Files: src/ex_docmd.c

Patch 8.1.0096

Problem: Inconsistent use of the word autocommands.
Solution: Don't use auto-commands or "auto commands".
Files: src/fileio.c

Patch 8.1.0097

Problem: Superfluous space before exclamation mark.
Solution: Remove the space. Don't translate debug message.
Files: src/regexp_nfa.c

Patch 8.1.0098

Problem: Segfault when pattern with \z() is very slow.
Solution: Check for NULL regprog. Add "nfa_fail" to test_override() to be able to test this. Fix that 'searchhl' resets called_emsg.
Files: src/syntax.c, runtime/doc/eval.txt, src/evalfunc.c, src/vim.h, src/testdir/test_syntax.vim, src/globals.h, src/screen.c, src/regexp.c, src/regexp_nfa.c

Patch 8.1.0099

Problem: Exclamation mark in error message not needed.
Solution: Remove the exclamation mark.
Files: src/regexp_nfa.c

Patch 8.1.0100

Problem: Terminal debugger: error when setting a watch point.
Solution: Don't try defining a sign for a watch point.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0101

Problem: No test for getcmdwintype().
Solution: Add a test. (Dominique Pelle, closes #3068)
Files: src/testdir/test_cmdline.vim

Patch 8.1.0102

Problem: Cannot build without syntax highlighting.
Solution: Add #ifdef around using reg_do_extmatch.
Files: src/regexp.c

Patch 8.1.0103

Problem: Long version string cannot be translated.
Solution: Build the string in init_longVersion().
Files: src/globals.h, src/version.h, src/version.c, src/proto/version.pro, src/main.c

Patch 8.1.0104

Problem: Can't build without the +eval feature.
Solution: Add #ifdef.
Files: src/regexp_nfa.c

Patch 8.1.0105

Problem: All tab stops are the same.
Solution: Add the variable tabstop feature. (Christian Brabandt, closes #2711)
Files: runtime/doc/change.txt, runtime/doc/options.txt, runtime/doc/various.txt, runtime/optwin.vim, src/beval.c, src/beval.h, src/buffer.c, src/charset.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/feature.h, src/gui_beval.c, src/gui_w32.c, src/hardcopy.c, src/message.c, src/misc1.c, src/ops.c, src/option.c, src/option.h, src/proto/misc1.pro, src/proto/option.pro, src/screen.c, src/structs.h, src/testdir/Make_all.mak, src/testdir/gen_opt_test.vim, src/testdir/test_breakindent.vim, src/testdir/test_vartabs.vim, src/version.c, src/workshop.c, src/Makefile

Patch 8.1.0106 (after 8.1.0103)

Problem: Build fails when HAVE_DATE_TIME is undefined.
Solution: Always define init_longVersion(). (Christian Brabandt, closes #3075)
Files: src/version.c

Patch 8.1.0107

Problem: Python: getting buffer option clears message. (Jacob Niehus)
Solution: Don't use aucmd_prebuf(). (closes #3079)
Files: src/option.c

Patch 8.1.0108

Problem: No Danish translations.
Solution: Add Danish message translations. (closes #3073) Move list of languages to a common makefile.
Files: src/po/Makefile, src/po/Make_cyg.mak, src/po/Make_mvc.mak, src/po/Make_ming.mak, src/po/Make_all.mak, src/po/da.po

Patch 8.1.0109

Problem: New po makefile missing from distribution.
Solution: Add it to the file list.
Files: Filelist

Patch 8.1.0110

Problem: File name not displayed with ":file" when 'F' is in '**shortmess**'.
Solution: Always display the file name when there is no argument (Christian Brabandt, closes #3070)
Files: src/ex_cmds.c, src/testdir/test_options.vim

Patch 8.1.0111

Problem: .po files do not use recommended names.
Solution: Give a warning if the recommended name is not used. Accept the recommended name for conversion. (Christian Brabandt, Ken Takata)
Files: src/po/Makefile, src/po/sjiscorr.c, src/po/check.vim

Patch 8.1.0112

Problem: No error when using bad arguments with searchpair().
Solution: Add error messages.
Files: src/evalfunc.c, src/testdir/test_search.vim

Patch 8.1.0113

Problem: Compiler warning for unused variable. (Yegappan Lakshmanan)
Solution: Add UNUSED. (Christian Brabandt)
Files: src/screen.c

Patch 8.1.0114

Problem: Confusing variable name.
Solution: Rename new_ts to new_vts_array. Change zero to NULL.
Files: src/ex_cmds.c, src/option.c

Patch 8.1.0115

Problem: The matchparen plugin may throw an error.
Solution: Change the skip argument from zero to "0".
Files: runtime/plugin/matchparen.vim

Patch 8.1.0116

Problem: Display problem with **'vartabstop'** and **'linebreak'**. (Chauca Fuentes)
Solution: Call tabstop_padding(). (Christian Brabandt, closes #3076)
Files: src/screen.c, src/testdir/test_vartabs.vim

Patch 8.1.0117

Problem: URL in install program still points to SourceForge.
Solution: Change it to www.vim.org. (closes #3100)
Files: src/dosinst.c

Patch 8.1.0118

Problem: Duplicate error message for put command.
Solution: Check return value of u_save(). (Jason Franklin)
Files: src/ops.c, src/testdir/test_messages.vim src/testdir/test_put.vim

Patch 8.1.0119

Problem: Failing test goes unnoticed because testdir/messages is not written.
Solution: Set **'nomodifiable'** only local to the buffer.
Files: src/testdir/test_put.vim

Patch 8.1.0120

Problem: Buffer **'modified'** set even when :sort has no changes.
Solution: Only set **'modified'** when lines are moved. (Jason Franklin)
Files: src/ex_cmds.c, src/testdir/test_sort.vim

Patch 8.1.0121

Problem: Crash when using ballooneval related to **'vartabstop'**.
Solution: Initialize balloonEval->vts to NULL. (Markus Braun)
Files: src/ex_cmds2.c, src/gui_beval.c, src/gui_w32.c, src/gui.c

Patch 8.1.0122

Problem: Translators don't always understand the maintainer message.
Solution: Add a comment that ends up in the generated po file. (Christian Brabandt, closes #3037)
Files: src/message.c

Patch 8.1.0123

Problem: MS-Windows: colors are wrong after setting **'notgc'**.
Solution: Only call control_console_color_rgb() for the win32 terminal.
(Nobuhiro Takasaki, closes #3107)
Files: src/option.c

Patch 8.1.0124

Problem: has('vcon') returns true even for non-win32 terminal.
Solution: Check the terminal type. (Nobuhiro Takasaki, closes #3106)
Files: src/evalfunc.c

Patch 8.1.0125

Problem: Virtual edit replace with multibyte fails at end of line. (Lukas Werling)
Solution: use ins_char() to add the character. (Christian Brabandt, closes #3114) Rename PCHAR() to PBYTE() to avoid mistakes like this.
Files: src/ops.c, src/testdir/test_virtualedit.vim, src/macros.h

Patch 8.1.0126

Problem: Various problems with **'vartabstop'**.
Solution: Fix memory leak. Fix crash. Add a few more tests. (Christian Brabandt, closes #3076)
Files: src/ex_cmds.c, src/option.c, src/screen.c, src/testdir/test_vartabs.vim

Patch 8.1.0127

Problem: Build failure when disabling the session feature. (Pawel Slowik)
Solution: Adjust #ifdef for vim_chdirfile().
Files: src/misc2.c

Patch 8.1.0128

Problem: Building with MinGW does not work out-of-the-box.
Solution: Add instructions for MSYS2. Set default WINVER. Add batch files to set \$PATH for MSYS2.
Files: src/Make_cyg_ming.mak, src/INSTALLpc.txt, src/msys32.bat, src/msys64.bat, Filelist

Patch 8.1.0129

Problem: Still some xterm-like terminals get a stray "p" on startup.
Solution: Consider all terminals that reply with a version smaller than 95 as not an xterm. (James McCoy)
Files: src/term.c

Patch 8.1.0130

Problem: ":profdel func" does not work if func was called already. (Dominique Pelle)
Solution: Reset uf_profiling and add a flag to indicate initialization was done.
Files: src/structs.h, src/userfunc.c

Patch 8.1.0131

Problem: :profdel is not tested.
Solution: Add a test. (Dominique Pelle, closes #3123)
Files: src/testdir/test_profile.vim

Patch 8.1.0132

Problem: Lua tests are old style.

Solution: Convert to new style tests. Improve coverage. (Dominique Pelle, closes #3091)

Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/Make_amiga.mak, src/testdir/Make_vms.mms,
src/testdir/test85.in, src/testdir/test_lua.vim

Patch 8.1.0133

Problem: tagfiles() can have duplicate entries.

Solution: Simplify the filename to make checking for duplicates work better. Add a test. (Dominique Pelle, closes #2979)

Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.1.0134

Problem: Lua interface does not support funcref.

Solution: Add funcref support. (Luis Carvalho)

Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.1.0135

Problem: Undo message delays screen update for **CTRL-O** u.

Solution: Add msg_attr_keep(). (closes #3125)

Files: src/message.c, src/proto.h, src/undo.c

Patch 8.1.0136

Problem: Lua tests don't cover new features.

Solution: Add more tests. (Dominique Pelle, closes #3130)

Files: runtime/doc/if_lua.txt, src/testdir/test_lua.vim

Patch 8.1.0137

Problem: CI does not run with TCL.

Solution: Add TCL to the travis config. (Dominique Pelle, closes #3133)

Files: .travis.yml

Patch 8.1.0138

Problem: Negative value of **'softtabstop'** not used correctly.

Solution: Use get_sts_value(). (Tom Ryder)

Files: src/edit.c, src/option.c, src/Makefile, src/testdir/test_tab.vim

Patch 8.1.0139

Problem: Lua tests fail on some platforms.

Solution: Accept a hex number with and without "0x". (Ken Takata, closes #3137)

Files: src/testdir/test_lua.vim

Patch 8.1.0140

Problem: Recording into a register has focus events. (Michael Naumann)

Solution: Don't record K_FOCUSGAINED and K_FOCUSLOST. (closes #3143)

Files: src/getchar.c

Patch 8.1.0141

Problem: :cexpr no longer jumps to the first error.

Solution: Use the quickfix list identifier. (Yegappan Lakshmanan,

closes #3092)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0142

Problem: Xterm and vt320 builtin termcap missing keypad keys.
Solution: Add the escape sequences. (Kouichi Iwamoto, closes #2973)
Files: src/term.c

Patch 8.1.0143

Problem: Matchit and matchparen don't handle E363.
Solution: Catch the E363 error. (Christian Brabandt)
Files: runtime/pack/dist/opt/matchit/plugin/matchit.vim,
runtime/plugin/matchparen.vim

Patch 8.1.0144

Problem: The :cd command does not have good test coverage.
Solution: Add more tests. (Dominique Pelle, closes #2972)
Files: src/testdir/test_cd.vim

Patch 8.1.0145

Problem: Test with grep is failing on MS-Windows.
Solution: Skip the test.
Files: src/testdir/test_quickfix.vim

Patch 8.1.0146

Problem: When \$LANG is set the compiler test may fail.
Solution: Unset \$LANG.
Files: src/testdir/test_compiler.vim

Patch 8.1.0147

Problem: Compiler warning when building with Python 3.7.
Solution: #undef PySlice_GetIndicesEx before redefining it. (Ozaki Kiichi,
closes #3153)
Files: src/if_python3.c

Patch 8.1.0148

Problem: Memory leak when using :tcl expr command.
Solution: Free the result of expression evaluation. (Dominique Pelle,
closes #3150)
Files: src/if_tcl.c

Patch 8.1.0149

Problem: The generated sessions file does not restore tabs properly if :lcd
was used in one of them.
Solution: Create the tab pages before setting the directory. (Yee Cheng
Chin, closes #3152)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0150

Problem: Insufficient test coverage for Tcl.
Solution: Add more tests. (Dominique Pelle, closes #3140)
Files: src/testdir/test_tcl.vim

Patch 8.1.0151

Problem: Mksession test fails on MS-Windows.
Solution: Always use an argument for :lcd.
Files: src/testdir/test_mksession.vim

Patch 8.1.0152

Problem: Cannot easily run individual tests on MS-Windows.
Solution: Move the list of tests to a separate file. Add a build rule in the MSVC makefile.
Files: Filelist, src/Makefile, src/Make_all.mak, src/Make_mvc.mak

Patch 8.1.0153 (after 8.1.0152)

Problem: Build with SHADOWDIR fails. (Elimar Riesebieter)
Solution: Create a link for Make_all.mak. (Tony Mechelynck)
Files: src/Makefile

Patch 8.1.0154

Problem: Crash with "set smarttab shiftwidth=0 softtabstop=-1".
Solution: Fall back to using '**tabstop**'. (closes #3155)
Files: src/edit.c, src/testdir/test_tab.vim

Patch 8.1.0155

Problem: Evim.man missing from the distribution.
Solution: Add it to the list.
Files: Filelist

Patch 8.1.0156

Problem: MS-Windows compiler warning.
Solution: Add a type cast. (Mike Williams)
Files: src/version.c

Patch 8.1.0157

Problem: Old iTerm2 is not recognized, resulting in stray output.
Solution: Recognize the termresponse.
Files: src/term.c

Patch 8.1.0158

Problem: GUI: input() fails if **CTRL-C** was pressed before. (Michael Naumann)
Solution: call vpeekc() to drop the **CTRL-C** from the input stream.
Files: src/ex_docmd.c

Patch 8.1.0159

Problem: Completion for user names does not work if a prefix is also a full matching name. (Nazri Ramliy)
Solution: Accept both full and partial matches. (Dominique Pelle)
Files: src/misc1.c, src/ex_docmd.c

Patch 8.1.0160

Problem: No Danish manual translations.
Solution: Add the Danish manual translations to the file list.
Files: Filelist

Patch 8.1.0161

Problem: Buffer not updated with '**autoread**' set if file was deleted. (Michael Naumann)

Solution: Don't set the timestamp to zero. (closes #3165)
Files: src/fileio.c, src/testdir/test_stat.vim

Patch 8.1.0162

Problem: Danish and German man pages are not installed. (Tony Mechelynck)
Solution: Adjust the makefile
Files: src/Makefile

Patch 8.1.0163

Problem: Insufficient testing for Tcl.
Solution: Add a few more tests. (Dominique Pelle, closes #3166)
Files: src/testdir/test_tcl.vim

Patch 8.1.0164

Problem: luaeval('vim.buffer().name') returns an error.
Solution: Return an empty string. (Dominique Pelle, closes #3167)
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.1.0165

Problem: :clist output can be very long.
Solution: Support filtering :clist entries. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0166

Problem: Using dict_add_nr_str() is clumsy.
Solution: Split into two functions. (Ozaki Kiichi, closes #3154)
Files: src/channel.c, src/dict.c, src/edit.c, src/evalfunc.c,
src/ex_cmds2.c, src/ops.c, src/option.c, src/proto/dict.pro,
src/quickfix.c, src/tag.c, src/terminal.c, src/undo.c

Patch 8.1.0167

Problem: Lock flag in new dictitem is reset in many places.
Solution: Always reset the lock flag.
Files: src/dict.c, src/channel.c, src/ex_cmds2.c, src/userfunc.c,
src/if_perl.xs, src/if_py_both.h

Patch 8.1.0168

Problem: Output of :marks is too short with multibyte chars. (Tony Mechelynck)
Solution: Get more bytes from the text line.
Files: src/mark.c, src/testdir/test_marks.vim

Patch 8.1.0169 (after 8.1.0165)

Problem: Calling message_filtered() a bit too often.
Solution: Only call message_filtered() when filtering is already false.
Files: src/quickfix.c, runtime/doc/quickfix.txt

Patch 8.1.0170

Problem: Invalid memory use with complicated pattern. (Andy Massimino)
Solution: Reallocate the list of listids when needed. (closes #3175)
Remove unnecessary function prototypes.
Files: src/regexp_nfa.c

Patch 8.1.0171

Problem: Typing **CTRL-W** n in a terminal window causes ml_get error.
Solution: When resizing the terminal outside of terminal_loop() make sure the snapshot is complete.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0172

Problem: **'viminfofile'** option does not behave like a file name.
Solution: Add the P_EXPAND flag. (closes #3178)
Files: src/option.c

Patch 8.1.0173

Problem: Compiler warning on MS-Windows.
Solution: Add type cast. (Mike Williams)
Files: src/libvterm/src/state.c

Patch 8.1.0174

Problem: After paging up and down fold line is wrong.
Solution: Correct the computation of w_topline and w_botline. (Hirohito Higashi)
Files: src/move.c, src/testdir/test_fold.vim

Patch 8.1.0175

Problem: Marks test fails in very wide window. (Vladimir Lomov)
Solution: Extend the text to match **'columns'**. (closes #3180, closes #3181)
Files: src/testdir/test_marks.vim

Patch 8.1.0176

Problem: Overlapping string argument for strcpy(). (Coverity)
Solution: Use STRMOVE() instead of STRCPY(). (Dominique Pelle, closes #3187)
Files: src/term.c

Patch 8.1.0177

Problem: Defining function in sandbox is inconsistent, cannot use :function but can define a lambda.
Solution: Allow defining a function in the sandbox, but also use the sandbox when executing it. (closes #3182)
Files: src/userfunc.c, src/ex_cmds.h

Patch 8.1.0178

Problem: Warning for passing pointer to non-pointer argument.
Solution: Use zero instead of NULL.
Files: src/if_ole.cpp

Patch 8.1.0179

Problem: Redundant condition for boundary check.
Solution: Remove the condition. (Dominique Pelle). Change FALSE to FAIL.
Files: src/undo.c

Patch 8.1.0180

Problem: Static analysis errors in Lua interface. (Coverity)
Solution: Check for NULL pointers.
Files: src/if_lua.c

Patch 8.1.0181

Problem: Memory leak with trailing characters in skip expression.
Solution: Free the return value.
Files: src/eval.c, src/testdir/test_search.vim

Patch 8.1.0182

Problem: Unicode standard was updated.
Solution: Include the changes. (Christian Brabandt)
Files: src/mbyte.c

Patch 8.1.0183

Problem: Lua API changed, breaking the build.
Solution: Adjust prototype of lua_rawgeti(). (Ken Takata, closes #3157, closes #3144)
Files: src/if_lua.c

Patch 8.1.0184

Problem: Not easy to figure out the window layout.
Solution: Add "wincol" and "winrow" to what getwininfo() returns.
Files: src/evalfunc.c, src/testdir/test_bufwintabinfo.vim, runtime/doc/eval.txt

Patch 8.1.0185

Problem: Running tests writes lua.vim even though it is not used.
Solution: Stop writing lua.vim.
Files: src/testdir/test1.in, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, src/testdir/Makefile

Patch 8.1.0186

Problem: Test for getwininfo() fails in GUI.
Solution: Account for missing tabline.
Files: src/testdir/test_bufwintabinfo.vim

Patch 8.1.0187 (after 8.1.0184)

Problem: getwininfo() and win_screenpos() return different numbers.
Solution: Add one to "wincol" and "winrow" from getwininfo().
Files: src/evalfunc.c, src/testdir/test_bufwintabinfo.vim, runtime/doc/eval.txt

Patch 8.1.0188

Problem: No test for ":cscope add".
Solution: Add a test. (Dominique Pelle, closes #3212)
Files: src/testdir/test_cscope.vim

Patch 8.1.0189

Problem: Function defined in sandbox not tested.
Solution: Add a test.
Files: src/testdir/test_functions.vim

Patch 8.1.0190

Problem: Perl refcounts are wrong.
Solution: Improve recounting. Add a test. (Damien)
Files: src/if_perl.xs, src/testdir/test_perl.vim

Patch 8.1.0191 (after 8.1.0190)

Problem: Perl test fails in 24 line terminal.

Solution: Create fewer windows.

Files: src/testdir/test_perl.vim

Patch 8.1.0192

Problem: Executing regexp recursively fails with a crash.

Solution: Move global variables into "rex".

Files: src/regexp.c, src/regexp.h, src/regexp_nfa.c

Patch 8.1.0193

Problem: Terminal debugger buttons don't always work. (Dominique Pelle)

Solution: Set '**cpo**' to its default value.

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0194

Problem: Possibly use of NULL pointer. (Coverity)

Solution: Reset the re_in_use flag earlier.

Files: src/regexp.c

Patch 8.1.0195

Problem: Terminal debugger commands don't always work. (Dominique Pelle)

Solution: Set '**cpo**' to its default value when defining commands. (Christian Brabandt)

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0196

Problem: Terminal debugger error with .gdbinit file.

Solution: Check two lines for the "new ui" response. (hint from Hirohito Higashi)

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0197

Problem: Windows GUI: title for search/replace is wrong.

Solution: Remove remark about doubling backslash. (closes #3230)

Files: src/gui_win32.c

Patch 8.1.0198

Problem: There is no hint that syntax is disabled for '**redrawtime**'.

Solution: Add a message.

Files: src/syntax.c

Patch 8.1.0199

Problem: spellbadword() does not check for caps error. (Dominique Pelle)

Solution: Adjust capcol when advancing.

Files: src/userfunc.c

Patch 8.1.0200

Problem: spellbadword() not tested.

Solution: Add a test. (Dominique Pelle, closes #3235)

Files: src/testdir/test_spell.vim

Patch 8.1.0201

Problem: Newer Python uses "importlib" instead of "imp".

Solution: Use "importlib" for newer Python versions. (Ozaki Kiichi, closes #3163)
Files: src/if_py_both.h, src/testdir/test87.in

Patch 8.1.0202

Problem: :version always shows +packages. (Takuya Fujiwara)
Solution: Add #ifdef (closes #3198) Also for has().
Files: src/version.c, src/evalfunc.c

Patch 8.1.0203

Problem: Building with Perl 5.28 fails on Windows.
Solution: Define Perl_mg_get. (closes #3196)
Files: src/if_perl.xs

Patch 8.1.0204

Problem: inputlist() is not tested.
Solution: Add a test. (Dominique Pelle, closes #3240)
Files: src/testdir/test_functions.vim

Patch 8.1.0205

Problem: Invalid memory access with invalid modeline.
Solution: Pass pointer limit. Add a test. (closes #3241)
Files: src/Make_all.mak, src/testdir/test_alot.vim, src/testdir/test_modeline.vim, src/option.c

Patch 8.1.0206 (after 8.1.0205)

Problem: Duplicate test function name.
Solution: Rename both functions.
Files: src/testdir/test_modeline.vim, src/testdir/test_glob2regpat.vim

Patch 8.1.0207

Problem: Need many menu translation files to cover regions.
Solution: When there is no region match, try without. (Christian Brabandt)
Files: runtime/menu.vim

Patch 8.1.0208 (after 8.1.0205)

Problem: File left behind after running individual test.
Solution: Delete the file.
Files: src/testdir/test_modeline.vim

Patch 8.1.0209

Problem: Stderr output from Ruby messes up display.
Solution: Turn the stderr output into a Vim message. (Masataka Pocke Kuwabara, closes #3238)
Files: src/if_ruby.c

Patch 8.1.0210

Problem: Still a few K&R function declarations.
Solution: Use ANSI function declarations (Hirohito Higashi)
Files: src/eval.c, src/evalfunc.c, src/list.c

Patch 8.1.0211

Problem: Expanding a file name "~" results in \$HOME. (Aidan Shafran)
Solution: Change "~" to "./~" before expanding. (closes #3072)

Files: src/testdir/test_expand.vim, src/ex_docmd.c, src/eval.c,
src/proto/eval.pro, src/evalfunc.c, src/if_cscope.c, src/misc1.c

Patch 8.1.0212

Problem: Preferred cursor column not set in interfaces.

Solution: Set w_set_curswant when setting the cursor. (David Hotham,
closes #3060)

Files: src/if_lua.c, src/if_mzsch.c, src/if_perl.xs, src/if_py_both.h,
src/if_ruby.c, src/if_tcl.c, src/testdir/test_lua.vim,
src/testdir/test_perl.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_ruby.vim,
src/testdir/test_tcl.vim

Patch 8.1.0213

Problem: **CTRL-W** CR does not work properly in a quickfix window.

Solution: Split the window if needed. (Jason Franklin)

Files: src/normal.c, src/proto/quickfix.pro, src/quickfix.c,
src/testdir/test_quickfix.vim, src/window.c

Patch 8.1.0214

Problem: +autochdir feature not reported by has() or :version.

Solution: Add the feature in the list.

Files: src/evalfunc.c, src/version.c

Patch 8.1.0215

Problem: No error if configure --with-x cannot configure X.

Solution: Check that when --with-x is used X can be configured.

Files: src/configure.ac, src/auto/configure

Patch 8.1.0216

Problem: Part of file not indented properly.

Solution: Adjust the indent. (Ken Takata)

Files: src/getchar.c

Patch 8.1.0217

Problem: Compiler warning for variable set but not used.

Solution: Move tilde_file inside #ifdef. (Hirohito Higashi, closes #3255)

Files: src/ex_docmd.c

Patch 8.1.0218

Problem: Cannot add matches to another window. (Qiming Zhao)

Solution: Add the "window" argument to matchadd() and matchaddpos().
(closes #3260)

Files: src/evalfunc.c, runtime/doc/eval.txt, src/testdir/test_match.vim

Patch 8.1.0219

Problem: Expanding ## fails to escape backtick.

Solution: Escape a backtick in a file name. (closes #3257)

Files: src/ex_docmd.c, src/testdir/test_edit.vim

Patch 8.1.0220

Problem: Ruby converts v:true and v:false to a number.

Solution: Use Qtrue and Qfalse instead. (Masataka Pocke Kuwabara,
closes #3259)

Files: src/if_ruby.c, src/testdir/test_ruby.vim

Patch 8.1.0221

Problem: Not enough testing for the Ruby interface.

Solution: Add more tests. (Dominique Pelle, closes #3252)

Files: runtime/doc/if_ruby.txt, src/testdir/test_ruby.vim

Patch 8.1.0222

Problem: Errors are reported for "make install".

Solution: Skip missing language files. (Christian Brabandt, closes #3254)

Files: src/installman.sh

Patch 8.1.0223

Problem: Completing shell command finds sub-directories in \$PATH.

Solution: Remove EW_DIR when completing an item in \$PATH. (Jason Franklin)

Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.1.0224

Problem: Hang in bracketed paste mode when t_PE not encountered.

Solution: Break out of the loop when got_int is set. (suggested by Christian Brabandt, closes #3146)

Files: src/edit.c

Patch 8.1.0225

Problem: Mode() does not indicate using **CTRL-O** from Insert mode.

Solution: Add "niI", "niR" and "niV" to mode() result. (closes #3000)

Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_functions.vim

Patch 8.1.0226

Problem: Too many #ifdefs.

Solution: Graduate the +vreplace feature, it's not much code and quite a few #ifdefs.

Files: runtime/doc/change.txt, runtime/doc/various.txt, src/edit.c,
src/evalfunc.c, src/gui.c, src/misc1.c, src/misc2.c, src/normal.c,
src/ops.c, src/screen.c, src/version.c, src/feature.h,
src/globals.h, src/macros.h, src/vim.h

Patch 8.1.0227

Problem: Spaces instead of tabs in makefile.

Solution: Use tabs and fix sorting. (Ken Takata)

Files: src/po/Make_all.mak

Patch 8.1.0228

Problem: Dropping files is ignored while Vim is busy.

Solution: Postpone the effect of dropping files until it's safe.

Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/gui.c, src/gui.h,
src/screen.c, src/main.c, src/gui_mac.c

Patch 8.1.0229

Problem: Crash when dumping profiling data.

Solution: Reset flag indicating that initialization was done.

Files: src/userfunc.c

Patch 8.1.0230

Problem: Directly checking **'buftype'** value.
Solution: Add the bt_normal() function. (Yegappan Lakshmanan)
Files: src/buffer.c, src/ex_docmd.c, src/fileio.c, src/proto/buffer.pro, src/quickfix.c

Patch 8.1.0231

Problem: :help -? goes to help for -+.
Solution: Add -? to list of special cases. (Hirohito Higashi)
Files: src/ex_cmds.c, src/testdir/test_help_tagjump.vim

Patch 8.1.0232

Problem: Ruby error does not include backtrace.
Solution: Add an error backtrace. (Masataka Pocke Kuwabara, closes #3267)
Files: src/if_ruby.c

Patch 8.1.0233

Problem: "safe" argument of call_vim_function() is always FALSE.
Solution: Remove the argument.
Files: src/eval.c, src/proto/eval.pro, src/edit.c, src/mbyte.c, src/normal.c, src/ex_getln.c

Patch 8.1.0234

Problem: Incorrect reference counting in Perl interface.
Solution: Call SvREFCNT_inc more often, add a test. (Damien)
Files: src/if_perl.xs, src/testdir/test_perl.vim

Patch 8.1.0235 (after 8.1.0231)

Problem: More help tags that jump to the wrong location.
Solution: Add more exceptions and a table for "expr-" tags. (Hirohito Higashi)
Files: src/ex_cmds.c, src/testdir/test_help_tagjump.vim

Patch 8.1.0236 (after 8.1.0232)

Problem: Ruby build fails when ruby_intern is missing.
Solution: Do not use ruby_intern2. (Ken Takata)
Files: src/if_ruby.c

Patch 8.1.0237

Problem: Ruby on Cygwin doesn't always work.
Solution: Use LIBRUBY_SO if LIBRUBY_ALIASES isn't set. (Ken Takata)
Files: src/configure.ac, src/auto/configure

Patch 8.1.0238

Problem: **'buftype'** is cleared when using ":term ++hidden cat". (Marcin Szamotulski)
Solution: Set the "options initialized" flag earlier. (closes #3278)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0239 (after 8.1.0236)

Problem: Now Ruby build fails on other systems.
Solution: Always define rb_intern. (Ken Takata, closes #3275)
Files: src/if_ruby.c

Patch 8.1.0240

Problem: g:actual_curbuf set in wrong scope. (Daniel Hahler)
Solution: Prepend the "g:" name space. (closes #3279)
Files: src/buffer.c

Patch 8.1.0241

Problem: Effect of ":tabmove N" is not clear.
Solution: Add a test that shows the behavior. (Christian Brabandt, closes #3288)
Files: src/testdir/test_tabpage.vim

Patch 8.1.0242

Problem: Insert mode completion may use an invalid buffer pointer. (Akib Nizam)
Solution: Check for ins_buf to be valid. (closes #3290)
Files: src/edit.c

Patch 8.1.0243

Problem: Using :term ++close ++hidden closes a window. (Marcin Szamotulski)
Solution: Don't close the window if only using it temporarily for unloading the terminal buffer. (closes #3287)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0244

Problem: No redraw when using a STOP signal on Vim and then a CONT signal.
Solution: Catch the CONT signal and force a redraw. (closes #3285)
Files: src/os_unix.c, src/term.c, src/proto/term.pro

Patch 8.1.0245

Problem: Calling setline() in TextChangedI autocmd breaks undo. (Jason Felice)
Solution: Don't save lines for undo when already saved. (closes #3291)
Files: src/edit.c, src/testdir/test_autocmd.vim

Patch 8.1.0246 (after 8.1.0245)

Problem: Build failure without the +eval feature.
Solution: Add #ifdef
Files: src/edit.c

Patch 8.1.0247

Problem: Python: error message for failing import is incorrect.
Solution: Adjust how modules are loaded. (Ozaki Kiichi, closes #3162)
Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok

Patch 8.1.0248

Problem: duplicated quickfix code.
Solution: Move the code to a function.
Files: src/quickfix.c

Patch 8.1.0249

Problem: GTK: when screen DPI changes Vim does not handle it.
Solution: Handle the gtk-xft-dpi signal. (Roel van de Kraats, closes #2357)
Files: src/gui_gtk_x11.c

Patch 8.1.0250

Problem: MS-Windows using VTP: windows size change incorrect.
Solution: Call SetConsoleScreenBufferSize() first. (Nobuhiro Takasaki, closes #3164)
Files: src/os_win32.c

Patch 8.1.0251

Problem: Using a full path is supported for **'directory'** but not for **'backupdir'**. (Mikolaj Machowski)
Solution: Support **'backupdir'** as well. (Christian Brabandt, closes #179)
Files: runtime/doc/options.txt, src/fileio.c, src/memline.c, src/proto/memline.pro, src/testdir/test_alot.vim, src/testdir/test_backup.vim, src/Make_all.mak

Patch 8.1.0252

Problem: Quickfix functions are too long.
Solution: Refactor. (Yegappan Lakshmanan, closes #2950)
Files: src/quickfix.c

Patch 8.1.0253

Problem: Saving and restoring window title does not always work.
Solution: Use the stack push and pop commands. (Kouichi Iwamoto, closes #3059)
Files: runtime/doc/term.txt, src/main.c, src/option.c, src/os_unix.c, src/proto/term.pro, src/term.c, src/term.h, src/vim.h, src/buffer.c, src/ex_docmd.c, src/option.c, src/os_amiga.c, src/os_mswin.c, src/os_win32.c

Patch 8.1.0254 (after 8.1.0253)

Problem: Cannot build on MS-Windows; Unused macro HAVE_HANDLE_DROP.
Solution: Adjust #ifdef. Delete the macro.
Files: src/main.c, src/vim.h

Patch 8.1.0255 (after 8.1.0251)

Problem: Backup test fails when using shadow directory.
Solution: Remove check for "src".
Files: src/testdir/test_backup.vim

Patch 8.1.0256 (after 8.1.0245)

Problem: Using setline() in TextChangedI splits undo.
Solution: Use another solution for undo not working properly.
Files: src/edit.c, src/testdir/test_autocmd.vim

Patch 8.1.0257

Problem: No test for pathshorten().
Solution: Add a test. (Dominique Pelle, closes #3295)
Files: src/testdir/test_functions.vim

Patch 8.1.0258

Problem: Not enough testing for the CompleteDone event.
Solution: Add a test. (closes #3297)
Files: src/testdir/test_ins_complete.vim

Patch 8.1.0259

Problem: No test for fixed quickfix issue.
Solution: Add a test. Clean up the code a bit. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0260

Problem: No LGTM logo in README file.
Solution: Add one. (Bas van Schaik, closes #3305)
Files: README.md

Patch 8.1.0261

Problem: Coverity complains about a negative array index.
Solution: When qf_id2nr() cannot find the list then don't set qf_curlist.
Files: src/quickfix.c

Patch 8.1.0262

Problem: Not enough testing for getftype().
Solution: Add a test. (Dominique Pelle, closes #3300)
Files: src/evalfunc.c, src/testdir/test_stat.vim

Patch 8.1.0263

Problem: Channel log doesn't show part of channel.
Solution: Add "sock", "out", "err" or "in". (Ozaki Kiichi, closes #3303)
Files: src/channel.c

Patch 8.1.0264

Problem: Backup tests fail when CWD is in /tmp.
Solution: Make '**backupskip**' empty. (Christian Brabandt, closes #3301)
Files: src/testdir/test_backup.vim

Patch 8.1.0265

Problem: The getcmdline() function is way too big.
Solution: Factor out the incremental search highlighting.
Files: src/ex_getln.c

Patch 8.1.0266

Problem: Parsing Ex address range is not a separate function.
Solution: Refactor do_one_cmd() to separate address parsing.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 8.1.0267

Problem: No good check if restoring quickfix list worked.
Solution: Let qf_restore_list() return OK/FAIL. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0268

Problem: File type checking has too many #ifdef.
Solution: Always define the S_IF macros. (Ken Takata, closes #3306)
Files: src/buffer.c, src/evalfunc.c, src/fileio.c, src/if_cscope.c,
src/os_unix.c, src/os_unix.h, src/vim.h

Patch 8.1.0269

Problem: Ruby Kernel.#p method always returns nil.
Solution: Copy p method implementation from Ruby code. (Masataka Pocke)

Kuwabara, closes #3315)
Files: src/if_ruby.c, src/testdir/test_ruby.vim

Patch 8.1.0270

Problem: Checking for a Tab in a line could be faster.
Solution: Use strchr() instead of strrchr(). (closes #3312)
Files: src/ex_cmds.c

Patch 8.1.0271

Problem: **'incsearch'** doesn't work for :s, :g or :v.
Solution: Also use **'incsearch'** for other commands that use a pattern.
Files: src/ex_getln.c, src/globals.h, src/screen.c,
src/testdir/test_search.vim

Patch 8.1.0272

Problem: Options test fails if temp var ends in slash. (Tom Briden)
Solution: Check for optional slash. (closes #3308)
Files: src/testdir/test_options.vim

Patch 8.1.0273

Problem: Invalid memory access when using **'incsearch'**.
Solution: Reset "patlen" when using previous search pattern.
Files: src/ex_getln.c

Patch 8.1.0274

Problem: **'incsearch'** triggers on ":source".
Solution: Check for the whole command name.
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0275

Problem: **'incsearch'** with :s doesn't start at cursor line.
Solution: Set cursor before parsing address. (closes #3318)
Also accept a match at the start of the first line.
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0276

Problem: No test for **'incsearch'** highlighting with :s.
Solution: Add a screendump test.
Files: src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_01.dump

Patch 8.1.0277

Problem: **'incsearch'** highlighting wrong in a few cases.
Solution: Fix using last search pattern. Restore highlighting when changing
command. (issue #3321)
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_02.dump,
src/testdir/dumps/Test_incsearch_substitute_03.dump

Patch 8.1.0278

Problem: **'incsearch'** highlighting does not accept reverse range.
Solution: Swap the range when needed. (issue #3321)
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_04.dump

Patch 8.1.0279

Problem: **'incsearch'** highlighting does not skip white space.
Solution: Skip white space after the command. (issue #3321)
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_05.dump

Patch 8.1.0280

Problem: **'incsearch'** highlighting does not work for ":g!/
Solution: Skip the exclamation mark. (Hirohito Higashi)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0281

Problem: Parsing command modifiers is not separated.
Solution: Move command modifier parsing to a separate function.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_cmds.h,
src/globals.h, src/feature.h

Patch 8.1.0282

Problem: **'incsearch'** does not work with command modifiers.
Solution: Skip command modifiers.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_getln.c,
src/testdir/test_search.vim

Patch 8.1.0283 (after 8.1.0282)

Problem: Missing test dump.
Solution: Add the dump file
Files: src/testdir/dumps/Test_incsearch_substitute_06.dump

Patch 8.1.0284

Problem: **'cursorline'** highlighting wrong with **'incsearch'**.
Solution: Move the cursor back if the match is outside the range.
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_07.dump
src/testdir/dumps/Test_incsearch_substitute_08.dump

Patch 8.1.0285

Problem: Compiler warning for conversion.
Solution: Add a type cast. (Mike Williams)
Files: src/ex_getln.c

Patch 8.1.0286

Problem: **'incsearch'** does not apply to :smagic and :snomagic.
Solution: Add support. (Hirohito Higashi)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0287

Problem: MAX is not defined everywhere.
Solution: Define MAX where needed.
Files: src/ex_getln.c

Patch 8.1.0288

Problem: Quickfix code uses cmdidx too often.
Solution: Add is_loclist_cmd(). (Yegappan Lakshmanan)

Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/quickfix.c

Patch 8.1.0289

Problem: Cursor moves to wrong column after quickfix jump.
Solution: Set the curswant flag. (Andy Massimino, closes #3331)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0290

Problem: "cit" on an empty HTML tag changes the whole tag.
Solution: Only adjust the area in Visual mode. (Andy Massimino, closes #3332)
Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.1.0291

Problem: **'incsearch'** highlighting not used for :sort.
Solution: Handle pattern in :sort command.
Files: src/ex_getln.c, src/testdir/test_search.vim, src/testdir/dumps/Test_incsearch_sort_01.dump

Patch 8.1.0292

Problem: MS-Windows: the text "self-installing" confuses some users.
Solution: Remove the text from the uninstall entry. (closes #3337)
Files: src/dosinst.c

Patch 8.1.0293

Problem: Checks for type of stack is cryptic.
Solution: Define IS_QF_STACK() and IS_LL_STACK(). (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0294

Problem: MS-Windows: sometimes uses short directory name.
Solution: Expand to long file name with correct caps. (Nobuhiro Takasaki, closes #3334)
Files: src/os_win32.c

Patch 8.1.0295

Problem: No **'incsearch'** highlighting for :vimgrep and similar commands.
Solution: Parse the :vimgrep command and similar ones to locate the search pattern. (Hirohito Higashi, closes #3344)
Files: src/ex_getln.c, src/testdir/test_search.vim, src/testdir/dumps/Test_incsearch_vimgrep_01.dump, src/testdir/dumps/Test_incsearch_vimgrep_02.dump, src/testdir/dumps/Test_incsearch_vimgrep_03.dump, src/testdir/dumps/Test_incsearch_vimgrep_04.dump, src/testdir/dumps/Test_incsearch_vimgrep_05.dump

Patch 8.1.0296

Problem: Command parsing for **'incsearch'** is a bit ugly.
Solution: Return when there is no pattern. Put common checks together.
Files: src/ex_getln.c

Patch 8.1.0297 (after 8.1.0294)

Problem: MS-Windows: tests fail, Vim crashes.
Solution: Fix long file name handling.

Files: src/os_win32.c

Patch 8.1.0298

Problem: Window resize test sometimes fails on Mac.

Solution: Add Test_popup_and_window_resize() to flaky tests.

Files: src/testdir/runtest.vim

Patch 8.1.0299 (after 8.1.0298)

Problem: misplaced comment

Solution: Remove comment

Files: src/testdir/runtest.vim

Patch 8.1.0300

Problem: The old window title might be freed twice. (Dominique Pelle)

Solution: Do not free "oldtitle" in a signal handler but set a flag to have it freed later.

Files: src/os_unix.c

Patch 8.1.0301

Problem: GTK: Input method popup displayed on wrong screen.

Solution: Add the screen position offset. (Ken Takata, closes #3268)

Files: src/gui_beval.c, src/gui_gtk_x11.c, src/mbyte.c,
src/proto/gui_gtk_x11.pro

Patch 8.1.0302

Problem: Crash when using :suspend and "fg".

Solution: Undo patch 8.1.0244.

Files: src/os_unix.c, src/term.c, src/proto/term.pro

Patch 8.1.0303

Problem: line2byte() is wrong for last line with **'noeol'** and **'nofixeol'**.

Solution: Fix off-by-one error. (Shane Harper, closes #3351)

Files: src/memline.c, src/testdir/test_functions.vim

Patch 8.1.0304

Problem: No redraw when using a STOP signal on Vim and then a CONT signal.

Solution: Catch the CONT signal and set the terminal to raw mode. This is like 8.1.0244 but without the screen redraw and a fix for multi-threading suggested by Dominique Pelle.

Files: src/os_unix.c, src/term.c, src/proto/term.pro

Patch 8.1.0305

Problem: Missing support for Lua 5.4 32 bits on Unix.

Solution: Define lua_newuserdatauv. (Kazunobu Kuriyama)

Files: src/if_lua.c

Patch 8.1.0306

Problem: Plural messages are not translated properly.

Solution: Add more usage of NGETTEXT(). (Sergey Alyoshin)

Files: src/vim.h, src/buffer.c, src/ex_cmds.c, src/ex_docmd.c,
src/fileio.c, src/misc1.c, src/ops.c

Patch 8.1.0307

Problem: There is no good way to get the window layout.

Solution: Add the winlayout() function. (Yegappan Lakshmanan)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/window.pro, src/window.c, src/testdir/test_window_id.vim

Patch 8.1.0308

Problem: A quick undo shows "1 seconds ago". (Tony Mechelynck)
Solution: Add singular/plural message.
Files: src/undo.c

Patch 8.1.0309

Problem: Profiling does not show a count for condition lines. (Daniel Hahler)
Solution: Count lines when not skipping. (Ozaki Kiichi, closes #2499)
Files: src/ex_docmd.c, src/testdir/test_profile.vim

Patch 8.1.0310

Problem: File info message not always suppressed with 'F' in '**shortmess**'. (Asheq Imran)
Solution: Save and restore msg_silent. (Christian Brabandt, closes #3221)
Files: src/buffer.c, src/memline.c, src/testdir/test_options.vim

Patch 8.1.0311

Problem: Filtering entries in a quickfix list is not easy.
Solution: Add the cfilter plugin. (Yegappan Lakshmanan)
Files: runtime/pack/dist/opt/cfilter/plugin/cfilter.vim, runtime/doc/quickfix.txt

Patch 8.1.0312

Problem: Wrong type for flags used in signal handlers.
Solution: Use sig_atomic_t. (Dominique Pelle, closes #3356)
Files: src/globals.h, src/os_unix.c, src/os_win32.h

Patch 8.1.0313

Problem: Information about a swap file is unavailable.
Solution: Add swapinfo(). (Enzo Ferber)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/memline.c, src/proto/memline.pro, src/testdir/test_swap.vim

Patch 8.1.0314 (after 8.1.0313)

Problem: Build failure without the +eval feature. (Brenton Horne)
Solution: Add #ifdef. Also add the "dirty" item.
Files: src/memline.c, runtime/doc/eval.txt, src/testdir/test_swap.vim

Patch 8.1.0315

Problem: Helpgrep with language doesn't work properly. (Takuya Fujiwara)
Solution: Check for the language earlier. (Hirohito Higashi)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0316

Problem: swapinfo() test fails on Travis.
Solution: Handle a long host name. (Ozaki Kiichi, closes #3361)
Also make the version check flexible. (James McCoy)
Files: src/testdir/test_swap.vim

Patch 8.1.0317

Problem: Cscope test fails when using shadow directory.
Solution: Resolve symlink in Vim. (James McCoy, closes #3364)
Files: src/testdir/test_cscope.vim

Patch 8.1.0318

Problem: The getftype() test may fail for char devices if the file disappeared in between the listing and the getftype() call.
Solution: Ignore empty result. (Ozaki Kiichi, closes #3360)
Files: src/testdir/test_stat.vim

Patch 8.1.0319

Problem: bzero() function prototype doesn't work for Android.
Solution: Add an #ifdef. (Elliott Hughes, closes #3365)
Files: src/osdef1.h.in

Patch 8.1.0320

Problem: Too much 'incsearch' highlight for pattern matching everything.
Solution: Add the skiplen to the command and remove the line range.
(Christian Brabandt) Check for empty pattern earlier.
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_09.dump

Patch 8.1.0321 (after 8.1.0320)

Problem: 'incsearch' regression: /\v highlights everything.
Solution: Put back the empty_pattern() check.
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_search_01.dump,
src/testdir/dumps/Test_incsearch_search_02.dump

Patch 8.1.0322

Problem: Test_copy_winopt() does not restore 'hidden'.
Solution: Restore the option, fix indent. (Ozaki Kiichi, closes #3367)
Files: src/testdir/test_options.vim

Patch 8.1.0323

Problem: Reverse order of VTP calls only needed the first time.
Solution: Add a flag to remember the state. (Nobuhiro Takasaki, closes #3366)
Files: src/os_win32.c

Patch 8.1.0324

Problem: Off-by-one error in cmdidx check. (Coverity)
Solution: Use ">=" instead of ">".
Files: src/ex_docmd.c

Patch 8.1.0325

Problem: Strings in swap file may not be NUL terminated. (Coverity)
Solution: Limit the length of the used string.
Files: src/memline.c

Patch 8.1.0326

Problem: Screen dump does not consider NUL and space equal.
Solution: Use temp variables instead of character from cell.
Files: src/terminal.c, src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.1.0327

Problem: The "g **CTRL-G**" command isn't tested much.
Solution: Add more tests. (Dominique Pelle, closes #3369)
Files: src/testdir/test_normal.vim

Patch 8.1.0328

Problem: inputlist() doesn't work with a timer. (Dominique Pelle)
Solution: Don't redraw when cmdline_row is zero. (Hirohito Higashi, closes #3239)
Files: src/misc1.c, src/screen.c

Patch 8.1.0329

Problem: Using inputlist() during startup results in garbage. (Dominique Pelle)
Solution: Make sure the xterm tracing is stopped when disabling the mouse.
Files: src/os_unix.c

Patch 8.1.0330

Problem: The qf_add_entries() function is too long.
Solution: Split in two parts. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0331

Problem: Insufficient test coverage for :mkview and :loadview.
Solution: Add tests. (Dominique Pelle, closes #3385)
Files: src/testdir/test_mksession.vim

Patch 8.1.0332

Problem: Get Gdk-Critical error on first balloon show.
Solution: Get screen geometry using the draw area widget. (Davit Samvelyan, closes #3386)
Files: src/gui_beval.c

Patch 8.1.0333

Problem: :mkview does not restore cursor properly after "\$". (Dominique Pelle)
Solution: Position the cursor with "normal! \$".
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0334

Problem: '**autowrite**' takes effect when buffer is not to be written.
Solution: Don't write buffers that are not supposed to be written. (Even Q Jones, closes #3391) Add tests for '**autowrite**'.
Files: src/ex_cmds2.c, src/testdir/test_writefile.vim

Patch 8.1.0335

Problem: mkview test fails on CI.
Solution: Attempt to force recomputing curswant after folding.
Files: src/testdir/test_mksession.vim

Patch 8.1.0336

Problem: mkview test still fails on CI.
Solution: Ignore curswant, don't see another solution.

Files: src/testdir/test_mksession.vim

Patch 8.1.0337

Problem: :file fails in quickfix command.

Solution: Allow :file without argument when curbuf_lock is set. (Jason Franklin)

Files: src/ex_docmd.c, src/testdir/test_quickfix.vim

Patch 8.1.0338

Problem: MS-Windows: VTP doesn't work properly with PowerShell.

Solution: Adjust the color index. (Nobuhiro Takasaki, closes #3347)

Files: src/os_win32.c

Patch 8.1.0339

Problem: Wrong highlight when '**incsearch**' set and cancelling :s.

Solution: Reset search line range. (Hirohito Higashi, Masamichi Abe)

Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_10.dump

Patch 8.1.0340

Problem: No test for :spellinfo.

Solution: Add a test. (Dominique Pelle, closes #3394)

Files: src/testdir/test_spell.vim

Patch 8.1.0341

Problem: :argadd in empty buffer changes the buffer name. (Pavol Juhas)

Solution: Don't re-use the current buffer when not going to edit the file.
(closes #3397) Do re-use the current buffer for :next.

Files: src/ex_cmds2.c, src/testdir/test_arglist.vim,
src/testdir/test_command_count.vim

Patch 8.1.0342

Problem: Crash when a callback deletes a window that is being used. (Ozaki Kiichi)

Solution: Do not unload a buffer that is being displayed while redrawing the
screen. Also avoid invoking callbacks while redrawing.
(closes #2107)

Files: src/buffer.c, src/misc2.c

Patch 8.1.0343

Problem: '**shellslash**' is not used for getcwd() with local directory.
(Daniel Hahler)

Solution: Call slash_adjust() later. (closes #3399)

Files: src/evalfunc.c

Patch 8.1.0344

Problem: '**hlsearch**' highlighting has a gap after /\$.

Solution: Remove suspicious code. (Ricky Zhou, closes #3400)

Files: src/screen.c, src/testdir/test_hlsearch.vim

Patch 8.1.0345

Problem: Cannot get the window id associated with the location list.

Solution: Add the "filewinid" argument to getloclist(). (Yegappan Lakshmanan, closes #3202)

Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.1.0346

Problem: Building with Aap is outdated and unused.
Solution: Remove the Aap build files.
Files: Filelist, src/main.aap, src/testdir/main.aap, src/config.aap.in,
runtime/macros/maze/main.aap

Patch 8.1.0347

Problem: Some tests fail on Solaris.
Solution: Skip writefile test. Fix path to libc.so. Improve test for Turkish
case change. (Libor Bukata, Bjorn Linse, closes #3403)
Files: src/testdir/test_functions.vim, src/testdir/test_normal.vim,
src/testdir/test_writefile.vim

Patch 8.1.0348

Problem: On Travis the slowest build is run last. (Dominique Pelle)
Solution: Reorder the build entries.
Files: .travis.yml

Patch 8.1.0349

Problem: Crash when wiping buffer in a callback.
Solution: Do not handle messages when only peeking for a character.
(closes #2107) Add "redraw_flag" to test_override().
Files: src/os_unix.c, src/os_win32.c, src/screen.c, src/evalfunc.c,
src/globals.h, runtime/doc/eval.txt

Patch 8.1.0350

Problem: Vim may block on ch_sendraw() when the job is sending data back to
Vim, which isn't read yet. (Nate Bosch)
Solution: Add the "noblock" option to job_start(). (closes #2548)
Files: src/channel.c, src/structs.h, src/testdir/test_channel.vim,
runtime/doc/channel.txt

Patch 8.1.0351

Problem: **'incsearch'** for `:/foo/s//<Esc>` changes last search pattern.
Solution: Save the last search pattern earlier.
Files: src/ex_docmd.c, src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0352

Problem: Browsing compressed tar files does not always work.
Solution: Use the "file" command to get the compression type.
Files: runtime/autoload/tar.vim

Patch 8.1.0353

Problem: An "after" directory of a package is appended to **'rtp'**, which
will be after the user's "after" directory. ()
Solution: Insert the package "after" directory before any other "after"
directory in **'rtp'**. (closes #3409)
Files: src/ex_cmds2.c, src/testdir/test_packadd.vim

Patch 8.1.0354 (after 8.1.0353)

Problem: Packadd test fails on MS-Windows.

Solution: Ignore difference between forward and backward slashes.
Files: src/testdir/test_packadd.vim

Patch 8.1.0355

Problem: Incorrect adjusting the popup menu for the preview window.
Solution: Compute position and height properly. (Ronan Pigott) Also show at least ten items. (closes #3414)
Files: src/popupmnu.c

Patch 8.1.0356

Problem: Using `:s` with `'incsearch'` prevents **CTRL-R CTRL-W**. (Boris Staletic)
Solution: When past the pattern put cursor back in the start position. (closes #3413)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.0357

Problem: Instructions for tests are outdated. (Jason Franklin)
Solution: Update the text.
Files: src/testdir/README.txt

Patch 8.1.0358

Problem: Crash when using `term_dumpwrite()` after the job finished.
Solution: Check for a finished job and give an error message.
Files: src/terminal.c

Patch 8.1.0359

Problem: No clue what test failed when using a screendump twice.
Solution: Add an extra argument to `VerifyScreenDump()`.
Files: src/testdir/screendump.vim

Patch 8.1.0360

Problem: Using an external diff program is slow and inflexible.
Solution: Include the `xdiff` library. (Christian Brabandt, closes #2732)
Use it by default.
Files: Filelist, runtime/doc/diff.txt, runtime/doc/options.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile, src/diff.c, src/structs.h, src/testdir/dumps/Test_diff_01.dump, src/testdir/dumps/Test_diff_02.dump, src/testdir/dumps/Test_diff_03.dump, src/testdir/dumps/Test_diff_04.dump, src/testdir/dumps/Test_diff_05.dump, src/testdir/dumps/Test_diff_06.dump, src/testdir/dumps/Test_diff_07.dump, src/testdir/dumps/Test_diff_08.dump, src/testdir/dumps/Test_diff_09.dump, src/testdir/dumps/Test_diff_10.dump, src/testdir/dumps/Test_diff_11.dump, src/testdir/dumps/Test_diff_12.dump, src/testdir/dumps/Test_diff_13.dump, src/testdir/dumps/Test_diff_14.dump, src/testdir/dumps/Test_diff_15.dump, src/testdir/dumps/Test_diff_16.dump, src/testdir/test_diffmode.vim, src/xdiff/COPYING, src/xdiff/xdiff.h, src/xdiff/xdiffi.c, src/xdiff/xdiffi.h,

src/xdiff/xemit.c, src/xdiff/xemit.h, src/xdiff/xhistogram.c,
src/xdiff/xinclude.h, src/xdiff/xmacros.h, src/xdiff/xpatience.c,
src/xdiff/xprepare.c, src/xdiff/xprepare.h, src/xdiff/xtypes.h,
src/xdiff/xutils.c, src/xdiff/xutils.h, src/xdiff/README.txt

Patch 8.1.0361

Problem: Remote user not used for completion. (Stucki)
Solution: Use \$USER too. (Dominique Pelle, closes #3407)
Files: src/misc1.c

Patch 8.1.0362

Problem: Cannot get the script line number when executing a function.
Solution: Store the line number besides the script ID. (Ozaki Kiichi,
closes #3362) Also display the line number with ":verbose set".
Files: runtime/doc/cmdline.txt, runtime/doc/eval.txt, src/Make_all.mak,
src/buffer.c, src/eval.c, src/evalfunc.c, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/getchar.c,
src/globals.h, src/main.c, src/menu.c, src/option.c,
src/proto/eval.pro, src/structs.h, src/syntax.c,
src/testdir/test_alot.vim, src/testdir/test_expand_func.vim,
src/testdir/test_maparg.vim, src/term.c src/userfunc.c

Patch 8.1.0363

Problem: Internal diff isn't used by default as advertised.
Solution: Add "internal" to the default value of '**diffopt**'.
Also add couple of files missing from the distribution.
Files: src/option.c, runtime/doc/options.txt, Filelist

Patch 8.1.0364

Problem: Compiler warning in xdiff code. (Yegappan Lakshmanan)
Solution: Initialize directly.
Files: src/xdiff/xemit.c, src/xdiff/README.txt

Patch 8.1.0365

Problem: Function profile doesn't specify where it was defined.
Solution: Show the script name and line number.
Files: src/userfunc.c, src/testdir/test_profile.vim

Patch 8.1.0366

Problem: Pieces of the xdiff code are not used.
Solution: Add "#if 0" to omit unused code.
Files: src/xdiff/xemit.c

Patch 8.1.0367

Problem: getchar(1) no longer processes pending messages. (Yasuhiro
Matsumoto)
Solution: Call parse_queued_messages().
Files: src/evalfunc.c

Patch 8.1.0368

Problem: GTK code has too many #ifdefs and building fails with GTK 2.10.
Solution: Always use gtk_widget_get_window() and define it for older GTK
versions. (Ken Takata, closes #3421)
Files: src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_f.c,

src/gui_gtk_x11.c, src/mbyte.c, src/vim.h

Patch 8.1.0369

Problem: Continuation lines cannot contain comments.
Solution: Support using "\ .
Files: src/ex_cmds2.c, src/testdir/test_eval_stuff.vim,
runtime/indent/vim.vim, runtime/doc/repeat.txt

Patch 8.1.0370

Problem: Not using internal diff if 'diffopt' is not changed.
Solution: Correct initialization of diff_flags. (Christian Brabandt)
Files: src/diff.c

Patch 8.1.0371

Problem: Argument types for select() may be wrong.
Solution: Use a configure macro. (Tobias Ulmer)
Files: src/config.h.in, src/configure.ac, src/auto/configure,
src/os_unix.c

Patch 8.1.0372

Problem: Screen updating slow when 'cursorline' is set.
Solution: Only redraw the old and new cursor line, not all lines.
Files: src/edit.c, src/move.c, src/screen.c, src/proto/screen.pro

Patch 8.1.0373 (after 8.1.0372)

Problem: Screen updating still slow when 'cursorline' is set.
Solution: Fix setting last_cursorline.
Files: src/move.c

Patch 8.1.0374

Problem: Moving the cursor is slow when 'relativenumber' is set.
Solution: Only redraw the number column, not all lines.
Files: src/screen.c, src/move.c

Patch 8.1.0375

Problem: Cannot use diff mode with Cygwin diff.exe. (Igor Forca)
Solution: Skip over unrecognized lines in the diff output.
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.1.0376

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize the variable.
Files: src/screen.c

Patch 8.1.0377

Problem: Xdiff doesn't use the Vim memory allocation functions.
Solution: Change the xdl_defines. Check for out-of-memory. Rename
"ignored" to "vim_ignored".
Files: src/xdiff/xdiff.h, src/xdiff/xpatience.c, src/xdiff/xdiffi.c,
src/channel.c, src/diff.c, src/evalfunc.c, src/ex_cmds.c,
src/fileio.c, src/main.c, src/mbyte.c, src/netbeans.c,
src/os_unix.c, src/os_win32.c, src/ui.c, src/window.c,
src/globals.h, src/term.c

Patch 8.1.0378

Problem: CI build failure.
Solution: Include vim.h as ../vim.h. Fix compiler warning.
Files: src/xdiff/xdiff.h, src/xdiff/xpatience.c

Patch 8.1.0379

Problem: Build dependencies are incomplete.
Solution: Update the build dependencies, mainly for xdiff. Adjust object directory for libvterm and xdiff.
Files: src/Makefile, src/configure.ac, src/auto/configure, src/libvterm/src/screen.c, src/libvterm/src/termscreen.c, src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.0380

Problem: "make proto" doesn't work well.
Solution: Define a few more types for cproto. Update proto files. Fix that workshop didn't build.
Files: src/vim.h, src/protodef.h, src/if_ruby.c, src/workshop.c, src/proto/digraph.pro, src/hardcopy.pro, src/proto/option.pro, src/proto/window.pro

Patch 8.1.0381

Problem: Variable declaration not at start of block.
Solution: Fix line ordering.
Files: src/xdiff/xpatience.c

Patch 8.1.0382

Problem: Some make programs can't handle dependency on "xdiff/./".
Solution: Strip it out.
Files: src/Makefile

Patch 8.1.0383

Problem: Missing source file rename.
Solution: Update the dependency.
Files: src/Make_mvc.mak

Patch 8.1.0384

Problem: Sign ordering depends on +netbeans feature.
Solution: Also order signs without +netbeans. (Christian Brabandt, closes #3224)
Files: src/structs.h, src/buffer.c

Patch 8.1.0385

Problem: Coveralls badge doesn't update.
Solution: Update the URL
Files: README.md

Patch 8.1.0386

Problem: Cannot test with non-default option value.
Solution: Add test_option_not_set().
Files: runtime/doc/eval.txt, src/option.c, src/proto/option.pro, src/evalfunc.c

Patch 8.1.0387

Problem: No test for 'ambwidth' detection.
Solution: Add a test.
Files: src/testdir/test_startup_utf8.vim

Patch 8.1.0388

Problem: Coverity complains about possible NULL pointer use.
Solution: Use get_tv_string() instead of get_tv_string_chk().
Files: src/evalfunc.c

Patch 8.1.0389

Problem: :behave command is not tested.
Solution: Add a test. (Dominique Pelle, closes #3429)
Files: src/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_behave.vim

Patch 8.1.0390

Problem: Scrollbars are not tested.
Solution: Add test_scrollbar() and a test.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_gui.vim

Patch 8.1.0391

Problem: Building in a shadow directory fails.
Solution: Don't link the xdiff directory but what's in it. (closes #3428)
Files: src/Makefile

Patch 8.1.0392

Problem: Error while typing :/foo/s// with 'incsearch' enabled.
Solution: Do not give search errors when highlighting matches.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_getln.c,
src/testdir/test_search.vim

Patch 8.1.0393

Problem: Not all white space difference options available.
Solution: Add "iblack", "iwhiteall" and "iwhiteeol" to 'diffopt'.
Files: src/diff.c, src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_17.dump,
src/testdir/dumps/Test_diff_18.dump,
src/testdir/dumps/Test_diff_19.dump,
src/testdir/dumps/Test_diff_20.dump

Patch 8.1.0394

Problem: Diffs are not always updated correctly.
Solution: When using internal diff update for any changes properly.
Files: src/structs.h, src/diff.c, src/proto/diff.pro, src/misc1.c,
src/main.c

Patch 8.1.0395

Problem: Compiler warning on 64-bit MS-Windows.
Solution: Add type cast. (Mike Williams)
Files: src/diff.c

Patch 8.1.0396

Problem: Another compiler warning on 64-bit MS-Windows.
Solution: Add type cast. (Mike Williams)

Files: src/xdiff/xutils.c

Patch 8.1.0397

Problem: No event triggered after updating diffs.

Solution: Add the DiffUpdated event.

Files: src/vim.h, src/diff.c, src/fileio.c,
src/testdir/test_diffmode.vim, runtime/doc/autocmd.txt

Patch 8.1.0398

Problem: No test for -o and -O command line arguments.

Solution: Add a test. (Dominique Pelle, closes #3438)

Files: src/testdir/test_startup.vim

Patch 8.1.0399

Problem: **'hlsearch'** highlight remains in other window after cancelling command.

Solution: Redraw all windows. Also remove unnecessary delays. (closes #3437)

Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_11.dump,
src/testdir/dumps/Test_incsearch_substitute_12.dump,
src/testdir/dumps/Test_incsearch_substitute_13.dump

Patch 8.1.0400

Problem: Using freed memory with :diffget.

Solution: Skip ex_diffupdate() while updating diffs. (closes #3442)

Files: src/diff.c

Patch 8.1.0401

Problem: Can't get swap name of another buffer.

Solution: Add swapname(). (Ozaki Kiichi, closes #3441)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_swap.vim

Patch 8.1.0402

Problem: The DiffUpdate event isn't triggered for :diffput.

Solution: Also trigger DiffUpdate for :diffget and :diffput.

Files: src/diff.c

Patch 8.1.0403

Problem: Header file missing from distribution.

Solution: Add src/protodef.h.

Files: Filelist

Patch 8.1.0404

Problem: Accessing invalid memory with long argument name.

Solution: Use item_count instead of checking for a terminating NULL.
(Dominique Pelle, closes #3444)

Files: src/testdir/test_arglist.vim, src/version.c

Patch 8.1.0405

Problem: Too many #ifdefs for GTK.

Solution: Define macros instead of using #ifdef. (Ken Takata, closes #3436)

Files: src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_f.c,
src/gui_gtk_x11.c, src/vim.h

Patch 8.1.0406

Problem: Several command line arguments are not tested.
Solution: Add tests for -A, -F, -H, -p and -V. (Dominique Pelle, closes #3446)
Files: src/testdir/test_startup.vim

Patch 8.1.0407

Problem: Quickfix code mixes using the stack and a list pointer.
Solution: Use a list pointer in more places. (Yegappan Lakshmanan, closes #3443)
Files: src/quickfix.c

Patch 8.1.0408

Problem: MSVC: cannot use the "x64" native compiler option.
Solution: Ignore case for %Platform%. Improve documentation. (Ken Takata)
Files: src/INSTALLpc.txt, src/msvc2015.bat

Patch 8.1.0409 (after 8.1.0406)

Problem: Startup test fails on MS-Windows.
Solution: Do the Arabic test in silent Ex mode. Loosen the check for -V2.
Files: src/testdir/test_startup.vim

Patch 8.1.0410

Problem: The ex_copen() function is too long.
Solution: Refactor to split off two functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0411

Problem: Renamed file missing from distribution.
Solution: Rename screen.c to termscreen.c (Zdenek Dohnal, closes #3449)
Files: Filelist

Patch 8.1.0412

Problem: Cannot build with GTK 2.4.
Solution: Add back a few #ifdefs. (Ken Takata, closes #3447)
Also support older GTK. (Tom Christensen)
Files: src/gui_gtk_x11.c

Patch 8.1.0413

Problem: Test output is duplicated or missing.
Solution: Adjust the MS-Windows and Unix test makefiles. (Ken Takata, closes #3452)
Files: src/testdir/Make_dos.mak, src/testdir/Makefile

Patch 8.1.0414

Problem: v:option_old and v:option_new are cleared when using :set in OptionSet autocmd. (Gary Johnson)
Solution: Don't trigger OptionSet recursively.
Files: src/option.c

Patch 8.1.0415

Problem: Not actually using 16 colors with vtp.
Solution: Always use 256 colors when vtp is used. (Nobuhiro Takasaki, closes #3432)

Files: src/option.c, src/term.c

Patch 8.1.0416

Problem: Sort doesn't report deleted lines.

Solution: Call msgmore(). (Christian Brabandt, closes #3454)

Files: src/ex_cmds.c, src/testdir/test_sort.vim

Patch 8.1.0417

Problem: Several command line arguments are not tested.

Solution: Add tests for -m, -M, -R and -Vfile. (Dominique Pelle, closes #3458)

Files: src/testdir/test_startup.vim

Patch 8.1.0418

Problem: MS-Windows: cannot separate Lua include and library directories.

Solution: Add LUA_LIBDIR and LUA_INCDIR. (Ken Takata, closes #3464)

Files: src/Make_cyg_ming.mak

Patch 8.1.0419

Problem: Cygwin: running cproto fails with -O2.

Solution: Strip -O2 for cproto. (Ken Takata, closes #3465)

Files: src/Makefile

Patch 8.1.0420

Problem: Generating vim.lib when using ActivePerl 5.20.3 or later.

Solution: Redefine XS_EXTERNAL(). (Ken Takata, closes #3462)

Files: src/if_perl.xs

Patch 8.1.0421

Problem: MS-Windows: Ruby path is wrong for Ruby 1.9 and later.

Solution: Let -I argument depend on Ruby version. (Ken Takata, closes #3461)

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.0422

Problem: Cannot create map file with MinGW.

Solution: Add support for \$MAP. (Ken Takata, closes #3460)

Files: src/Make_cyg_ming.mak

Patch 8.1.0423

Problem: MS-Windows: using dup-close for flushing a file.

Solution: Use _commit(). (Ken Takata, closes #3463)

Files: src/memfile.c, src/os_mac.h, src/os_win32.h

Patch 8.1.0424

Problem: Test output is very verbose, loading CI log is slow.

Solution: Redirect output to /dev/null. (Ken Takata, closes #3456)

Files: src/testdir/Makefile

Patch 8.1.0425

Problem: ml_get error and crash with appendbufline(). (Masashi Iizuka)

Solution: Set per-window buffer info. (Hirohito Higashi, closes #3455)

Files: src/buffer.c, src/testdir/test_bufline.vim

Patch 8.1.0426

Problem: Accessing invalid memory in SmcOpenConnection().
Solution: Reduce size of errorstring by one. (Dominique Pelle, closes #3469)
Files: src/os_unix.c, src/testdir/test_startup.vim

Patch 8.1.0427

Problem: MS-Windows GUI: using invalid encoded file name.
Solution: Drop the file name and return NULL. (Ken Takata, closes #3467)
Files: src/gui_w32.c

Patch 8.1.0428

Problem: The :suspend command is not tested.
Solution: Add a test. (Dominique Pelle, closes #3472)
Files: src/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_suspend.vim

Patch 8.1.0429 (after 8.1.0343)

Problem: No test for :lcd with 'shellslash'.
Solution: Add a test. (Daniel Hahler, closes #3475)
Files: src/testdir/test_getcwd.vim

Patch 8.1.0430

Problem: Xargadd file left behind after running test.
Solution: Delete the file. (Dominique Pelle)
Files: src/testdir/test_arglist.vim

Patch 8.1.0431

Problem: The qf_jump() function is too long.
Solution: Refactor to split it into several functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0432

Problem: Compiler warning for signed/unsigned.
Solution: Add type cast. (Mike Williams)
Files: src/xdiff/xemit.c

Patch 8.1.0433

Problem: Mapping can obtain text from inputsecret(). (Tommy Allen)
Solution: Disallow **CTRL-R** = and **CTRL-** e when using inputsecret().
Files: src/ex_getln.c

Patch 8.1.0434

Problem: copy_loclist() is too long.
Solution: Split in multiple functions. (Yegappan Lakshmanan)
Files: src/proto/quickfix.pro, src/quickfix.c, src/window.c

Patch 8.1.0435

Problem: Cursorline highlight not removed in some situation. (Vitaly Yashin)
Solution: Reset last_cursorline when resetting 'cursorline'. (Christian Brabandt, closes #3481)
Files: src/move.c, src/proto/move.pro, src/option.c

Patch 8.1.0436

Problem: Can get the text of inputsecret() with getcmdline(). (Tommy Allen)

Solution: Don't return the text.
Files: src/ex_getln.c

Patch 8.1.0437

Problem: May access freed memory when syntax HL times out. (Philipp Gesang)
Solution: Clear b_sst_first when clearing b_sst_array.
Files: src/syntax.c

Patch 8.1.0438

Problem: The ex_make() function is too long.
Solution: Split it into several functions. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0439

Problem: Recursive use of getcmdline() still not protected.
Solution: Instead of saving the command buffer when making a call which may cause recursiveness, save the buffer when actually being called recursively.
Files: src/ex_getln.c, src/proto/ex_getln.pro, src/getchar.c, src/main.c

Patch 8.1.0440

Problem: remove() with a range not sufficiently tested.
Solution: Add a test. (Dominique Pelle, closes #3497)
Files: src/testdir/test_listdict.vim

Patch 8.1.0441

Problem: Build failure without command line history.
Solution: Move cmdline_init() outside of #ifdef.
Files: src/ex_getln.c

Patch 8.1.0442

Problem: GUI: Cursor not drawn after ":redraw | sleep".
Solution: Flush the output. (closes #3496)
Files: src/ex_docmd.c

Patch 8.1.0443

Problem: Unnecessary static function prototypes.
Solution: Remove unnecessary prototypes.
Files: src/arabic.c, src/blowfish.c, src/buffer.c, src/charset.c, src/crypt_zip.c, src/digraph.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/gui.c, src/gui_at_fs.c, src/gui_athena.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/hangulin.c, src/hardcopy.c, src/if_cscope.c, src/if_mzsch.c, src/if_python3.c, src/if_xcmds.c, src/integration.c, src/json.c, src/main.c, src/mbyte.c, src/memline.c, src/message.c, src/misc1.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_unix.c, src/os_win32.c, src/pty.c, src/regexp.c, src/screen.c, src/search.c, src/sha256.c, src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c, src/term.c, src/ui.c, src/undo.c, src/version.c, src/window.c, src/workshop.c

Patch 8.1.0444

Problem: Unnecessary check for NULL pointer.
Solution: Remove check and call vim_free() directly.
Files: src/beval.c

Patch 8.1.0445

Problem: Setting '**term**' does not store location for termcap options.
Solution: Set the script context for termcap options that are changed when '**term**' is set.
Files: src/option.c, src/proto/option.pro, src/term.c, src/testdir/test_options.vim

Patch 8.1.0446

Problem: Options test fails in the GUI.
Solution: Don't try changing '**term**' in the GUI.
Files: src/testdir/test_options.vim

Patch 8.1.0447

Problem: GUI scrollbar test fails with Athena and Motif.
Solution: When not using on-the-fly scrolling call normal_cmd().
Files: src/evalfunc.c, src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 8.1.0448

Problem: Cursorline not removed when using '**cursorbind**'. (Justin Keyes)
Solution: Store the last cursor line per window. (closes #3488)
Files: src/testdir/test_diffmode.vim, src/testdir/dumps/Test_diff_with_cursorline_01.dump, src/testdir/dumps/Test_diff_with_cursorline_02.dump, src/testdir/dumps/Test_diff_with_cursorline_03.dump, src/structs.h, src/move.c

Patch 8.1.0449

Problem: When '**rnu**' is set folded lines are not displayed correctly. (Vitaly Yashin)
Solution: When only redrawing line numbers do draw folded lines. (closes #3484)
Files: src/screen.c, src/testdir/test_fold.vim, src/testdir/dumps/Test_folds_with_rnu_01.dump, src/testdir/dumps/Test_folds_with_rnu_02.dump

Patch 8.1.0450 (after patch 8.1.0449)

Problem: Build failure without the +fold feature.
Solution: Add #ifdef.
Files: src/screen.c

Patch 8.1.0451

Problem: Win32 console: keypad keys don't work.
Solution: Use numbers instead of characters to avoid the value becoming negative. (Mike Williams)
Files: src/os_win32.c

Patch 8.1.0452

Problem: MS-Windows: not finding intl.dll.
Solution: Also find intl.dll next to libintl.dll. (Ken Takata)

Files: src/os_win32.c, runtime/doc/mlang.txt

Patch 8.1.0453

Problem: MS-Windows: executable() is not reliable.

Solution: Use \$PATHEXT properly. (Yasuhiro Matsumoto, closes #3512)

Files: src/os_win32.c, src/testdir/test_functions.vim

Patch 8.1.0454

Problem: resolve() was not tested with a symlink cycle.

Solution: Add a test. (Dominique Pelle, closes #3513)

Files: src/testdir/test_functions.vim

Patch 8.1.0455

Problem: Checking for empty quickfix stack is not consistent.

Solution: Use qf_stack_empty(). (Yegappan Lakshmanan)

Files: src/quickfix.c

Patch 8.1.0456

Problem: Running test hangs when the input file is being edited.

Solution: Use a SwapExists autocommand to ignore editing the test script.

Files: src/testdir/Makefile, src/testdir/runtest.vim

Patch 8.1.0457 (after 8.1.0451)

Problem: Win32 console: key mappings don't work.

Solution: Use another solution for the keypad keys that doesn't break mappings. Some values will be negative. (Mike Williams)

Files: src/os_win32.c

Patch 8.1.0458

Problem: Ml_get error and crash when using "do".

Solution: Adjust cursor position also when diffupdate is not needed. (Hirohito Higashi)

Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.1.0459

Problem: Test_executable fails when there is a dog in the system.

Solution: Rename the dog. (Hirohito Higashi)

Files: src/testdir/test_functions.vim

Patch 8.1.0460

Problem: assert_fails() does not take a message argument

Solution: Add the argument.

Files: src/evalfunc.c, src/eval.c, src/testdir/test_assert.vim

Patch 8.1.0461

Problem: Quickfix code uses too many /* */ comments.

Solution: Change to // comments. (Yegappan Lakshmanan)

Files: src/quickfix.c

Patch 8.1.0462

Problem: When using ConPTY Vim can be a child process.

Solution: To find a Vim window use both EnumWindows() and EnumChildWindows(). (Nobuhiro Takasaki, closes #3521)

Files: src/os_mswin.c

Patch 8.1.0463

Problem: "simalt ~x" in .vimrc blocks swap file prompt.
Solution: Flush buffers before prompting. (Yasuhiro Matsumoto, closes #3518, closes #2192)
Files: src/memline.c

Patch 8.1.0464

Problem: MS-Windows: job_info() has cmd without backslashes. (Daniel Hahler)
Solution: Use rem_backslash(). (closes #3517, closes #3404) Add a test. (Yasuhiro Matsumoto)
Files: src/misc2.c, src/testdir/test_channel.vim

Patch 8.1.0465 (after 8.1.0452)

Problem: Client-server test fails.
Solution: Change logic in EnumWindows().
Files: src/os_mswin.c

Patch 8.1.0466 (after 8.1.0463)

Problem: Autocmd test fails.
Solution: Do call inchar() when flushing typeahead.
Files: src/vim.h, src/getchar.c, src/proto/getchar.pro, src/memline.c, src/message.c, src/misc1.c

Patch 8.1.0467 (after 8.1.0063)

Problem: Cannot build with Mac OS X 10.5.
Solution: Change #ifdef into #if. (Akshay Hegde, closes #3022)
Files: src/os_macosx.m

Patch 8.1.0468

Problem: MS-Windows: Filter command with pipe character fails. (Johannes Riecken)
Solution: Find the pipe character outside of quotes. (Yasuhiro Matsumoto, closes #1743, closes #3523)
Files: src/ex_cmds.c, src/testdir/test_filter_cmd.vim

Patch 8.1.0469

Problem: Too often indexing in qf_lists[].
Solution: Use a qf_list_T pointer. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0470

Problem: Pointer ownership around fname_expand() is unclear.
Solution: Allow b_ffname and b_sfname to point to the same allocated memory, only free one. Update comments.
Files: src/buffer.c, src/structs.h, src/fileio.c, src/ex_cmds.c

Patch 8.1.0471

Problem: Some tests are flaky or fail on some systems.
Solution: Increase waiting time for port number. Use "cmd /c" to execute "echo" on win32. (Ken Takata, closes #3534)
Files: src/testdir/shared.vim, src/testdir/test_channel.vim

Patch 8.1.0472

Problem: Dosinst command has a few flaws.
Solution: Register DisplayIcon, DisplayVersion and Publisher for the uninstaller. (closes #3485) Don't set '**diffexpr**' if internal diff is supported. Allow for using Vi compatible from the command line. Remove needless sleeps. Add comments in the generated _vimrc. (Ken Takata, closes #3525)
Files: src/dosinst.c

Patch 8.1.0473

Problem: User doesn't notice file does not exist when swap file does.
Solution: Add a **note** that the file cannot be found. Make the "still running" notice stand out.
Files: src/memline.c

Patch 8.1.0474

Problem: Directory where if_perl.c is written is inconsistent.
Solution: use auto/if_perl.c for MS-Windows. (Ken Takata, closes #3540)
Files: src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.0475

Problem: Memory not freed on exit when quit in autocmd.
Solution: Remember funccal stack when executing autocmd.
Files: src/structs.h, src/userfunc.c, src/proto/userfunc.pro, src/fileio.c, src/eval.c, src/ex_cmds2.c, src/main.c

Patch 8.1.0476

Problem: Memory leaks in test_escaped_glob.
Solution: Avoid failure when running the shell, use the sandbox.
Files: src/testdir/test_escaped_glob.vim

Patch 8.1.0477 (after 8.1.0475)

Problem: Tiny build fails.
Solution: Add a dummy declaration for funccal_entry_T.
Files: src/structs.h

Patch 8.1.0478

Problem: Cannot build with perl using MinGW.
Solution: Add -I. (Ken Takata, Cesar Romani)
Files: src/Make_cyg_ming.mak

Patch 8.1.0479

Problem: Failure when setting '**varsofttabstop**' to end in a comma. (Ralf Schandl)
Solution: Reject value with trailing comma. Add test for invalid values (closes #3544)
Files: src/testdir/test_vartabs.vim, src/option.c

Patch 8.1.0480

Problem: MinGW build file uses different -I flags than MVC.
Solution: Add -I to \$CFLAGS. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 8.1.0481

Problem: When "Terminal" highlight is reverted cursor doesn't show.
Solution: Get the colors of the "Terminal" group. (closes #3546)
Files: src/terminal.c

Patch 8.1.0482

Problem: MinGW "make clean" deletes all .exe files.
Solution: Only delete .exe files that it builds. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 8.1.0483

Problem: MinGW does not build tee.exe.
Solution: Add build instructions. (Yasuhiro Matsumoto, closes #3548)
Files: src/Make_cyg_ming.mak, src/tee/Makefile

Patch 8.1.0484

Problem: Some file types are not recognized.
Solution: Update the file type detection.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.0485

Problem: term_start() does not check if directory is accessible.
Solution: Add mch_access() call. (Jason Franklin)
Files: src/channel.c, src/testdir/test_terminal.vim

Patch 8.1.0486 (after 8.1.0485)

Problem: Can't build in MS-Windows.
Solution: Put mch_access() call inside #ifdef
Files: src/channel.c

Patch 8.1.0487

Problem: No menus specifically for the terminal window.
Solution: Add :tmenu. (Yee Cheng Chin, closes #3439) Add a menu test.
Files: runtime/delmenu.vim, runtime/doc/autocmd.txt, runtime/doc/gui.txt,
runtime/doc/index.txt, runtime/doc/terminal.txt,
runtime/doc/usr_42.txt, runtime/menu.vim, src/ex_cmdidxs.h,
src/ex_cmds.h, src/ex_docmd.c, src/menu.c, src/proto/menu.pro,
src/popupmnu.c, src/structs.h, src/testdir/test_menu.vim

Patch 8.1.0488

Problem: Using freed memory in quickfix code. (Dominique Pelle)
Solution: Add the quickfix_busy() flag to postpone deleting quickfix lists
until it is safe. (Yegappan Lakshmanan, closes #3538)
Files: src/quickfix.c, src/proto/quickfix.pro, src/misc2.c,
src/testdir/test_quickfix.vim

Patch 8.1.0489

Problem: Crash when autocmd clears vimpgrep location list.
Solution: Return from qf_jump_edit_buffer() early. (Yegappan Lakshmanan)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0490

Problem: MS-Windows: doesn't handle missing libwinpthread-1.dll.
Solution: Adjust Cygwin/MinGW build file. (Ken Takata, closes #2827)
Files: src/Make_cyg_ming.mak

Patch 8.1.0491

Problem: If a terminal dump has CR it is considered corrupt.
Solution: Ignore CR characters. (Nobuhiro Takasaki, closes #3558)
Files: src/terminal.c

Patch 8.1.0492

Problem: "Edit with existing Vim" list can get long.
Solution: Move the list to a submenu. (Ken Takata, closes #3561)
Files: src/GvimExt/gvimext.cpp

Patch 8.1.0493

Problem: argv() and argc() only work on the current argument list.
Solution: Add a window ID argument. (Yegappan Lakshmanan, closes #832)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_arglist.vim,
src/eval.c, src/proto/eval.pro

Patch 8.1.0494

Problem: Functions do not check for a window ID in other tabs.
Solution: Also find the window ID in other than the current tab.
Files: src/evalfunc.c

Patch 8.1.0495

Problem: :filter only supports some commands.
Solution: Add :filter support for more commands. (Marcin Szamotulski,
closes #2856)
Files: runtime/doc/various.txt, src/eval.c, src/mark.c, src/option.c,
src/syntax.c, src/testdir/test_filter_cmd.vim, src/userfunc.c

Patch 8.1.0496

Problem: No tests for indent files.
Solution: Add a mechanism for running indent file tests. Add a first test
for Vim indenting.
Files: runtime/indent/Makefile, runtime/indent/testdir/runtest.vim,
runtime/indent/testdir/cleantest.vim, runtime/indent/README.txt,
runtime/indent/testdir/README.txt, runtime/indent/testdir/vim.in,
runtime/indent/testdir/vim.ok, Filelist

Patch 8.1.0497

Problem: :%diffput changes order of lines. (Markus Braun)
Solution: Do adjust marks when using internal diff.
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.1.0498

Problem: /etc/gitconfig not recognized at a gitconfig file.
Solution: Add pattern to filetype detection. (closes #3568)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.0499

Problem: :2vimgrep causes an ml_get error
Solution: Pass tomatch pointer instead of value. (Yegappan Lakshmanan)
Files: src/ex_getln.c, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0500

Problem: Cleaning up in src/tee may not always work.
Solution: Use "rm" when appropriate. (Michael Soyka, closes #3571)
Files: src/tee/Makefile

Patch 8.1.0501

Problem: Cppcheck warns for using array index before bounds check.
Solution: Swap the conditions. (Dominique Pelle)
Files: src/memline.c

Patch 8.1.0502

Problem: Internal diff fails when diffing a context diff. (Hirohito Higashi)
Solution: Only use callback calls with one line. (closes #3581)
Files: src/diff.c, src/testdir/dumps/test_diff_of_diff_01.dump

Patch 8.1.0503

Problem: Missing change to diff test. (Hirohito Higashi)
Solution: Add the missing test function.
Files: src/testdir/test_diffmode.vim

Patch 8.1.0504

Problem: When **CTRL-C** is mapped it triggers InsertLeave.
Solution: Make **CTRL-C** behave the same way when typed or used in a mapping.
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.1.0505

Problem: Filter command test may fail if helplang is not set.
Solution: Set **'helplang'** for the test. (James McCoy, closes #3591)
Files: src/testdir/test_filter_cmd.vim

Patch 8.1.0506

Problem: Modeline test fails when run by root.
Solution: Set **'modeline'** for the test. (James McCoy, closes #3592)
Files: src/testdir/test_modeline.vim

Patch 8.1.0507

Problem: .raml files not properly detected.
Solution: Recognize .raml as raml instead of yaml. (closes #3594)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.0508

Problem: Suspend test fails when run by root.
Solution: Accept both '\$' and '#' for the prompt. (James McCoy, closes #3590)
Files: src/testdir/test_suspend.vim

Patch 8.1.0509

Problem: Checking cwd not accessible fails for root. (James McCoy)
Solution: Skip this part of the test for root. (closes #3595)
Files: src/testdir/test_terminal.vim

Patch 8.1.0510

Problem: Filter test fails when \$LANG is C.UTF-8.
Solution: Set **'helplang'** to "en" for any C language. (Christian Brabandt, closes #3577)
Files: src/option.c

Patch 8.1.0511

Problem: ml_get error when calling a function with a range.
Solution: Don't position the cursor after the last line.
Files: src/userfunc.c, src/testdir/test_functions.vim

Patch 8.1.0512

Problem: **'helplang'** default is inconsistent for C and C.UTF-8.
Solution: Don't accept a value unless it starts with two letters.
Files: src/ex_cmds2.c

Patch 8.1.0513

Problem: No error for set diffopt+=algorithm:.
Solution: Check for missing argument. (Hirohito Higashi, closes #3598)
Files: src/diff.c, src/testdir/gen_opt_test.vim

Patch 8.1.0514

Problem: **CTRL-W** ^ does not work when alternate buffer has no name.
Solution: Use another method to split and edit the alternate buffer. (Jason Franklin)
Files: src/testdir/test_normal.vim, src/testdir/test_window_cmd.vim, src/normal.c, src/window.c, runtime/doc/windows.txt

Patch 8.1.0515

Problem: Reloading a script gives errors for existing functions.
Solution: Allow redefining a function once when reloading a script.
Files: src/testdir/test_functions.vim, src/userfunc.c, src/structs.h, src/globals.h, src/buffer.c, src/ex_cmds2.c, src/main.c, src/option.c, runtime/doc/eval.txt

Patch 8.1.0516

Problem: :move command marks buffer modified when nothing changed.
Solution: Do not set **'modified'**. Add a test. (Jason Franklin)
Files: src/Make_all.mak, src/testdir/test_alot.vim, src/testdir/test_move.vim, src/ex_cmds.c

Patch 8.1.0517

Problem: Test_window_split_edit_alternate() fails on AppVeyor.
Solution: Disable the failing part for now.
Files: src/testdir/test_window_cmd.vim

Patch 8.1.0518

Problem: Test_window_split_edit_bufnr() fails on AppVeyor.
Solution: Disable the failing part for now.
Files: src/testdir/test_window_cmd.vim

Patch 8.1.0519

Problem: Cannot save and restore the tag stack.
Solution: Add gettagstack() and settagstack(). (Yegappan Lakshmanan, closes #3604)
Files: runtime/doc/eval.txt, runtime/doc/tagsrch.txt, runtime/doc/usr_41.txt, src/alloc.h, src/dict.c, src/evalfunc.c, src/list.c, src/misc2.c, src/proto/dict.pro, src/proto/list.pro, src/proto/misc2.pro, src/proto/tag.pro, src/tag.c,

src/testdir/test_tagjump.vim

Patch 8.1.0520

Problem: Screen diff test sometimes fails.
Solution: Add to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0521

Problem: Cannot build with +eval but without +quickfix.
Solution: Remove #ifdef for e_stringreq. (John Marriott)
Files: src/evalfunc.c

Patch 8.1.0522

Problem: :terminal does not show trailing empty lines.
Solution: Add empty lines. (Hirohito Higashi, closes #3605)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0523

Problem: Opening window from quickfix leaves empty buffer behind.
Solution: Add qf_jump_newwin(). (Yegappan Lakshmanan, closes #2574)
Files: src/proto/quickfix.pro, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.1.0524 (after 8.1.0522)

Problem: Terminal test fails on Windows.
Solution: Skip Test_terminal_does_not_truncate_last_newlines() for now.
Files: src/testdir/test_terminal.vim

Patch 8.1.0525 (after 8.1.0524)

Problem: Terminal test skips part on Windows.
Solution: Fix Test_terminal_does_not_truncate_last_newlines(). (Hirohito Higashi, closes #3606)
Files: src/Make_mvc.mak, src/testdir/test_terminal.vim

Patch 8.1.0526

Problem: Running out of signal stack in RealWaitForChar. (Vladimir Marek)
Solution: Make the fd_set variables static.
Files: src/os_unix.c

Patch 8.1.0527

Problem: Using 'shiftwidth' from wrong buffer for folding.
Solution: Use "buf" instead of "curbuf". (Christian Brabandt)
Files: src/fold.c

Patch 8.1.0528

Problem: Various typos in comments.
Solution: Fix the typos.
Files: src/fileio.c, src/gui.c, src/macros.h, src/screen.c, src/search.c,
src/spell.c, src/spellfile.c, src/vim.h, src/testdir/README.txt,
src/INSTALL, src/gui_athena.c, src/gui_gtk.c, src/gui_gtk_x11.c,
src/gui_motif.c, src/gui_xmew.c, src/if_tcl.c, src/os_amiga.c,
src/gui_w32.c, src/os_win32.c, src/gui_mac.c, src/os_vms_fix.com

Patch 8.1.0529

Problem: Flaky test sometimes fails in different ways.
Solution: When the second run gives a different error, try running the test again, up to five times.
Files: src/testdir/runtest.vim

Patch 8.1.0530

Problem: Channel and terminal tests that start a server can be flaky.
Solution: Add all channel and terminal tests that start a server to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0531

Problem: Flaky tests often fail with a common error message.
Solution: Add a pattern to match an error message indicating a flaky test.
Files: src/testdir/runtest.vim

Patch 8.1.0532

Problem: Cannot distinguish between quickfix and location list.
Solution: Add an explicit type variable. (Yegappan Lakshmanan)
Files: src/quickfix.c

Patch 8.1.0533

Problem: Screendump tests can be flaky.
Solution: Add VerifyScreenDump to the pattern of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0534

Problem: MS-Windows installer uses different \$HOME than Vim.
Solution: Use the Vim logic also in the MS-Windows installer. (Ken Takata, closes #3564)
Files: src/dosinst.c, src/misc1.c

Patch 8.1.0535

Problem: Increment/decrement might get interrupted by updating folds.
Solution: Disable fold updating for a moment. (Christian Brabandt, closes #3599)
Files: src/ops.c

Patch 8.1.0536

Problem: File time test fails when using NFS.
Solution: Use three file times instead of localtime(). (James McCoy, closes #3618)
Files: src/testdir/test_stat.vim

Patch 8.1.0537

Problem: ui_breakcheck() may be called recursively, which doesn't work.
Solution: When called recursively, just return. (James McCoy, closes #3617)
Files: src/ui.c

Patch 8.1.0538

Problem: Evaluating a modeline might invoke using a shell command. (Paul Huber)
Solution: Set the sandbox flag when setting options from a modeline.
Files: src/buffer.c

Patch 8.1.0539

Problem: Cannot build without the sandbox.
Solution: Set the secure option instead of using the sandbox. Also restrict the characters from **'spelllang'** that are used for LANG.vim. (suggested by Yasuhiro Matsumoto)
Files: runtime/doc/options.txt, src/buffer.c, src/option.c

Patch 8.1.0540

Problem: May evaluate insecure value when appending to option.
Solution: Set the secure flag when changing an option that was previously set insecurely. Also allow numbers for the characters from **'spelllang'** that are used for LANG.vim. (closes #3623)
Files: src/option.c

Patch 8.1.0541

Problem: Help message in dosinst.c is outdated.
Solution: Update the comment. (Ken Takata, closes #3626)
Files: src/dosinst.c

Patch 8.1.0542

Problem: shiftwidth() does not take **'vartabstop'** into account.
Solution: Use the cursor position or a position explicitly passed. Also make >> and << work better with **'vartabstop'**. (Christian Brabandt)
Files: runtime/doc/change.txt, runtime/doc/eval.txt, src/edit.c, src/evalfunc.c, src/normal.c, src/ops.c, src/option.c, src/proto/edit.pro, src/proto/option.pro, src/testdir/test_vartabs.vim

Patch 8.1.0543

Problem: Coverity warns for leaking memory and using wrong struct.
Solution: Free pointer when allocation fails. Change "boff" to "loff". (closes #3634)
Files: src/ex_getln.c, src/move.c

Patch 8.1.0544 (after 8.1.0540)

Problem: Setting **'filetype'** in a modeline causes an error (Hirohito Higashi).
Solution: Don't add the P_INSECURE flag when setting **'filetype'** from a modeline. Also for **'syntax'**.
Files: src/option.c, src/testdir/test_modeline.vim

Patch 8.1.0545

Problem: When executing indent tests user preferences interfere.
Solution: Add "--clean".
Files: runtime/indent/Makefile, runtime/indent/testdir/runtest.vim

Patch 8.1.0546

Problem: Modeline test with keymap fails.
Solution: Check that the keymap feature is available.
Files: src/testdir/test_modeline.vim

Patch 8.1.0547

Problem: Modeline test with keymap still fails.
Solution: Check that the keymap feature is available for the failure assert.
Files: src/testdir/test_modeline.vim

Patch 8.1.0548

Problem: Crash when job callback unloads a buffer. (James McCoy)
Solution: Don't round up the wait time to 10 msec in ui_inchar().
Files: src/ui.c

Patch 8.1.0549

Problem: Netbeans test depends on README.txt contents.
Solution: Use a generated file instead.
Files: src/testdir/test_netbeans.vim, src/testdir/test_netbeans.py

Patch 8.1.0550

Problem: Expression evaluation may repeat an error message. (Jason Franklin)
Solution: Increment did_emsg and check for the value when giving an error for the echo command.
Files: src/message.c, src/eval.c, src/testdir/test108.ok

Patch 8.1.0551 (after 8.1.0550)

Problem: Expression evaluation may repeat an error message. (Jason Franklin)
Solution: Check for the value of did_emsg when giving an error for the :execute command.
Files: src/eval.c

Patch 8.1.0552

Problem: Saved last search pattern may not be restored.
Solution: Call restore_last_search_pattern(). Add a check for balancing saving and restoring the last search pattern.
Files: src/ex_getln.c, src/search.c

Patch 8.1.0553

Problem: It is not easy to edit a script that was sourced.
Solution: Add a count to ":scriptnames", so that ":script 40" edits the script with script ID 40.
Files: src/ex_cmds.h, src/ex_cmds2.c, src/testdir/test_scriptnames.vim, src/Make_all.mak, src/testdir/Make_all.mak, runtime/doc/repeat.txt

Patch 8.1.0554

Problem: Popup menu overlaps with preview window.
Solution: Adjust the height computation. (Hirohito Higashi, closes #3414)
Files: src/popupmnu.c, src/testdir/test_popup.vim, src/testdir/dumps/Test_popup_and_previewwindow_01.dump

Patch 8.1.0555

Problem: Crash when last search pat is set but not last substitute pat.
Solution: Do not mix up last search pattern and last substitute pattern. (closes #3647)
Files: src/search.c, src/testdir/test_search.vim

Patch 8.1.0556

Problem: Saving/restoring search patterns share saved last_idx.
Solution: Use a separate saved last_idx for saving search patterns for functions and incremental search.
Files: src/search.c

Patch 8.1.0557

Problem: Termdebug: gdb may use X.Y for breakpoint number. (Ryou Ezo)
Solution: Handle X.Y breakpoint numbers. (Yasuhiro Matsumoto, close #3641)
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0558

Problem: Some MS-Windows instructions are outdated.
Solution: Update the uninstall instructions and the NSIS README. (Ken Takata, closes #3614) Also update remark about diff.exe.
Files: nsis/README.txt, uninstal.txt

Patch 8.1.0559

Problem: Command line completion not sufficiently tested.
Solution: Add more tests. (Dominique Pelle, closes #3622)
Files: src/testdir/test_arglist.vim, src/testdir/test_filetype.vim, src/testdir/test_history.vim, src/testdir/test_messages.vim, src/testdir/test_syntax.vim

Patch 8.1.0560

Problem: Cannot use address type "other" with user command.
Solution: Add "other" to the list. (Daniel Hahler, closes #3655) Also reject "%" for commands with "other". Add some more tests.
Files: src/ex_docmd.c, src/testdir/test_usercommands.vim

Patch 8.1.0561

Problem: MSVC error format has changed.
Solution: Make the space between the line number and colon optional.
Files: src/option.h

Patch 8.1.0562

Problem: Parsing of 'diffopt' is slightly wrong.
Solution: Fix the parsing and add a test. (Jason Franklin, Christian Brabandt)
Files: src/diff.c, src/testdir/test_diffmode.vim, src/testdir/dumps/Test_diff_09.dump, src/testdir/dumps/Test_diff_11.dump, src/testdir/screendump.vim

Patch 8.1.0563

Problem: Setting v:errors to a string give confusing error. (Christian Brabandt)
Solution: Change internal error into normal error message.
Files: src/eval.c

Patch 8.1.0564

Problem: Setting v:errors to wrong type still possible.
Solution: Return after giving an error message. (Christian Brabandt)
Files: src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.1.0565

Problem: Asan complains about reading before allocated block.
Solution: Workaround: Avoid offset from becoming negative.
Files: src/gui.c

Patch 8.1.0566

Problem: SGR not enabled for mintty because \$TERM is "xterm".
Solution: Detect mintty by the termresponse. (Ken Takata, closes #3667)
Files: src/term.c

Patch 8.1.0567 (after 8.1.0565)

Problem: Error for NUL byte in ScreenLines goes unnoticed.
Solution: Add an internal error message.
Files: src/gui.c

Patch 8.1.0568 (after 8.1.0567)

Problem: Error message for NUL byte in ScreenLines breaks Travis CI.
Solution: Use a normal message for now.
Files: src/gui.c

Patch 8.1.0569

Problem: Execute() always resets display column to zero. (Sha Liu)
Solution: Don't reset it to zero, restore the previous value. (closes #3669)
Files: src/evalfunc.c, src/testdir/test_execute_func.vim

Patch 8.1.0570

Problem: **'commentstring'** not used when adding fold marker. (Maxim Kim)
Solution: Only use empty **'comments'** middle when leader is empty. (Christian Brabandt, closes #3670)
Files: src/misc1.c, src/testdir/test_fold.vim

Patch 8.1.0571 (after 8.1.0569)

Problem: Non-silent execute() resets display column to zero.
Solution: Keep the display column as-is.
Files: src/evalfunc.c, src/testdir/test_execute_func.vim

Patch 8.1.0572

Problem: Stopping a job does not work properly on OpenBSD.
Solution: Do not use getpgid() to check the process group of the job process ID, always pass the negative process ID to kill(). (George Koehler, closes #3656)
Files: src/os_unix.c

Patch 8.1.0573

Problem: Cannot redefine user command without ! in same script
Solution: Allow redefining user command without ! in same script, like with functions.
Files: src/ex_docmd.c, src/testdir/test_usercommands.vim, runtime/doc/map.txt

Patch 8.1.0574

Problem: **'commentstring'** not used when adding fold marker in C.
Solution: Require white space before middle comment part. (mostly by Hirohito Higashi)
Files: src/misc1.c, src/testdir/test_fold.vim

Patch 8.1.0575

Problem: Termdebug: clearing multi-breakpoint does not work.
Solution: Delete all X.Y breakpoints. Keep more information about placed breakpoints. (Ozaki Kiichi, closes #3641)
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0576

Problem: Indent script tests pick up installed scripts.
Solution: Use current runtime indent scripts.
Files: runtime/indent/Makefile

Patch 8.1.0577

Problem: Tabpage right-click menu never shows "Close tab".
Solution: Always create the "Close tab" item but ignore the event if there is only one tab.
Files: src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c, src/gui.c

Patch 8.1.0578

Problem: Cannot disable arabic, rightleft and farsi in configure.
Solution: Add configure flags. (Diego Fernando Carrión, closes #1867)
Files: src/configure.ac, src/auto/configure, src/config.h.in, src/feature.h, src/Makefile

Patch 8.1.0579

Problem: Cannot attach properties to text.
Solution: First part of adding text properties.
Files: Filelist, runtime/doc/Makefile, runtime/doc/eval.txt, runtime/doc/textprop.txt, src/Make_all.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile, src/buffer.c, src/edit.c, src/evalfunc.c, src/feature.h, src/memline.c, src/misc1.c, src/misc2.c, src/proto.h, src/proto/memline.pro, src/proto/textprop.pro, src/screen.c, src/structs.h, src/testdir/Make_all.mak, src/testdir/test_textprop.vim, src/textprop.c, src/userfunc.c, src/version.c

Patch 8.1.0580

Problem: Invalid memory access when using text properties.
Solution: Disable text properties for now.
Files: src/feature.h

Patch 8.1.0581

Problem: Double free without the text properties feature.
Solution: Reset the dirty flag.
Files: src/memline.c

Patch 8.1.0582

Problem: Text properties are not enabled.
Solution: Fix sizeof argument and re-enable the text properties feature. Fix memory leak.
Files: src/feature.h, src/textprop.c

Patch 8.1.0583

Problem: Using illogical name for get_dict_number()/get_dict_string().

Solution: Rename to start with dict_.
Files: src/dict.c, src/proto/dict.pro, src/edit.c, src/eval.c,
src/evalfunc.c, src/quickfix.c, src/tag.c, src/terminal.c,
src/textprop.c

Patch 8.1.0584

Problem: With search **CTRL-L** does not pick up composing characters.
Solution: Check for composing characters. (Christian Brabandt, closes #3682)
[code change was accidentally included in 8.1.0579]
Files: src/testdir/test_search.vim

Patch 8.1.0585

Problem: Undo test may fail on MS-Windows.
Solution: Also handle lower case drive letters.
Files: src/testdir/test_undo.vim

Patch 8.1.0586

Problem: :digraph output is not easy to read.
Solution: Add highlighting for :digraphs. (Marcin Szamotulski, closes #3572)
Also add section headers for :digraphs!.
Files: src/ex_docmd.c, src/digraph.c, src/proto/digraph.pro,
src/ex_cmds.h, runtime/doc/digraph.txt

Patch 8.1.0587

Problem: GvimExt: realloc() failing is not handled properly.
Solution: Check for NULL return. (Jan-Jaap Korpershoek, closes #3689)
Files: src/GvimExt/gvimext.cpp

Patch 8.1.0588

Problem: Cannot define a sign with space in the text.
Solution: Allow for escaping characters. (Ben Jackson, closes #2967)
Files: src/ex_cmds.c, src/testdir/test_signs.vim

Patch 8.1.0589

Problem: Compilation error in gvimext.cpp.
Solution: Return a value. Also fix using uninitialized variable.
Files: src/GvimExt/gvimext.cpp, src/dosinst.c

Patch 8.1.0590

Problem: When a job ends the closed channels are not handled.
Solution: When a job is detected to have ended, check the channels again.
(closes #3530)
Files: src/channel.c, src/proto/channel.pro, src/misc2.c

Patch 8.1.0591

Problem: Channel sort test is flaky.
Solution: Do not check if the job is running, it may have be done very fast.
Files: src/testdir/test_channel.vim

Patch 8.1.0592

Problem: The libvterm tests are not run as part of Vim tests.
Solution: Add testing libvterm.
Files: src/Makefile, src/libvterm/Makefile

Patch 8.1.0593

Problem: Illegal memory access in libvterm test.
Solution: Fix off-by-one error.
Files: src/libvterm/src/vterm.c, src/libvterm/Makefile,
src/libvterm/t/run-test.pl

Patch 8.1.0594

Problem: Libvterm tests fail to run on Mac.
Solution: Only run libvterm tests on Linux.
Files: src/Makefile

Patch 8.1.0595

Problem: Libvterm tests are not run with coverage.
Solution: Adjust the Travis config. Show the actually run commands.
Files: .travis.yml, src/libvterm/Makefile

Patch 8.1.0596

Problem: Not all parts of printf() are tested.
Solution: Add a few more test cases. (Dominique Pelle, closes #3691)
Files: src/testdir/test_expr.vim

Patch 8.1.0597

Problem: Cannot run test_libvterm from the top directory.
Solution: Add test target in toplevel Makefile.
Files: Makefile

Patch 8.1.0598

Problem: Indent tests may use the wrong Vim binary.
Solution: Pass in the just built Vim binary.
Files: Makefile

Patch 8.1.0599

Problem: Without the +eval feature the indent tests don't work.
Solution: Skip the body of the tests.
Files: runtime/indent/testdir/cleantest.vim,
runtime/indent/testdir/runtest.vim

Patch 8.1.0600

Problem: Channel test is flaky.
Solution: Add test to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0601

Problem: A few compiler warnings.
Solution: Add type casts. (Mike Williams)
Files: src/GvimExt/gvimext.cpp, src/memline.c, src/textprop.c

Patch 8.1.0602

Problem: DirChanged is also triggered when the directory didn't change.
(Daniel Hahler)
Solution: Compare the current with the new directory. (closes #3697)
Files: src/ex_docmd.c, src/testdir/test_autocmd.vim, src/misc2.c,
src/testdir/test_autochdir.vim

Patch 8.1.0603

Problem: The :stop command is not tested.
Solution: Test :stop using a terminal window.
Files: src/testdir/test_terminal.vim, src/testdir/shared.vim

Patch 8.1.0604

Problem: Autocommand test fails on MS-Windows.
Solution: Use pathcmp() instead of strcmp() to check if a directory differs.
Files: src/ex_docmd.c, src/misc2.c

Patch 8.1.0605

Problem: Running make in the top directory echoes a comment.
Solution: Prefix with @. (closes #3698)
Files: Makefile

Patch 8.1.0606

Problem: **'cryptmethod'** defaults to a very old method.
Solution: Default to "blowfish2", it is now widely available.
Files: src/option.c, runtime/doc/options.txt

Patch 8.1.0607

Problem: Proto files are not in sync with the source code.
Solution: Update the proto files.
Files: src/os_mswin.c, src/proto/buffer.pro, src/proto/ex_cmds.pro,
src/proto/ex_getln.pro, src/proto/misc2.pro,
src/proto/userfunc.pro

Patch 8.1.0608

Problem: Coveralls is not updating.
Solution: Adjust path in Travis config.
Files: .travis.yml

Patch 8.1.0609

Problem: MS-Windows: unused variable, depending on the Ruby version.
Solution: Put ruby_sysinit and NtInitialize inside #ifdef and make them consistent. (Ken Takata)
Files: src/if_ruby.c

Patch 8.1.0610

Problem: MS-Windows ctags file list differs from Unix.
Solution: Define TAGS_FILES in the common makefile. (partly by Ken Takata)
Files: src/Make_all.mak, src/Makefile, src/Make_mvc.mak,
src/Make_cyg_ming.mak

Patch 8.1.0611

Problem: Crash when using terminal with long composing characters.
Solution: Make space for all characters. (Yasuhiro Matsumoto, closes #3619,
closes #3703)
Files: src/terminal.c

Patch 8.1.0612

Problem: Cannot use two global runtime dirs with configure.
Solution: Support a comma in --with-global-runtime. (James McCoy,
closes #3704)

Files: src/config.h.in, src/configure.ac, src/feature.h, src/os_unix.h,
src/auto/configure, src/Makefile

Patch 8.1.0613

Problem: When executing an insecure function the secure flag is stuck.
(Gabriel Barta)

Solution: Restore "secure" instead of decrementing it. (closes #3705)

Files: src/testdir/test_autocmd.vim, src/option.c, src/buffer.c

Patch 8.1.0614

Problem: Placing signs can be complicated.

Solution: Add functions for defining and placing signs. Introduce a group
name to avoid different plugins using the same signs. (Yegappan
Lakshmanan, closes #3652)

Files: runtime/doc/eval.txt, runtime/doc/sign.txt,
runtime/doc/usr_41.txt, src/alloc.h, src/buffer.c, src/evalfunc.c,
src/ex_cmds.c, src/globals.h, src/list.c, src/misc2.c,
src/netbeans.c, src/proto/buffer.pro, src/proto/ex_cmds.pro,
src/proto/list.pro, src/proto/misc2.pro, src/structs.h,
src/testdir/test_signs.vim, src/workshop.c

Patch 8.1.0615

Problem: Get_tv function names are not consistent.

Solution: Rename to tv_get.

Files: src/eval.c, src/proto/eval.pro, src/channel.c, src/dict.c,
src/evalfunc.c, src/list.c, src/message.c, src/tag.c,
src/terminal.c, src/textprop.c, src/window.c, src/ex_cmds.c,
src/os_unix.c, src/os_win32.c, src/json.c, src/regexp.c,
src/edit.c, src/misc2.c, src/popupmnu.c

Patch 8.1.0616

Problem: NSIS installer is outdated.

Solution: Use modern syntax, MUI2 and make it work better. Add translations.
(Guopeng Wen, Ken Takata, closes #3501)

Files: Filelist, nsis/gvim.nsi, nsis/icons/header.svg,
nsis/icons/welcome.svg, nsis/icons/header.bmp,
nsis/icons/un_header.bmp, nsis/icons/uninstall.bmp,
nsis/icons/welcome.bmp, nsis/lang/danish.nsi, nsis/lang/dutch.nsi,
nsis/lang/english.nsi, nsis/lang/german.nsi,
nsis/lang/italian.nsi, nsis/lang/japanese.nsi,
nsis/lang/simpchinese.nsi, nsis/lang/tradchinese.nsi,
src/dosinst.c

Patch 8.1.0617 (after 8.1.0616)

Problem: NSIS installer gets two files from the wrong directory.

Solution: Change \${VIMRT} to "..\".

Files: nsis/gvim.nsi

Patch 8.1.0618

Problem: term_getjob() does not return v:null as documented.

Solution: Do return v:null. (Damien) Add a test.

Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0619

Problem: :echomsg and :echoerr do not handle List and Dict like :echo does.
(Daniel Hahler)

Solution: Be more tolerant about the expression result type.

Files: src/eval.c, src/proto/eval.pro, src/evalfunc.c,
src/proto/evalfunc.pro, runtime/doc/eval.txt,
src/testdir/test_messages.vim, src/message.c

Patch 8.1.0620

Problem: Overruling CONF_ARGS from the environment no longer works. (Tony Mechelynck)

Solution: Do not define any CONF_ARGS by default.

Files: src/Makefile

Patch 8.1.0621

Problem: Terminal debugger does not handle unexpected debugger exit.

Solution: Check for debugger job ended and close unused buffers. (Damien)

Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.0622

Problem: Adding quickfix items marks items as valid errors. (Daniel Hahler)

Solution: Check when items are valid. (Yegappan Lakshmanan, closes #3683,
closes #3633)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0623

Problem: Iterating through window frames is repeated.

Solution: Define FOR_ALL_FRAMES. (Yegappan Lakshmanan)

Files: src/ex_docmd.c, src/globals.h, src/screen.c, src/window.c

Patch 8.1.0624 (after 8.1.0620)

Problem: Overruling CONF_ARGS from the environment still does not work.
(Tony Mechelynck)

Solution: Add back CONF_ARGS next to the new numbered ones.

Files: src/Makefile

Patch 8.1.0625

Problem: MS-Windows: terminal test fails in white console.

Solution: Accept both white and black background colors.

Files: src/testdir/test_terminal.vim

Patch 8.1.0626

Problem: MS-Windows: no resize to fit parent when using --windowid.

Solution: Pass FALSE for "mustset" in gui_set_shellsize(). (Agorgianitis
Loukas, closes #3616)

Files: src/gui.c

Patch 8.1.0627

Problem: Python cannot handle function name of script-local function.

Solution: Use <SNR> instead of the special byte code. (Ozaki Kiichi, closes
#3681)

Files: src/if_py_both.h, src/testdir/test_python2.vim,
src/testdir/test_python3.vim

Patch 8.1.0628

Problem: Compiler warning on MS-Windows.
Solution: Add type cast. (Mike Williams)
Files: src/if_py_both.h

Patch 8.1.0629

Problem: "gn" selects the wrong text with a multi-line match.
Solution: Get the end position from searchit() directly. (closes #3695)
Files: src/testdir/test_gn.vim, src/search.c, src/proto/search.pro,
src/edit.c, src/evalfunc.c, src/ex_docmd.c, src/ex_getln.c,
src/normal.c

Patch 8.1.0630

Problem: "wincmd p" does not work after using an autocmd window.
Solution: Store "prevwin" in aco_save_T. (Christian Brabandt, closes #3690)
Files: src/fileio.c, src/structs.h, src/testdir/test_window_cmd.vim

Patch 8.1.0631

Problem: Test for :stop fails on Arch.
Solution: Check five lines for the expected output. (closes #3714)
Files: src/testdir/test_terminal.vim

Patch 8.1.0632

Problem: Using sign group names is inefficient.
Solution: Store group names in a hash table and use a reference to them.
Also remove unnecessary use of ":exe" from the tests. (Yegappan
Lakshmanan, closes #3715)
Files: src/buffer.c, src/ex_cmds.c, src/structs.h,
src/testdir/test_signs.vim

Patch 8.1.0633

Problem: Crash when out of memory while opening a terminal window.
Solution: Handle out-of-memory more gracefully.
Files: src/terminal.c, src/libvterm/src/vterm.c,
src/libvterm/src/state.c, src/libvterm/src/termscreen.c

Patch 8.1.0634

Problem: Text properties cannot cross line boundaries.
Solution: Support multi-line text properties.
Files: src/textprop.c, src/testdir/test_textprop.vim,
runtime/doc/eval.txt

Patch 8.1.0635

Problem: Coverity complains about null pointer use.
Solution: Avoid using a null pointer.
Files: src/evalfunc.c

Patch 8.1.0636

Problem: line2byte() gives wrong values with text properties. (Bjorn Linse)
Solution: Compute byte offsets differently when text properties were added.
(closes #3718)
Files: src/structs.h, src/textprop.c, src/proto/textprop.pro,
src/memline.c, src/testdir/test_textprop.vim

Patch 8.1.0637

Problem: Nsis file no longer used.
Solution: Remove the file. (Ken Takata)
Files: nsis/vimrc.ini, Filelist

Patch 8.1.0638

Problem: Text property highlighting is off by one column. (Bjorn Linse)
Solution: Update text property highlighting earlier. Let it overrule syntax highlighting.
Files: src/structs.h, src/screen.c

Patch 8.1.0639

Problem: text properties test fails on MS-Windows
Solution: Set fileformat to "unix".
Files: src/testdir/test_textprop.vim

Patch 8.1.0640

Problem: Get E14 while typing command :tab with 'incsearch' set.
Solution: Do not give an error when looking for the command. (Hirohito Higashi)
Files: src/testdir/test_search.vim, src/ex_docmd.c

Patch 8.1.0641

Problem: No check for out-of-memory when converting regexp.
Solution: Bail out when lalloc() returns NULL. (John Marriott)
Files: src/regexp_nfa.c

Patch 8.1.0642

Problem: swapinfo() leaks memory. (Christian Brabandt)
Solution: Avoid allocating the strings twice.
Files: src/memline.c, src/dict.c, src/proto/dict.pro

Patch 8.1.0643

Problem: Computing byte offset wrong. (Bjorn Linse)
Solution: Use the right variable for array index.
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.1.0644

Problem: Finding next sign ID is inefficient.
Solution: Add next_sign_id. (Yegappan Lakshmanan, closes #3717)
Files: runtime/doc/eval.txt, src/buffer.c, src/evalfunc.c, src/ex_cmds.c, src/globals.h, src/main.c, src/proto/buffer.pro, src/structs.h, src/testdir/test_signs.vim

Patch 8.1.0645

Problem: Coverity warns for possible use of NULL pointer.
Solution: Check return value of vterm_obtain_screen().
Files: src/terminal.c

Patch 8.1.0646

Problem: Cannot build with Ruby 2.6.0.
Solution: Add rb_ary_detransient(). (Ozaki Kiichi, closes #3724)
Files: src/if_ruby.c

Patch 8.1.0647

Problem: MS-Windows: balloon_show() does not handle wide characters.
Solution: Use CreateWindowExW(). (Yasuhiro Matsumoto, closes #3708)
Files: src/gui_w32.c

Patch 8.1.0648

Problem: Custom operators can't act upon a forced motion. (Christian Wellenbrock)
Solution: Add the forced motion to the mode() result. (Christian Brabandt, closes #3490)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/globals.h, src/normal.c, src/testdir/test_mapping.vim

Patch 8.1.0649

Problem: setjmp() variables defined globally are used in one file.
Solution: Move the declarations to that file.
Files: src/globals.h, src/os_unix.c

Patch 8.1.0650

Problem: Command line argument -q [errorfile] is not tested.
Solution: Add a test. (Dominique Pelle, closes #3730)
Files: src/testdir/test_startup.vim

Patch 8.1.0651

Problem: :args \"foo works like :args without argument.
Solution: Fix check for empty argument. (closes #3728)
Files: src/ex_cmds2.c, src/testdir/test_arglist.vim

Patch 8.1.0652

Problem: Freeing memory for balloon eval too early.
Solution: Store the pointer in BalloonEval and free it later. (Yasuhiro Matsumoto, closes #3725)
Files: src/beval.h, src/gui_w32.c

Patch 8.1.0653 (after 8.1.0651)

Problem: Arglist test fails on MS-windows.
Solution: Only use a file name with a double quote on Unix.
Files: src/testdir/test_arglist.vim

Patch 8.1.0654

Problem: When deleting a line text property flags are not adjusted.
Solution: Adjust text property flags in preceding and following lines.
Files: src/memline.c, src/misc2.c, src/proto/misc2.pro, src/testdir/test_textprop.vim

Patch 8.1.0655

Problem: When appending a line text property flags are not added.
Solution: Add text properties to a newly added line.
Files: src/memline.c, src/testdir/test_textprop.vim, src/textprop.c

Patch 8.1.0656

Problem: Trying to reconnect to X server may cause problems.
Solution: Do not try reconnecting when exiting. (James McCoy)
Files: src/os_unix.c

Patch 8.1.0657 (after 8.1.0656)

Problem: Get error for using regexp recursively. (Dominique Pelle)

Solution: Do no check if connection is desired.

Files: src/os_unix.c

Patch 8.1.0658

Problem: Deleting signs and completion for :sign is insufficient.

Solution: Add deleting signs in a specified or any group from the current cursor location. Add group and priority to sign command completion. Add tests for different sign unplace commands. Update help text. Add tests for sign jump with group. Update help for sign jump. (Yegappan Lakshmanan, closes #3731)

Files: runtime/doc/sign.txt, src/buffer.c, src/evalfunc.c, src/ex_cmds.c, src/netbeans.c, src/proto/buffer.pro, src/proto/ex_cmds.pro, src/testdir/test_signs.vim

Patch 8.1.0659 (after 8.1.0658)

Problem: Build failure without the sign feature.

Solution: Put the sign struct declarations outside of the #ifdef.

Files: src/structs.h

Patch 8.1.0660

Problem: sign_unplace() may leak memory.

Solution: Free the group name before returning. Add a few more tests. (Yegappan Lakshmanan)

Files: src/evalfunc.c, src/testdir/test_signs.vim

Patch 8.1.0661

Problem: Clipboard regexp might be used recursively.

Solution: Check for recursive use and bail out.

Files: src/regexp.c, src/proto/regexp.pro, src/os_unix.c

Patch 8.1.0662

Problem: Needlessly searching for tilde in string.

Solution: Only check the first character. (James McCoy, closes #3734)

Files: src/misc1.c

Patch 8.1.0663

Problem: Text property display wrong when 'number' is set. (Dominique Pelle)

Solution: Compare with "vcol" instead of "col".

Files: src/screen.c

Patch 8.1.0664

Problem: Configure "fail-if-missing" does not apply to the enable-gui argument. (Rhialto)

Solution: Make configure fail if a GUI was specified and "fail-if-missing" is enabled and the GUI test fails.

Files: src/configure.ac, src/auto/configure

Patch 8.1.0665

Problem: Text property display wrong when 'spell' is set. (Dominique Pelle)

Solution: Remove unnecessary assignment to char_attr. Combine attributes if needed. Add a screenshot test.

Files: src/screen.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.0666 (after 8.1.0665)

Problem: Text property test fails.

Solution: Update screenshot.

Files: src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.0667 (after 8.1.0665)

Problem: Textprop test leaves file behind.

Solution: Delete the file. (Dominique Pelle, closes #3743)

Files: src/testdir/test_textprop.vim

Patch 8.1.0668

Problem: No test for overstrike mode in the command line.

Solution: Add a test. (Dominique Pelle, closes #3742)

Files: src/testdir/test_cmdline.vim

Patch 8.1.0669

Problem: The ex_sign() function is too long.

Solution: Refactor the function. Add a bit more testing. (Yegappan Lakshmanan, closes #3745)

Files: src/testdir/test_signs.vim, src/ex_cmds.c

Patch 8.1.0670

Problem: Macro for popup menu width is unused.

Solution: Remove it. (Hirohito Higashi)

Files: src/popupmnu.c

Patch 8.1.0671

Problem: Cursor in the wrong column after auto-formatting.

Solution: Check for deleting more spaces than adding. (closes #3748)

Files: src/ops.c, src/testdir/test_textformat.vim, src/mark.c,
src/proto/mark.pro, src/misc1.c

Patch 8.1.0672

Problem: The Lua interface doesn't know about v:null.

Solution: Add Lua support for v:null. (Uji, closes #3744)

Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.1.0673

Problem: Functionality for signs is spread out over several files.

Solution: Move most of the sign functionality into sign.c. (Yegappan Lakshmanan, closes #3751)

Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak,
src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak,
src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak,
src/Make_vms.mms, src/Makefile, src/README.txt, src/buffer.c,
src/evalfunc.c, src/ex_cmds.c, src/proto.h, src/proto/buffer.pro,
src/proto/ex_cmds.pro, src/proto/sign.pro, src/sign.c

Patch 8.1.0674

Problem: Leaking memory when updating a single line.

Solution: Do not call start_search_hl() twice.

Files: src/screen.c

Patch 8.1.0675

Problem: Text property column is screen columns is not practical.

Solution: Use byte values for the column.

Files: src/structs.h, src/textprop.c, src/proto/textprop.pro,
runtime/doc/eval.txt, runtime/doc/textprop.txt,
src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.0676

Problem: Textprop screendump test fails.

Solution: Add missing changes.

Files: src/screen.c

Patch 8.1.0677

Problem: Look-behind match may use the wrong line number. (Dominique Pelle)

Solution: Use the line number in regsave instead of the one in behind_pos,
we may be looking at the previous line. (closes #3749)

Files: src/regexp.c

Patch 8.1.0678

Problem: Text properties as not adjusted for inserted text.

Solution: Adjust text properties when inserting text.

Files: src/misc1.c, src/proto/misc1.pro, src/textprop.c,
src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.0679

Problem: Sign functions do not take buffer argument as documented.

Solution: Use get_buf_tv(). (Yegappan Lakshmanan, closes #3755)

Files: src/evalfunc.c, src/testdir/test_signs.vim

Patch 8.1.0680

Problem: Not easy to see what features are unavailable.

Solution: Highlight disabled features in the :version output. (Nazri Ramliy,
closes #3756)

Files: src/version.c

Patch 8.1.0681

Problem: Text properties as not adjusted for deleted text.

Solution: Adjust text properties when backspacing to delete text.

Files: src/edit.c, src/misc1.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.0682

Problem: Text properties are not adjusted when backspacing replaced text.

Solution: Keep text properties on text restored in replace mode.

Files: src/edit.c, src/textprop.c, src/globals.h,
src/testdir/test_textprop.vim

Patch 8.1.0683

Problem: Spell highlighting does not always end. (Gary Johnson)

Solution: Also reset char_attr when spell errors are highlighted.

Files: src/screen.c

Patch 8.1.0684

Problem: Warnings from 64-bit compiler.
Solution: Add type casts. (Mike Williams)
Files: src/memline.c, src/textprop.c

Patch 8.1.0685

Problem: get_buf_tv() is named inconsistently.
Solution: Rename it to tv_get_buf(). (Yegappan Lakshmanan, closes #3759)
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/terminal.c, src/textprop.c

Patch 8.1.0686

Problem: When 'y' is in '**coptions**' yanking for the clipboard changes redo.
Solution: Do not use the 'y' flag when "gui_yank" is TRUE. (Andy Massimino, closes #3760)
Files: src/normal.c

Patch 8.1.0687

Problem: Sentence text object in Visual mode is not tested.
Solution: Add a test. (Dominique Pelle, closes #3758)
Files: src/testdir/test_visual.vim

Patch 8.1.0688

Problem: Text properties are not restored by undo.
Solution: Also save text properties for undo.
Files: src/structs.h, src/undo.c, src/memline.c, src/proto/memline.pro

Patch 8.1.0689 (after 8.1.0688)

Problem: Undo with text properties not tested.
Solution: Add a test function.
Files: src/testdir/test_textprop.vim

Patch 8.1.0690

Problem: setline() and setbufline() do not clear text properties.
Solution: Clear text properties when setting the text.
Files: src/evalfunc.c, src/testdir/test_textprop.vim

Patch 8.1.0691

Problem: Text properties are not adjusted for :substitute.
Solution: Adjust text properties as well as possible.
Files: src/ex_cmds.c, src/textprop.c, src/proto/textprop.pro, src/testdir/test_textprop.vim

Patch 8.1.0692

Problem: If a buffer was deleted a channel can't write to it.
Solution: When the buffer exists but was unloaded, prepare it for writing. (closes #3764)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.1.0693 (after 8.1.0692)

Problem: Channel test fails sometimes.
Solution: Avoid race condition.

Files: src/testdir/test_channel.vim

Patch 8.1.0694

Problem: When using text props may free memory that is not allocated.
(Andy Massimino)

Solution: Allocate the line when adjusting text props. (closes #3766)

Files: src/textprop.c

Patch 8.1.0695

Problem: Internal error when using :popup.

Solution: When a menu only exists in Terminal mode give an error. (Naruhiko Nishino, closes #3765)

Files: runtime/doc/gui.txt, src/globals.h, src/menu.c, src/popupmnu.c,
src/testdir/test_popup.vim

Patch 8.1.0696

Problem: When test_edit fails '**insertmode**' may not be reset and the next test may get stuck. (James McCoy)

Solution: Always reset '**insertmode**' after executing a test. Avoid that an InsertCharPre autocommand or a '**complete**' function can change the state. (closes #3768)

Files: src/testdir/runtest.vim, src/edit.c

Patch 8.1.0697

Problem: ":sign place" requires the buffer argument.

Solution: Make the argument optional. Also update the help and clean up the sign test. (Yegappan Lakshmanan, closes #3767)

Files: runtime/doc/eval.txt, runtime/doc/sign.txt, src/sign.c,
src/testdir/test_signs.vim

Patch 8.1.0698

Problem: Clearing the window is used too often, causing the command line to be cleared when opening a tab. (Miroslav Koškár)

Solution: Use NOT_VALID instead of CLEAR. (suggested by Jason Franklin, closes #630) Also do this for a few other places where clearing the screen isn't really needed.

Files: src/window.c

Patch 8.1.0699

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)

Solution: Add a dummy init.

Files: src/edit.c

Patch 8.1.0700 (after 8.1.0698)

Problem: Using "gt" sometimes does not redraw a tab. (Jason Franklin)

Solution: Always set must_redraw in redraw_all_later().

Files: src/screen.c

Patch 8.1.0701

Problem: Sign message not translated and inconsistent spacing.

Solution: Add _() for translation. Add a space. (Ken Takata) Also use MSG_BUF_LEN instead of BUFSIZ.

Files: src/sign.c, src/testdir/test_signs.vim

Patch 8.1.0702

Problem: ":sign place" only uses the current buffer.
Solution: List signs for all buffers when there is no buffer argument.
Fix error message for invalid buffer name in sign_place().
(Yegappan Lakshmanan, closes #3774)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/sign.c,
src/testdir/test_signs.vim

Patch 8.1.0703

Problem: Compiler warnings with 64-bit compiler.
Solution: Change types, add type casts. (Mike Williams)
Files: src/textprop.c, src/undo.c

Patch 8.1.0704

Problem: Building with Ruby 2.6 gives compiler warnings.
Solution: Define a stub for rb_ary_detransient. (Ozaki Kiichi, closes #3779)
Files: src/if_ruby.c

Patch 8.1.0705

Problem: :colorscheme isn't tested enough
Solution: Improve test coverage of :colorscheme. (Dominique Pelle, closes
#3777) Remove unnecessary sleep.
Files: src/testdir/test_gui.vim

Patch 8.1.0706

Problem: Tabline is not always redrawn when something that is used in
'**tabline**' changes.
Solution: Add ":redrawtabline" so that a plugin can at least cause the
redraw when needed.
Files: runtime/doc/various.txt, runtime/doc/options.txt, src/ex_docmd.c,
src/ex_cmds.h, src/screen.c, src/proto/screen.pro,
src/ex_cmdidxs.h, src/testdir/test_tabline.vim

Patch 8.1.0707

Problem: Text property columns are not adjusted for changed indent.
Solution: Adjust text properties.
Files: src/misc1.c, src/testdir/test_textprop.vim

Patch 8.1.0708

Problem: Third argument for redrawWinline() is always FALSE.
Solution: Drop the argument. (neovim #9479)
Files: src/edit.c, src/move.c, src/screen.c, src/proto/screen.pro

Patch 8.1.0709

Problem: Windows are updated for every added/deleted sign.
Solution: Do not call update_debug_sign(). Only redraw when the line with
the sign is visible. (idea from neovim #9479)
Files: src/sign.c, src/screen.c, src/proto/screen.pro

Patch 8.1.0710

Problem: When using timers may wait for job exit quite long.
Solution: Return from ui_wait_for_chars_or_timer() when a job or channel
needs to be handled. (Ozaki Kiichi, closes #3783)
Files: src/ui.c, src/testdir/test_channel.vim

Patch 8.1.0711

Problem: Test files still use function!.
Solution: Remove the exclamation mark. Fix overwriting a function.
Files: src/testdir/test49.vim, src/testdir/test_autocmd.vim,
src/testdir/test_charsearch.vim,
src/testdir/test_charsearch_utf8.vim,
src/testdir/test_display.vim, src/testdir/test_edit.vim,
src/testdir/test_eval_func.vim, src/testdir/test_fnameescape.vim,
src/testdir/test_getcwd.vim, src/testdir/test_highlight.vim,
src/testdir/test_hlsearch.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_lambda.vim, src/testdir/test_listdict.vim,
src/testdir/test_listlbr.vim, src/testdir/test_listlbr_utf8.vim,
src/testdir/test_marks.vim, src/testdir/test_matchadd_conceal.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_messages.vim, src/testdir/test_number.vim,
src/testdir/test_options.vim, src/testdir/test_partial.vim,
src/testdir/test_smartindent.vim, src/testdir/test_substitute.vim,
src/testdir/test_system.vim, src/testdir/test_terminal.vim,
src/testdir/test_textobjects.vim, src/testdir/test_utf8.vim,
src/testdir/test_utf8_comparisons.vim,
src/testdir/test_vartabs.vim, src/testdir/test_vimscript.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_xxd.vim

Patch 8.1.0712

Problem: MS-Windows build instructions are a bit outdated.
Solution: Update the instructions. (Ken Takata)
Files: src/INSTALLpc.txt

Patch 8.1.0713

Problem: Images for NSIS take up too much space.
Solution: Put the images in a zip file.
Files: nsis/icons.zip, nsis/icons/disabled.bmp, nsis/icons/enabled.bmp,
nsis/icons/header.bmp, nsis/icons/header.svg,
nsis/icons/un_header.bmp, nsis/icons/uninstall.bmp,
nsis/icons/vim_16c.ico, nsis/icons/vim_uninst_16c.ico,
nsis/icons/welcome.bmp, nsis/icons/welcome.svg,
nsis/README.txt, Filelist, Makefile

Patch 8.1.0714

Problem: Unnecessary #if lines in GTK code.
Solution: Remove the #if. (Ken Takata, closes #3785)
Files: src/gui_beval.c, src/if_mzsch.c

Patch 8.1.0715

Problem: Superfluous call to redraw_win_later().
Solution: Remove the call.
Files: src/move.c

Patch 8.1.0716

Problem: Get warning message when **'completefunc'** returns nothing.
Solution: Allow for returning v:none to suppress the warning message.
(Yasuhiro Matsumoto, closes #3789)
Files: runtime/doc/insert.txt, src/edit.c,

src/testdir/test_ins_complete.vim

Patch 8.1.0717

Problem: There is no function for the ":sign jump" command.
Solution: Add the sign_jump() function. (Yegappan Lakshmanan, closes #3780)
Files: runtime/doc/eval.txt, runtime/doc/sign.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/sign.pro,
src/sign.c, src/testdir/test_signs.vim

Patch 8.1.0718

Problem: A couple compiler warnings.
Solution: Rename shadowed variables. Add UNUSED.
Files: src/misc1.c

Patch 8.1.0719

Problem: Too many #ifdefs.
Solution: Always build with the +visualextra feature.
Files: src/evalfunc.c, src/version.c, src/normal.c, src/ops.c,
src/feature.h, runtime/doc/various.txt

Patch 8.1.0720

Problem: Cannot easily change the current quickfix list index.
Solution: Add the "idx" argument to setqflist(). (Yegappan Lakshmanan,
closes #3701)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.1.0721

Problem: Conceal mode is not sufficiently tested.
Solution: Add screendump tests. Check all 'concealcursor' values.
Files: src/testdir/test_conceal.vim, src/Make_all.mak,
src/testdir/Make_all.mak
src/testdir/dumps/Test_conceal_two_windows_01.dump,
src/testdir/dumps/Test_conceal_two_windows_02.dump,
src/testdir/dumps/Test_conceal_two_windows_03.dump,
src/testdir/dumps/Test_conceal_two_windows_04.dump,
src/testdir/dumps/Test_conceal_two_windows_05.dump,
src/testdir/dumps/Test_conceal_two_windows_06i.dump,
src/testdir/dumps/Test_conceal_two_windows_06v.dump,
src/testdir/dumps/Test_conceal_two_windows_06c.dump,
src/testdir/dumps/Test_conceal_two_windows_06n.dump,
src/testdir/dumps/Test_conceal_two_windows_07i.dump,
src/testdir/dumps/Test_conceal_two_windows_07v.dump,
src/testdir/dumps/Test_conceal_two_windows_07c.dump,
src/testdir/dumps/Test_conceal_two_windows_07n.dump,
src/testdir/dumps/Test_conceal_two_windows_08i.dump,
src/testdir/dumps/Test_conceal_two_windows_08v.dump,
src/testdir/dumps/Test_conceal_two_windows_08c.dump,
src/testdir/dumps/Test_conceal_two_windows_08n.dump,
src/testdir/dumps/Test_conceal_two_windows_09i.dump,
src/testdir/dumps/Test_conceal_two_windows_09v.dump,
src/testdir/dumps/Test_conceal_two_windows_09c.dump,
src/testdir/dumps/Test_conceal_two_windows_09n.dump

Patch 8.1.0722

Problem: Cannot build without the virtualedit feature.
Solution: Make getviscol2() always available.
Files: src/misc2.c, src/proto/misc2.pro, src/ops.c

Patch 8.1.0723

Problem: Cannot run specific test when in src/testdir the same was as in the src directory.
Solution: Move build rule to src/testdir/Makefile.
Files: src/testdir/Make_all.mak, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/Makefile, src/Make_all.mak, src/testdir/Makefile, src/testdir/README.txt, src/Make_mvc.mak

Patch 8.1.0724

Problem: Build for MinGW fails.
Solution: Avoid specifying dependencies in included makefile.
Files: src/testdir/Make_all.mak, src/testdir/Makefile, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak

Patch 8.1.0725

Problem: Conceal mode is not completely tested.
Solution: Add tests for moving the cursor in Insert mode.
Files: src/testdir/test_conceal.vim, src/testdir/dumps/Test_conceal_two_windows_10.dump, src/testdir/dumps/Test_conceal_two_windows_11.dump, src/testdir/dumps/Test_conceal_two_windows_12.dump, src/testdir/dumps/Test_conceal_two_windows_13.dump

Patch 8.1.0726

Problem: Redrawing specifically for conceal feature.
Solution: Use generic redrawing methods.
Files: src/edit.c, src/gui.c, src/main.c, src/normal.c, src/screen.c, src/proto/screen.pro, src/window.c

Patch 8.1.0727

Problem: Compiler warning for sprintf() argument.
Solution: Add type cast.
Files: src/dosinst.c

Patch 8.1.0728

Problem: Cannot avoid breaking after a single space.
Solution: Add the 'p' flag to '**formatoptions**'. (Tom Ryder)
Files: runtime/doc/change.txt, src/edit.c, src/option.h, src/testdir/test_textformat.vim

Patch 8.1.0729

Problem: There is a SourcePre autocommand event but not a SourcePost.
Solution: Add the SourcePost autocommand event. (closes #3739)
Files: src/vim.h, src/fileio.c, src/ex_cmds2.c, runtime/doc/autocmd.txt, src/testdir/test_source.vim, src/testdir/Make_all.mak

Patch 8.1.0730

Problem: Compiler warning for get_buf_arg() unused.

Solution: Add #ifdef. (John Marriott)
Files: src/evalfunc.c

Patch 8.1.0731

Problem: JS encoding does not handle negative infinity.
Solution: Add support for negative infinity for JS encoding. (Dominique Pelle, closes #3792)
Files: runtime/doc/eval.txt, src/json.c, src/testdir/test_json.vim

Patch 8.1.0732

Problem: Cannot build without the eval feature.
Solution: Make a copy of the sourced file name.
Files: src/ex_cmds2.c

Patch 8.1.0733

Problem: Too many #ifdefs for the multibyte feature.
Solution: Tentatively always enable the multibyte feature. If you have a problem with this, please discuss on the Vim maillist.
Files: src/configure.ac, src/auto/configure, src/feature.h, src/Makefile, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.0734

Problem: The hlsearch state is not stored in a session file.
Solution: Add "nohlsearch" if appropriate. (Jason Franklin)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0735

Problem: Cannot handle binary data.
Solution: Add the Blob type. (Yasuhiro Matsumoto, closes #3638)
Files: runtime/doc/eval.txt, runtime/doc/if_perl.txt, runtime/doc/if_ruby.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile, src/blob.c, src/channel.c, src/eval.c, src/evalfunc.c, src/if_perl.xs, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/json.c, src/netbeans.c, src/proto.h, src/proto/blob.pro, src/proto/channel.pro, src/structs.h, src/testdir/Make_all.mak, src/vim.h, src/globals.h, src/testdir/test_blob.vim, src/testdir/test_channel.vim

Patch 8.1.0736

Problem: Code for Blob not sufficiently tested.
Solution: Add more tests. Fix uncovered crash. Add test_null_blob().
Files: src/testdir/test_blob.vim, src/testdir/test_assign.vim, src/eval.c, src/testdir/test_eval_stuff.vim, src/testdir/test_lambda.vim, runtime/doc/eval.txt, src/evalfunc.c, src/blob.c, src/testdir/test49.vim

Patch 8.1.0737

Problem: Compiler warning for uninitialized variable.
Solution: Add initialization. (John Marriott)
Files: src/eval.c

Patch 8.1.0738

Problem: Using freed memory, for loop over blob leaks memory.
Solution: Clear pointer after freeing memory. Decrement reference count

Files: after for loop over blob.
src/eval.c

Patch 8.1.0739

Problem: Text objects in not sufficiently tested.
Solution: Add a few more test cases. (Dominique Pelle, closes #3795)
Files: src/testdir/test_visual.vim

Patch 8.1.0740

Problem: Tcl test fails.
Solution: When the argument is empty don't give an error, instead rely on the error reporting higher up.
Files: src/eval.c

Patch 8.1.0741

Problem: Viminfo with Blob is not tested.
Solution: Extend the viminfo test. Fix reading a blob. Fixed storing a special variable value.
Files: src/testdir/test_viminfo.vim, src/eval.c, src/blob.c, src/proto/blob.pro

Patch 8.1.0742

Problem: Not all Blob operations are tested.
Solution: Add more testing for Blob.
Files: src/testdir/test_blob.vim, src/evalfunc.c, src/testdir/test_eval_stuff.vim

Patch 8.1.0743

Problem: Giving error messages is not flexible.
Solution: Add semsg(). Change argument from "char_u *" to "char *", also for msg() and get rid of most MSG macros. (Ozaki Kiichi, closes #3302) Also make emsg() accept a "char *" argument. Get rid of an enormous number of type casts.
Files: src/blob.c, src/blowfish.c, src/buffer.c, src/channel.c, src/crypt.c, src/dict.c, src/diff.c, src/digraph.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds.h, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/farsi.h, src/fileio.c, src/fold.c, src/getchar.c, src/globals.h, src/gui.c, src/gui_at_fs.c, src/gui_at_sb.c, src/gui_beval.c, src/gui_gtk_x11.c, src/gui_mac.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/hangulin.c, src/hardcopy.c, src/hashtab.c, src/if_cscope.c, src/if_lua.c, src/if_mzsch.c, src/if_perl.xs, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/if_xcmdsrv.c, src/json.c, src/list.c, src/main.c, src/mark.c, src/mbyte.c, src/memfile.c, src/memline.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/popupmnu.c, src/proto.h, src/proto/buffer.pro, src/proto/digraph.pro, src/proto/ex_docmd.pro, src/proto/ex_eval.pro, src/proto/ex_getln.pro, src/proto/hardcopy.pro, src/proto/mbyte.pro, src/proto/message.pro, src/proto/misc2.pro, src/proto/option.pro, src/proto/spell.pro, src/quickfix.c, src/regexp.c,

src/regexp_nfa.c, src/search.c, src/sign.c, src/spell.c,
src/spellfile.c, src/structs.h, src/syntax.c, src/tag.c,
src/term.c, src/terminal.c, src/textprop.c, src/ui.c, src/undo.c,
src/userfunc.c, src/version.c, src/vim.h, src/window.c,

Patch 8.1.0744 (after 8.1.0743)

Problem: Compiler warnings for signed/unsigned strings.
Solution: A few more type cast fixes.
Files: src/option.c, src/if_perl.xs, src/if_py_both.h, src/integration.c

Patch 8.1.0745

Problem: Compiler warnings for signed/unsigned string.
Solution: Remove type casts. (John Marriott)
Files: src/ex_docmd.c, src/mbyte.c

Patch 8.1.0746

Problem: Highlighting not updated with conceal and **'cursorline'**. (Jason Franklin)
Solution: Do not use a zero line number. Check if **'conceallevel'** is set for the current window.
Files: src/main.c, src/testdir/test_conceal.vim,
src/testdir/dumps/Test_conceal_cul_01.dump,
src/testdir/dumps/Test_conceal_cul_02.dump,
src/testdir/dumps/Test_conceal_cul_03.dump

Patch 8.1.0747

Problem: map() with a bad expression doesn't give an error. (Ingo Karkat)
Solution: Check for giving an error message. (closes #3800)
Files: src/eval.c, src/testdir/test_filter_map.vim

Patch 8.1.0748

Problem: Using sprintf() instead of semsg().
Solution: Use semsg(). Fix bug with E888. (Ozaki Kiichi, closes #3801)
Files: src/regexp.c

Patch 8.1.0749 (after 8.1.0747)

Problem: Error message contains garbage. (Dominique Pelle)
Solution: Use correct pointer to failed expression.
Files: src/eval.c

Patch 8.1.0750

Problem: When the last sign is deleted the signcolumn may not be removed even though **'signcolumn'** is "auto".
Solution: When deleting the last sign redraw the buffer. (Dominique Pelle, closes #3803, closes #3804)
Files: src/sign.c

Patch 8.1.0751

Problem: Some regexp errors are not tested.
Solution: Add a test function.
Files: src/testdir/test_regexp_latin.vim

Patch 8.1.0752

Problem: One more compiler warning for signed/unsigned string. (Tony

Mechelynck)
Solution: Remove type cast.
Files: src/ex_docmd.c

Patch 8.1.0753

Problem: printf format not checked for semsg().
Solution: Add GNUC attribute and fix reported problems. (Dominique Pelle, closes #3805)
Files: src/buffer.c, src/diff.c, src/eval.c, src/evalfunc.c, src/ex_docmd.c, src/if_cscope.c, src/netbeans.c, src/proto.h, src/proto/message.pro, src/quickfix.c, src/regexp_nfa.c, src/sign.c, src/spellfile.c, src/window.c, src/gui_x11.c

Patch 8.1.0754

Problem: Preferred column is lost when setting '**cursorcolumn**'.
Solution: Change option flag to P_RWINONLY. (Takayuki Kurosawa, closes #3806)
Files: src/option.c, src/testdir/test_cursor_func.vim

Patch 8.1.0755

Problem: Error message for get() on a Blob with invalid index.
Solution: Return an empty Blob, like get() on a List does.
Files: src/evalfunc.c, src/testdir/test_blob.vim

Patch 8.1.0756

Problem: copy() does not make a copy of a Blob.
Solution: Make a copy.
Files: src/eval.c, src/testdir/test_blob.vim

Patch 8.1.0757

Problem: Not enough documentation for Blobs.
Solution: Add a section about Blobs.
Files: runtime/doc/eval.txt

Patch 8.1.0758

Problem: Font number is always one instead of the actual.
Solution: Use "%d" instead of "1". (Ken Takata)
Files: src/gui_x11.c

Patch 8.1.0759

Problem: Showing two characters for tab is limited.
Solution: Allow for a third character for "tab:" in '**listchars**'. (Nathaniel Braun, Ken Takata, closes #3810)
Files: runtime/doc/options.txt, src/globals.h, src/message.c, src/option.c, src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.0760

Problem: No proper test for using '**termencoding**'.
Solution: Add a screendump test. Fix using double width characters in a screendump.
Files: src/terminal.c, src/testdir/test_termencoding.vim, src/testdir/Make_all.mak, src/testdir/dumps/Test_tenc_euc_jp_01.dump

Patch 8.1.0761

Problem: Default value for brief_wait is wrong.
Solution: Make the default FALSE. (Ozaki Kiichi, closes #3812, closes #3799)
Files: src/ui.c

Patch 8.1.0762

Problem: Compiler warning.
Solution: Add type cast. (Mike Williams)
Files: src/channel.c

Patch 8.1.0763

Problem: Nobody is using the Sun Workshop support.
Solution: Remove the Workshop support.
Files: runtime/doc/workshop.txt, runtime/doc/help.txt,
runtime/doc/netbeans.txt, src/Makefile, src/auto/configure,
src/beval.c, src/buffer.c, src/config.h.in, src/config.mk.in,
src/configure.ac, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds.h,
src/ex_docmd.c, src/feature.h, src/fileio.c, src/globals.h,
src/gui.c, src/gui_beval.c, src/gui_motif.c, src/gui_x11.c,
src/integration.c, src/integration.h, src/main.c, src/misc2.c,
src/nbdebug.c, src/netbeans.c, src/proto.h,
src/proto/workshop.pro, src/ui.c, src/version.c, src/vim.h,
src/workshop.c, src/workshop.h, src/wsdebug.c, src/wsdebug.h,
src/ex_cmdidxs.h

Patch 8.1.0764

Problem: List of distributed files is outdated.
Solution: Remove workshop files. Add blob files.
Files: Filelist

Patch 8.1.0765

Problem: String format of a Blob can't be parsed back.
Solution: Use 0z format.
Files: src/blob.c, src/eval.c, src/testdir/test_blob.vim

Patch 8.1.0766

Problem: Various problems when using Vim on VMS.
Solution: Various fixes. Define long_long_T. (Zoltan Arpadffy)
Files: src/eval.c, src/feature.h, src/fileio.c, src/gui_motif.c,
src/gui_x11.c, src/gui_xmebw.c, src/json.c, src/Make_vms.mms,
src/ops.c, src/os_vms_conf.h, src/vim.h, src/xdiff/xdiff.h,
src/xdiff/xinclude.h

Patch 8.1.0767

Problem: When deleting lines at the bottom signs are misplaced.
Solution: Properly update the line number of signs at the end of a buffer
after a delete/undo operation. (Yegappan Lakshmanan, closes #3798)
Files: src/sign.c, src/testdir/test_signs.vim

Patch 8.1.0768

Problem: Updating completions may cause the popup menu to flicker.
Solution: Avoid updating the text below the popup menu before drawing the
popup menu.
Files: src/popupmnu.c, src/proto/popupmnu.pro, src/edit.c, src/screen.c

Patch 8.1.0769

Problem: :stop is covered in two tests.
Solution: Remove Test_stop_in_terminal(). Make other test exit Vim cleanly.
(Ozaki Kiichi, closes #3814)
Files: src/testdir/test_terminal.vim, src/testdir/test_suspend.vim

Patch 8.1.0770

Problem: Inconsistent use of ELAPSED_FUNC.
Solution: Consistently use ELAPSED_FUNC. Also turn ELAPSED_TYPE into a typedef. (Ozaki Kiichi, closes #3815)
Files: src/channel.c, src/gui.c, src/misc1.c, src/os_unix.c, src/vim.h

Patch 8.1.0771

Problem: Some shell filetype patterns end in a star.
Solution: Make sure that patterns not ending in a star are preferred.
Files: runtime/filetype.vim, runtime/autoload/dist/ft.vim

Patch 8.1.0772

Problem: The sign_define_by_name() function is too long.
Solution: Split it into smaller functions. (Yegappan Lakshmanan, closes #3819)
Files: src/sign.c

Patch 8.1.0773

Problem: Not all crypt code is tested.
Solution: Disable unused crypt code. Add more test coverage.
Files: src/structs.h, src/crypt.c, src/testdir/test_crypt.vim, src/proto/crypt.pro, src/fileio.c

Patch 8.1.0774

Problem: VMS build is missing the blob file.
Solution: Add the blob file to the build rules. (Zoltan Arpadffy)
Files: src/Make_vms.mms, runtime/doc/os_vms.txt

Patch 8.1.0775

Problem: Matching too many files as zsh. (Danek Duvall)
Solution: Be more specific with zsh filetype patterns.
Files: runtime/filetype.vim

Patch 8.1.0776

Problem: Travis does not build a version without GUI on Linux.
Solution: Add an environment for tiny features without GUI.
Files: .travis.yml

Patch 8.1.0777

Problem: Win32: using pipes for channel does not work well.
Solution: Use a larger buffer and handle overlaps. (Yasuhiro Matsumoto, closes #3782)
Files: src/channel.c, src/os_win32.c

Patch 8.1.0778

Problem: Terminal test fails on MS-Windows.
Solution: Temporarily skip the test on MS-Windows. Do run it both in

terminal and GUI on other systems.
Files: src/testdir/test_terminal.vim

Patch 8.1.0779

Problem: Argument for message functions is inconsistent.
Solution: Make first argument to msg() "char *".
Files: src/buffer.c, src/crypt.c, src/edit.c, src/ex_cmds.c, src/eval.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/farsi.c, src/if_cscope.c, src/fileio.c, src/getchar.c, src/globals.h, src/gui.c, src/if_perl.xs, src/netbeans.c, src/gui_w32.c, src/hardcopy.c, src/if_mzsch.c, src/if_py_both.h, src/if_ruby.c, src/if_tcl.c, src/mark.c, src/mbyte.c, src/menu.c, src/memline.c, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/ops.c, src/option.c, src/os_amiga.c, src/os_unix.c, src/os_win32.c, src/proto/message.pro, src/quickfix.c, src/sign.c, src/regexp.c, src/ui.c, src/screen.c, src/search.c, src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c, src/term.c, src/undo.c, src/userfunc.c, src/version.c, src/vim.h, src/window.c, src/proto/eval.pro, src/evalfunc.c, src/ex_eval.c, src/farsi.h

Patch 8.1.0780

Problem: Terminal test fails on Mac.
Solution: Skip the test on Mac.
Files: src/testdir/test_terminal.vim

Patch 8.1.0781

Problem: Build error when using if_xcmdsrv.c.
Solution: Add missing part of 8.1.0779.
Files: src/if_xcmdsrv.c

Patch 8.1.0782

Problem: Win32: cursor blinks when Vim is not active.
Solution: Remove call to setActiveWindow(). (Yasuhiro Matsumoto, closes #3778)
Files: src/gui_w32.c, src/proto/gui_w32.pro, src/menu.c

Patch 8.1.0783

Problem: Compiler warning for signed/unsigned.
Solution: Add type cast. Change type of buffer. (Ozaki Kiichi, closes #3827)
Files: src/main.c, src/message.c

Patch 8.1.0784

Problem: Messy indent in if statement.
Solution: Improve structure of if statement. (Ozaki Kiichi, closes #3826)
Files: src/os_win32.c

Patch 8.1.0785

Problem: Depending on the configuration some functions are unused.
Solution: Add more #ifdefs, remove unused functions. (Dominique Pelle, closes #3822)
Files: src/buffer.c, src/channel.c, src/ex_cmds2.c, src/ex_docmd.c, src/fileio.c, src/getchar.c, src/gui_gtk_x11.c, src/hashtab.c, src/json.c, src/mbyte.c, src/message.c, src/misc1.c, src/misc2.c, src/ops.c, src/option.c, src/os_unix.c, src/proto/os_unix.pro,

src/proto/regexp.pro, src/proto/terminal.pro, src/regexp.c,
src/screen.c, src/search.c, src/syntax.c, src/term.c,
src/terminal.c, src/ui.c, src/userfunc.c

Patch 8.1.0786

Problem: ml_get error when updating the status line and a terminal had its
scrollback cleared. (Chris Patuzzo)
Solution: Check the cursor position when drawing the status line.
(closes #3830)
Files: src/buffer.c, src/testdir/test_terminal.vim

Patch 8.1.0787

Problem: Compiler warning for unused function. (Tony Mechelynck)
Solution: Tune #ifdef around setjmp functions.
Files: src/os_unix.c

Patch 8.1.0788

Problem: Cannot build with tiny features.
Solution: Adjust #ifdefs.
Files: src/os_unix.c

Patch 8.1.0789

Problem: Sourcing a session sets v:errmsg.
Solution: Use "%argdel" instead of "argdel *". (Jason Franklin)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0790

Problem: Code for creating tabpages in session is too complex.
Solution: Simplify the code. (Jason Franklin)
Files: src/ex_docmd.c

Patch 8.1.0791

Problem: A few compiler warnings on VMS.
Solution: Remove type cast. Adjust #ifdef. (Zoltan Arpadffy)
Files: src/os_unix.c, src/proto.h

Patch 8.1.0792

Problem: Popup menu is displayed on top of the cmdline window if it is
opened from Insert completion. (Bjorn Linse)
Solution: Remove the popup menu. Restore the cursor position.
(closes #3838)
Files: src/edit.c, src/ex_getln.c

Patch 8.1.0793

Problem: Incorrect error messages for functions that now take a Blob
argument.
Solution: Adjust the error messages. (Dominique Pelle, closes #3846)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/globals.h,
src/testdir/test_blob.vim, src/testdir/test_listdict.vim

Patch 8.1.0794

Problem: White space before "-Ntabmove" causes problems.
Solution: Skip whitespace. (Ozaki Kiichi, closes #3841)
Files: src/ex_docmd.c, src/testdir/test_tabpage.vim

Patch 8.1.0795 (after 8.1.0792)

Problem: Cannot build without popup menu.

Solution: Add #ifdef

Files: src/ex_getln.c

Patch 8.1.0796

Problem: MS-Windows 7: problem with named pipe on channel.

Solution: Put back the disconnect/connect calls. (Yasuhiro Matsumoto, closes #3833)

Files: src/channel.c, src/testdir/test_terminal.vim

Patch 8.1.0797

Problem: Error E898 is used twice.

Solution: Rename the Blob error to E899. (closes #3853)

Files: src/evalfunc.c, runtime/doc/eval.txt, src/testdir/test_listdict.vim

Patch 8.1.0798

Problem: Changing a blob while iterating over it works strangely.

Solution: Make a copy of the Blob before iterating.

Files: src/blob.c, src/proto/blob.pro, src/eval.c, src/testdir/test_blob.vim

Patch 8.1.0799

Problem: Calling deleted function; test doesn't work on Mac.

Solution: Wait for the function to be called before deleting it. Use a job to write to the pty, unless in the GUI. (Ozaki Kiichi, closes #3854)

Files: src/testdir/test_channel.vim, src/testdir/test_terminal.vim

Patch 8.1.0800

Problem: May use a lot of memory when a function creates a cyclic reference.

Solution: After saving a funccal many times, invoke the garbage collector. (closes #3835)

Files: src/userfunc.c

Patch 8.1.0801

Problem: MinGW: no hint that tests fail because of small terminal.

Solution: Add a rule for test1 that checks for "wrongtermsize". (msoyka-of-wharton)

Files: src/testdir/Make_ming.mak

Patch 8.1.0802

Problem: Negative index doesn't work for Blob.

Solution: Make it work, add a test. (closes #3856)

Files: src/blob.c, src/proto/blob.pro, src/eval.c, src/testdir/test_blob.vim

Patch 8.1.0803

Problem: Session file has problem with single quote in file name. (Jon Crowe)

Solution: Use a double quoted string. Add a test.

Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0804

Problem: Crash when setting v:errmsg to empty list. (Jason Franklin)

Solution: Separate getting value and assigning result.

Files: src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.1.0805

Problem: Too many #ifdefs.

Solution: Graduate FEAT_MBYTE, part 1.

Files: src/buffer.c, src/charset.c, src/diff.c, src/digraph.c,
src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c,
src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c,
src/fold.c, src/gui.c, src/gui_mac.c, src/gui_photon.c,
src/gui_w32.c

Patch 8.1.0806

Problem: Too many #ifdefs.

Solution: Graduate FEAT_MBYTE, part 2.

Files: src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/gui_w32.c,
src/gui_x11.c, src/hardcopy.c, src/if_xcmdsrv.c, src/json.c,
src/keyword_test.c, src/main.c, src/mbyte.c, src/memline.c,
src/message.c, src/misc1.c, src/misc2.c, src/move.c, src/normal.c,
src/ops.c, src/option.c, src/charset.c

Patch 8.1.0807

Problem: Session test fails on MS-Windows.

Solution: Don't try creating file with illegal name.

Files: src/testdir/test_mksession.vim

Patch 8.1.0808

Problem: MS-Windows: build error with GUI.

Solution: Remove "static".

Files: src/gui_w32.c

Patch 8.1.0809

Problem: Too many #ifdefs.

Solution: Graduate FEAT_MBYTE, part 3.

Files: src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_w32exe.c,
src/os_win32.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c,
src/screen.c

Patch 8.1.0810

Problem: Too many #ifdefs.

Solution: Graduate FEAT_MBYTE, part 4.

Files: src/getchar.c, src/search.c, src/sign.c, src/spell.c,
src/spellfile.c, src/syntax.c, src/tag.c, src/term.c, src/ui.c,
src/version.c, src/winclip.c, src/window.c, src/glbl_ime.cpp,
src/ex_cmds.h, src/globals.h, src/gui.h, src/if_py_both.h,
src/macros.h, src/option.h, src/os_mac.h, src/os_win32.h,
src/proto.h, src/spell.h, src/structs.h, src/vim.h

Patch 8.1.0811

Problem: Too many #ifdefs.

Solution: Graduate FEAT_MBYTE, the final chapter.
Files: src/feature.h, src/vim.h, src/crypt_zip.c, src/fileio.c,
src/message.c, src/spell.h, src/structs.h, src/config.h.in,
src/configure.ac, src/auto/configure, src/testdir/runtest.vim,
src/testdir/test_alot_utf8.vim, src/testdir/test_arabic.vim,
src/testdir/test_charsearch_utf8.vim,
src/testdir/test_cmdline.vim, src/testdir/test_digraph.vim,
src/testdir/test_display.vim, src/testdir/test_edit.vim,
src/testdir/test_erasebackward.vim,
src/testdir/test_expr_utf8.vim, src/testdir/test_functions.vim,
src/testdir/test_ga.vim, src/testdir/test_iminsert.vim,
src/testdir/test_increment_dbcs.vim, src/testdir/test_json.vim,
src/testdir/test_makeencoding.vim, src/testdir/test_maparg.vim,
src/testdir/test_mapping.vim, src/testdir/test_marks.vim,
src/testdir/test_match.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_mksession_utf8.vim, src/testdir/test_normal.vim,
src/testdir/test_plus_arg_edit.vim, src/testdir/test_profile.vim,
src/testdir/test_put.vim, src/testdir/test_regex_char_classes.vim,
src/testdir/test_regexp_utf8.vim, src/testdir/test_search.vim,
src/testdir/test_source_utf8.vim, src/testdir/test_spell.vim,
src/testdir/test_startup_utf8.vim,
src/testdir/test_termencoding.vim, src/testdir/test_terminal.vim,
src/testdir/test_utf8.vim, src/testdir/test_utf8_comparisons.vim,
src/testdir/test_viminfo.vim, src/testdir/test_virtualedit.vim,
src/testdir/test_visual.vim, src/testdir/test_wordcount.vim,
src/testdir/test_writefile.vim, src/appveyor.bat, src/os_macosx.m

Patch 8.1.0812

Problem: Unicode 16 feature is not useful and cannot be detected.
Solution: Remove UNICODE16.
Files: src/screen.c, src/vim.h, src/feature.h

Patch 8.1.0813

Problem: FileChangedShell not sufficiently tested.
Solution: Add a more comprehensive test case.
Files: src/testdir/test_autocmd.vim

Patch 8.1.0814

Problem: :mksession cannot handle a very long **'runtimepath'**. (Timothy Madden)
Solution: Expand each part separately, instead of the whole option at once. (Christian Brabandt, closes #3466)
Files: src/option.c, src/testdir/test_mksession.vim

Patch 8.1.0815

Problem: Dialog for file changed outside of Vim not tested.
Solution: Add a test. Move FileChangedShell test. Add 'L' flag to feedkeys().
Files: src/testdir/test_autocmd.vim, src/testdir/test_filechanged.vim, src/testdir/Make_all.mak, src/evalfunc.c, runtime/doc/eval.txt

Patch 8.1.0816

Problem: Test for **'runtimepath'** in session fails on MS-Windows.

Solution: Skip the test for now.
Files: src/testdir/test_mksession.vim

Patch 8.1.0817

Problem: "!=" command is not tested.
Solution: Add a test. (Dominique Pelle, closes #3859)
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_ex_equal.vim

Patch 8.1.0818

Problem: MS-Windows: cannot send large data with ch_sendraw().
Solution: Split write into several WriteFile() calls. (Yasuhiro Matsumoto,
closes #3823)
Files: src/channel.c, src/os_win32.c, src/testdir/test_channel.vim,
src/testdir/test_channel_pipe.py, src/vim.h

Patch 8.1.0819

Problem: A failed assert with a long string is hard to read.
Solution: Shorten the assert message.
Files: src/eval.c, src/testdir/test_assert.vim

Patch 8.1.0820

Problem: Test for sending large data over channel sometimes fails.
Solution: Handle that the job may have finished early. Also fix that file
changed test doesn't work in the GUI and reduce flakiness. (Ozaki
Kiichi, closes #3861)
Files: src/testdir/test_channel.vim, src/testdir/test_filechanged.vim

Patch 8.1.0821

Problem: Xxd "usage" output and other arguments not tested.
Solution: Add a test to trigger the usage output in various ways. Fix
uncovered problem.
Files: src/testdir/test_xxd.vim, src/xxd/xxd.c

Patch 8.1.0822

Problem: Peeking and flushing output slows down execution.
Solution: Do not update the mode message when global_busy is set. Do not
flush when only peeking for a character. (Ken Takata)
Files: src/getchar.c, src/screen.c, src/proto/screen.pro, src/edit.c

Patch 8.1.0823

Problem: Not sufficient testing of xxd.
Solution: Add some more test coverage.
Files: src/testdir/test_xxd.vim

Patch 8.1.0824

Problem: SunOS/Solaris has a problem with ttys.
Solution: Add mch_isatty() with extra handling for SunOS. (Ozaki Kiichi,
closes #3865)
Files: src/auto/configure, src/channel.c, src/config.h.in,
src/configure.ac, src/os_unix.c, src/proto/pty.pro, src/pty.c,
src/terminal.c

Patch 8.1.0825

Problem: Code for autocommands is mixed with file I/O code.
Solution: Move autocommand code to a separate file. (Yegappan Lakshmanan, closes #3863)
Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/README.txt, src/autocmd.c, src/fileio.c, src/globals.h, src/proto.h, src/proto/autocmd.pro, src/proto/fileio.pro

Patch 8.1.0826

Problem: Too many #ifdefs.
Solution: Graduate FEAT_VIRTUALEDIT. Adds about 10Kbyte to the code.
Files: src/buffer.c, src/charset.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/feature.h, src/globals.h, src/gui.c, src/if_py_both.h, src/macros.h, src/mark.c, src/mbyte.c, src/memline.c, src/menu.c, src/misc1.c, src/misc2.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/option.h, src/screen.c, src/search.c, src/spell.c, src/structs.h, src/tag.c, src/ui.c, src/undo.c, src/userfunc.c, src/version.c, src/vim.h, src/window.c

Patch 8.1.0827 (after 8.1.0825)

Problem: Missing dependency in Makefile.
Solution: Add dependency from autocmd.o on auto/osdef.h
Files: src/Makefile

Patch 8.1.0828

Problem: Still using FEAT_VIRTUALEDIT.
Solution: Remove last use of FEAT_VIRTUALEDIT.
Files: src/quickfix.c

Patch 8.1.0829

Problem: When **'hidden'** is set session creates extra buffers.
Solution: Move :badd commands to the end. (Jason Franklin)
Files: src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.1.0830

Problem: Test leaves directory behind on MS-Windows.
Solution: Close buffer before deleting directory.
Files: src/testdir/test_swap.vim

Patch 8.1.0831

Problem: Xxd test fails if man page has dos fileformat.
Solution: Make a copy with unix fileformat.
Files: src/testdir/test_xxd.vim

Patch 8.1.0832

Problem: confirm() is not tested.
Solution: Add a test. (Dominique Pelle, closes #3868)
Files: src/testdir/test_functions.vim

Patch 8.1.0833

Problem: Memory leak when jumps output is filtered.

Solution: Free the filtered name. (Dominique Pelle, closes #3869)
Files: src/mark.c

Patch 8.1.0834

Problem: GUI may wait too long before dealing with messages. Returning early may cause a mapping to time out.

Solution: Use the waiting loop from Unix also for the GUI.
(closes #3817, closes #3824)

Files: src/ui.c, src/proto/ui.pro, src/os_unix.c, src/gui.c,
src/testdir/screendump.vim

Patch 8.1.0835

Problem: GUI build fails on MS-Windows.

Solution: Adjust #ifdef.

Files: src/ui.c

Patch 8.1.0836

Problem: User completion test can fail on MS-Windows.

Solution: Allow for other names before "Administrator".

Files: src/testdir/test_cmdline.vim

Patch 8.1.0837

Problem: Timer interrupting cursorhold and mapping not tested.

Solution: Add tests with timers. (Ozaki Kiichi, closes #3871)

Files: src/testdir/test_autocmd.vim, src/testdir/test_mapping.vim

Patch 8.1.0838

Problem: Compiler warning for type conversion.

Solution: Add a type cast. (Mike Williams)

Files: src/channel.c

Patch 8.1.0839

Problem: When using VTP wrong colors after a color scheme change.

Solution: When VTP is active always clear after a color scheme change.
(Nobuhiro Takasaki, closes #3872)

Files: src/ex_docmd.c

Patch 8.1.0840

Problem: getchar(0) never returns a character in the terminal.

Solution: Call wait_func() at least once.

Files: src/ui.c, src/testdir/test_timers.vim, src/gui_gtk_x11.c,
src/gui_w32.c, src/gui_photon.c, src/gui_x11.c

Patch 8.1.0841

Problem: Travis config to get Lua on macOS is too complicated.

Solution: Use an addons entry. (Ozaki Kiichi, closes #3876)

Files: .travis.yml

Patch 8.1.0842

Problem: getchar_zero test fails on MS-Windows.

Solution: Disable the test for now.

Files: src/testdir/test_timers.vim

Patch 8.1.0843

Problem: Memory leak when running "make test_cd".
Solution: Free the stack element when failing. (Dominique Pelle, closes #3877)
Files: src/misc2.c

Patch 8.1.0844

Problem: When timer fails test will hang forever.
Solution: Use reltime() to limit waiting time. (Ozaki Kiichi, closes #3878)
Files: src/testdir/test_timers.vim

Patch 8.1.0845

Problem: Having job_status() free the job causes problems.
Solution: Do not actually free the job or terminal yet, put it in a list and free it a bit later. Do not use a terminal after checking the job status. (closes #3873)
Files: src/channel.c, src/terminal.c, src/proto/terminal.pro, src/misc2.c

Patch 8.1.0846

Problem: Not easy to recognize the system Vim runs on.
Solution: Add more items to the features list. (Ozaki Kiichi, closes #3855)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_channel.vim, src/testdir/test_functions.vim, src/testdir/test_terminal.vim, src/testdir/test_writefile.vim

Patch 8.1.0847

Problem: May use terminal after it was cleaned up.
Solution: Use the job pointer.
Files: src/terminal.c

Patch 8.1.0848

Problem: Cannot build with Ruby 1.8. (Tom G. Christensen)
Solution: Use rb_str_new2(). (Yasuhiro Matsumoto, closes #3883, closes #3884)
Files: src/if_ruby.c

Patch 8.1.0849

Problem: Cursorline highlight is not always updated.
Solution: Set w_last_cursorline when redrawing. Fix resetting cursor flags when using the popup menu.
Files: src/screen.c, src/popupmnu.c, src/testdir/test_highlight.vim, src/testdir/dumps/Test_cursorline_yank_01.dump

Patch 8.1.0850

Problem: Test for **'backupskip'** is not correct.
Solution: Split the option in parts and use expand(). (Michael Soyka)
Files: src/testdir/test_options.vim

Patch 8.1.0851

Problem: feedkeys() with "L" does not work properly.
Solution: Do not set typebuf_was_filled when using "L". (Ozaki Kiichi, closes #3885)
Files: src/evalfunc.c, src/testdir/test_autocmd.vim, src/testdir/test_mapping.vim, src/testdir/test_timers.vim

Patch 8.1.0852

Problem: findfile() and finddir() are not properly tested.
Solution: Extend the test and add more. (Dominique Pelle, closes #3880)
Files: src/testdir/test_findfile.vim

Patch 8.1.0853 (after 8.1.0850)

Problem: Options test fails on Mac.
Solution: Remove a trailing slash from \$TMPDIR.
Files: src/testdir/test_options.vim

Patch 8.1.0854

Problem: xxd does not work with more than 32 bit addresses.
Solution: Add support for 64 bit addresses. (Christer Jensen, closes #3791)
Files: src/xxd/xxd.c

Patch 8.1.0855

Problem: Cannot build xxd with MSVC 10.
Solution: Move declaration to start of block.
Files: src/xxd/xxd.c

Patch 8.1.0856

Problem: When scrolling a window other than the current one the cursorline highlighting is not always updated. (Jason Franklin)
Solution: Call redraw_for_cursorline() after scrolling. Only set w_last_cursorline when drawing the cursor line. Reset the lines to be redrawn also when redrawing the whole window.
Files: src/move.c, src/proto/move.pro, src/normal.c

Patch 8.1.0857

Problem: Indent functionality is not separated.
Solution: Move indent functionality into a new file. (Yegappan Lakshmanan, closes #3886)
Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/edit.c, src/indent.c, src/misc1.c, src/proto.h, src/proto/edit.pro, src/proto/indent.pro, src/proto/misc1.pro

Patch 8.1.0858

Problem: **'indentkeys'** and **'cinkeys'** defaults are different.
Solution: Make them the same, update docs. (close #3882)
Files: src/option.c, runtime/doc/options.txt, runtime/doc/indent.txt

Patch 8.1.0859

Problem: "%v" in **'errorformat'** does not handle multibyte characters.
Solution: Handle multibyte characters. (Yegappan Lakshmanan, closes #3700)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0860

Problem: Debug lines left in the code.
Solution: Delete the lines.
Files: src/edit.c

Patch 8.1.0861

Problem: Building with MinGW and static libc doesn't work.
Solution: Change the LIB argument. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 8.1.0862

Problem: No verbose version of character classes.
Solution: Add [[:ident:]], [[:keyword:]] and [[:fname:]]. (Ozaki Kiichi, closes #1373)
Files: runtime/doc/pattern.txt, src/regexp.c, src/regexp_nfa.c, src/testdir/test_regexp_utf8.vim

Patch 8.1.0863

Problem: Cannot see what signal caused a job to end.
Solution: Add "termsig" to job_info(). (Ozaki Kiichi, closes #3786)
Files: runtime/doc/eval.txt, src/channel.c, src/os_unix.c, src/structs.h, src/testdir/test_channel.vim

Patch 8.1.0864

Problem: Cannot have a local value for '**scrolloff**' and '**sidescrolloff**'. (Gary Holloway)
Solution: Make '**scrolloff**' and '**sidescrolloff**' global-local. (mostly by Aron Widforss, closes #3539)
Files: runtime/doc/options.txt, src/edit.c, src/ex_cmds.c, src/ex_docmd.c, src/gui.c, src/misc2.c, src/move.c, src/normal.c, src/option.c, src/proto/option.pro, src/option.h, src/search.c, src/structs.h, src/window.c, src/testdir/test_options.vim

Patch 8.1.0865

Problem: When '**listchars**' only contains "nbsp:X" it does not work.
Solution: Set extra_check when lcs_nbsp is set. (Ralf Schandl, closes #3889)
Files: src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.0866

Problem: Build file dependencies are outdated. (John Little)
Solution: Run "make proto" and "make depend".
Files: src/vim.h, src/Makefile, src/proto/sign.pro, src/proto/gui_w32.pro

Patch 8.1.0867

Problem: Cannot build Python interface with Python 2.4. (Tom G. Christensen)
Solution: Define PyBytes_FromStringAndSize. (Ken Takata, closes #3888)
Files: src/if_python.c

Patch 8.1.0868

Problem: Crash if triggering garbage collector after a function call. (Michael Henry)
Solution: Don't call the garbage collector right away, do it later. (closes #3894)
Files: src/userfunc.c

Patch 8.1.0869

Problem: Travis CI script is too complicated.
Solution: Add names to environments. Move appveyor script outside of src directory. (Ozaki Kiichi, closes #3890)

Files: .travis.yml, appveyor.yml, ci/appveyor.bat, src/appveyor.bat, Filelist

Patch 8.1.0870

Problem: Vim doesn't use the new ConPTY support in Windows 10.
Solution: Use ConPTY support, if available. (Nobuhiro Takasaki, closes #3794)
Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/terminal.txt, src/channel.c, src/evalfunc.c, src/globals.h, src/option.c, src/option.h, src/os_win32.c, src/proto/terminal.pro, src/structs.h, src/terminal.c, src/testdir/gen_opt_test.vim, src/testdir/test_autocmd.vim, src/testdir/test_mksession.vim, src/testdir/test_terminal.vim

Patch 8.1.0871

Problem: Build error when building with Ruby 2.6.0.
Solution: Change argument of rb_int2big_stub(). (Android Baumann, closes #3899)
Files: src/if_ruby.c

Patch 8.1.0872

Problem: Confusing condition.
Solution: Use "==" instead of "<=".
Files: src/gui_gtk_x11.c

Patch 8.1.0873

Problem: List if distributed files does not include the matchit autoload directory.
Solution: Add the directory.
Files: src/Filelist

Patch 8.1.0874

Problem: Using old style comments in new file.
Solution: Convert to // comments in new file. (Yegappan Lakshmanan)
Files: src/indent.c

Patch 8.1.0875

Problem: Not all errors of marks and findfile()/finddir() are tested.
Solution: Add more test coverage. (Dominique Pelle)
Files: src/testdir/test_findfile.vim, src/testdir/test_marks.vim

Patch 8.1.0876

Problem: Completion match not displayed when popup menu is not shown.
Solution: Call update_screen() when not displaying the popup menu to show the inserted match. (Ken Takata, Hirohito Higashi)
Files: src/edit.c

Patch 8.1.0877

Problem: New buffer used every time the quickfix window is opened.
Solution: Reuse the buffer. (Yegappan Lakshmanan, closes #3902)
Files: src/buffer.c, src/proto/quickfix.pro, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0878

Problem: Test for has('bsd') fails on some BSD systems.

Solution: Adjust the uname match. (James McCoy, closes #3909)
Files: src/testdir/test_functions.vim

Patch 8.1.0879

Problem: MS-Windows: temp name encoding can be wrong.
Solution: Convert from active code page to **'encoding'**. (Yasuhiro Matsumoto, closes #3520, closes #1698)
Files: src/fileio.c

Patch 8.1.0880

Problem: MS-Windows: inconsistent selection of winpty/conpty.
Solution: Name option **'termwintype'**, use ++type argument and "term_ptty" for term_start(). (Hirohito Higashi, closes #3915)
Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/terminal.txt, src/channel.c, src/option.c, src/option.h, src/structs.h, src/terminal.c, src/testdir/gen_opt_test.vim, runtime/optwin.vim, runtime/doc/quickref.txt

Patch 8.1.0881

Problem: Can execute shell commands in rvim through interfaces.
Solution: Disable using interfaces in restricted mode. Allow for writing file with writefile(), histadd() and a few others.
Files: runtime/doc/starting.txt, src/if_perl.xs, src/if_cmds.h, src/ex_cmds.c, src/ex_docmd.c, src/evalfunc.c, src/testdir/test_restricted.vim, src/testdir/Make_all.mak

Patch 8.1.0882 (after 8.1.0879)

Problem: Checking for FEAT_MBYTE which doesn't exist anymore. (Christ van Willegen)
Solution: Remove it.
Files: src/fileio.c

Patch 8.1.0883

Problem: Missing some changes for Ex commands.
Solution: Add missing changes in header file.
Files: src/ex_cmds.h

Patch 8.1.0884

Problem: Double check for bsd systems.
Solution: Delete the old line.
Files: src/testdir/test_functions.vim

Patch 8.1.0885

Problem: Test for restricted hangs on MS-Windows GUI.
Solution: Skip the test.
Files: src/testdir/test_restricted.vim

Patch 8.1.0886

Problem: Compiler warning for adding to NULL pointer and a condition that is always true.
Solution: Check for NULL pointer before adding. Remove useless "if". (Friedirch, closes #3913)
Files: src/dosinst.c, src/search.c

Patch 8.1.0887

Problem: The 'l' flag in :substitute is sticky.
Solution: Reset the flag. (Dominique Pelle, closes #3925)
Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.1.0888

Problem: The a: dict is not immutable as documented.
Solution: Make the a:dict immutable, add a test. (Ozaki Kiichi, Yasuhiro Matsumoto, closes #3929)
Files: src/eval.c, src/userfunc.c, src/testdir/test_let.vim, src/testdir/test_listdict.vim

Patch 8.1.0889

Problem: MS-Windows: a channel write may hang.
Solution: Check for WriteFile() not writing anything. (Yasuhiro Matsumoto, closes #3920)
Files: src/channel.c, src/testdir/test_channel.vim, src/testdir/test_channel_pipe.py

Patch 8.1.0890

Problem: Pty allocation wrong if using file for out channel and using null for in channel and null for error channel.
Solution: Correct using use_file_for_out in condition. (Ozaki Kiichi, closes #3917)
Files: src/os_unix.c, src/testdir/test_channel.vim

Patch 8.1.0891

Problem: Substitute command insufficiently tested.
Solution: Add more test coverage. (Dominique Pelle)
Files: src/testdir/test_substitute.vim

Patch 8.1.0892

Problem: Failure when closing a window when location list is in use.
Solution: Handle the situation gracefully. Make sure memory for 'switchbuf' is not freed at the wrong time. (Yegappan Lakshmanan, closes #3928)
Files: src/eval.c, src/evalfunc.c, src/proto/window.pro, src/quickfix.c, src/testdir/test_quickfix.vim, src/window.c

Patch 8.1.0893

Problem: Terminal test is a bit flaky.
Solution: Add test_terminal_no_cmd() to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0894

Problem: MS-Windows: resolve() does not return a reparse point.
Solution: Improve resolve(). (Yasuhiro Matsumoto, closes #3896)
Files: runtime/doc/eval.txt, src/buffer.c, src/evalfunc.c, src/os_mswin.c, src/proto/os_mswin.pro, src/testdir/test_functions.vim

Patch 8.1.0895 (after 8.1.0879)

Problem: MS-Windows: dealing with temp name encoding not quite right.

Solution: Use more wide functions. (Ken Takata, closes #3921)
Files: src/fileio.c

Patch 8.1.0896

Problem: Tests for restricted mode not run for MS-Windows GUI.
Solution: Make tests also work in MS-Windows GUI.
Files: src/testdir/test_restricted.vim

Patch 8.1.0897

Problem: Can modify a:000 when using a reference.
Solution: Make check for locked variable stricter. (Ozaki Kiichi, closes #3930)
Files: src/dict.c, src/eval.c, src/evalfunc.c, src/proto/eval.pro, src/testdir/test_channel.vim, src/testdir/test_let.vim, src/userfunc.c

Patch 8.1.0898

Problem: A messed up rgb.txt can crash Vim. (Pavel Cheremushkin)
Solution: Limit to 10000 entries. Also don't retry many times when the file cannot be read.
Files: src/term.c

Patch 8.1.0899

Problem: No need to check restricted mode for setwinvar().
Solution: Remove check_restricted().
Files: src/eval.c

Patch 8.1.0900

Problem: ConPTY may crash with 32-bit build.
Solution: Fix function declarations. (Ken Takata, closes #3943)
Files: src/terminal.c

Patch 8.1.0901

Problem: Index in getjumplist() may be wrong. (Epehien)
Solution: Call cleanup_jumplist() earlier. (Yegappan Lakshmanan, closes #3942)
Files: src/evalfunc.c, src/testdir/test_jumplist.vim

Patch 8.1.0902

Problem: Incomplete set of assignment operators.
Solution: Add /=, *= and %=. (Ozaki Kiichi, closes #3931)
Files: runtime/doc/eval.txt src/eval.c src/testdir/test_vimscript.vim

Patch 8.1.0903

Problem: Struct uses more bytes than needed.
Solution: Reorder members of regitem_S. (Dominique Pelle, closes #3936)
Files: src/regexp.c

Patch 8.1.0904

Problem: USE_LONG_FNAME never defined.
Solution: Remove using USE_LONG_FNAME. (Ken Takata, closes #3938)
Files: src/buffer.c, src/ex_cmds.c, src/fileio.c

Patch 8.1.0905

Problem: Complicated regexp causes a crash. (Kuang-che Wu)
Solution: Limit the recursiveness of addstate(). (closes #3941)
Files: src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0906

Problem: Using clumsy way to get console window handle.
Solution: Use GetConsoleWindow(). (Ken Takata, closes #3940)
Files: src/os_mswin.c

Patch 8.1.0907

Problem: CI tests on AppVeyor are failing.
Solution: Reduce the recursiveness limit for regexp.
Files: src/regexp_nfa.c

Patch 8.1.0908

Problem: Can't handle large value for %{nr}v in regexp. (Kuang-che Wu)
Solution: Give an error if the value is too large. (closes #3948)
Files: src/regexp_nfa.c

Patch 8.1.0909

Problem: MS-Windows: using ConPTY even though it is not stable.
Solution: When ConPTY version is unstable, prefer using winpty. (Ken Takata, closes #3949)
Files: runtime/doc/options.txt, src/os_win32.c, src/proto/os_win32.pro, src/terminal.c

Patch 8.1.0910

Problem: Crash with tricky search pattern. (Kuang-che Wu)
Solution: Check for running out of memory. (closes #3950)
Files: src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0911

Problem: Tag line with Ex command cannot have extra fields.
Solution: Recognize |;" as the end of the command. (closes #2402)
Files: runtime/doc/tagsrch.txt, src/tag.c, src/testdir/test_taglist.vim

Patch 8.1.0912

Problem: MS-Windows: warning for signed/unsigned.
Solution: Add type cast. (Nobuhiro Takasaki, closes #3945)
Files: src/terminal.c

Patch 8.1.0913

Problem: CI crashes when running out of memory.
Solution: Apply '**maxmempattern**' also to new regexp engine.
Files: src/regexp_nfa.c

Patch 8.1.0914

Problem: Code related to findfile() is spread out.
Solution: Put findfile() related code into a new source file. (Yegappan Lakshmanan, closes #3934)
Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/README.txt, src/findfile.c,

src/misc1.c, src/misc2.c, src/proto.h, src/proto/findfile.pro,
src/proto/misc1.pro, src/proto/misc2.pro, src/proto/window.pro,
src/window.c

Patch 8.1.0915

Problem: fsync() may not work properly on Mac.
Solution: Use fcntl() with F_FULLFSYNC. (suggested by Justin M. Keyes)
Files: src/fileio.c, src/proto/fileio.pro, src/evalfunc.c, src/memfile.c

Patch 8.1.0916

Problem: With Python 3.7 "find_module" is not made available.
Solution: Also add "find_module" with Python 3.7. (Joel Frederico,
closes #3954)
Files: src/if_py_both.h

Patch 8.1.0917

Problem: Double free when running out of memory.
Solution: Remove one free. (Ken Takata, closes #3955)
Files: src/userfunc.c

Patch 8.1.0918

Problem: MS-Windows: startup messages are not converted.
Solution: Convert messages when the current codepage differs from
'**encoding**'. (Yasuhiro Matsumoto, closes #3914)
Files: src/message.c, src/os_mswin.c, src/vim.h

Patch 8.1.0919

Problem: Compiler warnings.
Solution: Add type casts. (Mike Williams)
Files: src/message.c, src/regexp_nfa.c

Patch 8.1.0920

Problem: In Terminal-Normal mode job output messes up the window.
Solution: Postpone scrolling and updating the buffer when in Terminal-Normal
mode.
Files: src/terminal.c, src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_01.dump,
src/testdir/dumps/Test_terminal_02.dump,
src/testdir/dumps/Test_terminal_03.dump

Patch 8.1.0921

Problem: Terminal test sometimes fails; using memory after free.
Solution: Free memory a bit later. Add test to cover this. Disable flaky
screenshot test. (closes #3956)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0922

Problem: Terminal scrollbar test is flaky.
Solution: Wait a bit before running the tail command.
Files: src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_01.dump,
src/testdir/dumps/Test_terminal_02.dump,
src/testdir/dumps/Test_terminal_03.dump

Patch 8.1.0923

Problem: Terminal dump diff swap does not update file names.
Solution: Also swap the file name. Add a test.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.0924

Problem: Terminal scrollbar test still flaky.
Solution: Wait a bit longer before running the tail command.
Files: src/testdir/test_terminal.vim

Patch 8.1.0925

Problem: Terminal scrollbar test still still flaky.
Solution: Explicitly set the shell. Disable ruler. (Ozaki Kiichi, closes #3966)
Files: src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_01.dump,
src/testdir/dumps/Test_terminal_02.dump,
src/testdir/dumps/Test_terminal_03.dump

Patch 8.1.0926

Problem: No test for :wnext, :wNext and :wprevious.
Solution: Add a test. (Dominique Pelle, closes #3963)
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_wnext.vim

Patch 8.1.0927

Problem: USE_CR is never defined.
Solution: Remove usage of USE_CR. (Ken Takata, closes #3958)
Files: runtime/doc/options.txt, src/diff.c, src/evalfunc.c,
src/ex_cmds2.c, src/fileio.c, src/message.c, src/ops.c,
src/option.h, src/proto/ex_cmds2.pro, src/proto/fileio.pro,
src/tag.c

Patch 8.1.0928 (after 8.1.0927)

Problem: Stray log function call.
Solution: Remove the log function call.
Files: src/ex_cmds2.c

Patch 8.1.0929

Problem: No error when requesting ConPTY but it's not available.
Solution: Add an error message. (Hirohito Higashi, closes #3967)
Files: runtime/doc/terminal.txt, src/terminal.c

Patch 8.1.0930

Problem: Typo in Makefile.
Solution: Change ABORT_CLFAGS to ABORT_CFLAGS. (Kuang-che Wu, closes #3977)
Files: src/Makefile

Patch 8.1.0931

Problem: vtp_working included in GUI build but unused.
Solution: Adjust #ifdefs. (Ken Takata, closes #3971)
Files: src/os_win32.c

Patch 8.1.0932

Problem: Farsi support is outdated and unused.
Solution: Delete the Farsi support.
Files: Filelist, src/farsi.c, src/proto/farsi.pro, src/farsi.h, src/edit.c, src/main.c, src/normal.c, src/option.c, src/getchar.c, src/ex_cmds.c, src/search.c, src/ex_getln.c, src/charset.c, src/evalfunc.c, src/screen.c, src/window.c, src/globals.h, src/proto.h, farsi/README.txt, src/structs.h, farsi/fonts/DOS/far-a01.com, farsi/fonts/Sun0s/far-a01.fb, farsi/fonts/UNIXs/far-a01.f16, farsi/fonts/UNIXs/far-a01.pcf.gz, farsi/fonts/UNIXs/far-a01.pcf.Z, farsi/fonts/WINDOWS/far-a01.fon, src/Makefile, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/configure.ac, src/auto/configure, src/config.h.in, src/testdir/test_farsi.vim, src/version.c, src/testdir/Make_all.mak, runtime/doc/options.txt, runtime/doc/starting.txt, runtime/doc/quickref.txt, runtime/doc/farsi.txt

Patch 8.1.0933

Problem: When using VTP scroll region isn't used properly.
Solution: Make better use of the scroll region. (Nobuhiro Takasaki, closes #3974)
Files: src/os_win32.c, src/term.c

Patch 8.1.0934

Problem: Invalid memory access in search pattern. (Kuang-che Wu)
Solution: Check for incomplete equivalence class. (closes #3970)
Files: src/regexp.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0935

Problem: Old regexp engine may use invalid buffer for **'iskeyword'** or uninitialized buffer pointer. (Kuang-che Wu)
Solution: Set rex.reg_buf when compiling the pattern. (closes #3972)
Files: src/regexp.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0936

Problem: May leak memory when using **'vartabstop'**. (Kuang-che Wu)
Solution: Fix handling allocated memory for **'vartabstop'**. (closes #3976)
Files: src/option.c, src/buffer.c

Patch 8.1.0937

Problem: Invalid memory access in search pattern. (Kuang-che Wu)
Solution: Check for incomplete collation element. (Dominique Pelle, closes #3985)
Files: src/regexp.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0938

Problem: Background color is wrong in MS-Windows console when not using VTP.
Solution: Use g_attrCurrent. (Nobuhiro Takasaki, closes #3987)
Files: src/os_win32.c

Patch 8.1.0939

Problem: No completion for sign group names.

Solution: Add completion for sign group names and buffer names. (Yegappan Lakshmanan, closes #3980)
Files: src/sign.c, src/testdir/test_signs.vim

Patch 8.1.0940

Problem: MS-Windows console resizing not handled properly.
Solution: Handle resizing the console better. (Nobuhiro Takasaki, Ken Takata, closes #3968, closes #3611)
Files: src/ex_docmd.c, src/normal.c, src/os_win32.c, src/proto/os_win32.pro

Patch 8.1.0941

Problem: Macros for MS-Windows are inconsistent, using "32", "3264" and others.
Solution: Use MSWIN for all MS-Windows builds. Use FEAT_GUI_MSWIN for the GUI build. (Hirohito Higashi, closes #3932)
Files: src/GvimExt/gvimext.h, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_ivc.mak, src/Make_mvc.mak, src/beval.h, src/blowfish.c, src/channel.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/feature.h, src/fileio.c, src/getchar.c, src/glbl_ime.cpp, src/globals.h, src/gui.c, src/gui.h, src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_f.c, src/gui_gtk_x11.c, src/if_cscope.c, src/if_cscope.h, src/if_lua.c, src/if_mzsch.c, src/if_ole.cpp, src/if_perl.xs, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/macros.h, src/main.c, src/mbyte.c, src/memfile.c, src/memline.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/nbdebug.c, src/netbeans.c, src/normal.c, src/option.c, src/option.h, src/os_mswin.c, src/os_unix.c, src/os_w32exe.c, src/os_win32.c, src/os_win32.h, src/proto.h, src/screen.c, src/search.c, src/structs.h, src/syntax.c, src/term.c, src/terminal.c, src/ui.c, src/undo.c, src/version.c, src/vim.h, src/vim.rc, src/winclip.c

Patch 8.1.0942

Problem: Options window still checks for the multi_byte feature.
Solution: Remove the unnecessary check. (Dominique Pelle, closes #3990)
Files: runtime/optwin.vim

Patch 8.1.0943

Problem: Still a trace of Farsi support.
Solution: Remove defining macros.
Files: src/feature.h

Patch 8.1.0944

Problem: Format of nbdbg() arguments is not checked.
Solution: Add format attribute. Fix reported problems. (Dominique Pelle, closes #3992)
Files: src/nbdebug.h, src/netbeans.c

Patch 8.1.0945

Problem: Internal error when using pattern with NL in the range.
Solution: Use an actual newline for the range. (closes #3989) Also fix error message. (Dominique Pelle)

Files: src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.1.0946

Problem: Coveralls is not very useful.

Solution: Remove Coveralls badge, add badge for packages.

Files: README.md

Patch 8.1.0947

Problem: Using MSWIN before it is defined. (Cesar Romani)

Solution: Move the block that uses MSWIN to below including vim.h. (Ken Takata)

Files: src/if_ruby.c

Patch 8.1.0948

Problem: When built without +eval "Vim --clean" produces errors. (James McCoy)

Solution: Do not enable filetype detection.

Files: runtime/defaults.vim

Patch 8.1.0949

Problem: MS-Windows defines GUI macros different than other systems.

Solution: Swap FEAT_GUI and FEAT_GUI_MSWIN. (Hirohito Higashi, closes #3996)

Files: src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_ivc.mak, src/Make_mvc.mak, src/if_ole.cpp, src/vim.h, src/vim.rc

Patch 8.1.0950

Problem: Using :python sets 'pyxversion' even when not executed.

Solution: Check the "skip" flag. (Shane Harper, closes #3995)

Files: src/if_python.c, src/if_python3.c, src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.1.0951

Problem: Using WIN64 even though it is never defined.

Solution: Only use _WIN64. (Ken Takata, closes #3997)

Files: src/evalfunc.c

Patch 8.1.0952

Problem: Compilation warnings when building the MS-Windows installer.

Solution: Fix buffer sizes. (Yasuhiro Matsumoto, closes #3999)

Files: src/dosinst.c, src/dosinst.h, src/uninstal.c

Patch 8.1.0953

Problem: A very long file is truncated at 2³¹ lines.

Solution: Use LONG_MAX for MAXLNUM. (Dominique Pelle, closes #4011)

Files: src/vim.h

Patch 8.1.0954

Problem: Arguments of semsg() and siemsg() are not checked.

Solution: Add function prototype with __attribute__.

Files: src/message.c, src/proto/message.pro, src/proto.h

Patch 8.1.0955

Problem: Matchit autoload directory not in installer. (Chris Morgan)

Solution: Adjust the NSIS script. (Christian Brabandt, closes #4006)

Files: nsis/gvim.nsi

Patch 8.1.0956

Problem: Using context:0 in **'diffopt'** does not work well.

Solution: Make zero context do the same as one line context. (closes #4005)

Files: src/diff.c, src/testdir/test_diffmode.vim,
 src/testdir/dumps/Test_diff_06.0.dump,
 src/testdir/dumps/Test_diff_06.1.dump,
 src/testdir/dumps/Test_diff_06.2.dump

Patch 8.1.0957 (after 8.1.0915)

Problem: Mac: fsync fails on network share.

Solution: Check for ENOTSUP. (Yee Cheng Chin, closes #4016)

Files: src/fileio.c

Patch 8.1.0958

Problem: Compiling weird regexp pattern is very slow.

Solution: When reallocating post list increase size by 50%. (Kuang-che Wu,
 closes #4012) Make assert_inrange() accept float values.

Files: src/regexp_nfa.c, src/eval.c, src/testdir/test_regexp_latin.vim,
 src/testdir/test_assert.vim

Patch 8.1.0959

Problem: Sorting large numbers is not tested and does not work properly.

Solution: Add test. Fix comparing lines with and without a number.
 (Dominique Pelle, closes #4017)

Files: src/ex_cmds.c, src/testdir/test_sort.vim

Patch 8.1.0960

Problem: When using ConPTY garbage collection has undefined behavior.

Solution: Free the channel in a better way. (Nobuhiro Takasaki, closes #4020)

Files: src/channel.c

Patch 8.1.0961 (after 8.1.0957)

Problem: Mac: fsync may fail sometimes.

Solution: Do not check errno. (Yee Cheng Chin, closes #4025)

Files: src/fileio.c

Patch 8.1.0962

Problem: Building with MinGW and static libs doesn't work. (Salman Halim)

Solution: Add -lgcc. (Ken Takata)

Files: src/Make_cyg_ming.mak

Patch 8.1.0963

Problem: Illegal memory access when using **'incsearch'**.

Solution: Reset highlight_match when changing text. (closes #4022)

Files: src/testdir/test_search.vim, src/misc1.c,
 src/testdir/dumps/Test_incsearch_change_01.dump

Patch 8.1.0964

Problem: Cannot see in CI why a screenshot test failed.

Solution: Add info about the failure.

Files: src/testdir/screendump.vim

Patch 8.1.0965

Problem: Search test fails.
Solution: Wait a bit longer for the 'ambiwidth' redraw.
Files: src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_change_01.dump

Patch 8.1.0966

Problem: One terminal test is flaky.
Solution: Add to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.0967

Problem: Stray dependency in test Makefile.
Solution: Remove it. (Masato Nishihata, closes #4018)
Files: src/testdir/Makefile

Patch 8.1.0968

Problem: Crash when using search pattern \%Uffffffc23.
Solution: Limit character to INT_MAX. (closes #4009)
Files: src/regexp_nfa.c, src/testdir/test_search.vim

Patch 8.1.0969

Problem: Message written during startup is truncated.
Solution: Restore message after truncating. (closes #3969) Add a test.
(Yasuhiro Matsumoto)
Files: src/message.c, src/testdir/test_startup.vim

Patch 8.1.0970

Problem: Text properties test fails when 'encoding' is not utf-8.
Solution: Compare with original value of 'encoding'. (Christian Brabandt,
closes #3986)
Files: src/testdir/runtest.vim, src/testdir/test_textprop.vim

Patch 8.1.0971

Problem: Failure for selecting quoted text object moves cursor.
Solution: Restore the Visual selection on failure. (Christian Brabandt,
closes #4024)
Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.1.0972

Problem: Cannot switch from terminal window to next tabpage.
Solution: Make **CTRL-W** gt move to next tabpage.
Files: src/window.c, src/testdir/test_terminal.vim,
runtime/doc/terminal.txt

Patch 8.1.0973

Problem: Pattern with syntax error gives three error messages. (Kuang-che
Wu)
Solution: Remove outdated internal error. Don't fall back to other engine
after an error.(closes #4035)
Files: src/regexp_nfa.c, src/testdir/test_search.vim, src/regexp.c

Patch 8.1.0974

Problem: Cannot switch from terminal window to previous tabpage.

Solution: Make **CTRL-W** gT move to previous tabpage.
Files: src/window.c, src/testdir/test_terminal.vim,
runtime/doc/terminal.txt

Patch 8.1.0975

Problem: Using STRNCPY() wrongly. Warning for uninitialized variable.
Solution: Use mch_memmove(). Initialize variable. (Yasuhiro Matsumoto,
closes #3979)
Files: src/screen.c, src/textprop.c

Patch 8.1.0976

Problem: Dosinstall still has buffer overflow problems.
Solution: Adjust buffer sizes. (Yasuhiro Matsumoto, closes #4002)
Files: src/dosinst.c, src/dosinst.h, src/uninstal.c

Patch 8.1.0977

Problem: Blob not tested with Ruby.
Solution: Add more test coverage. Fixes a crash. (Dominique Pelle,
closes #4036)
Files: src/if_ruby.c, src/testdir/test_ruby.vim

Patch 8.1.0978

Problem: Blob not tested with Perl.
Solution: Add more test coverage. Fixes a crash. (Dominique Pelle,
closes #4037)
Files: src/if_perl.c, src/testdir/test_ruby.vim

Patch 8.1.0979

Problem: Compiler warning for unused functions. (Yasuhiro Matsumoto)
Solution: Adjust #ifdef.
Files: src/screen.c

Patch 8.1.0980

Problem: extend() insufficiently tested.
Solution: Add more tests. (Dominique Pelle, closes #4040)
Files: src/testdir/test_listdict.vim

Patch 8.1.0981

Problem: Pasting in terminal insufficiently tested.
Solution: Add more tests. (Dominique Pelle, closes #4040)
Files: src/testdir/test_terminal.vim

Patch 8.1.0982

Problem: update_cursor() called twice in :shell.
Solution: Remove one of the calls. (Yasuhiro Matsumoto, closes #4039)
Files: src/terminal.c

Patch 8.1.0983

Problem: Checking __CYGWIN32__ unnecessarily.
Solution: Remove the checks. (Ken Takata)
Files: src/evalfunc.c, src/os_unix.c, src/os_win32.c

Patch 8.1.0984

Problem: Unnecessary #ifdefs.

Solution: Remove the #ifdefs. (Ken Takata)
Files: src/winclip.c

Patch 8.1.0985

Problem: Crash with large number in regexp. (Kuang-che Wu)
Solution: Check for long becoming negative int. (closes #4042)
Files: src/regexp.c, src/testdir/test_search.vim

Patch 8.1.0986

Problem: rename() is not properly tested.
Solution: Add tests. (Dominique Pelle, closes #4061)
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_rename.vim

Patch 8.1.0987

Problem: Unnecessary condition in #ifdef.
Solution: Remove using CYGWIN32. (Ken Takata)
Files: src/os_unix.h, src/xxd/xd.c

Patch 8.1.0988

Problem: Deleting a location list buffer breaks location list window
functionality.
Solution: (Yegappan Lakshmanan, closes #4056)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.0989

Problem: Various small code ugliness.
Solution: Remove pointless NULL checks. Fix function calls. Fix typos.
(Dominique Pelle, closes #4060)
Files: src/buffer.c, src/crypt.c, src/evalfunc.c, src/ex_cmds2.c,
src/globals.h, src/gui_gtk_f.c, src/gui_gtk_x11.c, src/gui_mac.c,
src/ops.c, src/option.h, src/os_unix.c, src/os_win32.c,
src/popupmenu.c, src/regexp.c, src/ui.c, src/version.c

Patch 8.1.0990

Problem: Floating point exception with "%= 0" and "/= 0".
Solution: Avoid dividing by zero. (Dominique Pelle, closes #4058)
Files: src/eval.c, src/testdir/test_vimscript.vim

Patch 8.1.0991

Problem: Cannot build with FEAT_EVAL defined and FEAT_SEARCH_EXTRA
undefined, and with FEAT_DIFF defined and FEAT_EVAL undefined.
Solution: Add a couple of #ifdefs. (closes #4067)
Files: src/diff.c, src/search.c

Patch 8.1.0992

Problem: A :normal command while executing a register resets the
reg_executing() result.
Solution: Save and restore reg_executing. (closes #4066)
Files: src/ex_docmd.c, src/structs.h, src/testdir/test_functions.vim

Patch 8.1.0993

Problem: ch_read() may return garbage if terminating NL is missing.
Solution: Add terminating NUL. (Ozaki Kiichi, closes #4065)

Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.1.0994

Problem: Relative cursor position is not calculated correctly.

Solution: Always set topline, also when window is one line only.
(Robert Webb) Add more info to getwininfo() for testing.

Files: src/window.c, src/evalfunc.c, runtime/doc/eval.txt,
src/testdir/test_window_cmd.vim

Patch 8.1.0995

Problem: A getchar() call while executing a register resets the
reg_executing() result.

Solution: Save and restore reg_executing. (closes #4066)

Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.1.0996 (after 8.1.0994)

Problem: A few screendump tests fail because of scrolling.

Solution: Update the screendumps.

Files: src/testdir/dumps/Test_incsearch_substitute_11.dump,
src/testdir/dumps/Test_incsearch_substitute_12.dump,
src/testdir/dumps/Test_incsearch_substitute_13.dump

Patch 8.1.0997

Problem: Using GUI colors in vim.exe when **'termguicolors'** is off.

Solution: Add condition for **'termguicolors'** set. (Ken Takata, closes #4078)

Files: src/os_win32.c

Patch 8.1.0998

Problem: getcurpos() unexpectedly changes "curswant".

Solution: Save and restore "curswant". (closes #4069)

Files: src/evalfunc.c, src/testdir/test_visual.vim

Patch 8.1.0999

Problem: Use register one too often and not properly tested.

Solution: Do not always use register one when specifying a register.
(closes #4085) Add more tests.

Files: src/ops.c, src/testdir/test_registers.vim

Patch 8.1.1000

Problem: Indenting is off.

Solution: Make indenting consistent and update comments. (Ozaki Kiichi,
closes #4079)

Files: src/getchar.c, src/ops.c

Patch 8.1.1001

Problem: Visual area not correct when using **'cursorline'**.

Solution: Update w_last_cursorline also in Visual mode. (Hirohito Higashi,
closes #4086)

Files: src/screen.c, src/testdir/test_highlight.vim,
src/testdir/dumps/Test_cursorline_with_visualmode_01.dump

Patch 8.1.1002

Problem: "gf" does not always work when URL has a port number. (Jakob
Schöttl)

Solution: When a URL is recognized also accept ":". (closes #4082)
Files: src/findfile.c, src/testdir/test_gf.vim

Patch 8.1.1003

Problem: Playing back recorded key sequence mistakes key code.
Solution: Insert a <Nop> after the <Esc>. (closes #4068)
Files: src/getchar.c, src/testdir/test_registers.vim

Patch 8.1.1004

Problem: Function "luaV_setref()" not covered with tests.
Solution: Add a test. (Dominique Pelle, closes #4089)
Files: src/testdir/test_lua.vim

Patch 8.1.1005 (after 8.1.1003)

Problem: Test fails because t_F2 is not set.
Solution: Add try-catch.
Files: src/testdir/test_registers.vim

Patch 8.1.1006

Problem: Repeated code in quickfix support.
Solution: Move code to functions. (Yegappan Lakshmanan, closes #4091)
Files: src/quickfix.c

Patch 8.1.1007

Problem: Using closure may consume a lot of memory.
Solution: unreference items that are no longer needed. Add a test. (Ozaki Kiichi, closes #3961)
Files: src/testdir/Make_all.mak, src/testdir/test_memory_usage.vim, src/userfunc.c

Patch 8.1.1008

Problem: MS-Windows: HAVE_STDINT_H only defined for non-debug version.
Solution: Move definition of HAVE_STDINT_H up. (Taro Muraoka, closes #4109)
Files: src/Make_mvc.mak

Patch 8.1.1009

Problem: MS-Windows: some text is not baseline aligned.
Solution: Use bottom alignment. (Taro Muraoka, closes #4116, closes #1520)
Files: src/gui_dwrite.cpp

Patch 8.1.1010

Problem: Lua interface leaks memory.
Solution: Clear typeval after copying it.
Files: src/if_lua.c

Patch 8.1.1011

Problem: Indent from autoindent not removed from blank line. (Daniel Hahler)
Solution: Do not reset did_ai when text follows. (closes #4119)
Files: src/misc1.c, src/testdir/test_edit.vim

Patch 8.1.1012

Problem: Memory leak with E461.
Solution: Clear the typeval. (Dominique Pelle, closes #4111)
Files: src/eval.c

Patch 8.1.1013

Problem: MS-Windows: Scrolling fails when dividing the screen.
Solution: Position the cursor before calling ScrollConsoleScreenBuffer().
(Nobuhiro Takasaki, closes #4115)
Files: src/os_win32.c

Patch 8.1.1014

Problem: MS-Windows: /analyze only defined for non-debug version.
Solution: Move adding of /analyze up. (Ken Takata, closes #4114)
Files: src/Make_mvc.mak

Patch 8.1.1015

Problem: Quickfix buffer shows up in list, can't get buffer number.
Solution: Make the quickfix buffer unlisted when the quickfix window is closed. get the quickfix buffer number with getqflist().
(Yegappan Lakshmanan, closes #4113)
Files: runtime/doc/eval.txt, runtime/doc/quickfix.txt, src/quickfix.c, src/testdir/test_quickfix.vim, src/window.c

Patch 8.1.1016

Problem: MS-Windows: No color in shell when using "!" in **'guioptions'**.
Solution: Don't stop termcap when using a terminal window for the shell.
(Nobuhiro Takasaki, vim-jp, closes #4117)
Files: src/ex_cmds.c

Patch 8.1.1017

Problem: Off-by-one error in filetype detection.
Solution: Also check the last line of the file.
Files: runtime/autoload/dist/ft.vim

Patch 8.1.1018

Problem: Window cleared when entering Terminal-Normal twice. (Epeheien)
Solution: Don't cleanup scrollbar when there is no postponed scrollbar.
(Christian Brabandt, closes #4126)
Files: src/terminal.c

Patch 8.1.1019

Problem: Lua: may garbage collect function reference in use.
Solution: Keep the function name instead of the typeval. Make luaV_setref() handle funcref objects. (Ozaki Kiichi, closes #4127)
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.1.1020

Problem: Compiler warning for Python3 interface.
Solution: Add type cast. (Ozaki Kiichi, closes #4128, closes #4103)
Files: src/if_python3.c

Patch 8.1.1021

Problem: pyeval() and py3eval() leak memory.
Solution: Do not increase the reference count twice. (Ozaki Kiichi, closes #4129)
Files: src/if_python.c, src/if_python3.c

Patch 8.1.1022

Problem: May use NULL pointer when out of memory. (Coverity)
Solution: Check for blob_alloc() returning NULL.
Files: src/blob.c

Patch 8.1.1023

Problem: May use NULL pointer when indexing a blob. (Coverity)
Solution: Break out of loop after using index on blob
Files: src/eval.c

Patch 8.1.1024

Problem: Stray log calls in terminal code. (Christian Brabandt)
Solution: Remove the calls.
Files: src/terminal.c

Patch 8.1.1025

Problem: Checking NULL pointer after addition. (Coverity)
Solution: First check for NULL, then add the column.
Files: src/regexp.c

Patch 8.1.1026

Problem: Unused condition. (Coverity)
Solution: Remove the condition. Also remove unused #define.
Files: src/move.c

Patch 8.1.1027

Problem: Memory usage test sometimes fails.
Solution: Use 80% of before.last as the lower limit. (Christian Brabandt)
Files: src/testdir/test_memory_usage.vim

Patch 8.1.1028

Problem: MS-Windows: memory leak when creating terminal fails.
Solution: Free the command. (Ken Takata, closes #4138)
Files: src/os_win32.c

Patch 8.1.1029

Problem: DirectWrite doesn't take 'linespace' into account.
Solution: Include 'linespace' in the position. (Ken Takata, closes #4137)
Files: src/gui_dwrite.cpp, src/gui_w32.c

Patch 8.1.1030

Problem: Quickfix function arguments are inconsistent.
Solution: Pass a list pointer instead of info and index. (Yegappan Lakshmanan, closes #4135)
Files: src/quickfix.c

Patch 8.1.1031

Problem: Memory usage test may still fail.
Solution: Drop the unused min value. (Christian Brabandt)
Files: src/testdir/test_memory_usage.vim

Patch 8.1.1032

Problem: Warnings from clang static analyzer. (Yegappan Lakshmanan)
Solution: Fix relevant warnings.

Files: src/arabic.c, src/edit.c, src/eval.c, src/fileio.c, src/normal.c,
src/option.c, src/os_unix.c, src/regexp.c, src/screen.c,
src/channel.c, src/charset.c, src/message.c

Patch 8.1.1033

Problem: Memory usage test may still fail on some systems. (Elimar
Riesebieter)

Solution: Increase tolerance from 1% to 3%.

Files: src/testdir/test_memory_usage.vim

Patch 8.1.1034

Problem: Too many #ifdefs.

Solution: Merge FEAT_MOUSE_SGR into FEAT_MOUSE_XTERM / FEAT_MOUSE_TTY.

Files: src/evalfunc.c, src/misc2.c, src/os_unix.c, src/term.c,
src/version.c, src/feature.h

Patch 8.1.1035

Problem: prop_remove() second argument is not optional.

Solution: Fix argument count. Use "buf" instead of "curbuf". (closes #4147)

Files: src/evalfunc.c, src/testdir/test_textprop.vim, src/textprop.c

Patch 8.1.1036

Problem: Quickfix function arguments are inconsistent.

Solution: Pass a list pointer to more functions. (Yegappan Lakshmanan,
closes #4149)

Files: src/quickfix.c

Patch 8.1.1037

Problem: Memory usage test may still fail on some systems.

Solution: Increase tolerance from 3% to 20%.

Files: src/testdir/test_memory_usage.vim

Patch 8.1.1038

Problem: Arabic support excludes Farsi.

Solution: Add Farsi support to the Arabic support. (Ali Gholami Rudi,
Ameretat Reith)

Files: Filelist, src/arabic.c, src/arabic.h, src/globals.h, src/macros.h,
src/mbyte.c, src/proto/arabic.pro, src/proto/mbyte.pro,
src/Makefile, src/testdir/test_arabic.vim

Patch 8.1.1039

Problem: MS-Windows build fails.

Solution: Remove dependency on arabic.h

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms

Patch 8.1.1040

Problem: FEAT_TAG_ANYWHITE is not enabled in any build.

Solution: Remove the feature.

Files: src/feature.h, src/tag.c, src/evalfunc.c, src/version.c,
src/Make_vms.mms

Patch 8.1.1041

Problem: Test for Arabic no longer needed.

Solution: Remove the test for something that was intentionally left out.

Files: src/testdir/test_arabic.vim

Patch 8.1.1042

Problem: The paste test doesn't work properly in the Windows console.

Solution: Disable the test.

Files: src/testdir/test_paste.vim

Patch 8.1.1043

Problem: Lua interface does not support Blob.

Solution: Add support to Blob. (Ozaki Kiichi, closes #4151)

Files: runtime/doc/if_lua.txt, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.1.1044

Problem: No way to check the reference count of objects.

Solution: Add test_refcount(). (Ozaki Kiichi, closes #4124)

Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_vimscript.vim

Patch 8.1.1045

Problem: E315 ml_get error when using Python and hidden buffer.

Solution: Make sure the cursor position is valid. (Ben Jackson,
closes #4153, closes #4154)

Files: src/if_py_both.h, src/testdir/test_python2.vim,
src/testdir/test_python3.vim

Patch 8.1.1046

Problem: the "secure" variable is used inconsistently. (Justin M. Keyes)

Solution: Set it to one instead of incrementing.

Files: src/buffer.c, src/option.c

Patch 8.1.1047

Problem: WINCH signal is not tested.

Solution: Add a test. (Dominique Pelle, closes #4158)

Files: src/testdir/Make_all.mak, src/testdir/test_signals.vim

Patch 8.1.1048

Problem: Minor issues with tests.

Solution: Delete unused test OK file. Add missing entries in list of tests.
Fix readme file. (Masato Nishihata, closes #4160)

Files: src/testdir/test85.ok, src/testdir/Make_all.mak,
src/testdir/README.txt

Patch 8.1.1049

Problem: When user tries to exit with **CTRL-C** message is confusing.

Solution: Only mention ":qa!" when there is a changed buffer. (closes #4163)

Files: src/undo.c, src/proto/undo.pro, src/normal.c,
src/testdir/test_normal.vim

Patch 8.1.1050

Problem: Blank screen when DirectWrite failed.

Solution: Call redraw_later_clear() after recreating the Direct2D render
target. (Ken Takata, closes #4172)

Files: src/gui_dwrite.cpp

Patch 8.1.1051

Problem: Not all ways to switch terminal mode are tested.
Solution: Add more test cases.
Files: src/testdir/test_terminal.vim

Patch 8.1.1052

Problem: test for **CTRL-C** message sometimes fails
Solution: Make sure there are no changed buffers.
Files: src/testdir/test_normal.vim

Patch 8.1.1053

Problem: Warning for missing return statement. (Dominique Pelle)
Solution: Add return statement.
Files: src/undo.c

Patch 8.1.1054

Problem: Not checking return value of ga_grow(). (Coverity)
Solution: Only append when ga_grow() returns OK.
Files: src/if_lua.c

Patch 8.1.1055

Problem: **CTRL-G** U in Insert mode doesn't work to avoid splitting the undo sequence for shift-left and shift-right.
Solution: Also check dont_sync_undo for shifted cursor keys. (Christian Brabandt)
Files: src/edit.c, src/testdir/test_mapping.vim

Patch 8.1.1056

Problem: No eval function for Ruby.
Solution: Add rubyeval(). (Ozaki Kiichi, closes #4152)
Files: runtime/doc/eval.txt, runtime/doc/if_ruby.txt, src/evalfunc.c, src/if_ruby.c, src/proto/if_ruby.pro, src/testdir/test_ruby.vim

Patch 8.1.1057

Problem: Nsis config is too complicated.
Solution: Use "File /r" for the macros and pack directories. (Ken Takata, closes #4169)
Files: nsis/gvim.nsi

Patch 8.1.1058

Problem: Memory usage test may still fail on some systems.
Solution: Use 98% of the lower limit. (Christian Brabandt)
Files: src/testdir/test_memory_usage.vim

Patch 8.1.1059

Problem: MS-Windows: PlatformId() is called unnecessarily.
Solution: Remove calls to PlatformId(). (Ken Takata, closes #4170)
Files: src/os_win32.c

Patch 8.1.1060

Problem: MS-Windows: get_cmd_args() is no longer needed, get_cmd_argsW() is always used.
Solution: Remove get_cmd_args(). (Ken Takata, closes #4171)
Files: src/gui_w32.c, src/os_w32exe.c

Patch 8.1.1061

Problem: When substitute string throws error, substitute happens anyway.
Solution: Skip substitution when aborting. (closes #4161)
Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.1.1062

Problem: Quickfix code is repeated.
Solution: Define FOR_ALL_QFL_ITEMS(). Move some code to separate functions.
(Yegappan Lakshmanan, closes #4166)
Files: src/quickfix.c

Patch 8.1.1063

Problem: Insufficient testing for wildmenu completion.
Solution: Extend the test case. (Dominique Pelle, closes #4182)
Files: src/testdir/test_cmdline.vim

Patch 8.1.1064

Problem: No test for output conversion in the GTK GUI.
Solution: Add a simplistic test.
Files: src/testdir/test_gui.vim

Patch 8.1.1065

Problem: No test for using and deleting menu in the GUI.
Solution: Add a test.
Files: src/testdir/test_gui.vim

Patch 8.1.1066

Problem: VIMDLL isn't actually used.
Solution: Remove VIMDLL support.
Files: src/gui_w32.c, src/main.c, src/os_w32exe.c, src/Make_bc5.mak,
src/os_w32dll.c

Patch 8.1.1067

Problem: Issues added on github are unstructured.
Solution: Add a bug and feature request template. (Ken Takata, closes #4183)
Files: .github/ISSUE_TEMPLATE/feature_request.md,
.github/ISSUE_TEMPLATE/bug_report.md

Patch 8.1.1068

Problem: Cannot get all the information about current completion.
Solution: Add complete_info(). (Shougo, Hirohito Higashi, closes #4106)
Files: runtime/doc/eval.txt, runtime/doc/insert.txt,
runtime/doc/usr_41.txt, src/edit.c, src/evalfunc.c,
src/proto/edit.pro, src/testdir/test_popup.vim

Patch 8.1.1069

Problem: Source README file doesn't look nice on github.
Solution: Turn it into markdown, still readable as plain text.
(WenxuanHuang, closes #4141)
Files: src/README.txt, src/README.md, Filelist

Patch 8.1.1070

Problem: Issue templates are not good enough.

Solution: Rephrase to anticipate unexperienced users.
Files: .github/ISSUE_TEMPLATE/feature_request.md,
.github/ISSUE_TEMPLATE/bug_report.md

Patch 8.1.1071

Problem: Cannot get composing characters from the screen.
Solution: Add screenchars() and screenstring(). (partly by Ozaki Kiichi,
closes #4059)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/testdir/test_utf8.vim, src/testdir/view_util.vim

Patch 8.1.1072

Problem: Extending sign and foldcolumn below the text is confusing.
Solution: Let the sign and foldcolumn stop at the last text line, just like
the line number column. Also stop the command line window leader.
(Christian Brabandt, closes #3964)
Files: src/screen.c, src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_of_diff_01.dump,
src/testdir/dumps/Test_diff_01.dump,
src/testdir/dumps/Test_diff_02.dump,
src/testdir/dumps/Test_diff_03.dump,
src/testdir/dumps/Test_diff_04.dump,
src/testdir/dumps/Test_diff_05.dump,
src/testdir/dumps/Test_diff_06.dump,
src/testdir/dumps/Test_diff_06.0.dump,
src/testdir/dumps/Test_diff_06.1.dump,
src/testdir/dumps/Test_diff_06.2.dump,
src/testdir/dumps/Test_diff_10.dump,
src/testdir/dumps/Test_diff_11.dump,
src/testdir/dumps/Test_diff_12.dump,
src/testdir/dumps/Test_diff_13.dump,
src/testdir/dumps/Test_diff_14.dump,
src/testdir/dumps/Test_diff_15.dump,
src/testdir/dumps/Test_diff_16.dump,
src/testdir/dumps/Test_diff_17.dump,
src/testdir/dumps/Test_diff_18.dump,
src/testdir/dumps/Test_diff_19.dump,
src/testdir/dumps/Test_diff_20.dump,
src/testdir/dumps/Test_diff_with_cursorline_01.dump,
src/testdir/dumps/Test_diff_with_cursorline_02.dump,
src/testdir/dumps/Test_diff_with_cursorline_03.dump,
src/testdir/dumps/Test_folds_with_rnu_01.dump,
src/testdir/dumps/Test_folds_with_rnu_02.dump

Patch 8.1.1073

Problem: Space in number column is on wrong side with **'rightleft'** set.
Solution: Move the space to the text side. Add a test.
Files: src/screen.c, src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_of_diff_02.dump

Patch 8.1.1074

Problem: Python test doesn't wipe out hidden buffer.
Solution: Wipe out the buffer. (Ben Jackson, closes #4189)
Files: src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.1.1075

Problem: Function reference count wrong in Python code.
Solution: Use "0" instead of "N" for the arguments. (Ben Jackson, closes #4188)
Files: src/if_py_both.h

Patch 8.1.1076

Problem: File for Insert mode is much too big.
Solution: Split off the code for Insert completion. (Yegappan Lakshmanan, closes #4044)
Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/edit.c, src/evalfunc.c, src/globals.h, src/insexpand.c, src/misc2.c, src/proto.h, src/proto/edit.pro, src/proto/insexpand.pro, src/search.c, src/spell.c, src/structs.h, src/tag.c, src/vim.h

Patch 8.1.1077

Problem: reg_executing() is reset by calling input().
Solution: Implement a more generic way to save and restore reg_executing. (Ozaki Kiichi, closes #4192)
Files: src/evalfunc.c, src/ex_docmd.c, src/testdir/test_functions.vim

Patch 8.1.1078

Problem: When '**listchars**' is set a composing char on a space is wrong.
Solution: Separate handling a non-breaking space and a space. (Yasuhiro Matsumoto, closes #4046)
Files: src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.1079

Problem: No need for a separate ScreenLinesUtf8() test function.
Solution: Get the composing characters with ScreenLines().
Files: src/testdir/view_util.vim, src/testdir/test_listchars.vim, src/testdir/test_utf8.vim

Patch 8.1.1080

Problem: When a screendump test fails, moving the file is a hassle.
Solution: Instead of appending ".failed" to the file name, keep the same file name but put the screendump in the "failed" directory. Then the file name only needs to be typed once when moving a screendump.
Files: src/testdir/screendump.vim

Patch 8.1.1081

Problem: MS-Windows: cannot use fonts whose name cannot be represented in the current code page.
Solution: Use wide font functions. (Ken Takata, closes #4000)
Files: src/gui_w32.c, src/os_mswin.c, src/proto/gui_w32.pro, src/proto/os_mswin.pro

Patch 8.1.1082

Problem: "Conceal" match is mixed up with '**hlsearch**' match.

Solution: Check that a match is found, not a **'hlsearch'** item. (Andy Massimino, closes #4073)
Files: src/screen.c

Patch 8.1.1083

Problem: MS-Windows: hang when opening a file on network share.
Solution: Avoid using FindFirstFile(), use GetLongPathNameW(). (Ken Takata, closes #3923)
Files: src/os_win32.c

Patch 8.1.1084

Problem: Cannot delete a match from another window. (Paul Jolly)
Solution: Add window ID argument to matchdelete(), clearmatches(), getmatches() and setmatches(). (Andy Massimino, closes #4178)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_match.vim

Patch 8.1.1085

Problem: Compiler warning for possibly uninitialized variable. (Tony Mechelynck)
Solution: Make conditions more logical.
Files: src/arabic.c

Patch 8.1.1086

Problem: Too many curly braces.
Solution: Remove curly braces where they are not needed. (Hirohito Higashi, closes #3982)
Files: src/autocmd.c, src/buffer.c, src/encrypt_zip.c, src/dosinst.c, src/edit.c, src/insexpand.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/getchar.c, src/gui.c, src/gui_gtk.c, src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/if_mzsch.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/indent.c, src/libvterm/src/pen.c, src/macros.h, src/memline.c, src/menu.c, src/misc1.c, src/move.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_mswin.c, src/os_qnx.c, src/os_unix.c, src/os_win32.c, src/regexp_nfa.c, src/screen.c, src/spell.c, src/terminal.c

Patch 8.1.1087

Problem: tag stack is incorrect after **CTRL-T** and then :tag
Solution: Handle DT_TAG differently. (test by Andy Massimino, closes #3944, closes #4177)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.1.1088

Problem: Height of quickfix window not retained with vertical split.
Solution: Use frame_fixed_height() and frame_fixed_width(). (Hongbo Liu, closes #4013, closes #2998)
Files: src/testdir/test_winbuf_close.vim, src/window.c

Patch 8.1.1089

Problem: Tutor does not check \$LC_MESSAGES.
Solution: Let \$LC_MESSAGES overrule \$LANG. (Miklos Vajna, closes #4112)
Files: runtime/tutor/tutor.vim

Patch 8.1.1090

Problem: MS-Windows: modify_fname() has problems with some **'encoding'**.
Solution: Use GetLongPathNameW() instead of GetLongPathName(). (Ken Takata, closes #4007)
Files: src/eval.c

Patch 8.1.1091

Problem: MS-Windows: cannot use multibyte chars in environment var.
Solution: Use the wide API. (Ken Takata, closes #4008)
Files: src/misc1.c, src/testdir/test_let.vim

Patch 8.1.1092

Problem: Setting **'guifont'** when maximized resizes the Vim window. When **'guioptions'** contains "k" gvim may open with a tiny window.
Solution: Avoid un-maximizing when setting **'guifont'**. (Yee Cheng Chin, closes #3808)
Files: src/gui.c

Patch 8.1.1093

Problem: Support for outdated tags format slows down tag parsing.
Solution: Remove FEAT_TAG_OLDSTATIC.
Files: runtime/doc/tagsrch.txt, src/feature.h, src/tag.c, src/version.c

Patch 8.1.1094

Problem: Long line in tags file causes error.
Solution: Check for overlong line earlier. (Andy Massimino, closes #4051, closes #4084)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.1.1095

Problem: MS-Windows: executable() fails on very long filename.
Solution: Use much bigger buffer. (Ken Takata, closes #4015)
Files: src/os_win32.c, src/testdir/test_functions.vim

Patch 8.1.1096

Problem: MS-Windows: cannot distinguish BS and **CTRL-H**.
Solution: Add code for VK_BACK. (Linwei, closes #1833)
Files: src/term.c, src/os_win32.c

Patch 8.1.1097 (after 8.1.1092)

Problem: Motif build fails. (Paul Jolly)
Solution: Only use gui_mch_maximized() for MS-Windows. (closes #4194)
Files: src/gui.c

Patch 8.1.1098

Problem: Quickfix code duplication.
Solution: Refactor the qf_init_ext() function. (Yegappan Lakshmanan, closes #4193)
Files: src/README.md, src/quickfix.c

Patch 8.1.1099

Problem: The do_tag() function is too long.
Solution: Factor parts out to separate functions. Move simplify_filename() to a file where it fits better. (Andy Massimino, closes #4195)

Files: src/tag.c, src/proto/tag.pro, src/findfile.c,
src/proto/findfile.pro

Patch 8.1.1100

Problem: Tag file without trailing newline no longer works. (Marco Hinz)
Solution: Don't expect a newline at the end of the file. (closes #4200)
Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.1.1101

Problem: Signals test may fail in the GUI.
Solution: Skip the test for the GUI. (Yee Checng Chin, closes #4202)
Files: src/testdir/test_signals.vim

Patch 8.1.1102

Problem: Win32 exe file contains unused code.
Solution: Remove unused #ifdefs and code. (Ken Takata, closes #4198)
Files: src/os_w32exe.c

Patch 8.1.1103

Problem: MS-Windows: old API calls are no longer needed.
Solution: Always use the wide functions. (Ken Takata, closes #4199)
Files: src/glbl_ime.cpp, src/globals.h, src/gui_w32.c, src/misc1.c,
src/os_mswin.c, src/os_win32.c, src/vim.h,

Patch 8.1.1104

Problem: MS-Windows: not all environment variables can be used.
Solution: Use the wide version of WinMain() and main(). (Ken Takata,
closes #4206)
Files: src/Make_cyg.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak,
src/main.c, src/os_w32exe.c

Patch 8.1.1105

Problem: Long escape sequences may be split up.
Solution: Assume escape sequences can be up to 80 bytes long. (Nobuhiro
Takasaki, closes #4196)
Files: src/term.c

Patch 8.1.1106

Problem: No test for **'writedelay'**.
Solution: Add a test.
Files: src/testdir/test_options.vim

Patch 8.1.1107

Problem: No test for **'visualbell'**.
Solution: Add a test.
Files: src/testdir/test_options.vim

Patch 8.1.1108

Problem: Test for **'visualbell'** doesn't work.
Solution: Make **'belloff'** empty.
Files: src/testdir/test_options.vim

Patch 8.1.1109

Problem: Deleted file still in list of distributed files.

Solution: Remove the src/os_w32dll.c entry.
Files: Filelist

Patch 8.1.1110

Problem: Composing chars on space wrong when '**listchars**' is set.
Solution: Do not use "space" and "nbsp" entries of '**listchars**' when there is a composing character. (Yee Cheng Chin, closes #4197)
Files: src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.1111

Problem: It is not easy to check for infinity.
Solution: Add isinf(). (Ozaki Kiichi, closes #3787)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_float_func.vim

Patch 8.1.1112

Problem: Duplicate code in quickfix file.
Solution: Move code into functions. (Yegappan Lakshmanan, closes #4207)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.1113

Problem: Making an autocommand trigger once is not so easy.
Solution: Add the ++once argument. Also add ++nested as an alias for "nested". (Justin M. Keyes, closes #4100)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/testdir/test_autocmd.vim, src/globals.h

Patch 8.1.1114

Problem: Confusing overloaded operator "." for string concatenation.
Solution: Add ".." for string concatenation. Also "let a .= b".
Files: src/eval.c, src/testdir/test_eval_stuff.vim, runtime/doc/eval.txt

Patch 8.1.1115

Problem: Cannot build with older C compiler.
Solution: Move variable declaration to start of block.
Files: src/autocmd.c

Patch 8.1.1116

Problem: Cannot enforce a Vim script style.
Solution: Add the :scriptversion command. (idea by Yasuhiro Matsumoto, closes #3857)
Files: runtime/doc/repeat.txt, runtime/doc/eval.txt, src/eval.c, src/ex_cmds.h, src/evalfunc.c, src/ex_cmds2.c, src/proto/ex_cmds2.pro, src/structs.h, src/buffer.c, src/main.c, src/option.c, src/ex_cmdidxs.h, src/testdir/test_eval_stuff.vim

Patch 8.1.1117

Problem: Build failure without the +eval feature.
Solution: Add #ifdef.
Files: src/ex_cmds2.c

Patch 8.1.1118

Problem: A couple of conditions are hard to understand.
Solution: Split the conditions into pieces. (Ozaki Kiichi, closes #3879)

Files: src/getchar.c, src/os_unix.c

Patch 8.1.1119

Problem: No support for Windows on ARM64.

Solution: Add ARM64 support (Leendert van Doorn)

Files: src/GvimExt/Makefile, src/Make_mvc.mak, src/dosinst.c,
src/xpm/arm64/lib-vc14/libXpm.lib, Filelist, src/INSTALLpc.txt

Patch 8.1.1120

Problem: Cannot easily get directory entry matches.

Solution: Add the readdir() function. (Yasuhiro Matsumoto, closes #2439)

Files: runtime/doc/eval.txt, src/eval.c, src/evalfunc.c, src/misc1.c,
src/proto/eval.pro, src/testdir/test_functions.vim

Patch 8.1.1121

Problem: Test for term_gettitle() was disabled.

Solution: Enable the test and bail out only when it doesn't work. (Dominique Pelle, closes #3776)

Files: src/testdir/test_terminal.vim

Patch 8.1.1122

Problem: char2nr() does not handle composing characters.

Solution: Add str2list() and list2str(). (Ozaki Kiichi, closes #4190)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/testdir/test_utf8.vim

Patch 8.1.1123

Problem: No way to avoid filtering for autocomplete function, causing flickering of the popup menu.

Solution: Add the "equal" field to complete items. (closes #3887)

Files: runtime/doc/insert.txt, src/insexpand.c,
src/testdir/test_popup.vim

Patch 8.1.1124

Problem: Insert completion flags are mixed up.

Solution: Clean up flags use of ins_compl_add() and cp_flags.

Files: src/insexpand.c, src/proto/insexpand.pro, src/search.c, src/spell.c

Patch 8.1.1125

Problem: Libvterm does not handle the window position report.

Solution: Let libvterm call the fallback CSI handler when not handling CSI sequence. Handle the window position report in Vim.

Files: src/libvterm/src/state.c, src/terminal.c, src/ui.c,
src/proto/ui.pro, src/evalfunc.c, src/testdir/test_terminal.vim

Patch 8.1.1126

Problem: Build failure with +terminal but without tgetent.

Solution: Adjust #ifdef.

Files: src/ui.c

Patch 8.1.1127

Problem: getwinpos() doesn't work in terminal on MS-Windows console.

Solution: Adjust #ifdefs. Disable test for MS-Windows console.

Files: src/ui.c, src/term.c, src/terminal.c,

src/testdir/test_terminal.vim

Patch 8.1.1128

Problem: getwinpos() test does not work on MS-Windows.
Solution: Skip the test.
Files: src/testdir/test_terminal.vim

Patch 8.1.1129

Problem: When making a new screendump test have to create the file.
Solution: Continue creating the failed screendump, so it can be moved once it is correct.
Files: src/testdir/screendump.vim

Patch 8.1.1130

Problem: MS-Windows: warning for unused variable.
Solution: Remove the variable.
Files: src/evalfunc.c

Patch 8.1.1131

Problem: getwinpos() does not work in the MS-Windows console.
Solution: Implement getwinpos().
Files: src/ui.c, src/evalfunc.c, src/terminal.c,
src/testdir/test_terminal.vim

Patch 8.1.1132

Problem: getwinpos() test fails on MS-Windows.
Solution: Don't try running this test.
Files: src/testdir/test_terminal.vim

Patch 8.1.1133

Problem: Compiler warning for uninitialized struct member. (Yegappan Lakshmanan)
Solution: Add initializer field.
Files: src/globals.h

Patch 8.1.1134

Problem: Buffer for quickfix window is reused for another file.
Solution: Don't reuse the quickfix buffer. (Yegappan Lakshmanan)
Files: src/buffer.c, src/testdir/test_quickfix.vim

Patch 8.1.1135 (after 8.1.1134)

Problem: Build failure for small version. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/buffer.c

Patch 8.1.1136

Problem: Decoding of mouse click escape sequence is not tested.
Solution: Add a test for xterm and SGR using low-level input. Make low-level input execution with feedkeys() work.
Files: src/testdir/test_termcodes.vim, src/testdir/Make_all.mak,
src/evalfunc.c, src/ex_docmd.c

Patch 8.1.1137

Problem: Xterm mouse wheel escape sequence is not tested.

Solution: Add a test using low-level input. (Dominique Pelle, closes #4221)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1138

Problem: Plugins don't get notified when the popup menu changes.
Solution: Add the CompleteChanged event. (Qiming Zhao, Andy Massimino, closes #4176)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/dict.c, src/insexpand.c, src/popupmnu.c, src/proto/autocmd.pro, src/proto/dict.pro, src/proto/popupmnu.pro, src/testdir/test_popup.vim, src/vim.h

Patch 8.1.1139

Problem: No test for what is fixed in patch 8.1.0716.
Solution: Add a test. (Yasuhiro Matsumoto, closes #3797)
Files: src/testdir/test_ins_complete.vim

Patch 8.1.1140

Problem: Not easy to find out what neighbors a window has.
Solution: Add more arguments to winnr(). (Yegappan Lakshmanan, closes #3993)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/proto/window.pro, src/testdir/test_window_cmd.vim, src/window.c

Patch 8.1.1141

Problem: Terminal winpos test fails with very large terminal. (Dominique Pelle)
Solution: Compute the expected size more accurately. (closes #4228)
Files: src/testdir/test_terminal.vim

Patch 8.1.1142

Problem: No test for dragging the window separators with the mouse.
Solution: Add a test. (Dominique Pelle, closes #4226)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1143

Problem: May pass weird strings to file name expansion.
Solution: Check for matching characters. Disallow control characters.
Files: src/misc1.c, src/testdir/test_spell.vim, src/option.c, src/proto/option.pro, src/spell.c, src/testdir/test_escaped_glob.vim

Patch 8.1.1144 (after 8.1.1143)

Problem: Too strict checking of the 'spellfile' option.
Solution: Allow for a path.
Files: src/option.c, src/testdir/test_spell.vim

Patch 8.1.1145

Problem: Compiler warning for unused function. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/option.c

Patch 8.1.1146

Problem: In MS-Windows console colors in a terminal window are wrong.
Solution: Use the ansi index also for 16 colors. (Ken Takata)

Files: src/terminal.c

Patch 8.1.1147

Problem: Desktop file translations are requiring manual updates.

Solution: Use the .po files for desktop file translations. (Christian Brabandt)

Files: src/po/Makefile, src/po/gvim.desktop.in, src/po/vim.desktop.in, CONTRIBUTING.md, Filelist, runtime/vim.desktop, runtime/gvim.desktop

Patch 8.1.1148

Problem: **CTRL-L** with '**incsearch**' does not pick up char under cursor. (Smylers)

Solution: Do not compare the position with the cursor position. (Hirohito Higashi, closes #3620)

Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.1149

Problem: Building desktop files fails with older msgfmt.

Solution: Add autoconf check. Avoid always building the desktop files.

Files: src/configure.ac, src/auto/configure, src/po/Makefile, src/po/Make_all.mak, src/config.mk.in

Patch 8.1.1150

Problem: Generating desktop files not tested on Travis.

Solution: Install a newer msgfmt package. (Christian Brabandt)

Files: .travis.yml

Patch 8.1.1151

Problem: Build fails when using shadow directory.

Solution: Link the desktop.in files.

Files: src/Makefile

Patch 8.1.1152

Problem: Compiler warning with VS2019.

Solution: Specify different offset for "AMD64". (Ken Takata, closes #4235)

Files: src/GvimExt/Makefile

Patch 8.1.1153

Problem: Msgfmt complains about missing LINGUAS file. (Tony Mechelynck)

Solution: Add command to generate LINGUAS.

Files: src/po/Makefile

Patch 8.1.1154

Problem: Getting a newer msgfmt on Travis is too complicated.

Solution: Use a "sourceline" entry. (Ozaki Kiichi, closes #4236)

Files: .travis.yml

Patch 8.1.1155

Problem: Termcodes tests can be improved.

Solution: Add helper functions to simplify tests. Dragging statusline for xterm and sgr. (Dominique Pelle, closes #4237)

Files: src/testdir/test_termcodes.vim

Patch 8.1.1156

Problem: Unicode emoji and other image characters not recognized.
Solution: Add ranges for musical notation, game pieces, etc. (Martin Tournoij, closes #4238)
Files: src/mbyte.c

Patch 8.1.1157

Problem: Unicode tables are out of date.
Solution: Update to Unicode 12. (Christian Brabandt, closes #4240)
Files: src/mbyte.c

Patch 8.1.1158

Problem: Json encoded string is sometimes missing the final NUL.
Solution: Add the NUL. Also for log messages.
Files: src/json.c, src/channel.c, src/testdir/test_json.vim

Patch 8.1.1159

Problem: MS-Windows: with a silent (un)install \$VIM/_vimrc is removed.
Solution: Don't delete _vimrc in silent mode. (Ken Takata, closes #4242)
Files: nsis/gvim.nsi

Patch 8.1.1160

Problem: Termcodes test would fail in a very big terminal.
Solution: Bail out when the row is larger than what will work. (Dominique Pelle, closes #4246)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1161

Problem: Unreachable code.
Solution: Remove condition that will never be true. Add tests for all ANSI colors.
Files: src/terminal.c, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_all_ansi_colors.dump

Patch 8.1.1162

Problem: Incorrect coverage information; typo in color name.
Solution: Fix the typo. Set environment variables to have a nested Vim write the coverage info in another directory.
Files: src/testdir/test_terminal.vim, src/testdir/screendump.vim, src/testdir/dumps/Test_terminal_all_ansi_colors.dump

Patch 8.1.1163

Problem: Codecov does not report all the coverage information.
Solution: Make a second run with the nested execution output, expect that Codecov will merge the results.
Files: .travis.yml

Patch 8.1.1164

Problem: Gettitle test is failing when server name differs. (Kenta Sato)
Solution: Accept "VIM1" when 'autoservername' is used. (Dominique Pelle, closes #4250, closes #4249)
Files: src/testdir/test_terminal.vim

Patch 8.1.1165

Problem: No test for mouse clicks in the terminal tabpage line.
Solution: Add a test. (Dominique Pelle, closes #4247). Also init TabPageIdxs[], in case it's used before a redraw.
Files: src/screen.c, src/testdir/test_termcodes.vim

Patch 8.1.1166 (after 8.1.1164)

Problem: Gettitle test can still fail when another Vim is running.
Solution: Accept any server name number. (Dominique Pelle, closes #4252)
Files: src/testdir/test_terminal.vim

Patch 8.1.1167

Problem: No test for closing tab by click in tabline.
Solution: Add a test. Also fix that dragging window separator could fail in a large terminal. (Dominique Pelle, closes #4253)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1168

Problem: Not all screen update code of the terminal window is executed in tests.
Solution: Redraw before taking a screenshot.
Files: src/testdir/screendump.vim

Patch 8.1.1169

Problem: Writing coverage info in a separate dir is not needed.
Solution: Revert the changes to use a separate directory.
Files: .travis.yml, src/testdir/screendump.vim

Patch 8.1.1170

Problem: Terminal ANSI color test does not cover all colors.
Solution: Use the color number, the name is not always resulting in an ANSI color when t_Co is 256.
Files: src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_all_ansi_colors.dump

Patch 8.1.1171

Problem: Statusline test could fail in large terminal.
Solution: Make the test work on a huge terminal. (Dominique Pelle, closes #4255)
Files: src/testdir/test_statusline.vim

Patch 8.1.1172

Problem: Cursor properties were not fully tested.
Solution: Add a test. (Dominique Pelle, closes #4256)
Files: src/testdir/test_terminal.vim

Patch 8.1.1173

Problem: Suspend test has duplicated lines.
Solution: Use a function.
Files: src/testdir/test_suspend.vim

Patch 8.1.1174

Problem: Cannot build with Ruby 1.8. (Tom G. Christensen)
Solution: Include ruby/st.h. (Ozaki Kiichi, closes #4257)
Files: src/if_ruby.c

Patch 8.1.1175

Problem: No test for dragging a tab with the mouse and for creating a new tab by double clicking in the tabline.
Solution: Add two tests. (Dominique Pelle, closes #4258)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1176 (after 8.1.1175)

Problem: Test for dragging a tab is flaky.
Solution: Add a brief sleep.
Files: src/testdir/test_termcodes.vim

Patch 8.1.1177

Problem: .ts files are recognized as xml, while typescript is more common.
Solution: Recognize .ts files as typescript. (closes #4264)
Files: runtime/filetype.vim src/testdir/test_filetype.vim

Patch 8.1.1178

Problem: When mouse click tests fails value of **'ttymouse'** is unknown.
Solution: Add a message to the assert.
Files: src/testdir/test_termcodes.vim

Patch 8.1.1179

Problem: No test for mouse clicks in the fold column.
Solution: Add a test. (Dominique Pelle, closes #4261)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1180

Problem: Vim script debugger tests are old style.
Solution: Turn into new style tests. (Yegappan Lakshmanan, closes #4259)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test108.in, src/testdir/test108.ok, src/testdir/test_debugger.vim

Patch 8.1.1181

Problem: Tests for mouse clicks are a bit flaky when run in an interactive terminal.
Solution: Use "xterm2" instead of "xterm" for **'ttymouse'** to avoid spurious drag events.
Files: src/testdir/test_termcodes.vim

Patch 8.1.1182

Problem: Some function prototypes are outdated.
Solution: Update function prototypes. (Ken Takata, closes #4267)
Files: src/os_mswin.c, src/proto/ex_getln.pro, src/proto/gui_w32.pro, src/terminal.c, src/proto/terminal.pro, src/proto/window.pro, src/window.c

Patch 8.1.1183

Problem: Typos in VisVim comments.
Solution: Correct the typos. (Christ van Willegen)
Files: src/VisVim/Commands.cpp, src/VisVim/OleAut.cpp, src/VisVim/README_VisVim.txt

Patch 8.1.1184

Problem: Undo file left behind after running test.
Solution: Delete the undo file. (Dominique Pelle, closes #4279)
Files: src/testdir/test_filechanged.vim

Patch 8.1.1185

Problem: Mapping for **CTRL-X** is inconsistent.
Solution: Map **CTRL-X** to `"*d` also for the MS-Windows console. (Ken Takata, closes #4265)
Files: src/getchar.c

Patch 8.1.1186

Problem: `readdir()` allocates list twice.
Solution: Remove second allocation. Also check for zero length.
Files: src/evalfunc.c

Patch 8.1.1187

Problem: Cannot recognize Pipfile.
Solution: Use existing filetypes. (Charles Ross, closes #4280)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.1188

Problem: Not all Vim variables require the `v:` prefix.
Solution: When `scriptversion` is 3 all Vim variables can only be used with the `v:` prefix. (Ken Takata, closes #4274)
Files: src/eval.c, src/ex_cmds2.c, src/testdir/test_eval_stuff.vim, runtime/doc/eval.txt

Patch 8.1.1189

Problem: Mode is not cleared when leaving Insert mode.
Solution: Clear the mode when `got_int` is set. (Ozaki Kiichi, closes #4270)
Files: src/edit.c, src/testdir/test_buflines.vim, src/testdir/test_messages.vim

Patch 8.1.1190

Problem: `has('vimscrip-3')` does not work.
Solution: Add `"vimscrip-3"` to the list of features. (partly by Ken Takata)
Files: src/evalfunc.c, src/testdir/test_eval_stuff.vim

Patch 8.1.1191

Problem: Not all debug commands are covered by a test.
Solution: Add more tests. (Yegappan Lakshmanan, closes #4282)
Files: src/testdir/test_debugger.vim

Patch 8.1.1192

Problem: Mode is not cleared when leaving Insert mode with mapped Esc.
Solution: Clear the mode when `redraw_cmdline` is set. (closes #4269)
Files: src/globals.h, src/screen.c, src/testdir/test_messages.vim

Patch 8.1.1193

Problem: Typos and small problems in test files.
Solution: Small improvements.
Files: src/testdir/test_gn.vim, src/testdir/test_quotestar.vim, src/testdir/test_registers.vim, src/testdir/test_syntax.vim,

src/testdir/test_tabpage.vim, src/testdir/test_vartabs.vim

Patch 8.1.1194

Problem: Typos and small problems in source files.

Solution: Small fixes.

Files: src/channel.c, src/crypt.c, src/edit.c, src/regexp.h, src/tag.c, src/term.c, src/terminal.c, src/userfunc.c, src/installman.sh

Patch 8.1.1195

Problem: Vim script debugger functionality needs cleanup.

Solution: Move debugger code to a separate file. Add more tests. (Yegappan Lakshmanan, closes #4285)

Files: Filelist, src/Make_bc5.mak, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/debugger.c, src/ex_cmds2.c, src/proto.h, src/proto/debugger.pro, src/proto/ex_cmds2.pro

Patch 8.1.1196

Problem: Parallel build may fail.

Solution: Update dependencies.

Files: src/Makefile

Patch 8.1.1197

Problem: When starting with multiple tabs file messages is confusing.

Solution: Set '**shortmess**' when loading the other tabs. (Christian Brabandt)

Files: src/main.c, src/testdir/test_startup.vim, src/testdir/dumps/Test_start_with_tabs.dump

Patch 8.1.1198

Problem: Bracketed paste may remain active after Vim exists, because the terminal emulator restores the setting.

Solution: Set/reset bracketed paste mode before setting the terminal mode. (closes #3579)

Files: src/term.c

Patch 8.1.1199

Problem: No test for :abclear.

Solution: Add a test. (Dominique Pelle, closes #4292)

Files: src/testdir/test_mapping.vim

Patch 8.1.1200

Problem: Old style comments in debugger source.

Solution: Use new style comments. (Yegappan Lakshmanan, closes #4286)

Files: src/README.md, src/debugger.c

Patch 8.1.1201

Problem: Output of :command is hard to read.

Solution: Make some columns wider, some narrower. Truncate the command when listing all.

Files: src/ex_docmd.c, src/message.c, src/proto/message.pro, src/getchar.c, src/menu.c

Patch 8.1.1202

Problem: Always get regexp debugging logs when building with -DDEBUG.
Solution: By default do not create regexp debugging logs. (Ken Takata)
Files: src/regexp.c

Patch 8.1.1203

Problem: Some autocmd tests are old style.
Solution: Turn the tests into new style. (Yegappan Lakshmanan, closes #4295)
Files: src/Makefile, src/testdir/Make_all.mak,
src/testdir/Make_amiga.mak, src/testdir/Make_vms.mms,
src/testdir/test11.in, src/testdir/test11.ok,
src/testdir/test_autocmd.vim

Patch 8.1.1204

Problem: Output of :command with address completion is not nice.
Solution: Shorten the address completion names.
Files: src/ex_docmd.c, runtime/doc/map.txt

Patch 8.1.1205

Problem: A BufReadPre autocommand may cause the cursor to move.
Solution: Restore the cursor position after executing the autocommand,
unless the autocommand moved it. (Christian Brabandt,
closes #4302, closes #4294)
Files: src/autocmd.c, src/proto/window.pro, src/structs.h,
src/testdir/test_autocmd.vim, src/window.c

Patch 8.1.1206

Problem: User command parsing and listing not properly tested.
Solution: Add more tests. (Dominique Pelle, closes #4296)
Files: src/testdir/test_usercommands.vim

Patch 8.1.1207

Problem: Some compilers give warning messages.
Solution: Initialize variables, change printf() argument. (Christian
Brabandt, closes #4305)
Files: src/eval.c, src/screen.c, src/undo.c, src/window.c

Patch 8.1.1208

Problem: Links to repository use wrong file name.
Solution: Swap the file names. (Nahuel Ourthe, closes #4304)
Files: src/README.md

Patch 8.1.1209

Problem: Clever compiler warns for buffer being too small.
Solution: Make the buffer bigger (even though it's not really needed).
Files: src/evalfunc.c, src/syntax.c

Patch 8.1.1210

Problem: Support for user commands is spread out. No good reason to make
user commands optional.
Solution: Move user command support to usercmd.c. Always enable the
user_commands feature.
Files: src/usercmd.c, src/proto/usercmd.pro, Filelist, src/Make_bc5.mak,
src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_ivc.mak,

src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak,
src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/buffer.c, src/eval.c, src/evalfunc.c, src/ex_cmds.h,
src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_getln.c,
src/feature.h, src/macros.h, src/misc2.c, src/proto.h,
src/structs.h, src/version.c, runtime/doc/eval.txt,
runtime/doc/various.txt

Patch 8.1.1211

Problem: Not all user command code is tested.
Solution: Add more tests.
Files: src/testdir/test_usercommands.vim

Patch 8.1.1212

Problem: Signal PWR is not tested.
Solution: Test that PWR updates the swap file. (Dominique Pelle,
closes #4312)
Files: src/testdir/test_signals.vim

Patch 8.1.1213

Problem: "make clean" in top dir does not cleanup indent test output.
Solution: Clean the indent test output. Do not rely on the vim executable
for that. (closes #4307)
Files: Makefile, runtime/indent/Makefile,
runtime/indent/testdir/cleantest.vim

Patch 8.1.1214

Problem: Old style tests.
Solution: Move tests from test14 to new style test files. (Yegappan
Lakshmanan, closes #4308)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test14.in, src/testdir/test14.ok,
src/testdir/test_edit.vim, src/testdir/test_normal.vim,
src/testdir/test_search.vim, src/testdir/test_substitute.vim,
src/testdir/test_visual.vim

Patch 8.1.1215

Problem: "make clean" does not remove generated src/po files.
Solution: Remove the files for "make clean". (Christian Brabandt)
Files: src/po/Makefile

Patch 8.1.1216

Problem: Mouse middle click is not tested.
Solution: Add a test. (Dominique Pelle, closes #4310)
Files: src/testdir/test_termcodes.vim

Patch 8.1.1217

Problem: MS-Windows: no space reserved for font quality name.
Solution: Add quality_name length if present. (Ken Takata, closes #4311)
Files: src/gui_w32.c

Patch 8.1.1218

Problem: Cannot set a directory for a tab page.
Solution: Add the tab-local directory. (Yegappan Lakshmanan, closes #4212)

Files: runtime/doc/autocmd.txt, runtime/doc/editing.txt,
runtime/doc/eval.txt, runtime/doc/index.txt,
runtime/doc/options.txt, runtime/doc/usr_22.txt,
runtime/doc/usr_41.txt, src/eval.c, src/evalfunc.c,
src/ex_cmdidxs.h, src/ex_cmds.h, src/ex_docmd.c, src/if_py_both.h,
src/proto/eval.pro, src/proto/ex_docmd.pro, src/structs.h,
src/testdir/test_getcwd.vim, src/testdir/test_mksession.vim,
src/window.c

Patch 8.1.1219

Problem: Not checking for NULL return from alloc().
Solution: Add checks. (Martin Kunev, closes #4303, closes #4174)
Files: src/beval.c, src/blowfish.c, src/crypt.c, src/crypt_zip.c,
src/ops.c, src/option.c, src/popupmnu.c, src/proto/blowfish.pro,
src/proto/crypt_zip.pro, src/gui_gtk_f.c, src/gui_gtk_x11.c,
src/libvterm/src/state.c, src/libvterm/src/termscreen.c

Patch 8.1.1220 (after 8.1.1219)

Problem: Build fails on MS-Windows.
Solution: Move declaration to start of block.
Files: src/libvterm/src/state.c

Patch 8.1.1221

Problem: Filtering does not work when listing marks.
Solution: Implement filtering marks. (Marcin Szamotulski, closes #3895)
Files: runtime/doc/various.txt, src/mark.c,
src/testdir/test_filter_cmd.vim

Patch 8.1.1222 (after 8.1.1219)

Problem: Build still fails on MS-Windows.
Solution: Move another declaration to start of block.
Files: src/libvterm/src/state.c

Patch 8.1.1223

Problem: Middle mouse click test fails without a clipboard.
Solution: Check if the clipboard can be used. (Dominique Pelle, Christian Brabandt) Also use WorkingClipboard() instead of checking for the "clipboard" feature.
Files: src/testdir/test_termcodes.vim, src/testdir/test_quotestar.vim

Patch 8.1.1224

Problem: MS-Windows: cannot specify font weight.
Solution: Add the "W" option to '**guifont**'. (closes #4309) Move GUI font explanation out of options.txt.
Files: runtime/doc/options.txt, runtime/doc/gui.txt,
runtime/doc/mbyte.txt, src/gui_w32.c, src/os_mswin.c

Patch 8.1.1225

Problem: Cannot create a pty to use with :terminal on FreeBSD.
Solution: Add support for posix_openpt(). (Ozaki Kiichi, closes #4306, closes #4289)
Files: src/configure.ac, src/config.h.in, src/auto/configure, src/pty.c

Patch 8.1.1226

Problem: `{not in Vi}` remarks get in the way of useful help text.
Solution: Make a list of all Vi options, instead of mentioning what Vi does not have. Update the help text for options.
Files: runtime/doc/vi_diff.txt, runtime/doc/options.txt

Patch 8.1.1227

Problem: Duplicate entries in the generated .desktop files. (Ralf Schandl)
Solution: Remove translated entries from the .in files. (closes #4313)
Files: src/po/gvim.desktop.in, src/po/vim.desktop.in

Patch 8.1.1228

Problem: Not possible to process tags with a function.
Solution: Add tagfunc() (Christian Brabandt, Andy Massimino, closes #4010)
Files: runtime/doc/options.txt, runtime/doc/tagsrch.txt,
runtime/optwin.vim, src/buffer.c, src/dict.c, src/ex_cmds.c,
src/globals.h, src/insexpand.c, src/normal.c, src/option.c,
src/option.h, src/proto/dict.pro, src/structs.h, src/tag.c,
src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_tagfunc.vim, src/vim.h, src/window.c

Patch 8.1.1229

Problem: Warning for posix_openpt() not declared. (Tony Mechelynck)
Solution: Add declaration.
Files: src/pty.c

Patch 8.1.1230

Problem: A lot of code is shared between vim.exe and gvim.exe.
Solution: Optionally put the shared code in vim.dll. (Ken Takata, closes #4287)
Files: Filelist, nsis/gvim.nsi, runtime/doc/gui_w32.txt,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/channel.c,
src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/feature.h,
src/fileio.c, src/getchar.c, src/globals.h, src/gui.c, src/gui.h,
src/gui_gtk_x11.c, src/gui_w32.c, src/if_mzsch.c, src/main.c,
src/mbyte.c, src/memline.c, src/message.c, src/misc2.c,
src/normal.c, src/option.c, src/os_mswin.c, src/os_w32dll.c,
src/os_w32exe.c, src/os_win32.c, src/os_win32.h,
src/proto/gui.pro, src/proto/gui_w32.pro, src/proto/misc2.pro,
src/proto/os_mswin.pro, src/proto/os_win32.pro, src/syntax.c,
src/term.c, src/terminal.c, src/ui.c, src/version.c, src/vim.rc

Patch 8.1.1231

Problem: Asking about existing swap file unnecessarily.
Solution: When it is safe, delete the swap file. Remove HAS_SWAP_EXISTS_ACTION, it is always defined. (closes #1237)
Files: src/memline.c, src/globals.h, src/buffer.c, src/ex_cmds.c,
src/fileio.c, src/main.c, src/testdir/test_swap.vim,
runtime/doc/usr_11.txt, src/os_win32.c, src/proto/os_win32.pro,
src/os_unix.c, src/proto/os_unix.pro

Patch 8.1.1232

Problem: Can't build on MS-Windows.
Solution: Define process_still_running.
Files: src/memline.c, src/os_win32.c, src/proto/os_win32.pro,

src/os_unix.c, src/proto/os_unix.pro

Patch 8.1.1233

Problem: Cannot build tiny version.
Solution: Remove #ifdef for verb_msg().
Files: src/message.c

Patch 8.1.1234

Problem: Swap file test fails on MS-Windows.
Solution: Only compare the tail of the file names.
Files: src/testdir/test_swap.vim

Patch 8.1.1235

Problem: Compiler warnings for using STRLEN() value.
Solution: Cast to int. (Christian Brabandt, Mike Williams)
Files: src/tag.c

Patch 8.1.1236

Problem: sjiscorr.c not found in shadow directory. (Tony Mechelynck)
Solution: Link po/*.c files with "make shadow".
Files: src/Makefile

Patch 8.1.1237

Problem: Error for using "compl", reserved word in C++.
Solution: Rename to "complp". (suggestion by Ken Takata)
Files: src/usercmd.c, src/proto/usercmd.pro

Patch 8.1.1238

Problem: MS-Windows: compiler warning for sprintf() format.
Solution: Change %d to %ld. (Ken Takata)
Files: src/gui_w32.c

Patch 8.1.1239

Problem: Key with byte sequence containing CSI does not work.
Solution: Do not recognize CSI as special unless the GUI is active. (Ken Takata, closes #4318)
Files: src/getchar.c

Patch 8.1.1240

Problem: Runtime desktop files are overwritten by build. (Tony Mechelynck)
Solution: Instead of copying the files find them with "make install".
Files: src/Makefile, src/po/Makefile

Patch 8.1.1241

Problem: Ex command info contains confusing information.
Solution: When using the NOTADR flag use ADDR_OTHER for the address type. Cleanup code using NOTADR. Check for errors in create_cmdidxs.vim. Adjust Makefile to see the errors.
Files: src/ex_cmds.h, src/ex_docmd.c, src/Makefile, src/create_cmdidxs.vim, src/usercmd.c, src/ex_cmds.c, src/window.c, src/testdir/test_usercommands.vim

Patch 8.1.1242

Problem: No cmdline redraw when tabpages have different 'cmdheight'.

Solution: redraw the command line when `'cmdheight'` changes when switching tabpages. (closes #4321)

Files: src/testdir/test_tabpage.vim, src/window.c,
src/testdir/dumps/Test_tabpage_cmdheight.dump,
src/testdir/screendump.vim

Patch 8.1.1243 (after 8.1.1241)

Problem: Compiler warnings for incomplete switch statement. (Tony Mechelynck)

Solution: Add ADDR_QUICKFIX to the list.

Files: src/ex_docmd.c

Patch 8.1.1244

Problem: No tests for CTRL-mouse-click.

Solution: Add a few tests. (Dominique Pelle, closes #4323)

Files: src/testdir/test_termcodes.vim

Patch 8.1.1245

Problem: `":copen 10"` sets height in full-height window. (Daniel Hahler)

Solution: Don't set the height if the quickfix window is full height. (closes #4325)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.1246

Problem: Cannot handle negative mouse coordinate from urxvt.

Solution: Accept '-' where a digit is expected. (Vincent Vinel, closes #4326)

Files: src/term.c

Patch 8.1.1247

Problem: Urxvt mouse codes are not tested.

Solution: Also set `'ttymouse'` to "urxvt" in the termcodes test.

Files: src/testdir/test_termcodes.vim

Patch 8.1.1248

Problem: No test for dec mouse.

Solution: Add some tests for dec mouse. Add "no_query_mouse".

Files: src/evalfunc.c, src/globals.h, src/os_unix.c,
src/testdir/test_termcodes.vim, runtime/doc/eval.txt

Patch 8.1.1249

Problem: Compiler warning for uninitialized variable.

Solution: Initialize it. (Christian Brabandt)

Files: src/regexp_nfa.c

Patch 8.1.1250

Problem: No test for netterm mouse.

Solution: Add some tests for netterm mouse.

Files: src/testdir/test_termcodes.vim

Patch 8.1.1251

Problem: No test for completion of mapping keys.

Solution: Add a test. Also clean up the code.

Files: src/getchar.c, src/term.c, src/proto/term.pro,

src/testdir/test_cmdline.vim

Patch 8.1.1252

Problem: Not all mapping completion is tested.
Solution: Add a few more mapping completion tests.
Files: src/testdir/test_cmdline.vim

Patch 8.1.1253 (after 8.1.1252)

Problem: Mapping completion test fails.
Solution: Fix expected output.
Files: src/testdir/test_cmdline.vim

Patch 8.1.1254

Problem: Mapping completion contains dead code.
Solution: Remove the code.
Files: src/term.c, src/testdir/test_cmdline.vim

Patch 8.1.1255

Problem: Building desktop files fails on FreeBSD. (Adam Weinberger)
Solution: Avoid using non-portable construct in Makefile. (closes #4332)
Files: src/po/Makefile

Patch 8.1.1256

Problem: Cannot navigate through errors relative to the cursor.
Solution: Add :cabove, :cbelow, :labove and :lbelow. (Yegappan Lakshmanan, closes #4316)
Files: runtime/doc/index.txt, runtime/doc/quickfix.txt, src/ex_cmdidxs.h, src/ex_cmds.h, src/ex_docmd.c, src/proto/quickfix.pro, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.1257

Problem: MSVC: name of object directory not always right.
Solution: Adjust comment. Don't use different directory for DIRECTX. Do use different directory for USE_MSVCRT. (Ken Takata, closes #4333)
Files: src/Make_mvc.mak

Patch 8.1.1258

Problem: The "N files to edit" message can not be suppressed.
Solution: Suppress the message with --not-a-term. (closes #4320)
Files: src/main.c

Patch 8.1.1259

Problem: Crash when exiting early. (Ralf Schandl)
Solution: Only pop/push the title when it was set. (closes #4334)
Files: src/os_unix.c, src/misc2.c, src/usercmd.c, src/tag.c

Patch 8.1.1260

Problem: Comparing with pointer instead of value.
Solution: Add a "*". (Ken Takata, closes #4336)
Files: src/usercmd.c

Patch 8.1.1261

Problem: No error for quickfix commands with negative range.
Solution: Add ADDR_UNSIGNED and use it for quickfix commands. Make

Files: assert_fails() show the command if the error doesn't match.
src/ex_cmds.h, src/ex_docmd.c, src/testdir/test_quickfix.vim,
runtime/doc/quickfix.txt, src/eval.c, src/quickfix.c,
src/proto/quickfix.pro, src/ex_cmds2.c

Patch 8.1.1262

Problem: Cannot simulate a mouse click in a test.
Solution: Add test_setmouse().
Files: src/evalfunc.c, runtime/doc/eval.txt, runtime/doc/usr_41.txt

Patch 8.1.1263

Problem: Mouse clicks in WinBar not tested.
Solution: Add a test for clicking on the WinBar entries.
Files: src/testdir/test_winbar.vim

Patch 8.1.1264

Problem: Crash when closing window from WinBar click. (Ben Jackson)
Solution: Check that window pointer is still valid. (closes #4337)
Files: src/menu.c

Patch 8.1.1265

Problem: When GPM mouse support is enabled double clicks in xterm do not work.
Solution: Use KS_GPM_MOUSE for GPM mouse events.
Files: src/term.c, src/os_unix.c, src/keymap.h

Patch 8.1.1266

Problem: Winbar test doesn't test enough.
Solution: Check that the WinBar actually shows up. Correct check for clicks with no effect. (Ben Jackson, closes #4338)
Files: src/testdir/test_winbar.vim

Patch 8.1.1267

Problem: Cannot check if GPM mouse support is working.
Solution: Add the "mouse_gpm_enable" feature.
Files: src/evalfunc.c, src/os_unix.c, src/proto/os_unix.pro,
runtime/doc/eval.txt

Patch 8.1.1268

Problem: Map completion test fails in GUI.
Solution: Skip the test that fails.
Files: src/testdir/test_cmdline.vim

Patch 8.1.1269

Problem: MS-Windows GUI: multibyte chars with a 0x80 byte do not work when compiled with VIMDLL.
Solution: Adjust the condition for fixing the input buffer. (Ken Takata, closes #4330)
Files: src/getchar.c

Patch 8.1.1270

Problem: Cannot see current match position.
Solution: Show "3/44" when using the "n" command and "S" is not in '**shortmess**'. (Christian Brabandt, closes #4317)

Files: runtime/doc/options.txt, runtime/doc/pattern.txt, src/option.c,
src/option.h, src/search.c, src/testdir/Make_all.mak,
src/testdir/test_search_stat.vim

Patch 8.1.1271 (after 8.1.1270)

Problem: Compiler warnings for use of STRNCPY(). (John Marriott)

Solution: Use mch_memmove() instead of STRNCPY().

Files: src/search.c

Patch 8.1.1272

Problem: Click on WinBar of other window not tested.

Solution: Add a test case.

Files: src/testdir/test_winbar.vim

Patch 8.1.1273

Problem: Compiler warning in direct write code.

Solution: Add a type cast.

Files: src/gui_dwrite.cpp

Patch 8.1.1274

Problem: After :unmenu can still execute the menu with :emenu.

Solution: Do not execute a menu that was disabled for the specified mode.

Files: src/menu.c, src/testdir/test_menu.vim

Patch 8.1.1275

Problem: Cannot navigate to errors before/after the cursor.

Solution: Add the :cbefore and :cafter commands. (Yegappan Lakshmanan,
closes #4340)

Files: runtime/doc/index.txt, runtime/doc/quickfix.txt, src/ex_cmdidxs.h,
src/ex_cmds.h, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.1276

Problem: Cannot combine text properties with syntax highlighting.

Solution: Add the "combine" field to prop_type_add(). (closes #4343)

Files: runtime/doc/eval.txt, runtime/doc/textprop.txt, src/screen.c,
src/structs.h, src/testdir/test_textprop.vim

Patch 8.1.1277 (after 8.1.1276)

Problem: Missing screenshot update.

Solution: Update the screenshot.

Files: src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.1278 (after 8.1.1276)

Problem: Missing change for "combine" field.

Solution: Also change the textprop implementation.

Files: src/textprop.c

Patch 8.1.1279

Problem: Cannot set '**spelllang**' to "sr@latin". (Bojan Stipic)

Solution: Allow using '@' in '**spelllang**'. (closes #4342)

Files: src/option.c, src/testdir/gen_opt_test.vim

Patch 8.1.1280

Problem: Remarks about functionality not in Vi clutters the help.

Solution: Move all info about what is new in Vim or already existed in Vi to vi_diff.txt. Remove {not in Vi} remarks. (closes #4268) Add "noet" to the help files modeline. Also include many other help file improvements.

Files: runtime/doc/vi_diff.txt, runtime/doc/arabic.txt, runtime/doc/autocmd.txt, runtime/doc/change.txt, runtime/doc/channel.txt, runtime/doc/cmdline.txt, runtime/doc/debugger.txt, runtime/doc/debug.txt, runtime/doc/develop.txt, runtime/doc/diff.txt, runtime/doc/digraph.txt, runtime/doc/editing.txt, runtime/doc/eval.txt, runtime/doc/farsi.txt, runtime/doc/filetype.txt, runtime/doc/fold.txt, runtime/doc/ft_ada.txt, runtime/doc/ft_rust.txt, runtime/doc/ft_sql.txt, runtime/doc/gui.txt, runtime/doc/gui_w32.txt, runtime/doc/gui_x11.txt, runtime/doc/hangulin.txt, runtime/doc/hebrew.txt, runtime/doc/helphelp.txt, runtime/doc/help.txt, runtime/doc/howto.txt, runtime/doc/if_cscop.txt, runtime/doc/if_lua.txt, runtime/doc/if_mzsch.txt, runtime/doc/if_ole.txt, runtime/doc/if_perl.txt, runtime/doc/if_pyth.txt, runtime/doc/if_ruby.txt, runtime/doc/if_sniff.txt, runtime/doc/if_tcl.txt, runtime/doc/indent.txt, runtime/doc/index.txt, runtime/doc/insert.txt, runtime/doc/intro.txt, runtime/doc/map.txt, runtime/doc/mbyte.txt, runtime/doc/message.txt, runtime/doc/mlang.txt, runtime/doc/motion.txt, runtime/doc/netbeans.txt, runtime/doc/options.txt, runtime/doc/os_390.txt, runtime/doc/os_amiga.txt, runtime/doc/os_beos.txt, runtime/doc/os_dos.txt, runtime/doc/os_mac.txt, runtime/doc/os_mint.txt, runtime/doc/os_msdos.txt, runtime/doc/os_os2.txt, runtime/doc/os_qnx.txt, runtime/doc/os_risc.txt, runtime/doc/os_unix.txt, runtime/doc/os_vms.txt, runtime/doc/os_win32.txt, runtime/doc/pattern.txt, runtime/doc/pi_getscript.txt, runtime/doc/pi_gzip.txt, runtime/doc/pi_logipat.txt, runtime/doc/pi_netrw.txt, runtime/doc/pi_paren.txt, runtime/doc/pi_spec.txt, runtime/doc/pi_tar.txt, runtime/doc/pi_vimball.txt, runtime/doc/pi_zip.txt, runtime/doc/print.txt, runtime/doc/quickfix.txt, runtime/doc/quickref.txt, runtime/doc/quotes.txt, runtime/doc/recover.txt, runtime/doc/remote.txt, runtime/doc/repeat.txt, runtime/doc/rileft.txt, runtime/doc/russian.txt, runtime/doc/scroll.txt, runtime/doc/sign.txt, runtime/doc/spell.txt, runtime/doc/sponsor.txt, runtime/doc/starting.txt, runtime/doc/syntax.txt, runtime/doc/tabpage.txt, runtime/doc/tagsrch.txt, runtime/doc/terminal.txt, runtime/doc/term.txt, runtime/doc/textprop.txt, runtime/doc/tips.txt, runtime/doc/todo.txt, runtime/doc/uganda.txt, runtime/doc/undo.txt, runtime/doc/usr_01.txt, runtime/doc/usr_02.txt, runtime/doc/usr_03.txt, runtime/doc/usr_04.txt, runtime/doc/usr_05.txt, runtime/doc/usr_06.txt,

runtime/doc/usr_07.txt, runtime/doc/usr_08.txt,
runtime/doc/usr_09.txt, runtime/doc/usr_10.txt,
runtime/doc/usr_11.txt, runtime/doc/usr_12.txt,
runtime/doc/usr_20.txt, runtime/doc/usr_21.txt,
runtime/doc/usr_22.txt, runtime/doc/usr_23.txt,
runtime/doc/usr_24.txt, runtime/doc/usr_25.txt,
runtime/doc/usr_26.txt, runtime/doc/usr_27.txt,
runtime/doc/usr_28.txt, runtime/doc/usr_29.txt,
runtime/doc/usr_30.txt, runtime/doc/usr_31.txt,
runtime/doc/usr_32.txt, runtime/doc/usr_40.txt,
runtime/doc/usr_41.txt, runtime/doc/usr_43.txt,
runtime/doc/usr_44.txt, runtime/doc/usr_45.txt,
runtime/doc/usr_90.txt, runtime/doc/usr_toc.txt,
runtime/doc/varioues.txt, runtime/doc/version4.txt,
runtime/doc/version5.txt, runtime/doc/version6.txt,
runtime/doc/version7.txt, runtime/doc/version8.txt,
runtime/doc/visual.txt, runtime/doc/windows.txt, runtime/doc/tags

Patch 8.1.1281

Problem: Cannot specify a count with :chistory.
Solution: Add a count to :chistory and :lhistory. (Yegappan Lakshmanan, closes #4344)
Files: runtime/doc/quickfix.txt, src/ex_cmds.h, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.1282

Problem: Running make in src/po leaves LINGUAS file behind. (Ken Takata)
Solution: Delete LINGUAS after running msgfmt.
Files: src/po/Makefile

Patch 8.1.1283

Problem: Delaying half a second after the top-bot message.
Solution: Instead of the delay add "W" to the search count.
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.1.1284

Problem: Detecting *.tmpl as htmlcheetah is outdated.
Solution: Use the generic name "template". (closes #4348)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.1285

Problem: Test17 is old style.
Solution: Turn into new style test. (Yegappan Lakshmanan, closes #4347)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test17.in, src/testdir/test17.ok, src/testdir/test17a.in, src/testdir/test_checkpath.vim, src/testdir/test_gf.vim

Patch 8.1.1286

Problem: Running tests leaves XTest_tabpage_cmdheight file behind.
Solution: Delete the right file. (closes #4350)
Files: src/testdir/test_tabpage.vim

Patch 8.1.1287

Problem: Cannot build with +eval but without +mouse.
Solution: Add #ifdefs around f_test_setmouse(). (John Marriott)
Files: src/evalfunc.c

Patch 8.1.1288

Problem: Search stats don't show for mapped command.
Solution: Remove SEARCH_PEEK from searchit flags. Add a test. (Christian Brabandt)
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.1.1289

Problem: May not have enough space to add "W" to search stats.
Solution: Reserve a bit more space. (Christian Brabandt)
Files: src/search.c

Patch 8.1.1290

Problem: .hgignore and .gitignore are either distributed or in git, not both.
Solution: Add .gitignore to the distribution and .hgignore to git. Update the entries. (Christian Brabandt, Ken Takata)
Files: .gitignore, .hgignore, Filelist

Patch 8.1.1291

Problem: Not easy to change directory and restore.
Solution: Add the chdir() function. (Yegappan Lakshmanan, closes #4358)
Files: runtime/doc/eval.txt, runtime/doc/todo.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/ex_docmd.c,
src/if_py_both.h, src/proto/ex_docmd.pro, src/structs.h,
src/testdir/test_cd.vim

Patch 8.1.1292

Problem: Invalid command line arguments not tested.
Solution: Add a test. (Dominique Pelle, closes #4346)
Files: src/testdir/test_startup.vim

Patch 8.1.1293

Problem: MSVC files are no longer useful for debugging. Newer Visual Studio versions cannot read them.
Solution: Delete the files. (Ken Takata, closes #4357)
Files: Filelist, src/Make_dvc.mak, src/Make_ivc.mak,
runtime/doc/debug.txt, src/INSTALLpc.txt, src/Make_mvc.mak

Patch 8.1.1294

Problem: MS-Windows: Some fonts return wrong average char width.
Solution: Compute the average ourselves. (Ken Takata, closes #4356)
Files: src/gui_w32.c

Patch 8.1.1295

Problem: When vimrun.exe does not exist external command may fail.
Solution: Use "cmd /c" twice to get the same behavior. (Ken Takata, closes #4355)
Files: src/os_win32.c

Patch 8.1.1296

Problem: Crash when using invalid command line argument.
Solution: Check for options not being initialized.
Files: src/term.c, src/testdir/test_startup.vim

Patch 8.1.1297

Problem: Invalid argument test fails without GTK.
Solution: Test `-display` and `--display` separately.
Files: src/testdir/test_startup.vim

Patch 8.1.1298

Problem: Invalid argument test fails without X clipboard.
Solution: Test `-display` only with the `+xterm_clipboard` feature.
Files: src/testdir/test_startup.vim

Patch 8.1.1299

Problem: "extends" from **'listchars'** is used when **'list'** is off. (Hiroyuki Yoshinaga)
Solution: Only use the "extends" character when **'list'** is on. (Hirohito Higashi, closes #4360)
Files: src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.1300

Problem: In a terminal **'ballooneval'** does not work right away.
Solution: Flush output after drawing the balloon. Add the `<Ignore>` key code. Add a test.
Files: src/ex_cmds2.c, src/testdir/test_balloon.vim, src/misc2.c, src/testdir/Make_all.mak, src/testdir/dumps/Test_balloon_eval_term_01.dump

Patch 8.1.1301

Problem: When compiled with VIMDLL some messages are not shown.
Solution: Set/reset `gui.in_use` and `gui.starting` as needed. (Ken Takata, closes #4361)
Files: src/gui_w32.c, src/main.c, src/message.c

Patch 8.1.1302

Problem: `v:beval_text` is not tested in Visual mode.
Solution: Add a screenshot of the balloon in Visual mode.
Files: src/testdir/test_balloon.vim, src/normal.c, src/testdir/dumps/Test_balloon_eval_term_01.dump, src/testdir/dumps/Test_balloon_eval_term_02.dump

Patch 8.1.1303

Problem: Not possible to hide a balloon.
Solution: Hide the balloon when `balloon_show()` is called with an empty string or list. Add `balloon_gettext()`.
Files: src/evalfunc.c, src/popupmnu.c, src/gui_beval.c, src/gui_w32.c, src/beval.h, src/testdir/test_balloon.vim, runtime/doc/eval.txt

Patch 8.1.1304

Problem: MS-Windows: compiler warning for unused value.
Solution: Adjust `#ifdefs`. (Ken Takata, closes #4363)
Files: src/gui.c

Patch 8.1.1305

Problem: There is no easy way to manipulate environment variables.
Solution: Add environ(), getenv() and setenv(). (Yasuhiro Matsumoto, closes #2875)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/testdir/Make_all.mak, src/testdir/test_envIRON.vim

Patch 8.1.1306

Problem: Borland support is outdated and doesn't work.
Solution: Remove Borland support, there are other (free) compilers available. (Thomas Dzedzic, Ken Takata, closes #4364)
Files: .gitignore, .hgignore, Filelist, runtime/doc/debug.txt, runtime/doc/develop.txt, runtime/doc/usr_90.txt, src/GvimExt/Make_bc5.mak, src/GvimExt/gvimext.cpp, src/GvimExt/gvimext.rc, src/INSTALLpc.txt, src/Make_bc5.mak, src/dosinst.c, src/dosinst.h, src/evalfunc.c, src/ex_cmds.c, src/ex_getln.c, src/gui_w32.c, src/if_ole.cpp, src/if_py_both.h, src/main.c, src/mark.c, src/message.c, src/misc1.c, src/misc2.c, src/normal.c, src/option.c, src/os_mswin.c, src/os_w32exe.c, src/os_win32.c, src/os_win32.h, src/proto.h, src/screen.c, src/spell.c, src/spellfile.c, src/syntax.c, src/userfunc.c, src/vim.h, src/vim.rc, src/vimrun.c, src/xxd/Make_bc5.mak, src/xxd/xxd.c

Patch 8.1.1307

Problem: Cannot reconnect to the X server after it restarted.
Solution: Add the :xrestore command. (Adrian Kocis, closes #844)
Files: runtime/doc/index.txt, runtime/doc/various.txt, src/os_unix.c, src/proto/os_unix.pro, src/globals.h, src/ex_cmds.h, src/ex_cmdidxs.h, src/ex_docmd.c, src/testdir/test_paste.vim

Patch 8.1.1308

Problem: The Normal highlight is not defined when compiled with GUI.
Solution: Always define Normal. (Christian Brabandt, closes #4072)
Files: runtime/doc/syntax.txt, src/syntax.c, src/testdir/test_highlight.vim

Patch 8.1.1309 (after 8.1.1308)

Problem: Test for Normal highlight fails on MS-Windows GUI.
Solution: Skip the test for MS-Windows GUI.
Files: src/testdir/test_highlight.vim

Patch 8.1.1310

Problem: Named function arguments are never optional.
Solution: Support optional function arguments with a default value. (Andy Massimino, closes #3952)
Files: runtime/doc/eval.txt, src/structs.h, src/testdir/test_user_func.vim, src/userfunc.c

Patch 8.1.1311

Problem: Aborting an autocmd with an exception is not tested.
Solution: Add a test. Also shows how to abort a command by throwing an exception.
Files: src/testdir/test_autocmd.vim

Patch 8.1.1312

Problem: Coverity warning for using uninitialized variable.
Solution: Clear exarg_T.
Files: src/quickfix.c, src/channel.c, src/ex_cmds2.c

Patch 8.1.1313

Problem: Warnings for using localtime() and ctime().
Solution: Use localtime_r() if available. Avoid using ctime().
Files: src/configure.ac, src/auto/configure, src/config.h.in,
src/evalfunc.c, src/nbdebug.c, src/undo.c, src/memline.c,
src/proto/memline.pro, src/hardcopy.c

Patch 8.1.1314

Problem: MSVC makefile is not nicely indented.
Solution: Adjust spaces in preprocessor directives. (Ken Takata)
Files: src/Make_mvc.mak

Patch 8.1.1315

Problem: There is always a delay if a termrequest is never answered.
Solution: When the response is not received within two seconds consider the
request to have failed.
Files: src/term.c

Patch 8.1.1316

Problem: Duplicated localtime() call.
Solution: Delete one.
Files: src/undo.c

Patch 8.1.1317

Problem: Output from Travis can be improved.
Solution: Add section headers. Handle errors better. (Ozaki Kiichi,
closes #4098)
Files: .travis.yml, configure

Patch 8.1.1318

Problem: Code for text changes is in a "misc" file.
Solution: Move the code to change.c.
Files: src/misc1.c, src/proto/misc1.pro, src/change.c,
src/proto/change.pro, src/proto.h, src/memline.c, Filelist,
src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_manx.mak,
src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak,
src/Make_vms.mms, src/Makefile, src/README.md

Patch 8.1.1319

Problem: Computing function length name in many places.
Solution: compute name length in call_func().
Files: src/eval.c, src/userfunc.c, src/channel.c, src/evalfunc.c,
src/ex_cmds2.c, src/regexp.c, src/terminal.c

Patch 8.1.1320

Problem: It is not possible to track changes to a buffer.
Solution: Add listener_add() and listener_remove(). No docs or tests yet.
Files: src/structs.h, src/change.c, src/proto/change.pro

Patch 8.1.1321

Problem: No docs or tests for listener functions.
Solution: Add help and tests for listener_add() and listener_remove().
Invoke the callbacks before redrawing.
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt,
src/testdir/test_listener.vim, src/testdir/Make_all.mak,
src/change.c, src/screen.c, src/evalfunc.c, src/proto/evalfunc.pro

Patch 8.1.1322

Problem: Cygwin makefile is not nicely indented.
Solution: Adjust spaces in preprocessor directives. (Ken Takata)
Files: src/Make_cyg_ming.mak

Patch 8.1.1323

Problem: **'mouse'** option is reset when using GPM mouse.
Solution: Add flag for GPM mouse.
Files: src/term.c

Patch 8.1.1324

Problem: Stray comma in VMS makefile.
Solution: Remove the comma. (Naruhiko Nishino, closes #4368)
Files: src/Make_vms.mms

Patch 8.1.1325

Problem: Cannot build with +eval but without +channel and +timers. (John Marriott)
Solution: Adjust #ifdef for get_callback().
Files: src/evalfunc.c, src/testdir/test_autocmd.vim

Patch 8.1.1326

Problem: No test for listener with partial.
Solution: Add a test. Add example to help.
Files: src/testdir/test_listener.vim, runtime/doc/eval.txt

Patch 8.1.1327

Problem: Unnecessary scroll after horizontal split.
Solution: Don't adjust to fraction if all the text fits in the window.
(Martin Kunev, closes #4367)
Files: src/testdir/test_window_cmd.vim, src/window.c

Patch 8.1.1328

Problem: No test for listener with undo operation.
Solution: Add a test.
Files: src/testdir/test_listener.vim

Patch 8.1.1329

Problem: Plans for popup window support are spread out.
Solution: Add a first version of the popup window help.
Files: runtime/doc/popup.txt, runtime/doc/Makefile, runtime/doc/help.txt

Patch 8.1.1330

Problem: Using bold attribute in terminal changes the color. (Jason Franklin)

Solution: Don't set the "bold-highbright" flag in vterm unless the terminal supports less than 16 colors.
Files: src/terminal.c, src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_all_ansi_colors.dump

Patch 8.1.1331

Problem: Test 29 is old style.
Solution: Turn it into a new style test. (Yegappan Lakshmanan, closes #4370)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test29.in, src/testdir/test29.ok,
src/testdir/test_backspace_opt.vim, src/testdir/test_join.vim

Patch 8.1.1332

Problem: Cannot flush change listeners without also redrawing. The line numbers in the list of changes may become invalid.
Solution: Add listener_flush(). Invoke listeners before adding a change that makes line numbers invalid.
Files: src/evalfunc.c, src/change.c, src/proto/change.pro,
src/screen.c, runtime/doc/eval.txt, src/testdir/test_listener.vim

Patch 8.1.1333

Problem: Text properties don't always move after changes.
Solution: Update properties before reporting changes to listeners. Move text property when splitting a line.
Files: src/change.c, src/ex_cmds.c, src/textprop.c,
src/proto/textprop.pro, src/testdir/test_textprop.vim

Patch 8.1.1334

Problem: When buffer is hidden "F" in '**shortmess**' is not used.
Solution: Check the "F" flag in '**shortmess**' when the buffer is already loaded. (Jason Franklin) Add test_getvalue() to be able to test this.
Files: src/buffer.c, src/evalfunc.c, src/testdir/test_options.vim,
runtime/doc/eval.txt

Patch 8.1.1335

Problem: Listener callback is called after inserting text.
Solution: Flush the changes before inserting or deleting a line. Store changes per buffer.
Files: src/change.c, src/proto/change.pro, src/memline.c,
src/structs.h, src/testdir/test_listener.vim

Patch 8.1.1336

Problem: Some eval functionality is not covered by tests.
Solution: Add a few more test cases. (Masato Nishihata, closes #4374)
Files: src/testdir/test_buflines.vim, src/testdir/test_cindent.vim,
src/testdir/test_cursor_func.vim, src/testdir/test_delete.vim,
src/testdir/test_expand_func.vim, src/testdir/test_float_func.vim,
src/testdir/test_fnamemodify.vim, src/testdir/test_functions.vim

Patch 8.1.1337

Problem: Get empty text prop when splitting line just after text prop.
Solution: Do not create an empty text prop at the start of the line.
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.1.1338

Problem: Hang when concealing the '>' shown for a wide char that doesn't fit in the last cell.

Solution: Put back the pointer when the '>' is not going to be displayed. (closes #4377)

Files: src/screen.c

Patch 8.1.1339

Problem: Installer needs to product name et al.

Solution: Add a few lines to the NSIS installer script. (Ken Takata)

Files: nsis/gvim.nsi

Patch 8.1.1340

Problem: Attributes from '**cursorline**' overwrite textprop.

Solution: Combine the attributes. (closes #3912)

Files: src/screen.c, src/textprop.c, src/testdir/test_textprop.vim, src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.1341

Problem: Text properties are lost when joining lines.

Solution: Move the text properties to the joined line.

Files: src/ops.c, src/textprop.c, src/proto/textprop.pro, src/testdir/test_textprop.vim, src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.1342

Problem: Using freed memory when joining line with text property.

Solution: Use already computed length.

Files: src/ops.c

Patch 8.1.1343

Problem: Text properties not adjusted for Visual block mode delete.

Solution: Call adjust_prop_columns(). (closes #4384)

Files: src/ops.c, src/textprop.c, src/testdir/test_textprop.vim, src/misc1.c, src/testdir/dumps/Test_textprop_vis_01.dump, src/testdir/dumps/Test_textprop_vis_02.dump

Patch 8.1.1344

Problem: Coverity complains about possibly using a NULL pointer and copying a string into a fixed size buffer.

Solution: Check for NULL, even though it should not happen. Use vim_strncpy() instead of strcpy().

Files: src/change.c, src/memline.c

Patch 8.1.1345

Problem: Stuck in sandbox with ":s/../../\=Function/gn".

Solution: Don't skip over code to restore sandbox. (Christian Brabandt)

Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.1.1346

Problem: Error for Python exception does not show useful info.

Solution: Show the last line instead of the first one. (Ben Jackson, closes #4381)

Files: src/if_py_both.h, src/testdir/test86.ok, src/testdir/test87.ok,
src/testdir/test_python2.vim, src/testdir/test_python3.vim,
src/testdir/test_pyx2.vim, src/testdir/test_pyx3.vim

Patch 8.1.1347 (after 8.1.1327)

Problem: Fractional scroll position not restored after closing window.

Solution: Do restore fraction if topline is not one.

Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.1.1348

Problem: Running tests may cause the window to move.

Solution: Correct the reported window position for the offset with the
position after ":winpos". Works around an xterm bug.

Files: src/testdir/test_edit.vim

Patch 8.1.1349

Problem: If writing runs into a conversion error the backup file is
deleted. (Arseny Nasokin)

Solution: Don't delete the backup file if the file was overwritten and a
conversion error occurred. (Christian Brabandt, closes #4387)

Files: src/fileio.c, src/testdir/test_writefile.vim

Patch 8.1.1350

Problem: "W" for wrapping not shown when more than 99 matches.

Solution: Adjust check for length. (Masato Nishihata, closes #4388)

Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.1.1351

Problem: Text property wrong after :substitute.

Solution: Save for undo before changing any text properties.

Files: src/testdir/test_textprop.vim, src/ex_cmds.c, src/textprop.c,
src/proto/textprop.pro, src/change.c, src/edit.c, src/misc1.c,
src/ops.c

Patch 8.1.1352

Problem: Undofile() reports wrong name. (Francisco Giordano)

Solution: Clean up the name before changing path separators. (closes #4392,
closes #4394)

Files: src/evalfunc.c, src/testdir/test_undo.vim

Patch 8.1.1353 (after 8.1.1352)

Problem: Undo test fails on Mac.

Solution: Expect "private" on the Mac.

Files: src/testdir/test_undo.vim

Patch 8.1.1354

Problem: Getting a list of text lines is clumsy.

Solution: Add the =<< assignment. (Yegappan Lakshmanan, closes #4386)

Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_let.vim

Patch 8.1.1355

Problem: Obvious mistakes are accepted as valid expressions.

Solution: Be more strict about parsing numbers. (Yasuhiro Matsumoto,
closes #3981)

Files: src/charset.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c,
src/ex_getln.c, src/json.c, src/misc2.c, src/ops.c, src/option.c,
src/proto/charset.pro, src/testdir/test_expr.vim,
src/testdir/test_json.vim

Patch 8.1.1356

Problem: Some text in heredoc assignment ends the text. (Ozaki Kiichi)
Solution: Recognize "let v =<<" and skip until the end.
Files: src/userfunc.c, src/testdir/test_let.vim

Patch 8.1.1357

Problem: Test 37 is old style.
Solution: Turn it into a new style test. (Yegappan Lakshmanan, closes #4398)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test37.in, src/testdir/test37.ok,
src/testdir/test_scrollbind.vim

Patch 8.1.1358

Problem: Cannot enter character with a CSI byte.
Solution: Only check "gui.in_use" when VIMDLL is defined. (Ken Takata,
closes #4396)
Files: src/getchar.c

Patch 8.1.1359

Problem: Text property wrong after :substitute with backslash.
Solution: Adjust text property columns when removing backslashes.
(closes #4397)
Files: src/ex_cmds.c, src/testdir/test_textprop.vim, src/vim.h,
src/textprop.c, src/proto/textprop.pro, src/change.c, src/edit.c,
src/misc1.c, src/ops.c

Patch 8.1.1360 (after Patch 8.1.1345)

Problem: Buffer left '**nomodifiable**' after :substitute. (Ingo Karkat)
Solution: Save the value of '**modifiable**' earlier. (Christian Brabandt,
closes #4403)
Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.1.1361

Problem: Python setuptools don't work with Python 3.
Solution: Add dummy implementation for find_module. (Joel Frederico,
closes #4402, closes #3984)
Files: src/if_py_both.h

Patch 8.1.1362

Problem: Code and data in tests can be hard to read.
Solution: Use the new heredoc style. (Yegappan Lakshmanan, closes #4400)
Files: src/testdir/test_autocmd.vim, src/testdir/test_balloon.vim,
src/testdir/test_buflines.vim, src/testdir/test_cindent.vim,
src/testdir/test_conceal.vim, src/testdir/test_exit.vim,
src/testdir/test_fold.vim, src/testdir/test_goto.vim,
src/testdir/test_join.vim, src/testdir/test_mksession_utf8.vim,
src/testdir/test_normal.vim, src/testdir/test_profile.vim,
src/testdir/test_quickfix.vim, src/testdir/test_startup.vim,
src/testdir/test_terminal.vim, src/testdir/test_xxd.vim

Patch 8.1.1363

Problem: ":vert options" does not make a vertical split.
Solution: Pass the right modifiers in \$OPTWIN_CMD. (Ken Takata, closes #4401)
Files: src/ex_cmds2.c, src/testdir/test_options.vim

Patch 8.1.1364

Problem: Design for popup window support needs more details.
Solution: Add details about using a window and buffer. Rename popup_show() to popup_create() and add popup_show() and popup_hide().
Files: runtime/doc/popup.txt

Patch 8.1.1365

Problem: Source command doesn't check for the sandbox. (Armin Razmjou)
Solution: Check for the sandbox when sourcing a file.
Files: src/getchar.c, src/testdir/test_source.vim

Patch 8.1.1366

Problem: Using expressions in a modeline is unsafe.
Solution: Disallow using expressions in a modeline, unless the **'modelineexpr'** option is set. Update help, add more tests.
Files: runtime/doc/options.txt, src/option.c, src/option.h, src/testdir/test_modeline.vim, src/testdir/test49.in

Patch 8.1.1367 (after 8.1.1366)

Problem: can set **'modelineexpr'** in modeline.
Solution: Add P_SECURE flag.
Files: src/option.c, src/testdir/test_modeline.vim

Patch 8.1.1368 (after 8.1.1366)

Problem: Modeline test fails with python but without pythonhome.
Solution: Correct test argument.
Files: src/testdir/test_modeline.vim

Patch 8.1.1369

Problem: Get E484 when using system() during GUI startup.
Solution: Check "gui.starting". (Ken Takata)
Files: src/os_win32.c

Patch 8.1.1370

Problem: Not using the new github feature for donations.
Solution: Add a Sponsor button. (closes #4417)
Files: .github/FUNDING.yml

Patch 8.1.1371

Problem: Cannot recover from a swap file.
Solution: Do not expand environment variables in the swap file name. Do not check the extension when we already know a file is a swap file. (Ken Takata, closes #4415, closes #4369)
Files: src/buffer.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/gui.c, src/if_cscope.c, src/main.c, src/memline.c, src/misc1.c, src/proto/memline.pro, src/proto/misc1.pro, src/search.c, src/spell.c, src/spellfile.c, src/tag.c,

src/testdir/test_swap.vim, src/vim.h

Patch 8.1.1372

Problem: When evaluating **'statusline'** the current window is unknown.
(Daniel Hahler)
Solution: Set "g:actual_curwin" for %{} items. Set "g:statusline_winid" when evaluating %!. (closes #4406, closes #3299)
Files: src/buffer.c, runtime/doc/options.txt, src/testdir/test_statusline.vim

Patch 8.1.1373

Problem: "[p" in Visual mode puts in wrong line.
Solution: Call nv_put() instead of duplicating the functionality.
(closes #4408)
Files: src/normal.c, src/testdir/test_put.vim

Patch 8.1.1374

Problem: Check for file changed triggers too often.
Solution: Don't use "b_p_ar" when it is negative.
Files: src/fileio.c

Patch 8.1.1375

Problem: Without "TS" in **'shortmess'** get a hit-enter prompt often.
Solution: Always truncate the search message. Also avoid putting it in the message history. (closes #4413)
Files: src/search.c, src/main.c, src/testdir/test_search_stat.vim

Patch 8.1.1376

Problem: Warnings for size_t/int mixups.
Solution: Change types, add type casts. (Mike Williams)
Files: src/search.c, src/textprop.c

Patch 8.1.1377

Problem: MS-Windows GUI uses wrong shell command for bash. (Robert Bogomip)
Solution: Check that **'shellcmdflag'** is "/c". (Ken Takata, closes #4418)
Files: src/os_win32.c

Patch 8.1.1378

Problem: Delete() can not handle a file name that looks like a pattern.
Solution: Use readdir() instead of appending "/*" and expanding wildcards.
(Ken Takata, closes #4424, closes #696)
Files: src/testdir/test_functions.vim, src/evalfunc.c, src/fileio.c, src/proto/fileio.pro

Patch 8.1.1379 (after 8.1.1374)

Problem: Filechanged test hangs.
Solution: Do not check **'autoread'**.
Files: src/fileio.c, src/testdir/test_filechanged.vim

Patch 8.1.1380

Problem: MS-Windows building VIMDLL with MSVC: SUBSYSTEM is not set.
Solution: Invert condition. (Ken Takata, closes #4422)
Files: src/Make_mvc.mak

Patch 8.1.1381

Problem: MS-Windows: missing build dependency.
Solution: Make gui_dwrite.cpp depend on gui_dwrite.h. (Ken Takata, closes #4423)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.1382

Problem: Error when editing test file.
Solution: Remove part of modeline.
Files: src/testdir/test_vimscript.vim, src/testdir/test49.vim, src/testdir/test49.in

Patch 8.1.1383

Problem: Warning for size_t/int mixup.
Solution: Change type. (Mike Williams)
Files: src/search.c

Patch 8.1.1384

Problem: Using "int" for alloc() often results in compiler warnings.
Solution: Use "size_t" and remove type casts. Remove alloc_check(), Vim only works with 32 bit ints anyway.
Files: src/misc2.c, src/proto/misc2.pro, src/change.c, src/ex_cmds.c, src/netbeans.c, src/autocmd.c, src/buffer.c, src/change.c, src/channel.c, src/charset.c, src/debugger.c, src/dict.c, src/diff.c, src/digraph.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/fileio.c, src/findfile.c, src/fold.c, src/getchar.c, src/gui.c, src/gui_at_fs.c, src/gui_gtk.c, src/gui_gtk_x11.c, src/gui_motif.c, src/gui_w32.c, src/hashtab.c, src/if_cscope.c, src/if_perlsfio.c, src/if_python3.c, src/if_xcmdsrv.c, src/indent.c, src/insexpand.c, src/main.c, src/mbyte.c, src/memfile.c, src/memline.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/netbeans.c, src/ops.c, src/option.c, src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_vms.c, src/os_win32.c, src/quickfix.c, src/regexp.c, src/screen.c, src/spell.c, src/spellfile.c, src/syntax.c, src/term.c, src/undo.c, src/usercmd.c, src/userfunc.c, src/version.c, src/winclip.c

Patch 8.1.1385

Problem: Signed/unsigned compiler warning.
Solution: Use STRLEN() instead of strlen().
Files: src/fileio.c

Patch 8.1.1386

Problem: Unnecessary type casts for lalloc().
Solution: Remove type casts. Change lalloc(size, TRUE) to alloc(size).
Files: src/buffer.c, src/change.c, src/channel.c, src/diff.c, src/edit.c, src/eval.c, src/ex_cmds.c, src/ex_getln.c, src/fileio.c, src/getchar.c, src/gui_mac.c, src/insexpand.c, src/gui_w32.c, src/gui_x11.c, src/menu.c, src/netbeans.c, src/ops.c, src/os_mswin.c, src/os_amiga.c, src/os_qnx.c, src/os_unix.c, src/os_win32.c, src/popupmnu.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/screen.c, src/search.c, src/sign.c,

src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c,
src/terminal.c, src/textprop.c, src/ui.c, src/undo.c,
src/userfunc.c, src/winclip.c, src/window.c

Patch 8.1.1387

Problem: Calling prop_add() in an empty buffer doesn't work. (Dominique Pelle)

Solution: Open the memline before adding a text property. (closes #4412)

Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.1.1388

Problem: Errors when calling prop_remove() for an unloaded buffer.

Solution: Bail out when the buffer is not loaded. Add a few more tests for failing when the buffer number is invalid.

Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.1.1389

Problem: Changes are not flushed when end and start overlap. (Paul Jolly)

Solution: When end of a previous changes overlaps with start of a new change, first flush listeners.

Files: src/change.c, src/testdir/test_listener.vim

Patch 8.1.1390

Problem: Search stats are off when using count or offset.

Solution: Recompute the stats when needed. (Masato Nishihata, closes #4410)

Files: src/testdir/test_search_stat.vim, src/search.c

Patch 8.1.1391

Problem: No popup window support.

Solution: Add initial code for popup windows. Add the **'wincolor'** option.

Files: Filelist, runtime/doc/popup.txt, runtime/doc/options.txt,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms,
src/Makefile, src/autocmd.c, src/buffer.c, src/ex_cmds.h,
src/ex_cmdidxs.h, src/proto/buffer.pro, src/eval.c src/evalfunc.c
src/feature.h, src/globals.h, src/option.c, src/option.h,
src/popupwin.c, src/proto.h, src/proto/popupwin.pro,
src/proto/window.pro, src/screen.c, src/structs.h, src/terminal.c,
src/testdir/Make_all.mak, src/testdir/dumps/Test_popupwin_01.dump,
src/testdir/test_popupwin.vim, src/vim.h, src/window.c

Patch 8.1.1392 (after 8.1.1391)

Problem: Build failure in tiny version.

Solution: Define ex_popupclear to ex_ni if not implemented. Add UNUSED.

Files: src/ex_docmd.c, src/window.c

Patch 8.1.1393

Problem: Unnecessary type casts.

Solution: Remove type casts from alloc() and lalloc() calls. (Mike Williams)

Files: src/channel.c, src/crypt.c, src/dict.c, src/dosinst.c,
src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c,
src/ex_getln.c, src/fileio.c, src/findfile.c, src/if_ole.cpp,
src/if_py_both.h, src/list.c, src/message.c, src/misc1.c,
src/misc2.c, src/ops.c, src/os_vms.c, src/os_win32.c,
src/quickfix.c, src/regexp_nfa.c, src/screen.c, src/search.c,

src/sign.c, src/syntax.c, src/tag.c, src/term.c, src/terminal.c,
src/textprop.c

Patch 8.1.1394

Problem: Not restoring t_F2 in registers test.
Solution: Assign to &t_F2 instead of t_F2. (Andy Massimino, closes #4434)
Files: src/testdir/test_registers.vim

Patch 8.1.1395

Problem: Saving for undo may access invalid memory. (Dominique Pelle)
Solution: Set ml_line_len also when returning a constant string.
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.1.1396

Problem: **'wincolor'** does not apply to lines below the buffer.
Solution: Also apply **'wincolor'** to the "~" lines and the number column.
Files: src/screen.c, src/testdir/test_highlight.vim,
src/testdir/dumps/Test_wincolor_01.dump

Patch 8.1.1397

Problem: Build fails in tiny version.
Solution: Always define hl_combine_attr().
Files: src/syntax.c

Patch 8.1.1398

Problem: Duplicate line in MSVC build file.
Solution: Remove the line. (Ken Takata, closes #4436)
Files: src/Make_mvc.mak

Patch 8.1.1399

Problem: Popup windows not adjusted when switching tabs.
Solution: Save and restore first_tab_popupwin. Fix closing a tabpage.
Files: src/window.c, src/popupwin.c, src/proto/popupwin.pro,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_02.dump,
src/testdir/dumps/Test_popupwin_03.dump,
src/testdir/dumps/Test_popupwin_04.dump

Patch 8.1.1400

Problem: Using global pointer for tab-local popups is clumsy.
Solution: Use the pointer in tabpage_T.
Files: src/popupwin.c, src/globals.h, src/eval.c, src/screen.c,
src/window.c

Patch 8.1.1401

Problem: Misspelled mkspellmem as makespellmem.
Solution: Drop duplicate help entry, fix test. (Naruhiko Nishino, Yasuhiro
Matsumoto, closes #4437)
Files: runtime/doc/options.txt, src/testdir/test_modeline.vim

Patch 8.1.1402

Problem: "timer" option of popup windows not supported.
Solution: Implement the "timer" option. (Yasuhiro Matsumoto, closes #4439)
Files: src/structs.h, src/testdir/test_popupwin.vim, src/popupwin.c,

src/window.c, runtime/doc/popup.txt

Patch 8.1.1403

Problem: Cannot build without the timer feature.

Solution: Add #ifdef.

Files: src/structs.h, src/window.c, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.1404

Problem: Cannot change the patch level when building with NSIS.

Solution: Use \$PATCHLEVEL if defined. (Christian Brabandt)

Files: nsis/gvim.nsi

Patch 8.1.1405

Problem: "highlight" option of popup windows not supported.

Solution: Implement the "highlight" option.

Files: src/option.c, src/proto/option.pro, src/diff.c src/popupwin.c,
runtime/doc/popup.txt, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_01.dump,
src/testdir/dumps/Test_popupwin_03.dump

Patch 8.1.1406

Problem: popup_hide() and popup_show() not implemented yet.

Solution: Implement the functions.

Files: src/popupwin.c, src/proto/popupwin.pro, src/evalfunc.c,
src/structs.h, runtime/doc/popup.txt, src/screen.c, src/vim.h,
src/testdir/test_popupwin.vim

Patch 8.1.1407

Problem: Popup_create() does not support text properties.

Solution: Support the third form of the text argument.

Files: src/textprop.c, src/proto/textprop.pro, src/popupwin.c,
src/testdir/test_popupwin.vim, src/screen.c,
src/testdir/dumps/Test_popupwin_02.dump,
src/testdir/dumps/Test_popupwin_03.dump,
src/testdir/dumps/Test_popupwin_04.dump,
runtime/doc/popup.txt

Patch 8.1.1408

Problem: PFL_HIDDEN conflicts with system header file. (Ken Takata)

Solution: Rename to POPF_HIDDEN.

Files: src/popupwin.c, src/screen.c, src/vim.h

Patch 8.1.1409

Problem: Coverity warns for using uninitialized memory.

Solution: Add a condition to clearing the growarray.

Files: src/json.c

Patch 8.1.1410

Problem: Popup_move() is not implemented yet.

Solution: Implement it. (Yasuhiro Matsumoto, closes #4441) Improve the
positioning and resizing.

Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,
src/screen.c, src/structs.h, src/proto/popupwin.pro,

src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_05.dump

Patch 8.1.1411

Problem: Coverity warns for divide by zero.
Solution: Make sure width is larger than zero.
Files: src/charset.c

Patch 8.1.1412

Problem: Test 30 is old style.
Solution: Turn it into a new style test. (Yegappan Lakshmanan, closes #4440)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test30.in, src/testdir/test30.ok,
src/testdir/test_fileformat.vim

Patch 8.1.1413

Problem: Error when the drive of the swap file was disconnected.
Solution: Try closing and re-opening the swap file. (partly by Joe Orost,
closes #4378)
Files: src/memfile.c, src/structs.h, src/testdir/test_startup.vim

Patch 8.1.1414

Problem: Alloc() returning "char_u *" causes a lot of type casts.
Solution: Have it return "void *". (Mike Williams) Define ALLOC_ONE() to
check the simple allocations.
Files: src/autocmd.c, src/blob.c, src/blowfish.c, src/buffer.c,
src/change.c, src/channel.c, src/crypt.c, src/crypt_zip.c,
src/dict.c, src/diff.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c,
src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c,
src/fileio.c, src/findfile.c, src/getchar.c, src/gui_gtk.c,
src/gui_gtk_x11.c, src/gui_mac.c, src/gui_motif.c,
src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/hardcopy.c,
src/hashtab.c, src/if_ctype.c, src/if_mzsch.c, src/if_perlsfio.c,
src/if_py_both.h, src/if_python3.c, src/if_xcmdsrv.c,
src/insexpand.c, src/list.c, src/mark.c, src/mbyte.c,
src/memfile.c, src/memfile_test.c, src/memline.c, src/message.c,
src/misc2.c, src/netbeans.c, src/normal.c, src/ops.c,
src/option.c, src/os_amiga.c, src/os_mac_conv.c, src/os_mswin.c,
src/os_unix.c, src/os_vms.c, src/os_win32.c, src/popupmnu.c,
src/proto/misc2.pro, src/quickfix.c, src/regexp.c,
src/regexp_nfa.c, src/screen.c, src/search.c, src/sign.c,
src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c, src/term.c,
src/terminal.c, src/textprop.c, src/ui.c, src/undo.c,
src/userfunc.c, src/version.c, src/winclip.c, src/window.c,
src/vim.h, src/testdir/test_ctype.vim

Patch 8.1.1415 (after 8.1.1414)

Problem: Build error in MS-Windows GUI.
Solution: Fix the LALLOC_MULT() argument.
Files: src/gui_w32.c

Patch 8.1.1416

Problem: Popup_getposition() not implemented yet.
Solution: Implement it. (Yasuhiro Matsumoto, closes #4449)

Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,
src/proto/popupwin.pro, src/testdir/test_popupwin.vim

Patch 8.1.1417

Problem: MS-Windows: resolve() does not resolve all components of the path.
(David Briscoe)

Solution: Do not bail out for a reparse point. (Yasuhiro Matsumoto,
closes #4211, closes #4447)

Files: src/os_mswin.c, src/testdir/test_functions.vim

Patch 8.1.1418

Problem: Win_execute() is not implemented yet.

Solution: Implement it.

Files: src/evalfunc.c, src/popupwin.c, src/testdir/test_execute_func.vim,
runtime/doc/popup.txt, runtime/doc/eval.txt

Patch 8.1.1419

Problem: Listener callbacks may be called recursively.

Solution: Set "updating_screen" while listener callbacks are invoked.

Files: src/change.c, src/screen.c, src/proto/screen.pro, src/ui.c

Patch 8.1.1420

Problem: Popup window size only uses first line length.

Solution: Use the longest line. (Ben Jackson, closes #4451) Also deal with
wrapping lines.

Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1421

Problem: Drawing "~" line in popup window.

Solution: Just draw text in the last line of the popup window.

Files: src/screen.c, src/structs.h, src/popupwin.c,
src/proto/popupwin.pro, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_05.dump,
src/testdir/dumps/Test_popupwin_06.dump

Patch 8.1.1422

Problem: Popup_getoptions() not implemented yet.

Solution: Implement it. (closes #4452)

Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,
src/proto/popupwin.pro, src/testdir/test_popupwin.vim

Patch 8.1.1423

Problem: Popup windows use options from current window and buffer.

Solution: Clear all local options when creating a popup window.

Files: src/popupwin.c, src/option.c, src/proto/option.pro,
src/testdir/test_popupwin.vim

Patch 8.1.1424

Problem: Crash when popup menu is deleted while waiting for char.

Solution: Bail out when pum_array was cleared.

Files: src/popupmnu.c

Patch 8.1.1425

Problem: Win_execute() does not set window pointers properly.

Solution: Use `switch_win_noblock()`. Also execute autocommands in a popup window.
Files: `src/window.c`, `src/proto/window.pro`, `src/evalfunc.c`, `src/autocmd.c`

Patch 8.1.1426

Problem: No test for syntax highlight in popup window.
Solution: Add a screenshot test. Update associated documentation. Avoid **'buftype'** being reset by `setbufvar()`.
Files: `runtime/doc/eval.txt`, `src/testdir/test_popupwin.vim`,
`src/testdir/dumps/Test_popupwin_10.dump`,
`src/testdir/dumps/Test_popupwin_11.dump`

Patch 8.1.1427 (after 8.1.1426)

Problem: Popup window screenshot test fails.
Solution: Add missing change to popup window code.
Files: `src/popupwin.c`

Patch 8.1.1428

Problem: `Popup_atcursor()` not implemented yet.
Solution: Implement it. (Yasuhiro Matsumoto, closes #4456)
Files: `runtime/doc/popup.txt`, `src/evalfunc.c`, `src/popupwin.c`,
`src/proto/popupwin.pro`, `src/testdir/test_popupwin.vim`

Patch 8.1.1429

Problem: "pos" option of popup window not supported yet.
Solution: Implement the option. Rename `popup_getposition()` to `popup_getpos()`.
Files: `src/structs.h`, `src/popupwin.c`, `src/proto/popupwin.pro`,
`runtime/doc/popup.txt`

Patch 8.1.1430

Problem: Popup window option "wrap" not supported.
Solution: Implement it.
Files: `src/popupwin.c`, `src/testdir/test_popupwin.vim`,
`src/testdir/dumps/Test_popupwin_wrap.dump`,
`src/testdir/dumps/Test_popupwin_nowrap.dump`

Patch 8.1.1431

Problem: Popup window listed as "Scratch".
Solution: List them as "Popup".
Files: `src/buffer.c`, `src/popupwin.c`, `src/testdir/test_popupwin.vim`,
`runtime/doc/popup.txt`, `runtime/doc/windows.txt`

Patch 8.1.1432 (after 8.1.1429)

Problem: Can't build with eval feature.
Solution: Add missing rename.
Files: `src/evalfunc.c`

Patch 8.1.1433

Problem: `Win_execute()` may leave popup window focused, eventually leading to a crash. (Bjorn Linse)
Solution: When previous window was closed, go to the first window.
Files: `src/window.c`, `src/testdir/test_popupwin.vim`

Patch 8.1.1434

Problem: Test 3 is old style.
Solution: Turn into a new style test. (Yegappan Lakshmanan, closes #4460)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test3.in, src/testdir/test3.ok,
src/testdir/test_cindent.vim

Patch 8.1.1435

Problem: Memory usage test is a bit too flaky.
Solution: Adjust the tolerances a bit. (Christian Brabandt)
Files: src/testdir/test_memory_usage.vim

Patch 8.1.1436

Problem: Writefile test fails when run under /tmp.
Solution: Adjust '**backupskip**'. (Kenta Sato, closes #4462)
Files: src/testdir/test_writefile.vim

Patch 8.1.1437

Problem: Code to handle callbacks is duplicated.
Solution: Add callback_T and functions to deal with it.
Files: src/structs.h, src/evalfunc.c, src/proto/evalfunc.pro,
src/change.c, src/channel.c, src/proto/channel.pro, src/buffer.c,
src/userfunc.c, src/proto/userfunc.pro, src/eval.c,
src/ex_cmds2.c, src/popupwin.c

Patch 8.1.1438

Problem: Some commands cause trouble in a popup window.
Solution: Add NOT_IN_POPUP_WINDOW.
Files: src/macros.h, src/popupwin.c, src/proto/popupwin.pro,
src/ex_docmd.c, src/ex_cmds2.c, src/window.c,
src/testdir/test_popupwin.vim

Patch 8.1.1439

Problem: Json_encode() is very slow for large results.
Solution: In the growarray use a growth of at least 50%. (Ken Takata,
closes #4461)
Files: src/misc2.c

Patch 8.1.1440

Problem: Win_execute() test fails.
Solution: Adjust the expected error number. Move to popup test.
Files: src/testdir/test_execute_func.vim, src/testdir/test_popupwin.vim

Patch 8.1.1441

Problem: Popup window filter not yet implemented.
Solution: Implement the popup filter.
Files: src/structs.h, runtime/doc/popup.txt, src/popupwin.c,
src/proto/popupwin.pro, src/window.c, src/getchar.c, src/screen.c,
src/misc2.c, src/proto/misc2.pro, src/vim.h,
src/testdir/test_popupwin.vim

Patch 8.1.1442

Problem: Popup windows not considered when the Vim window is resized.
(Ben Jackson)

Solution: Reallocate the w_lines structure. (closes #4467)
Files: src/screen.c

Patch 8.1.1443

Problem: Popup window padding and border not implemented yet.
Solution: Implement padding and border. Add core position and size to popup_getpos().
Files: src/structs.h, src/popupwin.c, src/screen.c,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_20.dump, runtime/doc/popup.txt

Patch 8.1.1444

Problem: Not using double line characters for popup border.
Solution: Use double line characters if using utf-8.
Files: src/screen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.1.1445

Problem: Popup window border highlight not implemented yet.
Solution: Implement the "borderhighlight" option.
Files: src/structs.h, src/popupwin.c, src/window.c, src/screen.c,
src/testdir/test_popupwin.vim, runtime/doc/popup.txt,
src/testdir/dumps/Test_popupwin_22.dump

Patch 8.1.1446

Problem: Popup window callback not implemented yet.
Solution: Implement the callback.
Files: runtime/doc/popup.txt, src/popupwin.c, src/structs.h,
src/evalfunc.c, src/window.c, src/testdir/test_popupwin.vim

Patch 8.1.1447

Problem: Not allowed to create an empty popup.
Solution: Remove restriction that there is some text. (closes #4470)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1448

Problem: Statusline is sometimes drawn on top of popup.
Solution: Redraw popups after the statusline. (Naruhiko Nishino,
closes #4468)
Files: src/screen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_behind.dump

Patch 8.1.1449

Problem: Popup text truncated at end of screen.
Solution: Move popup left if needed. Add the "fixed" property to disable that. (Ben Jackson, closes #4466)
Files: runtime/doc/popup.txt, src/popupwin.c, src/structs.h,
src/testdir/test_popupwin.vim

Patch 8.1.1450

Problem: Popup window positioning wrong when using padding or borders.
Solution: Fix computing the position.
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_corners.dump

Patch 8.1.1451

Problem: **CTRL-L** does not clear screen with a popup window.
Solution: Do not change the type to NOT_VALID. Redraw all windows.
(closes #4471)
Files: src/screen.c

Patch 8.1.1452

Problem: Line and col property of popup windows not properly checked.
Solution: Check for "+" or "-" sign.
Files: src/popupwin.c, src/dict.c, src/proto/dict.pro,
src/window.c, src/testdir/test_popupwin.vim

Patch 8.1.1453

Problem: Popup window "moved" property not implemented yet.
Solution: Implement it.
Files: src/main.c, src/edit.c, src/gui.c, src/globals.h, src/structs.h,
src/screen.c, src/popupwin.c, src/proto/popupwin.pro,
src/testdir/test_popupwin.vim, runtime/doc/popup.txt

Patch 8.1.1454

Problem: Build failure without the conceal feature.
Solution: Remove #ifdef.
Files: src/autocmd.c

Patch 8.1.1455

Problem: Popup_atcursor() not completely implemented.
Solution: Add the default for the "moved" property.
Files: src/popupwin.c, src/normal.c, src/vim.h,
src/testdir/test_popupwin.vim

Patch 8.1.1456

Problem: WinBar not redrawn after scrolling one line.
Solution: Exclude the winbar height when deciding what to redraw.
(closes #4473)
Files: src/screen.c, src/testdir/test_winbar.vim

Patch 8.1.1457

Problem: Cannot reuse a buffer when loading a screen dump.
Solution: Add the "bufnr" option.
Files: runtime/doc/eval.txt, src/structs.h, src/channel.c,
src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.1458

Problem: Crash when using gtags. (issue #4102)
Solution: Check for negative row or col in screen_puts_len(). (Christian
Brabandt)
Files: src/screen.c

Patch 8.1.1459

Problem: Popup window border looks bad when 'ambiwidth' is "double".
(Yasuhiro Matsumoto)
Solution: Only use line drawing characters when 'ambiwidth' is "single".
(Ken Takata, closes #4477)

Files: src/screen.c

Patch 8.1.1460

Problem: Popup window border characters may be wrong.

Solution: Reset the border characters for each popup. Correct use of 'ambiwidth'.

Files: src/screen.c

Patch 8.1.1461

Problem: Tests do not run or are not reliable on some systems.

Solution: Use "findstr" instead of "grep" on MS-Windows. Clear PROMPT_COMMAND in the terminal test. Delete temp file. Wait for output after executing a debug command. (Yegappan Lakshmanan, closes #4479)

Files: src/testdir/test_debugger.vim, src/testdir/test_envIRON.vim, src/testdir/test_filetype.vim, src/testdir/test_source.vim, src/testdir/test_terminal.vim

Patch 8.1.1462

Problem: MS-Windows: using special character requires quoting.

Solution: Add quotes. (Ken Takata)

Files: src/testdir/test_envIRON.vim

Patch 8.1.1463

Problem: Gcc warns for uninitialized variable.

Solution: Put usage inside "if". (Ken Takata)

Files: src/textprop.c

Patch 8.1.1464

Problem: Only 4-digit rgb termresponse is recognized.

Solution: Also recognize 2-digit rgb response. (closes #4486)

Files: src/term.c, src/test_termcodes.vim

Patch 8.1.1465

Problem: Allocating wrong amount of memory. (Yegappan Lakshmanan)

Solution: Use sizeof() for right type of struct.

Files: src/memfile_test.c

Patch 8.1.1466

Problem: Not updating priority on existing sign.

Solution: Set the sign priority. Add a test. (Yegappan Lakshmanan)

Files: src/sign.c, src/testdir/test_signs.vim, runtime/doc/eval.txt, runtime/doc/sign.txt

Patch 8.1.1467 (after 8.1.1465)

Problem: Cscope test fails.

Solution: Update expected text.

Files: src/testdir/test_cscope.vim

Patch 8.1.1468

Problem: The generated desktop files may be invalid.

Solution: Check validity with desktop-file-validate. (Christian Brabandt, Will Thompson, closes #4480)

Files: src/po/Makefile

Patch 8.1.1469

Problem: No test for checking the cursor style response.
Solution: Add a simple test. Also include the missing part of 8.1.1464.
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.1.1470

Problem: New Unicode character U+32FF missing from double-width table.
Solution: Add the character.
Files: src/mbyte.c

Patch 8.1.1471

Problem: **'background'** not correctly set for 2-digit rgb termresponse.
Solution: Adjust what digit to use. (closes #4495)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.1.1472

Problem: Add_termcap_entry() is not tested.
Solution: Add a simple test.
Files: src/testdir/test_termcodes.vim

Patch 8.1.1473

Problem: New resolve() implementation causes problem for plugins.
Solution: Only resolve a reparse point after checking it is needed. (Ken Takata, closes #4492)
Files: src/os_mswin.c, src/testdir/test_functions.vim

Patch 8.1.1474

Problem: **'ttybuiltin'** is not tested.
Solution: At least test that it doesn't break things.
Files: src/testdir/test_termcodes.vim

Patch 8.1.1475

Problem: Search string not displayed when **'rightleft'** is set.
Solution: Clear the right part of the old text. (closes #4488, closes #4489)
Files: src/search.c, src/testdir/test_search.vim

Patch 8.1.1476

Problem: No statistics displayed after running tests.
Solution: Summarize the test results. (Christian Brabandt, closes #4391)
Also make it possible to report a skipped file.
Files: src/Makefile, src/testdir/Makefile, src/testdir/summarize.vim,
src/testdir/runtest.vim, src/testdir/test_arabic.vim,
src/testdir/test_autochdir.vim, src/testdir/test_balloon.vim

Patch 8.1.1477

Problem: Test summary fails in the tiny version.
Solution: set **'nocompatible'**.
Files: Filelist, src/testdir/summarize.vim

Patch 8.1.1478

Problem: Still an error when running tests with the tiny version.
Solution: Do not try reading test.log
Files: src/testdir/Makefile, src/testdir/summarize.vim

Patch 8.1.1479

Problem: Change included for debugging only.
Solution: Restore the REDIR_TEST_TO_NULL line.
Files: src/testdir/Makefile

Patch 8.1.1480

Problem: Desktop file check doesn't run on CI.
Solution: Install the desktop-file-utils packages. (Christian Brabandt, closes #4498)
Files: .travis.yml

Patch 8.1.1481

Problem: Length for two-digit rgb termresponse is off by one.
Solution: Adjust the length. (closes #4494)
Files: src/term.c

Patch 8.1.1482

Problem: No test for wincol() depending on the 'number' option.
Solution: Add a couple of tests. (Christian Brabandt, closes #4500)
Files: src/testdir/test_gui.vim

Patch 8.1.1483

Problem: Skipped tests are not properly listed.
Solution: Throw a "Skipped" exception instead of using ":finish" or ":return".
Files: src/testdir/test_breakindent.vim, src/testdir/test_cdo.vim, src/testdir/test_channel.vim, src/testdir/test_clientserver.vim, src/testdir/test_balloon.vim, src/testdir/test_conceal.vim, src/testdir/test_debugger.vim, src/testdir/test_diffmode.vim, src/testdir/test_fold.vim, src/testdir/test_highlight.vim, src/testdir/test_popup.vim, src/testdir/test_popupwin.vim, src/testdir/test_search.vim, src/testdir/test_startup.vim, src/testdir/test_startup_utf8.vim, src/testdir/test_syntax.vim, src/testdir/test_tabpage.vim, src/testdir/test_termencoding.vim, src/testdir/test_terminal.vim, src/testdir/test_textprop.vim, src/testdir/test_timers.vim

Patch 8.1.1484

Problem: Some tests are slow.
Solution: Add timing to the test messages. Fix double free when quitting in VimLeavePre autocmd.
Files: src/testdir/runtest.vim, src/eval.c

Patch 8.1.1485

Problem: Double free when garbage_collect() is used in autocommand.
Solution: Have garbage collection also set the copyID in funccal_stack.
Files: src/eval.c, src/userfunc.c

Patch 8.1.1486

Problem: A listener change is merged even when it adds a line. (Paul Jolly)
Solution: Do not merge a change that adds or removes a line. (closes #4490)
Files: src/change.c, src/testdir/test_listener.vim

Patch 8.1.1487

Problem: Older msgfmt cannot generate proper .desktop file.
Solution: Add a configure check to not use this msgfmt version. (Ken Takata)
Files: src/configure.ac, src/auto/configure

Patch 8.1.1488

Problem: Summary of tests has incorrect failed count.
Solution: Add to the failed count instead of setting it. (Christian Brabandt)
Files: src/testdir/summarize.vim

Patch 8.1.1489

Problem: Sign order wrong when priority was changed.
Solution: Reorder signs when priority is changed. (Yegappan Lakshmanan, closes #4502)
Files: src/quickfix.c, src/sign.c, src/testdir/test_signs.vim

Patch 8.1.1490

Problem: When a single test fails the exit code is not set. (Daniel Hahler)
Solution: Add an exit command. (closes #4506)
Files: src/testdir/Makefile

Patch 8.1.1491

Problem: When skipping over code after an exception was thrown expression evaluation is aborted after a function call. (Ingo Karkat)
Solution: Do not fail if not executing the expression. (closes #4507)
Files: src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.1.1492

Problem: MS-Windows: when "!" is in 'guioptions' "!:start" fails.
Solution: Do not use a terminal window when the shell command begins with "!:start". (Yasuhiro Matsumoto, closes #4504)
Files: src/misc2.c, src/os_win32.c

Patch 8.1.1493

Problem: Redrawing with popups is slow and causes flicker.
Solution: Avoid clearing and redrawing using a zindex mask.
Files: src/globals.h, src/screen.c, src/proto/screen.pro, src/popupwin.c, src/popupmnu.c

Patch 8.1.1494 (after 8.1.1493)

Problem: Build failure.
Solution: Add missing changes.
Files: src/structs.h

Patch 8.1.1495 (after 8.1.1494)

Problem: Memory access error.
Solution: Use the correct size for clearing the popup mask.
Files: src/screen.c

Patch 8.1.1496

Problem: Popup window height is not recomputed.
Solution: Recompute the height when needed.
Files: src/popupwin.c, src/testdir/dumps/Test_popupwin_06.dump

Patch 8.1.1497

Problem: Accessing memory beyond allocated space.
Solution: Check column before accessing popup mask.
Files: src/screen.c

Patch 8.1.1498

Problem: ":write" increments b:changedtick even though nothing changed.
(Daniel Hahler)
Solution: Only increment b:changedtick if the modified flag is reset.
Files: src/change.c, src/proto/change.pro, runtime/doc/eval.txt,
src/buffer.c, src/ex_cmds2.c, src/fileio.c, src/memline.c,
src/undo.c

Patch 8.1.1499

Problem: Ruler not updated after popup window was removed.
Solution: use popup_mask in screen_puts().
Files: src/screen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_07.dump,
src/testdir/dumps/Test_popupwin_08.dump

Patch 8.1.1500

Problem: Wrong shell command when building with VIMDLL and "!" in
'guioptions'.
Solution: Add check for GUI in use. (Ken Takata)
Files: src/misc2.c

Patch 8.1.1501

Problem: New behavior of b:changedtick not tested.
Solution: Add a few test cases. (Daniel Hahler)
Files: src/testdir/test_changedtick.vim

Patch 8.1.1502

Problem: Cannot play any sound.
Solution: Use libcanberra if available. Add sound functions.
Files: src/configure.ac, src/auto/configure, src/config.h.in,
src/Makefile, src/sound.c, src/proto/sound.pro, src/proto.h,
src/evalfunc.c, src/feature.h, runtime/doc/eval.txt, Filelist,
src/version.c, src/testdir/test_sound.vim, src/testdir/silent.wav,
src/testdir/Make_all.mak, .travis.yml

Patch 8.1.1503

Problem: Sound test fails on Travis.
Solution: Set AUDIODEV to "null".
Files: .travis.yml

Patch 8.1.1504

Problem: Sound test still fails on Travis.
Solution: Add more lines to the install section.
Files: .travis.yml

Patch 8.1.1505

Problem: Running "make clean" twice gives errors.
Solution: Add "-f" to "rm". (closes #4516)
Files: src/testdir/Makefile

Patch 8.1.1506

Problem: Syntax error in Travis config.
Solution: Set AUDIODEV in another section.
Files: .travis.yml

Patch 8.1.1507

Problem: Sound test still fails on Travis.
Solution: Try another dummy sound approach.
Files: .travis.yml

Patch 8.1.1508

Problem: Sound keeps failing on Travis.
Solution: Throw a skipped exception in the test.
Files: src/testdir/test_sound.vim

Patch 8.1.1509

Problem: Cmdline_row can become negative, causing a crash.
Solution: Make sure cmdline_row does not become negative. (closes #4102)
Files: src/misc1.c

Patch 8.1.1510

Problem: A plugin cannot easily expand a command like done internally.
Solution: Add the expandcmd() function. (Yegappan Lakshmanan, closes #4514)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/testdir/test_expand.vim

Patch 8.1.1511

Problem: Matches in a popup window are not displayed properly.
Solution: Do display matches in a popup window. (closes #4517)
Files: src/screen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_matches.dump

Patch 8.1.1512

Problem: ch_evalexpr() hangs when used recursively. (Paul Jolly)
Solution: Change ch_block_id from a single number to a list of IDs to wait on.
Files: src/channel.c, src/structs.h

Patch 8.1.1513

Problem: All popup functionality is in functions, except :popupclear.
Solution: Add popup_clear() for consistency. Also rename sound_stopall() to sound_clear().
Files: src/ex_cmds.h, src/ex_cmdidxs.h, src/evalfunc.c, src/popupwin.c,
src/proto/popupwin.pro, src/sound.c, src/proto/sound.pro,
src/testdir/test_popupwin.vim src/testdir/test_sound.vim,
runtime/doc/eval.txt runtime/doc/popup.txt

Patch 8.1.1514 (after 8.1.1492)

Problem: MS-Windows: wrong shell command with ! in 'guioptions'.
Solution: Do not check for ! in 'guioptions' when applying 'shellxquote'.
(Yasuhiro Matsumoto, closes #4519)
Files: src/misc2.c

Patch 8.1.1515

Problem: Memory leak reported for sound when build with EXITFREE.
Solution: Free sound stuff when exiting.
Files: src/misc2.c

Patch 8.1.1516

Problem: Time reported for a test measured wrong.
Solution: Move the computation to the end of RunTheTest(). (Ozaki Kiichi, closes #4520)
Files: src/testdir/runtest.vim

Patch 8.1.1517

Problem: When a popup changes all windows are redrawn.
Solution: Only update the lines that were affected. Add a file for profiling popup windows efficiency.
Files: src/screen.c, src/proto/screen.pro, src/ui.c, src/popupwin.c, src/globals.h, src/testdir/popupbounce.vim, Filelist

Patch 8.1.1518

Problem: Crash when setting **'columns'** while a popup is visible.
Solution: Recompute all positions when clearing the screen. (closes #4467)
Files: src/screen.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_04a.dump

Patch 8.1.1519

Problem: **'backupskip'** may contain duplicates.
Solution: Add the P_NODUP flag. (Tom Ryder)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.1.1520

Problem: Popup windows are ignored when dealing with mouse position
Solution: Find the mouse position inside a popup window. Allow for modeless selection.
Files: src/ui.c, src/proto/ui.pro, src/popupwin.c, src/proto/popupwin.pro, src/screen.c, src/beval.c, src/edit.c, src/evalfunc.c, src/gui.c, src/normal.c, src/structs.h

Patch 8.1.1521

Problem: When a popup window is closed the buffer remains.
Solution: Wipe out the buffer.
Files: src/window.c, src/testdir/test_popupwin.vim

Patch 8.1.1522

Problem: Popup_notification() not implemented yet.
Solution: Implement it.
Files: src/popupwin.c, src/proto/popupwin.pro, src/evalfunc.c, src/structs.h, src/testdir/test_popupwin.vim, runtime/doc/popup.txt, src/testdir/dumps/Test_popupwin_notify_01.dump, src/testdir/dumps/Test_popupwin_notify_02.dump

Patch 8.1.1523

Problem: Cannot show range of buffer lines in popup window.
Solution: Add the "firstline" property. (closes #4523)
Files: src/popupwin.c, src/structs.h, runtime/doc/popup.txt,

src/testdir/test_popupwin.vim,
testdir/dumps/Test_popupwin_firstline.dump

Patch 8.1.1524

Problem: Tests are silently skipped.
Solution: Throw an exception for skipped tests in more places.
Files: src/testdir/test_assert.vim, src/testdir/test_paste.vim,
src/testdir/shared.vim, src/testdir/test_crypt.vim,
src/testdir/test_cscope.vim, src/testdir/test_digraph.vim,
src/testdir/test_float_func.vim, src/testdir/test_gui.vim,
src/testdir/test_gui_init.vim, src/testdir/test_history.vim,
src/testdir/test_langmap.vim, src/testdir/test_listlbr.vim,
src/testdir/test_listlbr_utf8.vim, src/testdir/test_lua.vim,
src/testdir/test_makeencoding.vim,
src/testdir/test_matchadd_conceal.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_memory_usage.vim, src/testdir/test_menu.vim,
src/testdir/test_mksession.vim,
src/testdir/test_mksession_utf8.vim,
src/testdir/test_netbeans.vim, src/testdir/test_paste.vim,
src/testdir/test_perl.vim, src/testdir/test_profile.vim,
src/testdir/test_prompt_buffer.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_pyx2.vim,
src/testdir/test_pyx3.vim, src/testdir/test_quickfix.vim,
src/testdir/test_quotestar.vim, src/testdir/test_reftime.vim,
src/testdir/test_ruby.vim, src/testdir/test_sha256.vim,
src/testdir/test_shortpathname.vim, src/testdir/test_signals.vim,
src/testdir/test_signs.vim, src/testdir/test_spell.vim,
src/testdir/test_syntax.vim, src/testdir/test_tcl.vim,
src/testdir/test_termcodes.vim, src/testdir/test_terminal.vim,
src/testdir/test_terminal_fail.vim,
src/testdir/test_textobjects.vim, src/testdir/test_textprop.vim,
src/testdir/test_timers.vim, src/testdir/test_vartabs.vim,
src/testdir/test_winbar.vim, src/testdir/test_windows_home.vim,
src/testdir/test_xxd.vim

Patch 8.1.1525

Problem: Cannot move a popup window with the mouse.
Solution: Add the "drag" property and make it possible to drag a popup window by its border.
Files: src/popupwin.c, src/proto/popupwin.pro, src/structs.h, src/ui.c,
src/window.c, src/proto/window.pro, runtime/doc/popup.txt

Patch 8.1.1526

Problem: No numerical value for the patchlevel.
Solution: Add v:versionlong.
Files: src/version.c, src/eval.c, src/vim.h, runtime/doc/eval.txt,
src/testdir/test_eval_stuff.vim

Patch 8.1.1527

Problem: When moving a popup window over the command line it is not redrawn.
Solution: Redraw the command line. Move popup redrawing code to the popupwin file.

Files: src/screen.c, src/proto/screen.pro, src/popupwin.c,
src/proto/popupwin.pro, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_drag_01.dump,
src/testdir/dumps/Test_popupwin_drag_02.dump

Patch 8.1.1528

Problem: Popup_any_visible() is unused.
Solution: Remove it.
Files: src/popupwin.c, src/proto/popupwin.pro

Patch 8.1.1529

Problem: Libcanberra is linked with even when not used.
Solution: Have configure check for libcanberra only when wanted.
(suggestions by Libor Bukata)
Files: src/feature.h, src/configure.ac, src/auto/configure, src/Makefile

Patch 8.1.1530

Problem: Travis config is not optimal.
Solution: Remove system conditions. Do not use excluding matrix. Cache OSX
results. (Ozaki Kiichi, closes #4521)
Files: .travis.yml

Patch 8.1.1531

Problem: Clipboard type name is inconsistent.
Solution: Rename VimClipboard to Clipboard_T.
Files: src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/gui_mac.c,
src/proto/gui_mac.pro, src/gui_x11.c, src/proto/gui_x11.pro,
src/ops.c, src/proto/ops.pro, src/os_qnx.c, src/proto/os_qnx.pro,
src/os_unix.c, src/proto/os_unix.pro, src/ui.c, src/proto/ui.pro,
src/winclip.c, src/proto/winclip.pro, src/globals.h, src/proto.h

Patch 8.1.1532 (after 8.1.1531)

Problem: Build fails.
Solution: Add missing changes.
Files: src/vim.h

Patch 8.1.1533

Problem: GUI build fails on Mac.
Solution: Change VimClipboard type in non-C file.
Files: src/os_macosx.m

Patch 8.1.1534

Problem: Modeless selection in popup window selects too much.
Solution: Restrict the selection to inside of the popup window.
Files: src/vim.h, src/ui.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_select_01.dump,
src/testdir/dumps/Test_popupwin_select_02.dump

Patch 8.1.1535 (after 8.1.1534)

Problem: Popup select test fails on Mac.
Solution: Skip test if clipboard feature not available.
Files: src/testdir/test_popupwin.vim

Patch 8.1.1536 (after 8.1.1534)

Problem: Popup select test still fails on Mac.
Solution: Set '**clipboard**' to "autoselect"
Files: src/testdir/test_popupwin.vim

Patch 8.1.1537

Problem: Using "tab" for popup window can be confusing.
Solution: Use "tabpage". (Hirohito Higashi, closes #4532)
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.1538

Problem: Cannot specify highlighting for notifications.
Solution: Use the PopupNotification group if it exists. Add a minimal width to notifications.
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_notify_01.dump,
src/testdir/dumps/Test_popupwin_notify_02.dump

Patch 8.1.1539

Problem: Not easy to define a variable and lock it.
Solution: Add ":const". (Ryuichi Hayashida, closes #4541)
Files: runtime/doc/eval.txt, src/eval.c, src/ex_cmdidxs.h, src/ex_cmds.h,
src/proto/eval.pro, src/testdir/Make_all.mak,
src/testdir/test_const.vim

Patch 8.1.1540 (after 8.1.1539)

Problem: Cannot build without the +eval feature.
Solution: Define ex_const if needed.
Files: src/ex_docmd.c

Patch 8.1.1541

Problem: Check for ASAN is not reliable.
Solution: Check the version output. (Dominique Pelle, closes #4543)
Files: src/testdir/test_memory_usage.vim

Patch 8.1.1542

Problem: An OptionSet autocommand does not get enough info.
Solution: Add v:option_command, v:option_oldlocal and v:option_oldglobal. (Latrice Wilgus, closes #4118)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt,
runtime/doc/version8.txt, src/eval.c, src/option.c, src/structs.h,
src/testdir/test_autocmd.vim, src/vim.h

Patch 8.1.1543

Problem: Const test fails with small features.
Solution: Don't unlet non-existing variables.
Files: src/testdir/test_const.vim

Patch 8.1.1544

Problem: Some balloon tests don't run when they can.
Solution: Split GUI balloon tests off into a separate file. (Ozaki Kiichi, closes #4538) Change the feature check into a command for consistency.

Files: Filelist, src/testdir/Make_all.mak, src/testdir/check.vim, src/testdir/test_arabic.vim, src/testdir/test_balloon.vim, src/testdir/test_balloon_gui.vim, src/testdir/test_crypt.vim, src/testdir/test_csscope.vim, src/testdir/test_digraph.vim, src/testdir/test_float_func.vim, src/testdir/test_gui.vim, src/testdir/test_gui_init.vim, src/testdir/test_history.vim, src/testdir/test_langmap.vim, src/testdir/test_listlbr.vim, src/testdir/test_listlbr_utf8.vim, src/testdir/test_lua.vim, src/testdir/test_makeencoding.vim, src/testdir/test_matchadd_conceal.vim, src/testdir/test_matchadd_conceal_utf8.vim, src/testdir/test_memory_usage.vim, src/testdir/test_menu.vim, src/testdir/test_mksession.vim, src/testdir/test_mksession_utf8.vim, src/testdir/test_netbeans.vim, src/testdir/test_paste.vim, src/testdir/test_perl.vim, src/testdir/test_popupwin.vim, src/testdir/test_profile.vim, src/testdir/test_prompt_buffer.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim, src/testdir/test_pyx2.vim, src/testdir/test_pyx3.vim, src/testdir/test_quickfix.vim, src/testdir/test_quotestar.vim, src/testdir/test_reftime.vim, src/testdir/test_ruby.vim, src/testdir/test_sha256.vim, src/testdir/test_shortpathname.vim, src/testdir/test_signals.vim, src/testdir/test_signs.vim, src/testdir/test_spell.vim, src/testdir/test_syntax.vim, src/testdir/test_tcl.vim, src/testdir/test_termcodes.vim, src/testdir/test_terminal.vim, src/testdir/test_terminal_fail.vim, src/testdir/test_textobjects.vim, src/testdir/test_textprop.vim, src/testdir/test_timers.vim, src/testdir/test_vartabs.vim, src/testdir/test_winbar.vim, src/testdir/test_windows_home.vim, src/testdir/test_xxd.vim

Patch 8.1.1545

Problem: When the screen is too small there is no message about that.
(Daniel Hahler)
Solution: Do not use :cquit. (closes #4534)
Files: src/testdir/runtest.vim

Patch 8.1.1546

Problem: In some tests **'tags'** is set but not restored. (Daniel Hahler)
Solution: Restore **'tags'**. (closes #4535)
Files: src/testdir/test_autocmd.vim, src/testdir/test_cmdline.vim, src/testdir/test_options.vim, src/testdir/test_tagcase.vim, src/testdir/test_tagjump.vim, src/testdir/test_taglist.vim

Patch 8.1.1547

Problem: Functionality of bt_nofile() is confusing.
Solution: Split into bt_nofile() and bt_nofilename().
Files: src/buffer.c, src/proto/buffer.pro, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/fileio.c, src/popupmnu.c, src/quickfix.c

Patch 8.1.1548

Problem: Popup_dialog() is not implemented.
Solution: Implement popup_dialog() and popup_filter_yesno().
Files: src/popupwin.c, src/proto/popupwin.pro, src/evalfunc.c,

src/structs.h, src/globals.h, src/testdir/test_popupwin.vim,
runtime/doc/popup.txt

Patch 8.1.1549 (after 8.1.1547)

Problem: Quickfix test fails.
Solution: Negate result of bt_quickfix().
Files: src/quickfix.c

Patch 8.1.1550

Problem: When a popup has left padding text may be cut off.
Solution: Add the border and padding when computing the size.
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_20.dump,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.1.1551

Problem: Warning for shadowing popup_dragwin. (Dominique Pelle)
Solution: Add missing change.
Files: src/ui.c

Patch 8.1.1552

Problem: Cursor position is wrong after sign column appears or disappears.
(Yegappan Lakshmanan)
Solution: Call changed_line_abv_curs() instead of changed_cline_bef_curs().
Files: src/sign.c, src/testdir/test_signs.vim,
src/testdir/dumps/Test_sign_cursor_01.dump,
src/testdir/dumps/Test_sign_cursor_02.dump

Patch 8.1.1553

Problem: Not easy to change the text in a popup window.
Solution: Add popup_settext(). (Ben Jackson, closes #4549)
Also display a space for an empty popup.
Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,
src/proto/popupwin.pro,
src/testdir/dumps/Test_popup_settext_01.dump,
src/testdir/dumps/Test_popup_settext_02.dump,
src/testdir/dumps/Test_popup_settext_03.dump,
src/testdir/dumps/Test_popup_settext_04.dump,
src/testdir/dumps/Test_popup_settext_05.dump,
src/testdir/dumps/Test_popup_settext_06.dump,
src/testdir/test_popupwin.vim

Patch 8.1.1554 (after 8.1.1539)

Problem: Docs and tests for :const can be improved.
Solution: Improve documentation, add a few more tests. (Ryuichi Hayashida,
closes #4551)
Files: runtime/doc/eval.txt, src/testdir/test_const.vim

Patch 8.1.1555

Problem: NOT_IN_POPUP_WINDOW is confusing. (Andy Massimino)
Solution: Rename to ERROR_IF_POPUP_WINDOW().
Files: src/popupwin.c, src/proto/popupwin.pro, src/macros.h,
src/ex_cmds2.c, src/ex_docmd.c, src/window.c

Patch 8.1.1556

Problem: The command displayed to show a failing screenshot does not include the "testdir" directory.
Solution: Prefix the directory name so that it can be copy-pasted.
Files: src/testdir/screendump.vim

Patch 8.1.1557

Problem: Compiler warning for unused variables in tiny version. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/option.c

Patch 8.1.1558

Problem: Popup_menu() and popup_filter_menu() are not implemented yet.
Solution: Implement the functions. Fix that centering didn't take the border and padding into account.
Files: runtime/doc/popup.txt, src/popupwin.c, src/proto/popupwin.pro, src/evalfunc.c, src/screen.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_menu_01.dump, src/testdir/dumps/Test_popupwin_menu_02.dump, src/testdir/dumps/Test_popupwin_menu_03.dump, src/testdir/dumps/Test_popupwin_drag_01.dump, src/testdir/dumps/Test_popupwin_drag_02.dump

Patch 8.1.1559

Problem: Popup window title property not implemented yet.
Solution: Implement the title property.
Files: runtime/doc/popup.txt, src/popupwin.c, src/structs.h, src/window.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_menu_01.dump, src/testdir/dumps/Test_popupwin_menu_02.dump, src/testdir/dumps/Test_popupwin_title.dump

Patch 8.1.1560

Problem: Popup window hidden option not implemented yet.
Solution: Implement the hidden option.
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1561

Problem: Popup_setoptions() is not implemented yet.
Solution: Implement popup_setoptions(). Also add more fields to popup_getoptions().
Files: runtime/doc/popup.txt, src/popupwin.c, src/proto/popupwin.pro, src/dict.c, src/proto/dict.pro, src/evalfunc.c, src/testdir/test_popupwin.vim, src/testdir/runtest.vim

Patch 8.1.1562

Problem: Popup window not always redrawn after popup_setoptions().
Solution: Force a redraw.
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_23.dump

Patch 8.1.1563

Problem: Crash when using closures.

Solution: Set reference in varlist of funccal when running the garbage collector. (Ozaki Kiichi, closes #4554, closes #4547)
Files: src/testdir/test_vimscript.vim, src/userfunc.c

Patch 8.1.1564

Problem: Sign column takes up space. (Adam Stankiewicz)
Solution: Optionally put signs in the number column. (Yegappan Lakshmanan, closes #4555, closes #4515)
Files: runtime/doc/options.txt, src/option.c, src/screen.c, src/testdir/test_signs.vim

Patch 8.1.1565

Problem: MS-Windows: no sound support.
Solution: Add sound support for MS-Windows. (Yasuhiro Matsumoto, Ken Takata, closes #4522)
Files: runtime/doc/eval.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/sound.c, src/testdir/test_sound.vim

Patch 8.1.1566

Problem: Error message when terminal closes while it is not in the current tab.
Solution: Also set "do_set_w_closing" when using the special autocommand window. (closes #4552)
Files: src/terminal.c

Patch 8.1.1567

Problem: localtime_r() does not respond to \$TZ changes.
Solution: If \$TZ changes then call tzset(). (Tom Ryder)
Files: src/auto/configure, src/config.h.in, src/configure.ac, src/evalfunc.c, src/memline.c, src/proto/memline.pro, src/testdir/test_functions.vim, src/undo.c

Patch 8.1.1568 (after 8.1.1567)

Problem: Strftime() test fails on MS-Windows.
Solution: Skip the check for using the \$TZ environment variable.
Files: src/testdir/test_functions.vim

Patch 8.1.1569

Problem: Cannot build with signs but without diff feature.
Solution: Move #ifdef. (Tom Ryder)
Files: src/screen.c

Patch 8.1.1570

Problem: Icon signs not displayed properly in the number column.
Solution: Display them properly. (Yegappan Lakshmanan, closes #4559)
Files: src/gui.c, src/screen.c, src/testdir/test_signs.vim

Patch 8.1.1571

Problem: textprop highlight starts too early if just after a tab.
Solution: Check if still drawing a previous character. (closes #4558)
Files: src/screen.c, src/testdir/test_textprop.vim, src/testdir/dumps/Test_textprop_tab.dump

Patch 8.1.1572 (after 8.1.1569)

Problem: Compiler warnings with tiny build. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/screen.c

Patch 8.1.1573 (after 8.1.1571)

Problem: Textprop test fails if screenshots do not work.
Solution: Add check for screenshots working.
Files: src/testdir/test_textprop.vim

Patch 8.1.1574

Problem: Tabpage option not yet implemented for popup window.
Solution: Implement tabpage option, also for popup_getoptions().
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.1575

Problem: Callbacks may be garbage collected.
Solution: Set reference in callbacks. (Ozaki Kiichi, closes #4564)
Files: src/buffer.c, src/channel.c, src/eval.c, src/ex_cmds2.c,
src/popupwin.c, src/proto/buffer.pro, src/proto/popupwin.pro,
src/terminal.c, src/testdir/test_listener.vim,
src/testdir/test_popupwin.vim, src/testdir/test_prompt_buffer.vim,
src/userfunc.c

Patch 8.1.1576

Problem: Compiler warning for unused argument.
Solution: Add "UNUSED" annotation. (Dominique Pelle, closes #4570)
Files: src/ui.c

Patch 8.1.1577

Problem: Command line redrawn for +arabic without Arabic characters.
(Dominique Pelle)
Solution: Check if there actually are any Arabic characters. Do redraw
after displaying incsearch. (closes #4569)
Files: src/ex_getln.c

Patch 8.1.1578

Problem: MS-Windows: pathdef.c should depend on build options.
Solution: Generate pathdef.c in the object directory. Fix dependencies.
(Ken Takata, closes #4565)
Files: .gitignore, .hgignore, src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.1579

Problem: Dict and list could be GC'ed while displaying error in a timer.
(Yasuhiro Matsumoto)
Solution: Block garbage collection when executing a timer. Add
test_garbagecollect_soon(). Add "no_wait_return" to
test_override(). (closes #4571)
Files: src/dict.c, src/testdir/test_timers.vim, src/evalfunc.c,
runtime/doc/eval.txt

Patch 8.1.1580

Problem: Cannot make part of a popup transparent.
Solution: Add the "mask" option.

Files: runtime/doc/popup.txt, src/popupwin.c, src/screen.c, src/structs.h, src/window.c, src/ui.c, src/vim.h, src/globals.h, src/testdir/dumps/Test_popupwin_mask_1.dump, src/testdir/dumps/Test_popupwin_mask_2.dump

Patch 8.1.1581

Problem: Shared functions for testing are disorganised.
Solution: Group functions in script files. (Ozaki Kiichi, closes #4573)
Files: Filelist, src/testdir/screendump.vim, src/testdir/shared.vim, src/testdir/term_util.vim, src/testdir/test_mksession.vim, src/testdir/test_suspend.vim, src/testdir/test_terminal.vim, src/testdir/test_timers.vim, src/testdir/view_util.vim

Patch 8.1.1582

Problem: Cannot build with +textprop but without +timers.
Solution: Add #ifdef. (Ola Söder, closes #4574)
Files: src/popupwin.c

Patch 8.1.1583

Problem: Set_ref_in_list() only sets ref in items.
Solution: Rename to set_ref_in_list_items() to avoid confusion.
Files: src/eval.c, src/proto/eval.pro, src/if_lua.c, src/popupwin.c, src/userfunc.c, src/if_py_both.h

Patch 8.1.1584

Problem: The evalfunc.c file is getting too big.
Solution: Move channel and job related functions to channel.c.
Files: src/channel.c, src/evalfunc.c, src/proto/channel.pro

Patch 8.1.1585

Problem: :let-heredoc does not trim enough.
Solution: Trim indent from the contents based on the indent of the first line. Use let-heredoc in more tests.
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_balloon.vim, src/testdir/test_cindent.vim, src/testdir/test_const.vim, src/testdir/test_debugger.vim, src/testdir/test_functions.vim, src/testdir/test_goto.vim, src/testdir/test_gui.vim, src/testdir/test_highlight.vim, src/testdir/test_join.vim, src/testdir/test_let.vim, src/testdir/test_memory_usage.vim, src/testdir/test_messages.vim, src/testdir/test_mksession_utf8.vim, src/testdir/test_normal.vim, src/testdir/test_popup.vim, src/testdir/test_popupwin.vim, src/testdir/test_profile.vim, src/testdir/test_quickfix.vim, src/testdir/test_xxd.vim

Patch 8.1.1586

Problem: Error number used in two places.
Solution: Renumber one. (Ken Takata)
Files: runtime/doc/popup.txt, src/popupwin.c

Patch 8.1.1587

Problem: Redraw problem when sign icons in the number column.
Solution: Clear and redraw when changing related options. Right align the sign icon in the GUI. (Yegappan Lakshmanan, closes #4578)

Files: src/gui.c, src/option.c

Patch 8.1.1588

Problem: In :let-heredoc line continuation is recognized.

Solution: Do not consume line continuation. (Ozaki Kiichi, closes #4580)

Files: src/autocmd.c, src/digraph.c, src/eval.c, src/evalfunc.c,
src/ex_cmds.c, src/ex_cmds.h, src/ex_cmds2.c, src/ex_docmd.c,
src/ex_getln.c, src/normal.c, src/ops.c, src/proto/autocmd.pro,
src/proto/ex_cmds2.pro, src/proto/ex_docmd.pro,
src/proto/ex_getln.pro, src/proto/userfunc.pro,
src/testdir/test_let.vim, src/testdir/test_startup.vim,
src/userfunc.c

Patch 8.1.1589

Problem: Popup window does not indicate scroll position.

Solution: Add a scrollbar.

Files: runtime/doc/popup.txt, src/popupwin.c, src/structs.h,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_firstline.dump,
src/testdir/dumps/Test_popupwin_scroll_1.dump,
src/testdir/dumps/Test_popupwin_scroll_2.dump,
src/testdir/dumps/Test_popupwin_scroll_3.dump,
src/testdir/dumps/Test_popupwin_scroll_4.dump

Patch 8.1.1590

Problem: Popup window test fails.

Solution: Add "scrollbar" to expected result.

Files: src/testdir/test_popupwin.vim

Patch 8.1.1591

Problem: On error garbage collection may free memory in use.

Solution: Reset may_garbage_collect when evaluating expression mapping.
Add tests. (Ozaki Kiichi, closes #4579)

Files: src/ex_cmds2.c, src/getchar.c, src/testdir/test_mapping.vim,
src/testdir/test_timers.vim, src/testdir/test_vimscript.vim

Patch 8.1.1592

Problem: May start file dialog while exiting.

Solution: Ignore the "browse" modifier when exiting. (Ozaki Kiichi,
closes #4582)

Files: src/ex_cmds.c, src/terminal.c

Patch 8.1.1593

Problem: Filetype not detected for C++ header files without extension.

Solution: Recognize the file by the Emacs file mode. (Dmitry Ilyin,
closes #4593)

Files: runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.1.1594

Problem: May still start file dialog while exiting.

Solution: Ignore the "browse" modifier in another place when exiting.
(Ozaki Kiichi, closes #4582)

Files: src/ex_cmds.c

Patch 8.1.1595

Problem: MS-Windows with VIMDLL: colors wrong in the GUI.
Solution: Do not set the terminal colors when not using the GUI. (Ken Takata, closes #4588)
Files: src/syntax.c

Patch 8.1.1596

Problem: When resizing the screen may draw popup in wrong position. (Masato Nishihata)
Solution: Check the popup is not outside of the screen. (fixes #4592)
Files: src/popupwin.c

Patch 8.1.1597

Problem: Cannot scroll a popup window with the mouse.
Solution: If the popup window has a scrollbar let the mouse scroll wheel scroll the window.
Files: runtime/doc/popup.txt, src/normal.c, src/popupwin.c, src/screen.c, src/testdir/dumps/Test_popupwin_firstline.dump, src/testdir/dumps/Test_popupwin_scroll_1.dump, src/testdir/dumps/Test_popupwin_scroll_2.dump, src/testdir/dumps/Test_popupwin_scroll_3.dump, src/testdir/dumps/Test_popupwin_scroll_5.dump, src/testdir/dumps/Test_popupwin_scroll_6.dump, src/testdir/dumps/Test_popupwin_scroll_7.dump

Patch 8.1.1598

Problem: Update to test file missing.
Solution: Update the popup window test file.
Files: src/testdir/test_popupwin.vim

Patch 8.1.1599

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Add a dummy assignment.
Files: src/popupwin.c, src/normal.c

Patch 8.1.1600

Problem: Cannot specify highlighting for popup window scrollbar.
Solution: Add "scrollbarhighlight" and "thumbhighlight" options.
Files: src/popupwin.c, src/structs.h, src/window.c, src/testdir/dumps/Test_popupwin_scroll_5.dump, src/testdir/dumps/Test_popupwin_scroll_6.dump, src/testdir/dumps/Test_popupwin_scroll_7.dump

Patch 8.1.1601

Problem: Missing changes to popup window test file.
Solution: Add those changes.
Files: src/testdir/test_popupwin.vim

Patch 8.1.1602

Problem: Popup window cannot overflow on the left or right.
Solution: Only set the "fixed" option when it is in the dict. Set w_leftcol to allow for the popup overflowing on the left and use it when applying the mask.
Files: src/popupwin.c

Patch 8.1.1603

Problem: Crash when using unknown highlighting in text property.
Solution: Check for zero highlight ID.
Files: src/screen.c, src/testdir/test_textprop.vim

Patch 8.1.1604

Problem: Popup window scroll test is flaky.
Solution: Add a delay between scroll events.
Files: src/testdir/test_popupwin.vim

Patch 8.1.1605

Problem: Vim may delay processing messages on a json channel. (Pontus Leitzler)
Solution: Try parsing json when checking if there is readahead.
Files: src/channel.c

Patch 8.1.1606

Problem: On a narrow screen ":hi" output is confusing.
Solution: Insert a space between highlight group name and "xxx". (Masato Nishihaga, closes #4599)
Files: src/syntax.c, src/testdir/test_highlight.vim

Patch 8.1.1607

Problem: Popup window scrollbar does not respond to click.
Solution: Mouse click in scrollbar scrolls by one line.
Files: src/popupwin.c, src/proto/popupwin.pro, src/structs.h, src/ui.c, src/normal.c, runtime/doc/popup.txt, src/testdir/dumps/Test_popupwin_scroll_8.dump, src/testdir/dumps/Test_popupwin_scroll_9.dump

Patch 8.1.1608

Problem: The evalfunc.c file is too big.
Solution: Move sign functionality to sign.c.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/sign.c, src/proto/sign.pro

Patch 8.1.1609

Problem: The user cannot easily close a popup window.
Solution: Add the "close" property. (mostly by Masato Nishihata, closes #4601)
Files: runtime/doc/popup.txt, src/popupwin.c, src/proto/popupwin.pro, src/structs.h, src/testdir/dumps/Test_popupwin_close_01.dump, src/testdir/dumps/Test_popupwin_close_02.dump, src/testdir/dumps/Test_popupwin_close_03.dump, src/testdir/test_popupwin.vim, src/ui.c

Patch 8.1.1610

Problem: There is no way to add or load a buffer without side effects.
Solution: Add the bufadd() and bufload() functions.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.1.1611

Problem: Bufadd() reuses existing buffer without a name.
Solution: When the name is empty always create a new buffer.
Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.1.1612

Problem: Cannot show an existing buffer in a popup window.
Solution: Support buffer number argument in popup_create().
Files: src/buffer.c, src/proto/buffer.pro, src/evalfunc.c,
src/popupwin.c, src/vim.h, src/normal.c, src/screen.c, src/ui.c,
src/window.c, src/testdir/test_popupwin.vim, runtime/doc/popup.txt

Patch 8.1.1613

Problem: Popup window test fails with Athena and Motif.
Solution: Compute the highlight attribute when the GUI is not active.
Files: src/syntax.c

Patch 8.1.1614

Problem: **'numberwidth'** can only go up to 10.
Solution: Allow up to 20. (Charlie Stanton, closes #4584)
Files: runtime/doc/options.txt, src/option.c, src/screen.c,
src/testdir/gen_opt_test.vim, src/testdir/test_options.vim

Patch 8.1.1615

Problem: Crash when passing buffer number to popup_create(). (Yasuhiro Matsumoto)
Solution: Initialize the window properly.
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1616

Problem: Build failure with gcc on Amiga.
Solution: Add missing header includes. (Ola Söder, closes #4603)
Files: src/os_amiga.h

Patch 8.1.1617

Problem: No test for popup window with mask and position fixed.
Solution: Add a couple of screenshots. Fix detected problems.
Files: src/popupwin.c, src/structs.h, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_mask_1.dump,
src/testdir/dumps/Test_popupwin_mask_2.dump,
src/testdir/dumps/Test_popupwin_mask_3.dump,
src/testdir/dumps/Test_popupwin_mask_4.dump

Patch 8.1.1618

Problem: Amiga-like systems quickly run out of stack.
Solution: Reserve a Megabyte stack. (Ola Söder, closes #4608)
Files: src/os_amiga.c

Patch 8.1.1619

Problem: Tests are not run with GUI on Travis.
Solution: Add a testgui job. (Ozaki Kiichi, closes #4609)
Files: .travis.yml, src/testdir/test_highlight.vim,
src/testdir/test_mapping.vim, src/testdir/test_timers.vim

Patch 8.1.1620

Problem: No test for popup window with border and mask.
Solution: Add this popup window, fix problems.
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_mask_1.dump,
src/testdir/dumps/Test_popupwin_mask_2.dump,
src/testdir/dumps/Test_popupwin_mask_3.dump,
src/testdir/dumps/Test_popupwin_mask_4.dump

Patch 8.1.1621

Problem: Amiga: time.h included twice.
Solution: Remove include from evalfunc.c, move outside of #ifdef in
os_amiga.h. (Ola Söder, closes #4607)
Files: src/evalfunc.c, src/os_amiga.h

Patch 8.1.1622

Problem: Wrong width if displaying a lot of lines in a popup window.
Solution: Accurately compute the line overflow.
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_firstline.dump

Patch 8.1.1623

Problem: Display wrong with signs in narrow number column.
Solution: Increase the numbercolumn width if needed. (Yegappan Lakshmanan,
closes #4606)
Files: src/option.c, src/screen.c, src/sign.c, src/testdir/test_signs.vim

Patch 8.1.1624

Problem: When testing in the GUI may try to run gvim in a terminal.
Solution: Add the -v argument. (Yee Cheng Chin, closes #4605) Don't skip
tests that work now.
Files: src/testdir/shared.vim, src/testdir/term_util.vim,
src/testdir/test_mapping.vim, src/testdir/test_timers.vim

Patch 8.1.1625

Problem: Script line numbers are not exactly right.
Solution: Handle heredoc and continuation lines better. (Ozaki Kiichi,
closes #4611, closes #4511)
Files: src/ex_cmds2.c, src/proto/ex_cmds2.pro,
src/testdir/test_vimscript.vim, src/userfunc.c

Patch 8.1.1626

Problem: No test for closing a popup window with a modified buffer.
Solution: Add a test. Add "popups" to getbufinfo().
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_popupwin.vim

Patch 8.1.1627

Problem: Header file contains mixed comment style.
Solution: Use // style comments.
Files: src/structs.h

Patch 8.1.1628

Problem: Popup window functions not in list of functions.
Solution: Add popup window functions to the list of functions. Reorganise

the popup window help.
Files: runtime/doc/eval.txt, runtime/doc/popup.txt,
runtime/doc/usr_41.txt

Patch 8.1.1629

Problem: Terminal function help is in the wrong file.
Solution: Move the function details to terminal.txt.
Files: runtime/doc/eval.txt, runtime/doc/terminal.txt

Patch 8.1.1630

Problem: Various small problems.
Solution: Various small improvements.
Files: src/gui_beval.c, src/list.c, src/menu.c, src/message.c,
src/misc2.c, src/testdir/test_terminal.vim, src/os_vms_conf.h,
src/testdir/Make_vms.mms

Patch 8.1.1631

Problem: Displaying signs is inefficient.
Solution: Avoid making multiple calls to get information about a placed
sign. (Yegappan Lakshmanan, closes #4586)
Files: src/proto/sign.pro, src/screen.c, src/sign.c, src/structs.h

Patch 8.1.1632

Problem: Build with EXITFREE but without +arabic fails.
Solution: Rename the function and adjust #ifdefs. (closes #4613)
Files: src/ex_getln.c, src/proto/ex_getln.pro, src/misc2.c

Patch 8.1.1633

Problem: Cannot generate prototypes with X11 but without GUI.
Solution: Include X11/Intrinsic.h.
Files: src/gui.h

Patch 8.1.1634

Problem: Terminal test fails when term_getansicolors() is missing.
Diff test fails without +rightleft. (Dominique Pelle)
Solution: Check if term_getansicolors() is supported. (closes #4597)
Files: src/testdir/test_terminal.vim, src/testdir/test_diffmode.vim

Patch 8.1.1635

Problem: Warnings for unused variables in small version. (John Marriott)
Solution: Adjust #ifdefs.
Files: src/screen.c

Patch 8.1.1636

Problem: Crash when popup has fitting scrollbar. (Trygve Aaberge)
Solution: Don't divide by zero if the scrollbar just fits. (closes #4615)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1637

Problem: After running tests and clean the XfakeHOME directory remains.
Solution: Use "rm -rf". (Hirohito Higashi)
Files: src/testdir/Makefile, src/testdir/Make_amiga.mak

Patch 8.1.1638

Problem: Running tests leaves some files behind.
Solution: Delete the files. (Ozaki Kiichi, closes #4617)
Files: src/testdir/test_functions.vim, src/testdir/test_popupwin.vim

Patch 8.1.1639

Problem: Changing an autoload name into a script file name is inefficient.
Solution: Remember the last replaced #. (Ozaki Kiichi, closes #4618)
Files: src/eval.c

Patch 8.1.1640

Problem: The CursorHold autocommand takes down a balloon. (Paul Jolly)
Solution: Ignore the CursorHold pseudo-key.
Files: src/getchar.c, src/testdir/test_balloon.vim,
src/testdir/dumps/Test_balloon_eval_term_01.dump,
src/testdir/dumps/Test_balloon_eval_term_01a.dump

Patch 8.1.1641

Problem: Garbage collection may run at a wrong moment. (Trygve Aaberge)
Solution: Postpone garbage collection while parsing messages. (closes #4620)
Files: src/misc2.c

Patch 8.1.1642 (after 8.1.0374)

Problem: May use uninitialized variable. (Patrick Palka)
Solution: Initialize variables earlier. (closes #4623)
Files: src/screen.c, src/testdir/test_number.vim

Patch 8.1.1643

Problem: Sign placement is wrong when '**foldcolumn**' is set.
Solution: Adjust the column computation. (Yee Cheng Chin, closes #4627)
Files: src/gui.c

Patch 8.1.1644

Problem: Sound test does not work on Travis.
Solution: Use "sg" command to enable audio. (Ozaki Kiichi, closes #4624)
Files: .travis.yml

Patch 8.1.1645

Problem: Cannot use a popup window for a balloon.
Solution: Add popup_beval(). Add the "mousemoved" property. Add the screenpos() function.
Files: src/popupwin.c, src/proto/popupwin.pro, src/move.c,
src/proto/move.pro, src/beval.c, src/proto/beval.pro,
src/evalfunc.c, src/popupmnu.c, src/normal.c,
src/testdir/test_popupwin.vim, src/testdir/test_cursor_func.vim,
runtime/doc/popup.txt, runtime/doc/eval.txt,
runtime/doc/usr_41.txt,
src/testdir/dumps/Test_popupwin_beval_1.dump,
src/testdir/dumps/Test_popupwin_beval_2.dump,
src/testdir/dumps/Test_popupwin_beval_3.dump

Patch 8.1.1646 (after 8.1.1645)

Problem: build failure
Solution: Add changes to structure.
Files: src/structs.h

Patch 8.1.1647

Problem: Build error with GTK and hangulinput feature, im_get_status() defined twice. (Dominique Pelle)
Solution: Adjust im_get_status(). (closes #4628)
Files: src/hangulin.c, src/mbyte.c

Patch 8.1.1648

Problem: MS-Windows: build error with normal features.
Solution: Adjust #ifdef for find_word_under_cursor().
Files: src/beval.c, src/proto/beval.pro

Patch 8.1.1649

Problem: Illegal memory access when closing popup window.
Solution: Get w_next before closing the window.
Files: src/popupwin.c

Patch 8.1.1650

Problem: Warning for using uninitialized variable. (Tony Mechelynck)
Solution: Simplify the code by always using the mouse coordinates.
Files: src/beval.c

Patch 8.1.1651

Problem: Suspend test is flaky on some systems.
Solution: Wait for the shell prompt to show. (Yee Cheng Chin, closes #4632)
Files: src/testdir/test_suspend.vim

Patch 8.1.1652

Problem: GUI: popup window doesn't close on mouse movement. (Paul Jolly)
Solution: Generate mouse-move events when a popup window is visible.
Files: src/gui.c, src/globals.h

Patch 8.1.1653

Problem: Ubsan warns for possibly passing NULL pointer.
Solution: Skip code when length is zero. (Dominique Pelle, closes #4631)
Files: src/channel.c

Patch 8.1.1654

Problem: GUI: screen updates from '**balloonexpr**' are not displayed.
Solution: Update the screen if needed. Also avoid the cursor being displayed in the wrong position.
Files: src/beval.c

Patch 8.1.1655

Problem: Popup window border drawn wrong with multibyte char. (Marcin Szamotulski)
Solution: Correct check in mb_fix_col(). (closes #4635)
Files: src/mbyte.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_24.dump

Patch 8.1.1656

Problem: Popup window width is wrong when using Tabs. (Paul Jolly)
Solution: Count tabs correctly. (closes #4637)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,

src/testdir/dumps/Test_popupwin_11.dump

Patch 8.1.1657

Problem: Terminal: screen updates from 'balloonexpr' are not displayed.

Solution: Update the screen if needed. Fix the word position for "mousemoved".

Files: src/beval.c, src/proto/beval.pro, src/popupwin.c, src/normal.c, src/proto/normal.pro

Patch 8.1.1658

Problem: Debug statements included in patch.

Solution: Remove the debug statements.

Files: src/normal.c, src/popupwin.c

Patch 8.1.1659

Problem: Popup window "mousemoved" values not correct.

Solution: Convert text column to mouse column.

Files: src/popupwin.c, runtime/doc/popup.txt

Patch 8.1.1660

Problem: Assert_fails() does not fail inside try/catch.

Solution: Set trylevel to zero. (Ozaki Kiichi, closes #4639)

Files: src/eval.c, src/testdir/test_assert.vim

Patch 8.1.1661

Problem: Cannot build with +textprop but without +balloon_eval.

Solution: Adjust #ifdefs. (closes #4645)

Files: src/proto.h

Patch 8.1.1662

Problem: Cannot build uninstal.exe with some version of MinGW.

Solution: Add -lole32. (Rene Nyffenegger, closes #4646)

Files: src/Make_cyg_ming.mak

Patch 8.1.1663

Problem: Compiler warning for using size_t.

Solution: Add type cast. (Mike Williams)

Files: src/popupwin.c

Patch 8.1.1664

Problem: GUI resize may cause changing Rows at a bad time. (Dominique Pelle)

Solution: Postpone resizing while updating the screen.

Files: src/term.c

Patch 8.1.1665

Problem: Crash when popup window with mask is below the screen.

Solution: Correct boundary check.

Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_mask_5.dump

Patch 8.1.1666

Problem: Click in popup window scrollbar with border doesn't scroll.

Solution: Correct column for the border. (Naruhiko Nishino, closes #4650)

Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_scroll_9.dump

Patch 8.1.1667

Problem: Flags for Ex commands may clash with other symbols.

Solution: Prepend with EX_.

Files: src/ex_cmds.h, src/evalfunc.c, src/ex_docmd.c, src/ex_getln.c,
src/usercmd.c, src/syntax.c

Patch 8.1.1668

Problem: Popup window test is a bit flaky on some systems.

Solution: Clear the command line. (Naruhiko Nishino, closes #4656)

Files: src/testdir/test_popupwin.vim

Patch 8.1.1669

Problem: Travis: test results section is closed even when some tests failed.

Solution: Only close the section on success. (Daniel Hahler, closes #4659)

Files: .travis.yml

Patch 8.1.1670

Problem: Sign column not always properly aligned.

Solution: Use "col" only after it was calculated. (Yee Cheng Chin,
closes #4649)

Files: src/gui.c

Patch 8.1.1671

Problem: Copying a blob may result in it being locked.

Solution: Reset v_lock. (Ken Takata, closes #4648)

Files: src/blob.c, src/testdir/test_blob.vim

Patch 8.1.1672 (after 8.1.1667)

Problem: "make cmdidxs" doesn't work.

Solution: Update macro names. (Naruhiko Nishino, closes #4660)

Files: src/create_cmdidxs.vim

Patch 8.1.1673

Problem: Cannot easily find the popup window at a certain position.

Solution: Add popup_locate().

Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,
src/proto/popupwin.pro, src/testdir/test_popupwin.vim

Patch 8.1.1674

Problem: Script to check a colorscheme can be improved.

Solution: Match the whole group name. Don't warn for what is usually omitted.

Files: runtime/colors/tools/check_colors.vim

Patch 8.1.1675

Problem: Listener list not correctly updated on listener_remove().

Solution: Only set "prev" when not removing a listener. Return one if the
listener was found and removed.

Files: src/change.c

Patch 8.1.1676

Problem: "maxwidth" of popup window does not always work properly.
Solution: Adjust the computation. (Naruhiko Nishino, closes #4653)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_menu_maxwidth_1.dump

Patch 8.1.1677

Problem: Tests get stuck when running into an existing swapfile.
Solution: Set v:swapchoice to "q" and report an error. (Daniel Hahler,
closes #4644)
Files: src/testdir/runtest.vim

Patch 8.1.1678

Problem: When using popup_menu() does not scroll to show the selected line.
Solution: Scroll the text. (Naruhiko Nishino, closes #4651)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_menu_scroll_1.dump,
src/testdir/dumps/Test_popupwin_menu_scroll_2.dump,
src/testdir/dumps/Test_popupwin_menu_scroll_3.dump,
src/testdir/dumps/Test_popupwin_menu_scroll_4.dump,
src/testdir/dumps/Test_popupwin_menu_scroll_5.dump,
src/testdir/dumps/Test_popupwin_menu_scroll_6.dump

Patch 8.1.1679

Problem: Test using SwapExists autocommand file may fail.
Solution: Remove the SwapExists autocommand.
Files: src/testdir/test_window_cmd.vim

Patch 8.1.1680

Problem: The command table is not well aligned.
Solution: Adjust indent.
Files: src/ex_cmds.h

Patch 8.1.1681

Problem: Insert stray "{" when listener gets buffer line. (Paul Jolly)
Solution: Flush the cached line after invoking listeners. (closes #4455)
Files: src/memline.c, src/testdir/test_listener.vim

Patch 8.1.1682

Problem: Placing a larger number of signs is slow.
Solution: Add functions for dealing with a list of signs. (Yegappan
Lakshmanan, closes #4636)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/proto/sign.pro, src/sign.c, src/testdir/test_signs.vim

Patch 8.1.1683

Problem: Dictionary with string keys is longer than needed.
Solution: Use `*{key: val}` for literal keys.
Files: runtime/doc/eval.txt, src/eval.c, src/dict.c, src/proto/dict.pro,
src/testdir/test_listdict.vim, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_07.dump,
src/testdir/dumps/Test_popupwin_mask_2.dump,
src/testdir/dumps/Test_popupwin_mask_3.dump,
src/testdir/dumps/Test_popupwin_mask_4.dump,
src/testdir/dumps/Test_popupwin_mask_5.dump,

src/testdir/dumps/Test_popupwin_scroll_2.dump,
src/testdir/dumps/Test_popupwin_scroll_3.dump,
src/testdir/dumps/Test_popupwin_scroll_4.dump

Patch 8.1.1684

Problem: Profiling functionality is spread out.
Solution: Put profiling functionality in profiler.c. (Yegappan Lakshmanan, closes #4666)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_dice.mak, src/Make_manx.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_sas.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/ex_cmds2.c, src/globals.h, src/profiler.c, src/proto.h, src/proto/ex_cmds2.pro, src/proto/profiler.pro, src/proto/userfunc.pro, src/structs.h, src/userfunc.c

Patch 8.1.1685

Problem: Missing file in distributed file list.
Solution: Add profiler.pro
Files: Filelist

Patch 8.1.1686

Problem: "*" of "*{" is recognized as multiply operator. (Yasuhiro Matsumoto)
Solution: Check for the "{".
Files: src/eval.c, src/testdir/test_listdict.vim

Patch 8.1.1687

Problem: The evalfunc.c file is too big.
Solution: Move testing support to a separate file.
Files: Filelist, src/evalfunc.c, src/eval.c, src/proto/eval.pro, src/testing.c, src/proto/testing.pro, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/proto.h

Patch 8.1.1688

Problem: Old makefiles are no longer useful.
Solution: Delete the makefiles, they most likely don't work anyway.
Files: Filelist, src/Make_dice.mak, src/Make_manx.mak, src/Make_sas.mak

Patch 8.1.1689

Problem: Profiling code is spread out.
Solution: Move more profiling code to profiler.c. (Yegappan Lakshmanan, closes #4668)
Files: src/ex_cmds2.c, src/profiler.c, src/proto/ex_cmds2.pro, src/proto/profiler.pro, src/proto/userfunc.pro, src/structs.h, src/userfunc.c

Patch 8.1.1690

Problem: Default padding for popup window menu is too much.
Solution: Only add padding left and right.
Files: runtime/doc/popup.txt, src/popupwin.c, src/testdir/dumps/Test_popupwin_menu_01.dump, src/testdir/dumps/Test_popupwin_menu_02.dump, src/testdir/dumps/Test_popupwin_menu_maxwidth_1.dump,

```
src/testdir/dumps/Test_popupwin_menu_scroll_1.dump,  
src/testdir/dumps/Test_popupwin_menu_scroll_2.dump,  
src/testdir/dumps/Test_popupwin_menu_scroll_3.dump,  
src/testdir/dumps/Test_popupwin_menu_scroll_4.dump,  
src/testdir/dumps/Test_popupwin_menu_scroll_5.dump,  
src/testdir/dumps/Test_popupwin_menu_scroll_6.dump
```

Patch 8.1.1691

Problem: Diff test fails on some systems. (Elimar Riesebieter)
Solution: Add a term_wait() call.
Files: src/testdir/test_diffmode.vim

Patch 8.1.1692

Problem: Using *{} for literal dict is not backwards compatible. (Yasuhiro Matsumoto)
Solution: Use ~{} instead.
Files: runtime/doc/eval.txt runtime/doc/popup.txt, src/eval.c,
src/testdir/test_listdict.vim src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_07.dump,
src/testdir/dumps/Test_popupwin_mask_2.dump,
src/testdir/dumps/Test_popupwin_mask_3.dump,
src/testdir/dumps/Test_popupwin_mask_4.dump,
src/testdir/dumps/Test_popupwin_mask_5.dump,
src/testdir/dumps/Test_popupwin_scroll_2.dump,
src/testdir/dumps/Test_popupwin_scroll_3.dump,
src/testdir/dumps/Test_popupwin_scroll_4.dump

Patch 8.1.1693

Problem: Syntax coloring and highlighting is in one big file.
Solution: Move the highlighting to a separate file. (Yegappan Lakshmanan, closes #4674)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/globals.h, src/highlight.c, src/proto.h,
src/proto/highlight.pro, src/proto/syntax.pro, src/structs.h,
src/syntax.c

Patch 8.1.1694

Problem: The RUN_VIM variable is longer than needed.
Solution: Shorten RUN_VIM. (Daniel Hahler, closes #4643)
Files: src/testdir/Makefile, src/testdir/shared.vim

Patch 8.1.1695

Problem: Windows 10: crash when cursor is at bottom of terminal.
Solution: Position the cursor before resizing. (Yasuhiro Matsumoto, closes #4679)
Files: src/os_win32.c

Patch 8.1.1696

Problem: MSVC: link command line is too long.
Solution: Use the @<< mechanism to pass the arguments via a file. (Christian Brabandt)
Files: src/Make_mvc.mak

Patch 8.1.1697

Problem: Cannot build with MSVC.
Solution: Remove the backslashes after the @<< mechanism.
Files: src/Make_mvc.mak

Patch 8.1.1698

Problem: Appveyor build with MSVC fails.
Solution: Remove the sed command
Files: ci/appveyor.bat

Patch 8.1.1699

Problem: Highlight_ga can be local instead of global.
Solution: Move highlight_ga into highlight.c. (Yegappan Lakshmanan, closes #4675)
Files: src/globals.h, src/highlight.c, src/proto/highlight.pro, src/structs.h, src/syntax.c

Patch 8.1.1700

Problem: Listener callback called for the wrong buffer.
Solution: Invoke listeners before calling ml_append_int().
Files: src/memline.c

Patch 8.1.1701

Problem: Appveyor build with MSVC fails puts progress bar in log.
Solution: Adjust the sed command. (Ken Takata)
Files: ci/appveyor.bat

Patch 8.1.1702

Problem: Compiler warning for uninitialized variable.
Solution: Initialize it. (Christian Brabandt)
Files: src/gui.c

Patch 8.1.1703

Problem: Breaking out of loop by checking window pointer is insufficient.
Solution: Check the window ID and the buffer number. (closes #4683)
Files: src/misc2.c

Patch 8.1.1704

Problem: C-R C-W does not work after C-G when using **'incsearch'**.
Solution: Put cursor at end of the match. (Yasuhiro Matsumoto, closes #4664)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.1.1705

Problem: Using ~{} for a literal dict is not nice.
Solution: Use #{} instead.
Files: runtime/doc/eval.txt runtime/doc/popup.txt, src/eval.c, src/testdir/test_listdict.vim src/testdir/test_popupwin.vim

Patch 8.1.1706

Problem: Typo in #ifdef.
Solution: Change PROT to PROTO.
Files: src/beval.c

Patch 8.1.1707

Problem: Coverity warns for possibly using a NULL pointer.
Solution: Change the logic to make sure no NULL pointer is used.
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1708

Problem: Coverity warns for using uninitialized variable.
Solution: Set the start col when col is set.
Files: src/beval.c

Patch 8.1.1709

Problem: Coverity warns for possibly using a NULL pointer.
Solution: Make sure no NULL pointer is used.
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1710

Problem: Coverity found dead code.
Solution: Remove merging of listener changes.
Files: src/change.c

Patch 8.1.1711

Problem: Listener callback called at the wrong moment
Solution: Invoke listeners before calling ml_delete_int(). (closes #4657)
Files: src/memline.c

Patch 8.1.1712

Problem: Signs in number column cause text to be misaligned.
Solution: Improve alignment. (Yasuhiro Matsumoto, closes #4694)
Files: src/screen.c, src/testdir/test_signs.vim

Patch 8.1.1713

Problem: Highlighting cursor line only works with popup_menu().
Solution: Add the "cursorline" property. (Naruhiko Nishino, closes #4671)
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/dumps/Test_popupwin_cursorline_1.dump,
src/testdir/dumps/Test_popupwin_cursorline_2.dump,
src/testdir/dumps/Test_popupwin_cursorline_3.dump,
src/testdir/dumps/Test_popupwin_cursorline_4.dump,
src/testdir/dumps/Test_popupwin_cursorline_5.dump,
src/testdir/dumps/Test_popupwin_cursorline_6.dump,
src/testdir/dumps/Test_popupwin_menu_filter_1.dump,
src/testdir/dumps/Test_popupwin_menu_filter_2.dump,
src/testdir/dumps/Test_popupwin_menu_filter_3.dump,
src/testdir/dumps/Test_popupwin_menu_filter_4.dump,
src/testdir/test_popupwin.vim, src/vim.h

Patch 8.1.1714

Problem: Cannot preview a file in a popup window.
Solution: Add the **'previewpopup'** option.
Files: runtime/doc/windows.txt, runtime/doc/options.txt, src/popupwin.c,
src/proto/popupwin.pro, src/option.c, src/option.h, src/ex_cmds.c,
src/testdir/dumps/Test_popupwin_previewpopup_1.dump,
src/testdir/dumps/Test_popupwin_previewpopup_2.dump,
src/ex_docmd.c, src/testdir/gen_opt_test.vim

Patch 8.1.1715

Problem: Emoji characters are seen as word characters for spelling. (Gautam Iyer)
Solution: Exclude class 3 from word characters.
Files: src/spell.c

Patch 8.1.1716

Problem: Old style comments are wasting space
Solution: Use new style comments in option header file. (closes #4702)
Files: src/option.h

Patch 8.1.1717

Problem: Last char in menu popup window highlighted.
Solution: Do not highlight an extra character twice.
Files: src/screen.c, src/testdir/test_popupwin.vim,
 src/testdir/dumps/Test_popupwin_menu_04.dump

Patch 8.1.1718

Problem: Popup menu highlighting does not look good.
Solution: Highlight the whole window line. Fix that sign line HL is not displayed in a window with a background color.
Files: src/popupwin.c, src/sign.c, src/proto/sign.pro, src/screen.c,
 src/testdir/dumps/Test_popupwin_menu_scroll_1.dump,
 src/testdir/dumps/Test_popupwin_menu_scroll_2.dump,
 src/testdir/dumps/Test_popupwin_menu_scroll_3.dump,
 src/testdir/dumps/Test_popupwin_menu_scroll_4.dump,
 src/testdir/dumps/Test_popupwin_menu_scroll_5.dump,
 src/testdir/dumps/Test_popupwin_menu_scroll_6.dump,
 src/testdir/dumps/Test_popupwin_menu_01.dump,
 src/testdir/dumps/Test_popupwin_menu_02.dump,
 src/testdir/dumps/Test_popupwin_menu_04.dump

Patch 8.1.1719

Problem: Popup too wide when **'showbreak'** is set.
Solution: Set window width when computing line length. (closes #4701)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
 src/testdir/dumps/Test_popupwin_showbreak.dump

Patch 8.1.1720

Problem: Crash with very long %[] pattern. (Reza Mirzazade farkhani)
Solution: Check for reg_toolong. (closes #4703)
Files: src/regexp.c, src/testdir/test_regexp_utf8.vim

Patch 8.1.1721

Problem: Build failure with normal features without netbeans interface.
Solution: Enable signs when using the text properties feature.
Files: src/feature.h

Patch 8.1.1722

Problem: Error when scriptversion is 2 a making a dictionary access.
Solution: Parse the subscript even when not evaluating the sub-expression. (closes #4704)
Files: src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.1.1723

Problem: Heredoc assignment has no room for new features. (FUJIWARA Takuya)
Solution: Require the marker does not start with a lower case character.
(closes #4705)
Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_let.vim

Patch 8.1.1724

Problem: Too much overhead checking for **CTRL-C** while processing text.
Solution: Increase BREAKCHECK_SKIP. Remove the difference for when built
with the GUI. (suggested by Andy Massimino, closes #4708)
Files: src/misc1.c, src/screen.c, src/feature.h

Patch 8.1.1725

Problem: MS-Windows: E325 message may use incorrect date format.
Solution: Convert strftime() result to '**encoding**'. Also make the message
translatable. (Ken Takata, closes #4685, closes #4681)
Files: src/memline.c

Patch 8.1.1726

Problem: The eval.txt help file is too big.
Solution: Split off testing support to testing.txt. Move function details
to where the functionality is explained.
Files: runtime/doc/Makefile, runtime/doc/eval.txt,
runtime/doc/testing.txt, runtime/doc/sign.txt,
runtime/doc/textprop.txt, runtime/doc/help.txt,
runtime/doc/channel.txt, runtime/doc/tags

Patch 8.1.1727

Problem: Code for viminfo support is spread out.
Solution: Move to code to viminfo.c. (Yegappan Lakshmanan, closes #4686)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/buffer.c,
src/eval.c, src/ex_cmds.c, src/ex_docmd.c, src/globals.h,
src/proto.h, src/proto/buffer.pro, src/proto/eval.pro,
src/proto/ex_cmds.pro, src/proto/viminfo.pro, src/structs.h,
src/viminfo.c

Patch 8.1.1728

Problem: Wrong place for command line history viminfo support.
Solution: Move it to viminfo.c.
Files: src/ex_getln.c, src/proto/ex_getln.pro, src/viminfo.c,
src/structs.h

Patch 8.1.1729

Problem: Heredoc with trim not properly handled in function.
Solution: Allow for missing indent. (FUJIWARA Takuya, closes #4713)
Files: src/userfunc.c, src/testdir/test_let.vim

Patch 8.1.1730

Problem: Wrong place for mark viminfo support.
Solution: Move it to viminfo.c. (Yegappan Lakshmanan, closes #4716)
Files: src/README.md, src/mark.c, src/proto/mark.pro,
src/proto/viminfo.pro, src/structs.h, src/viminfo.c

Patch 8.1.1731

Problem: Command line history not read from viminfo on startup.
Solution: Get history length after initializing it.
Files: src/viminfo.c, src/testdir/test_viminfo.vim

Patch 8.1.1732

Problem: Completion in cmdwin does not work for buffer-local commands.
Solution: Use the right buffer. (closes #4711)
Files: src/usercmd.c, src/testdir/test_ins_complete.vim

Patch 8.1.1733

Problem: The man ftplugin leaves an empty buffer behind.
Solution: Don't make new window and edit, use split. (Jason Franklin)
Files: runtime/ftplugin/man.vim, src/testdir/test_man.vim

Patch 8.1.1734

Problem: The evalfunc.c file is too big.
Solution: Move some functions to other files.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/json.c,
src/proto/json.pro src/window.c, src/proto/window.pro,
src/highlight.c, src/proto/highlight.pro, src/globals.h

Patch 8.1.1735 (after 8.1.1734)

Problem: Can't build with tiny features.
Solution: Add missing #ifdefs.
Files: src/json.c, src/highlight.c

Patch 8.1.1736

Problem: Viminfo support is spread out.
Solution: Move more viminfo code to viminfo.c. (Yegappan Lakshmanan,
closes #4717) Reorder code to make most functions static.
Files: src/fileio.c, src/ops.c, src/option.c, src/proto/ops.pro,
src/proto/option.pro, src/proto/search.pro, src/proto/viminfo.pro,
src/search.c, src/structs.h, src/viminfo.c, src/ex_cmds.c,
src/proto/ex_cmds.pro

Patch 8.1.1737

Problem: :args command that outputs one line gives more prompt.
Solution: Only output line break if needed. (Daniel Hahler, closes #4715)
Files: src/version.c, src/testdir/test_arglist.vim

Patch 8.1.1738

Problem: Testing lambda with timer is slow.
Solution: Do not test timer accuracy, only that it works. (Daniel Hahler,
closes #4723)
Files: src/testdir/test_lambda.vim

Patch 8.1.1739

Problem: Deleted match highlighting not updated in other window.
Solution: Mark the window for refresh. (closes #4720) Also fix that
ambi-width check clears with wrong attributes.
Files: src/term.c, src/highlight.c, src/testdir/test_match.vim,
src/testdir/dumps/Test_matchdelete_1.dump

Patch 8.1.1740

Problem: Exepath() doesn't work for "bin/cat".
Solution: Check for any path separator. (Daniel Hahler, closes #4724, closes #4710)
Files: src/evalfunc.c, src/os_unix.c, src/testdir/test_functions.vim

Patch 8.1.1741

Problem: Cleared/added match highlighting not updated in other window. (Andy Massimino)
Solution: Mark the right window for refresh.
Files: src/highlight.c, src/testdir/test_match.vim, src/testdir/dumps/Test_matchclear_1.dump, src/testdir/dumps/Test_matchadd_1.dump

Patch 8.1.1742

Problem: Still some match functions in evalfunc.c.
Solution: Move them to highlight.c.
Files: src/evalfunc.c, src/highlight.c, src/proto/highlight.pro, src/ex_docmd.c

Patch 8.1.1743

Problem: **'hlsearch'** and match highlighting in the wrong place.
Solution: Move highlighting from inside screen functions to highlight.c.
Files: src/screen.c, src/highlight.c, src/proto/highlight.pro

Patch 8.1.1744

Problem: Build error without the conceal feature.
Solution: Define variables also without the conceal feature.
Files: src/screen.c

Patch 8.1.1745

Problem: Compiler warning for unused argument.
Solution: Add UNUSED. Change comments to new style.
Files: src/highlight.c

Patch 8.1.1746

Problem: ":dl" is seen as ":dlist" instead of ":delete".
Solution: Do not use cmdidxs2[] if the length is 1. (closes #4721)
Files: src/ex_docmd.c, src/testdir/test_excmd.vim, src/testdir/Make_all.mak

Patch 8.1.1747

Problem: Compiler warning for unused variables. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/screen.c

Patch 8.1.1748 (after 8.1.1737)

Problem: :args output is not aligned.
Solution: Output a line break after the last item in a row.
Files: src/version.c

Patch 8.1.1749

Problem: Coverity warns for using negative index.
Solution: Move using index inside "if".

Files: src/viminfo.c

Patch 8.1.1750

Problem: Depending on the terminal width :version may miss a line break.

Solution: Add a line break when needed.

Files: src/version.c

Patch 8.1.1751

Problem: When redrawing popups plines_win() may be called often.

Solution: Pass a cache to mouse_comp_pos().

Files: src/ui.c, src/proto/ui.pro, src/beval.c, src/evalfunc.c,
src/popupwin.c

Patch 8.1.1752

Problem: Resizing hashtable is inefficient.

Solution: Avoid resizing when the final size is predictable.

Files: src/hashtab.c, src/proto/hashtab.pro, src/popupwin.c

Patch 8.1.1753

Problem: Use of popup window mask is inefficient.

Solution: Precompute and cache the mask.

Files: src/popupwin.c

Patch 8.1.1754 (after 8.1.1753)

Problem: Build failure.

Solution: Add missing change to window struct.

Files: src/structs.h

Patch 8.1.1755

Problem: Leaking memory when using a popup window mask.

Solution: Free the cached mask.

Files: src/window.c

Patch 8.1.1756

Problem: Autocommand that splits window messes up window layout.

Solution: Disallow splitting a window while closing one. In ":all" give an error when moving a window will not work.

Files: src/buffer.c, src/window.c, src/testdir/test_window_cmd.vim

Patch 8.1.1757

Problem: Text added with appendbuflines() to another buffer isn't displayed.

Solution: Update topline. (partly by Christian Brabandt, closes #4718)

Files: src/evalfunc.c, src/testdir/test_buflines.vim,
src/testdir/dumps/Test_appendbuflines_1.dump

Patch 8.1.1758

Problem: Count of g\$ not used correctly when text is not wrapped.

Solution: Do use the count. (Christian Brabandt, closes #4729, closes #4566)

Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.1.1759

Problem: No mode char for terminal mapping from maparg().

Solution: Check for TERMINAL mode. (closes #4735)

Files: src/getchar.c, src/testdir/test_maparg.vim

Patch 8.1.1760

Problem: Extra line break for wrapping output of :args.
Solution: Avoid the extra line break. (Daniel Hahler, closes #4737)
Files: src/version.c, src/testdir/test_arglist.vim

Patch 8.1.1761

Problem: Filetype "vuejs" causes problems for some users.
Solution: Rename to "vue".
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.1762

Problem: Some filetype rules are in the wrong place.
Solution: Move to the right place. Add a few more tests.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.1763

Problem: Evalfunc.c is still too big.
Solution: Move dict and list functions to a better place.
Files: src/evalfunc.c, src/dict.c, src/proto/dict.pro, src/list.c,
src/proto/list.pro, src/blob.c, src/proto/blob.pro

Patch 8.1.1764

Problem: ":browse oldfiles" is not tested.
Solution: Add a test.
Files: src/testdir/test_viminfo.vim

Patch 8.1.1765

Problem: get(func, dict, def) does not work properly.
Solution: Handle NULL dict better. (Takuya Fujiwara, closes #4734)
Files: src/evalfunc.c, src/testdir/test_getvar.vim,
src/testdir/test_partial.vim

Patch 8.1.1766

Problem: Code for writing session file is spread out.
Solution: Put it in one file. (Yegappan Lakshmanan, closes #4728)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/eval.c, src/ex_docmd.c, src/misc2.c, src/proto.h,
src/proto/eval.pro, src/proto/misc2.pro, src/proto/session.pro,
src/session.c

Patch 8.1.1767

Problem: FEAT_SESSION defined separately.
Solution: Make FEAT_SESSION depend on FEAT_EVAL.
Files: src/feature.h, src/session.c, src/evalfunc.c, src/ex_docmd.c,
src/gui_gtk_x11.c, src/proto.h

Patch 8.1.1768

Problem: Man plugin changes setting in current window.
Solution: Set options later. (Jason Franklin)
Files: runtime/ftplugin/man.vim, src/testdir/test_man.vim

Patch 8.1.1769

Problem: **'shellslash'** is also used for completion.
Solution: Add the **'completeslash'** option. (Yasuhiro Matsumoto, closes #3612)
Files: runtime/doc/options.txt, src/ex_getln.c, src/insexpand.c,
src/option.c, src/option.h, src/structs.h,
src/testdir/test_ins_complete.vim

Patch 8.1.1770

Problem: Cannot get the window ID of the popup preview window.
Solution: Add popup_getpreview().
Files: src/evalfunc.c, src/popupwin.c, src/proto/popupwin.pro,
runtime/doc/eval.txt, runtime/doc/popup.txt,
src/testdir/dumps/Test_popupwin_previewpopup_3.dump

Patch 8.1.1771

Problem: Options test fails on MS-Windows.
Solution: Add correct and incorrect values for **'completeslash'**.
Files: src/testdir/gen_opt_test.vim

Patch 8.1.1772

Problem: Options test still fails on MS-Windows.
Solution: Check buffer-local value of **'completeslash'**.
Files: src/option.c

Patch 8.1.1773

Problem: The preview popup window may be too far to the right.
Solution: Keep it inside the screen. Also keep the close button and
scrollbar visible if possible.
Files: src/popupwin.c, src/proto/popupwin.pro, src/ex_cmds.c,
src/screen.c, src/vim.h, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_previewpopup_1.dump,
src/testdir/dumps/Test_popupwin_previewpopup_2.dump,
src/testdir/dumps/Test_popupwin_previewpopup_3.dump,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump

Patch 8.1.1774

Problem: Test is silently skipped.
Solution: Throw "Skipped".
Files: src/testdir/test_ins_complete.vim

Patch 8.1.1775

Problem: Error message may be empty in filetype test.
Solution: Use v:exception instead. (Daniel Hahler, closes #4744)
Files: src/testdir/test_filetype.vim

Patch 8.1.1776

Problem: Text added with a job to another buffer isn't displayed.
Solution: Update topline after adding a line. (closes #4745)
Files: src/channel.c, src/testdir/test_channel.vim, src/testdir/check.vim,
src/testdir/dumps/Test_job_buffer_scroll_1.dump

Patch 8.1.1777

Problem: Useless checks for job feature in channel test.
Solution: Remove the checks. Remove ch_log() calls.
Files: src/testdir/test_channel.vim

Patch 8.1.1778

Problem: Not showing the popup window right border is confusing.
Solution: Also show the border when there is no close button. (closes #4747)
Files: src/popupwin.c, src/testdir/dumps/Test_popupwin_20.dump,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.1.1779

Problem: Not showing the popup window right border is confusing.
Solution: Also show the border when **'wrap'** is off. (closes #4747)
Files: src/popupwin.c, src/testdir/dumps/Test_popupwin_20.dump,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.1.1780

Problem: Warning for file no longer available is repeated every time Vim is
focused. (Brian Armstrong)
Solution: Only give the message once. (closes #4748)
Files: src/fileio.c

Patch 8.1.1781

Problem: Amiga: no builtin OS readable version info.
Solution: Add a "version" variable. (Ola Söder, closes #4753)
Files: src/os_amiga.c

Patch 8.1.1782

Problem: MS-Windows: system() has temp file error with **'noshelltemp'**.
Solution: Check s_dont_use_vimrun. (Ken Takata, closes #4754)
Files: src/os_win32.c

Patch 8.1.1783

Problem: MS-Windows: compiler test may fail when using %:S.
Solution: Reset **'shellslash'**.
Files: src/testdir/test_compiler.vim

Patch 8.1.1784

Problem: MS-Windows: resolve() does not work if serial nr duplicated.
Solution: Use another method to get the full path. (Ken Takata, closes #4661)
Files: src/os_mswin.c

Patch 8.1.1785

Problem: Map functionality mixed with character input.
Solution: Move the map functionality to a separate file. (Yegappan
Lakshmanan, closes #4740) Graduate the +localmap feature.
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/buffer.c, src/feature.h, src/evalfunc.c, src/ex_docmd.c,
src/getchar.c, src/map.c, src/proto.h, src/proto/getchar.pro,
src/proto/map.pro, src/version.c, src/structs.h

Patch 8.1.1786

Problem: Double click in popup scrollbar starts selection.
Solution: Ignore the double click.
Files: src/ui.c, src/popupwin.c, src/proto/popupwin.pro

Patch 8.1.1787

Problem: Cannot resize a popup window.
Solution: Allow for resizing by dragging the lower right corner.
Files: runtime/doc/popup.txt, src/popupwin.c, src/structs.h, src/vim.h,
src/ui.c src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_drag_01.dump,
src/testdir/dumps/Test_popupwin_drag_02.dump,
src/testdir/dumps/Test_popupwin_drag_03.dump,
src/testdir/dumps/Test_popupwin_previewpopup_1.dump,
src/testdir/dumps/Test_popupwin_previewpopup_2.dump,
src/testdir/dumps/Test_popupwin_previewpopup_3.dump,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump

Patch 8.1.1788 (after 8.1.1787)

Problem: missing changes in proto file
Solution: Update proto file.
Files: src/proto/popupwin.pro

Patch 8.1.1789

Problem: Cannot see file name of preview popup window.
Solution: Add the file name as the title.
Files: src/ex_cmds.c, src/popupwin.c, src/proto/popupwin.pro,
src/fileio.c,
src/testdir/dumps/Test_popupwin_previewpopup_1.dump,
src/testdir/dumps/Test_popupwin_previewpopup_2.dump,
src/testdir/dumps/Test_popupwin_previewpopup_3.dump,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump,
src/testdir/dumps/Test_popupwin_previewpopup_5.dump

Patch 8.1.1790

Problem: :mkvimrc is not tested.
Solution: Add a test.
Files: src/testdir/test_mksession.vim

Patch 8.1.1791

Problem: **'completeslash'** also applies to globpath().
Solution: Add the WILD_IGNORE_COMPLETESLASH flag. (test by Yasuhiro
Matsumoto, closes #4760)
Files: src/testdir/test_ins_complete.vim, src/ex_getln.c, src/evalfunc.c,
src/vim.h

Patch 8.1.1792

Problem: The vgetorpeek() function is too long.
Solution: Split off the part that handles mappings.
Files: src/getchar.c

Patch 8.1.1793

Problem: Mixed comment style in globals.
Solution: Use // comments where appropriate.
Files: src/globals.h

Patch 8.1.1794 (after 8.1.1792)

Problem: Tests are flaky.
Solution: Undo the change to vgetorpeek().

Files: src/getchar.c

Patch 8.1.1795

Problem: No syntax HL after splitting windows with :bufdo. (Yasuhiro Matsumoto)

Solution: Trigger Syntax autocommands in buffers that are active. (closes #4761)

Files: src/vim.h, src/option.c, src/ex_cmds2.c, src/testdir/test_syntax.vim

Patch 8.1.1796

Problem: :argdo is not tested

Solution: Add a test.

Files: src/testdir/test_arglist.vim

Patch 8.1.1797 (after 8.1.1794)

Problem: The vgetorpeek() function is too long.

Solution: Split off the part that handles mappings, with fix.

Files: src/getchar.c

Patch 8.1.1798

Problem: Warning for unused variable in tiny version. (Tony Mechelynck)

Solution: Move inside #ifdef. Reformat code.

Files: src/getchar.c

Patch 8.1.1799

Problem: Cannot avoid mapping for a popup window.

Solution: Add the "mapping" property, default TRUE.

Files: runtime/doc/popup.txt, src/getchar.c, src/popupwin.c, src/vim.h, src/proto/popupwin.pro, src/testdir/test_popupwin.vim

Patch 8.1.1800

Problem: Function call functions have too many arguments.

Solution: Pass values in a funcexe_T struct.

Files: src/eval.c, src/structs.h, src/userfunc.c, src/proto/userfunc.pro, src/list.c, src/regexp.c, src/terminal.c, src/change.c, src/ex_cmds2.c, src/popupwin.c, src/channel.c

Patch 8.1.1801

Problem: Cannot build without the +eval feature.

Solution: Always define funcexe_T.

Files: src/structs.h

Patch 8.1.1802

Problem: Missing change to call_callback().

Solution: Add missing change.

Files: src/sound.c

Patch 8.1.1803

Problem: All builtin functions are global.

Solution: Add the method call operator ->. Implemented for a limited number of functions.

Files: runtime/doc/eval.txt, src/eval.c, src/structs.h, src/userfunc.c, src/globals.h, src/evalfunc.c, src/proto/evalfunc.pro,

src/testdir/test_method.vim, src/testdir/Make_all.mak

Patch 8.1.1804

Problem: No test for display updating without a scroll region.

Solution: Add a test.

Files: src/testdir/test_display.vim, src/testdir/check.vim,
src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_scroll_no_region_1.dump,
src/testdir/dumps/Test_scroll_no_region_2.dump,
src/testdir/dumps/Test_scroll_no_region_3.dump

Patch 8.1.1805

Problem: Au_did_filetype is declared twice.

Solution: Remove it from autocmd.c. (closes #4767)

Files: src/globals.h, src/autocmd.c

Patch 8.1.1806

Problem: Test for display updating doesn't check without statusline.

Solution: Add screenshots without a status line.

Files: src/testdir/test_display.vim,
src/testdir/dumps/Test_scroll_no_region_4.dump,
src/testdir/dumps/Test_scroll_no_region_5.dump,
src/testdir/dumps/Test_scroll_no_region_6.dump

Patch 8.1.1807

Problem: More functions can be used as a method.

Solution: Add append(), appendbuflines(), assert_equal(), etc.
Also add the :eval command.

Files: runtime/doc/eval.txt, runtime/doc/testing.txt, src/evalfunc.c,
src/testdir/test_method.vim, src/ex_cmds.h, src/ex_eval.c,
src/proto/ex_eval.pro, src/ex_cmdidxs.h

Patch 8.1.1808

Problem: Build failure for tiny version.

Solution: Define ex_eval to ex_ni. Clean up the ordering a bit.

Files: src/ex_docmd.c

Patch 8.1.1809

Problem: More functions can be used as a method.

Solution: Add has_key(), split(), str2list(), etc.

Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_method.vim,
src/testdir/test_diffmode.vim, src/testdir/test_syntax.vim,
src/testdir/test_system.vim

Patch 8.1.1810

Problem: Popup_getoptions() is missing an entry for "mapping".

Solution: Add the entry.

Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1811

Problem: Popup window color cannot be set to "Normal".

Solution: Check for non-empty 'wincolor' instead of zero attribute.
(closes #4772)

Files: src/screen.c, src/testdir/test_popupwin.vim,

src/testdir/dumps/Test_popupwin_20.dump,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.1.1812

Problem: Reading a truncated undo file hangs Vim.
Solution: Check for reading EOF. (closes #4769)
Files: src/undo.c, src/testdir/test_undo.vim

Patch 8.1.1813

Problem: ATTENTION prompt for a preview popup window.
Solution: Close the popup window if aborting the buffer load. Avoid getting the ATTENTION dialog.
Files: src/tag.c, src/ex_cmds.c, src/memline.c, src/vim.h, runtime/doc/windows.txt

Patch 8.1.1814

Problem: A long title in a popup window overflows.
Solution: Truncate the title. (closes #4770)
Files: src/testdir/test_popupwin.vim, src/popupwin.c, src/testdir/dumps/Test_popupwin_longtitle_1.dump, src/testdir/dumps/Test_popupwin_longtitle_2.dump

Patch 8.1.1815

Problem: Duplicating info for internal functions.
Solution: Use one table to list internal functions.
Files: src/evalfunc.c

Patch 8.1.1816

Problem: Cannot use a user defined function as a method.
Solution: Pass the base as the first argument to the user defined function after "->". (partly by FUJIWARA Takuya)
Files: src/eval.c, src/userfunc.c, src/testdir/test_user_func.vim, src/testdir/test_autoload.vim, src/testdir/sautest/autoload/foo.vim

Patch 8.1.1817

Problem: Github contribution text is incomplete.
Solution: Update the text.
Files: CONTRIBUTING.md

Patch 8.1.1818

Problem: Unused variable.
Solution: Remove the variable. (Mike Williams)
Files: src/sound.c

Patch 8.1.1819

Problem: :pedit does not work with a popup preview window.
Solution: Avoid aborting with an error. (fixes #4777) Also double check that after prepare_tagpreview() the current window is not a popup window.
Files: src/ex_docmd.c, src/popupmenu.c, src/search.c, src/tag.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_previewpopup_6.dump, src/testdir/dumps/Test_popupwin_previewpopup_7.dump,

src/testdir/dumps/Test_popupwin_previewpopup_8.dump

Patch 8.1.1820

Problem: Using expr->FuncRef() does not work.
Solution: Make FuncRef work as a method.
Files: src/eval.c, src/userfunc.c, src/testdir/test_method.vim

Patch 8.1.1821

Problem: No test for wrong number of method arguments.
Solution: Add a test.
Files: src/testdir/test_method.vim

Patch 8.1.1822

Problem: Confusing error message when range is not allowed.
Solution: With ADDR_NONE give e_norange. Change e_invaddr to e_invrage for consistency.
Files: src/ex_docmd.c, src/globals.h, src/testdir/test_excmd.vim

Patch 8.1.1823

Problem: Command line history code is spread out.
Solution: Put the code in a new file. (Yegappan Lakshmanan, closes #4779)
Also graduate the +cmdline_hist feature.
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/cmdhist.c, src/ex_getln.c, src/proto.h, src/proto/cmdhist.pro, src/proto/ex_getln.pro, src/version.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/misc2.c, src/normal.c, src/ops.c, src/option.c, src/search.c, src/tag.c, src/usercmd.c, src/viminfo.c, src/feature.h, src/globals.h

Patch 8.1.1824

Problem: Crash when correctly spelled word is very long. (Ben Kraft)
Solution: Check word length before copying. (closes #4778)
Files: src/spell.c, src/testdir/test_spell.vim

Patch 8.1.1825

Problem: Allocating more memory than needed for extended structs.
Solution: Use offsetof() instead of sizeof(). (Dominique Pelle, closes #4785)
Files: src/dict.c

Patch 8.1.1826

Problem: Tests use hand coded feature and option checks.
Solution: Use the commands from check.vim in more tests.
Files: src/testdir/check.vim, src/testdir/shared.vim, src/testdir/test_autochdir.vim, src/testdir/test_autocmd.vim, src/testdir/test_balloon.vim, src/testdir/test_breakindent.vim, src/testdir/test_buflines.vim, src/testdir/test_cdo.vim, src/testdir/test_channel.vim, src/testdir/test_clientserver.vim, src/testdir/test_conceal.vim, src/testdir/test_cscope.vim, src/testdir/test_debugger.vim, src/testdir/test_filechanged.vim, src/testdir/test_fold.vim, src/testdir/test_functions.vim, src/testdir/test_gui.vim, src/testdir/test_gui_init.vim, src/testdir/test_highlight.vim, src/testdir/test_mapping.vim,

src/testdir/test_match.vim, src/testdir/test_memory_usage.vim,
src/testdir/test_options.vim, src/testdir/test_paste.vim,
src/testdir/test_popup.vim, src/testdir/test_search.vim,
src/testdir/test_signals.vim, src/testdir/test_startup.vim,
src/testdir/test_syntax.vim, src/testdir/test_termcodes.vim,
src/testdir/test_terminal.vim, src/testdir/test_timers.vim,
src/testdir/test_vimscript.vim

Patch 8.1.1827

Problem: Allocating more memory than needed for extended structs.

Solution: Use offsetof() instead of sizeof(). (Dominique Pelle,
closes #4786)

Files: src/getchar.c, src/regexp.c, src/sign.c, src/structs.h,
src/syntax.c, src/textprop.c, src/userfunc.c

Patch 8.1.1828

Problem: Not strict enough checking syntax of method invocation.

Solution: Check there is no white space inside ->method().

Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_method.vim

Patch 8.1.1829

Problem: Difference in screenshots.

Solution: Update screenshots. Change checks in a few more tests.
(closes #4789)

Files: src/testdir/test_balloon_gui.vim,
src/testdir/test_shortpathname.vim,
src/testdir/test_windows_home.vim,
src/testdir/dumps/Test_popupwin_previewpopup_1.dump,
src/testdir/dumps/Test_popupwin_previewpopup_2.dump,
src/testdir/dumps/Test_popupwin_previewpopup_3.dump,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump,
src/testdir/dumps/Test_popupwin_previewpopup_5.dump

Patch 8.1.1830

Problem: Travis does not report error when tests fail.

Solution: Explicitly do "exit 1".

Files: .travis.yml

Patch 8.1.1831

Problem: Confusing skipped message.

Solution: Drop "run" from "run start the GUI".

Files: src/testdir/check.vim

Patch 8.1.1832

Problem: Win_execute() does not work in other tab. (Rick Howe)

Solution: Take care of the tab. (closes #4792)

Files: src/testdir/test_execute_func.vim, src/evalfunc.c, src/window.c,
src/proto/window.pro

Patch 8.1.1833

Problem: Allocating a bit too much when spellbadword() does not find a bad
word.

Solution: Reset "len" when going to the next word. (Daniel Hahler,
closes #4788)

Files: src/evalfunc.c

Patch 8.1.1834

Problem: Cannot use a lambda as a method.

Solution: Implement ->{lambda}(). (closes #4768)

Files: runtime/doc/eval.txt, src/eval.c, src/testdir/test_method.vim

Patch 8.1.1835

Problem: Cannot use printf() as a method.

Solution: Pass the base as the second argument to printf().

Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_method.vim

Patch 8.1.1836

Problem: Inaccurate memory estimate for Amiga-like OS.

Solution: Adjust #ifdef for AvailMem(). (Ola Söder, closes #4797)

Files: src/os_amiga.c

Patch 8.1.1837

Problem: Popup test fails if clipboard is supported but not working.

Solution: Add the "clipboard_working" feature. Also use Check commands instead of "if" and "throw". And remove stray ch_logfile().

Files: src/testdir/test_popupwin.vim, src/evalfunc.c, runtime/doc/eval.txt

Patch 8.1.1838

Problem: There is :spellwrong and :spellgood but not :spellrare.

Solution: Add :spellrare. (Martin Tournioij, closes #4291)

Files: runtime/doc/spell.txt, src/ex_cmdidxs.h, src/ex_cmds.h, src/normal.c, src/proto/spellfile.pro, src/spellfile.c, src/spell.h, src/testdir/Make_all.mak, src/testdir/test_normal.vim, src/testdir/test_spellfile.vim

Patch 8.1.1839

Problem: Insufficient info when test fails because of screen size.

Solution: Report the detected screen size.

Files: src/testdir/runtest.vim

Patch 8.1.1840

Problem: Testing: WorkingClipboard() is not accurate.

Solution: Check feature clipboard_working instead.

Files: src/testdir/shared.vim, src/testdir/test_paste.vim, src/testdir/test_quotestar.vim, src/testdir/test_termcodes.vim

Patch 8.1.1841

Problem: No test for Ex shift commands.

Solution: Add a test. (Dominique Pelle, closes #4801)

Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim, src/testdir/test_shift.vim

Patch 8.1.1842

Problem: Test listed as flaky should no longer be flaky.

Solution: Remove Test_popup_and_window_resize from the list of flaky tests. (Daniel Hahler, close #4807)

Files: src/testdir/runtest.vim

Patch 8.1.1843

Problem: Might be freeing memory that was not allocated.
Solution: Have next_fenc() set the fenc_allocated flag. (closes #4804)
Files: src/fileio.c

Patch 8.1.1844

Problem: Buffer no longer unloaded when adding text properties to it.
Solution: Do not create the memfile. (closes #4808)
Files: runtime/doc/textprop.txt, src/testdir/test_textprop.vim,
src/textprop.c

Patch 8.1.1845

Problem: May use NULL pointer when running out of memory.
Solution: Do not clear popup buffers when NULL. (closes #4802)
Files: src/screen.c

Patch 8.1.1846

Problem: Inconsistently using GetVimCommand() and v:prospath. (Daniel Hahler)
Solution: Use GetVimCommand(). (closes #4806)
Files: src/testdir/test_autocmd.vim, src/testdir/test_gui.vim,
src/testdir/test_normal.vim, src/testdir/test_profile.vim,
src/testdir/test_suspend.vim, src/testdir/test_system.vim,
src/testdir/test_vimscript.vim

Patch 8.1.1847

Problem: Suspend test is failing.
Solution: Do not use GetVimCommandClean().
Files: src/testdir/test_suspend.vim

Patch 8.1.1848

Problem: '**langmap**' is not used for **CTRL-W** command in terminal.
Solution: Push the command in the typeahead buffer instead of the stuff buffer. (closes #4814)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.1849

problem: Some insert complete functions in the wrong file.
Solution: Move complete functions to insexpand.c. (Yegappan Lakshmanan, closes #4815)
Files: src/evalfunc.c, src/insexpand.c, src/proto/insexpand.pro

Patch 8.1.1850

Problem: Focus may remain in popup window.
Solution: Change focus if needed.
Files: src/popupmnu.c

Patch 8.1.1851

Problem: Crash when sound_playfile() callback plays sound.
Solution: Invoke callback later from event loop.
Files: src/testdir/test_sound.vim, src/ui.c, src/sound.c,
src/proto/sound.pro, src/feature.h, src/os_unix.c, src/ex_docmd.c,
src/misc2.c

Patch 8.1.1852

Problem: Timers test is flaky.
Solution: Accept a larger count. Add test to list of flaky tests.
Files: src/testdir/test_timers.vim, src/testdir/runtest.vim

Patch 8.1.1853

Problem: Timers test is still flaky.
Solution: Compute the time to sleep more accurately.
Files: src/ex_docmd.c

Patch 8.1.1854

Problem: Now another timer test is flaky.
Solution: Add test to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.1855

Problem: Another failing timer test.
Solution: Assert that timers are finished by the end of the test. Rename test functions to make them easier to find.
Files: src/testdir/test_timers.vim, src/testdir/runtest.vim

Patch 8.1.1856

Problem: popup preview test fails sometimes. (Christian Brabandt)
Solution: Clear the command line.
Files: src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_previewpopup_6.dump

Patch 8.1.1857

Problem: Cannot use modifier with multibyte character.
Solution: Allow using a multibyte character, although it doesn't work everywhere.
Files: src/misc2.c, src/testdir/test_mapping.vim

Patch 8.1.1858

Problem: Test for multibyte mapping fails on some systems.
Solution: Test in another way.
Files: src/testdir/test_mapping.vim

Patch 8.1.1859

Problem: Timer test sometimes fails on Mac.
Solution: Show more info when it fails.
Files: src/testdir/test_timers.vim

Patch 8.1.1860

Problem: Map timeout test is flaky.
Solution: Add test to list of flaky tests. Increase timeout.
Files: src/testdir/runtest.vim, src/testdir/test_mapping.vim

Patch 8.1.1861

Problem: Only some assert functions can be used as a method.
Solution: Allow using most assert functions as a method.
Files: runtime/doc/testing.txt, src/evalfunc.c,
src/testdir/test_assert.vim

Patch 8.1.1862

Problem: Coverity warns for not using return value.
Solution: Add "(void)" to avoid the warning.
Files: src/normal.c

Patch 8.1.1863

Problem: Confusing error when using a builtin function as method while it does not support that.
Solution: Add a specific error message.
Files: src/vim.h, src/evalfunc.c, src/userfunc.c, src/testdir/test_method.vim

Patch 8.1.1864

Problem: Still a timer test that is flaky on Mac.
Solution: Adjust the sleep times.
Files: src/testdir/test_timers.vim

Patch 8.1.1865

Problem: Spellrare and spellrepall in the wrong order.
Solution: Put spellrare below spellrepall. (closes #4820)
Files: runtime/doc/spell.txt, src/ex_cmds.h

Patch 8.1.1866

Problem: Modeless selection in GUI does not work properly.
Solution: Avoid going beyond the end of the line. (closes #4783)
Files: src/ui.c

Patch 8.1.1867

Problem: Still a timer test that is flaky on Mac.
Solution: Loop with a sleep instead of one fixed sleep.
Files: src/testdir/test_timers.vim

Patch 8.1.1868

Problem: Multibyte characters in '**listchars**' don't work correctly if '**linebreak**' is also enabled. (Martin Tournoij)
Solution: Make it work correctly. (Christian Brabandt, closes #4822, closes #4812)
Files: src/screen.c, src/testdir/test_listchars.vim

Patch 8.1.1869

Problem: Code for the argument list is spread out.
Solution: Put argument list code in arglist.c. (Yegappan Lakshmanan, closes #4819)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/arglist.c, src/buffer.c, src/evalfunc.c, src/ex_cmds2.c, src/ex_docmd.c, src/proto.h, src/proto/arglist.pro, src/proto/buffer.pro, src/proto/ex_cmds2.pro, src/proto/ex_docmd.pro

Patch 8.1.1870

Problem: Using :pedit from a help file sets the preview window to help filetype. (Wang Shidong)

Solution: Do not set "keep_help_flag". (closes #3536)
Files: src/ex_docmd.c, src/testdir/test_window_cmd.vim

Patch 8.1.1871 (after 8.1.1866)
Problem: Modeless selection in GUI still not correct.
Solution: Fix max_col.
Files: src/ui.c

Patch 8.1.1872
Problem: When Vim exits because of a signal, VimLeave is not triggered.
(Daniel Hahler)
Solution: Unblock autocommands when triggering VimLeave. (closes #4818)
Files: src/main.c

Patch 8.1.1873 (after 8.1.1872)
Problem: Cannot build tiny version.
Solution: Remove #ifdef for is_autocmd_blocked().
Files: src/autocmd.c

Patch 8.1.1874
Problem: Modeless selection in popup window overlaps scrollbar.
Solution: Subtract scrollbar from max_col. (closes #4773)
Files: src/ui.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_select_01.dump

Patch 8.1.1875
Problem: Cannot get size and position of the popup menu.
Solution: Add pum_getpos(). (Ben Jackson, closes #4827)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_popup.vim

Patch 8.1.1876
Problem: proto file missing from distribution
Solution: Add the file.
Files: Filelist

Patch 8.1.1877
Problem: Graduated features scattered.
Solution: Put graduated and obsolete features together.
Files: src/feature.h

Patch 8.1.1878
Problem: Negative float before method not parsed correctly.
Solution: Apply "!" and "-" in front of expression before using ->.
Files: src/eval.c, src/proto/eval.pro, src/userfunc.c,
src/testdir/test_method.vim

Patch 8.1.1879
Problem: More functions can be used as methods.
Solution: Make float functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_float_func.vim

Patch 8.1.1880

Problem: Cannot show extra info for completion in a popup window.
Solution: Add the "popup" entry in '**completeopt**'.
Files: runtime/doc/options.txt, src/popupmnu.c, src/ex_cmds.c,
src/proto/ex_cmds.pro, src/ex_docmd.c, src/search.c, src/tag.c,
src/popupwin.c, src/proto/popupwin.pro, src/option.c, src/vim.h,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_infopopup_1.dump,
src/testdir/dumps/Test_popupwin_infopopup_2.dump,
src/testdir/dumps/Test_popupwin_infopopup_3.dump,
src/testdir/dumps/Test_popupwin_infopopup_4.dump

Patch 8.1.1881

Problem: Popup window test fails in some configurations.
Solution: Check that screendumps can be made.
Files: src/testdir/test_popupwin.vim

Patch 8.1.1882

Problem: Cannot specify properties of the info popup window.
Solution: Add the '**completepopup**' option. Default to PmenuSel highlight.
Files: runtime/doc/options.txt, runtime/doc/insert.txt, src/option.c,
src/popupwin.c, src/proto/popupwin.pro, src/option.h,
src/testdir/test_popupwin.vim, src/screen.c,
src/testdir/dumps/Test_popupwin_infopopup_1.dump,
src/testdir/dumps/Test_popupwin_infopopup_2.dump,
src/testdir/dumps/Test_popupwin_infopopup_3.dump

Patch 8.1.1883

Problem: Options test fails.
Solution: Add entry for '**completepopup**'.
Files: src/testdir/gen_opt_test.vim

Patch 8.1.1884

Problem: Cannot use mouse scroll wheel in popup in Insert mode. Mouse
clicks in popup close the popup menu.
Solution: Check if the mouse is in a popup window. Do not let mouse events
close the popup menu. (closes #4544)
Files: src/edit.c, src/popupmnu.c, src/insexpand.c

Patch 8.1.1885

Problem: Comments in libvterm are inconsistent.
Solution: Use // comments. Also update the table of combining characters.
Files: src/libvterm/bin/unterm.c, src/libvterm/bin/vterm-ctrl.c,
src/libvterm/bin/vterm-dump.c, src/libvterm/include/vterm.h,
src/libvterm/include/vterm_keycodes.h,
src/libvterm/src/encoding.c, src/libvterm/src/keyboard.c,
src/libvterm/src/mouse.c, src/libvterm/src/parser.c,
src/libvterm/src/pen.c, src/libvterm/src/rect.h,
src/libvterm/src/state.c, src/libvterm/src/unicode.c,
src/libvterm/src/utf8.h, src/libvterm/src/vterm.c,
src/libvterm/src/vterm_internal.h, src/libvterm/src/termscreen.c

Patch 8.1.1886

Problem: Command line expansion code is spread out.
Solution: Move the code to cmdexpand.c. (Yegappan Lakshmanan, closes #4831)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/cmdexpand.c, src/evalfunc.c, src/ex_getln.c, src/proto.h, src/proto/cmdexpand.pro, src/proto/ex_getln.pro, src/structs.h

Patch 8.1.1887

Problem: The +cmdline_compl feature is not in the tiny version.

Solution: Graduate the +cmdline_compl feature.

Files: src/cmdexpand.c, src/arglist.c, src/autocmd.c, src/buffer.c, src/cmdhist.c, src/eval.c, src/evalfunc.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/feature.h, src/highlight.c, src/if_cscope.c, src/map.c, src/menu.c, src/misc1.c, src/misc2.c, src/option.c, src/sign.c, src/syntax.c, src/tag.c, src/term.c, src/usercmd.c, src/userfunc.c, src/version.c, src/globals.h, src/option.h, src/structs.h, runtime/doc/cmdline.txt

Patch 8.1.1888

Problem: More functions can be used as methods.

Solution: Make various functions usable as a method.

Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_vimscript.vim, src/testdir/test_balloon_gui.vim, src/testdir/test_popup.vim, src/testdir/test_functions.vim, src/testdir/test_hide.vim, src/testdir/test_arglist.vim

Patch 8.1.1889

Problem: Coverity warns for using a NULL pointer.

Solution: Use zero for column if pos is NULL.

Files: src/netbeans.c

Patch 8.1.1890

Problem: Ml_get error when deleting fold marker.

Solution: Check that the line number is not below the last line. Adjust the fold when deleting the empty line. (Christian Brabandt, closes #4834)

Files: src/fold.c, src/normal.c, src/testdir/test_fold.vim

Patch 8.1.1891

Problem: Functions used in one file are global.

Solution: Add "static". (Yegappan Lakshmanan, closes #4840)

Files: src/autocmd.c, src/buffer.c, src/change.c, src/channel.c, src/charset.c, src/dict.c, src/digraph.c, src/eval.c, src/ex_cmds.c, src/ex_eval.c, src/fileio.c, src/findfile.c, src/getchar.c, src/gui.c, src/indent.c, src/json.c, src/list.c, src/mark.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/ops.c, src/option.c, src/popupwin.c, src/profiler.c, src/proto/autocmd.pro, src/proto/buffer.pro, src/proto/change.pro, src/proto/channel.pro, src/proto/charset.pro, src/proto/dict.pro, src/proto/eval.pro, src/proto/ex_cmds.pro, src/proto/ex_eval.pro, src/proto/fileio.pro, src/proto/findfile.pro, src/proto/getchar.pro, src/proto/gui.pro, src/proto/indent.pro, src/proto/json.pro, src/proto/list.pro, src/proto/mark.pro, src/proto/menu.pro, src/proto/message.pro, src/proto/misc1.pro, src/proto/misc2.pro, src/proto/ops.pro, src/proto/option.pro, src/proto/popupwin.pro, src/proto/profiler.pro,

src/proto/quickfix.pro, src/proto/spell.pro, src/proto/term.pro,
src/proto/textprop.pro, src/proto/ui.pro, src/proto/undo.pro,
src/proto/window.pro, src/quickfix.c, src/regexp.c, src/spell.c,
src/term.c, src/textprop.c, src/ui.c, src/undo.c, src/window.c

Patch 8.1.1892

Problem: Missing index entry and option menu for **'completepopup'**.
Solution: Add the entries. Adjust #ifdefs to avoid dead code.
Files: runtime/doc/quickref.txt, runtime/optwin.vim, src/option.c,
src/option.h, src/popupwin.c

Patch 8.1.1893

Problem: Script to summarize test results can be improved.
Solution: Use "silent" for substitute to avoid reporting number of matches.
Remove duplicate "set nocp". (Daniel Hahler, closes #4845)
Files: src/testdir/summarize.vim

Patch 8.1.1894

Problem: Not checking for out-of-memory of autoload_name().
Solution: Check for NULL. (Dominique Pelle, closes #4846)
Files: src/eval.c

Patch 8.1.1895

Problem: Using NULL pointer when out of memory.
Solution: Bail out or skip the code using the pointer. (Zu-Ming Jiang,
closes #4805, closes #4843, closes #4939, closes #4844)
Files: src/message.c, src/highlight.c, src/buffer.c, src/ops.c

Patch 8.1.1896

Problem: Compiler warning for unused variable.
Solution: Add #ifdef. (John Marriott) Missing part of 8.1.1892.
Files: src/popupmnu.c

Patch 8.1.1897

Problem: May free memory twice when out of memory.
Solution: Check that backslash_half_save() returns a different pointer.
(Dominique Pelle, closes #4847)
Files: src/cmdexpand.c, src/misc1.c

Patch 8.1.1898

Problem: Crash when out of memory during startup.
Solution: When out of memory message given during initialisation bail out.
(closes #4842)
Files: src/misc2.c

Patch 8.1.1899

Problem: sign_place() does not work as documented.
Solution: Make it accept line numbers like line(). (Yegappan Lakshmanan,
closes #4848)
Files: src/sign.c, src/testdir/test_signs.vim

Patch 8.1.1900

Problem: Sign test fails in the GUI.
Solution: Catch and ignore the exception.

Files: src/testdir/test_signs.vim

Patch 8.1.1901

Problem: The +insert_expand feature is not always available.

Solution: Graduate the +insert_expand feature.

Files: src/feature.h, src/autocmd.c, src/buffer.c, src/change.c, src/charset.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_getln.c, src/getchar.c, src/gui.c, src/highlight.c, src/indent.c, src/insexpand.c, src/misc2.c, src/move.c, src/option.c, src/popupmnu.c, src/screen.c, src/search.c, src/spell.c, src/tag.c, src/term.c, src/userfunc.c, src/version.c, src/globals.h, src/option.h, src/proto.h, src/structs.h, src/vim.h, runtime/doc/change.txt, runtime/doc/index.txt, runtime/doc/insert.txt, runtime/doc/options.txt

Patch 8.1.1902

Problem: Cannot have an info popup without a border.

Solution: Add the "border" item to **'completepopup'**.

Files: src/popupwin.c, src/proto/popupwin.pro, src/popupmnu.c, src/testdir/test_popupwin.vim, src/testdir/gen_opt_test.vim, src/testdir/dumps/Test_popupwin_infopopup_nb_1.dump

Patch 8.1.1903

Problem: Cannot build without the +eval feature.

Solution: Add missing #ifdefs

Files: src/insexpand.c, src/popupmnu.c

Patch 8.1.1904

Problem: Cannot have an info popup align with the popup menu.

Solution: Add the "align" item to **'completepopup'**.

Files: src/popupwin.c, src/popupmnu.c, src/vim.h, runtime/doc/insert.txt, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_infopopup_align_1.dump, src/testdir/dumps/Test_popupwin_infopopup_align_2.dump, src/testdir/dumps/Test_popupwin_infopopup_align_3.dump

Patch 8.1.1905

Problem: Cannot set all properties of the info popup.

Solution: Add popup_findinfo(). Rename popup_getpreview() to popup_findpreview().

Files: src/popupwin.c, src/proto/popupwin.pro, src/ex_cmds.c, src/ex_docmd.c, src/popupmnu.c, src/evalfunc.c, runtime/doc/popup.txt, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_infopopup_align_3.dump

Patch 8.1.1906

Problem: Info popup size is sometimes incorrect.

Solution: Compute the position and size after setting the content.

Files: src/popupmnu.c

Patch 8.1.1907

Problem: Wrong position for info popup with scrollbar on the left.

Solution: Take the scrollbar into account.

Files: src/popupwin.c, src/testdir/test_popupwin.vim,

src/testdir/dumps/Test_popupwin_infopopup_5.dump,
src/testdir/dumps/Test_popupwin_cursorline_3.dump,
src/testdir/dumps/Test_popupwin_cursorline_4.dump,
src/testdir/dumps/Test_popupwin_cursorline_5.dump,
src/testdir/dumps/Test_popupwin_cursorline_6.dump,
src/testdir/dumps/Test_popupwin_menu_filter_1.dump,
src/testdir/dumps/Test_popupwin_menu_filter_2.dump,
src/testdir/dumps/Test_popupwin_menu_filter_3.dump,
src/testdir/dumps/Test_popupwin_menu_filter_4.dump

Patch 8.1.1908

Problem: Every popup window consumes a buffer number.
Solution: Recycle buffers only used for popup windows. Do not list popup window buffers.
Files: src/popupwin.c, src/window.c, src/vim.h, src/buffer.c,
src/proto/buffer.pro, src/ex_docmd.c,
src/testdir/test_popupwin.vim

Patch 8.1.1909

Problem: More functions can be used as methods.
Solution: Make a few more functions usable as a method.
Files: runtime/doc/eval.txt, runtime/doc/testing.txt, src/evalfunc.c,
src/testdir/test_popupwin.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_buflines.vim, src/testdir/test_assert.vim

Patch 8.1.1910

Problem: Redrawing too much when toggling '**relativenumber**'.
Solution: Only clear when '**signcolumn**' is set to "number". (Yegappan Lakshmanan, closes #4852)
Files: src/option.c

Patch 8.1.1911

Problem: More functions can be used as methods.
Solution: Make a few more functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test69.in,
src/testdir/test69.ok, src/testdir/test_functions.vim

Patch 8.1.1912

Problem: More functions can be used as methods.
Solution: Make channel and job functions usable as a method.
Files: runtime/doc/channel.txt, src/evalfunc.c,
src/testdir/test_channel.vim

Patch 8.1.1913

Problem: Not easy to compute the space on the command line.
Solution: Add v:echospace. (Daniel Hahler, closes #4732)
Files: src/vim.h, src/eval.c, src/option.c, runtime/doc/eval.txt,
src/testdir/test_messages.vim

Patch 8.1.1914

Problem: Command line expansion code is spread out.
Solution: Move set_one_cmd_context(). (Yegappan Lakshmanan, closes #4855)
Files: src/cmdexpand.c, src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 8.1.1915

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_functions.vim, src/testdir/test_cd.vim,
src/testdir/test_cindent.vim, src/testdir/test_match.vim,
src/testdir/test_popup.vim, src/testdir/test_cursor_func.vim,
src/testdir/test_method.vim, src/testdir/test_bufline.vim,
src/testdir/test_diffmode.vim

Patch 8.1.1916

Problem: Trying to allocate negative amount of memory when closing a popup.
Solution: Check the rows are not out of bounds. Don't finish a selection if
it was never started.
Files: src/ui.c

Patch 8.1.1917

Problem: Non-current window is not redrawn when moving popup. (Ben Jackson)
Solution: Redraw all windows under a popup. (closes #4860)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_drag_01.dump,
src/testdir/dumps/Test_popupwin_drag_02.dump,
src/testdir/dumps/Test_popupwin_drag_03.dump

Patch 8.1.1918

Problem: Redrawing popups is inefficient.
Solution: Fix the logic to compute what window lines to redraw. Make it
work below the last line. Remove redrawing all windows.
Files: src/popupwin.c

Patch 8.1.1919

Problem: Using current window option values when passing a buffer to
popup_create().
Solution: Clear the window-local options. (closes #4857)
Files: src/option.c, src/proto/option.pro, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.1920

Problem: Cannot close a popup by the X when a filter consumes all events.
Solution: Check for a click on the close button before invoking filters.
(closes #4858)
Files: src/popupwin.c, src/proto/popupwin.pro, src/ui.c,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_close_04.dump,
src/testdir/dumps/Test_popupwin_close_05.dump

Patch 8.1.1921

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_expand.vim,
src/testdir/test_expand_func.vim, src/testdir/test_expr.vim,
src/testdir/test_findfile.vim, src/testdir/test_fnameescape.vim,
src/testdir/test_fnamemodify.vim, src/testdir/test_fold.vim,
src/testdir/test_functions.vim, src/testdir/test_search.vim,

src/testdir/test_vimscript.vim

Patch 8.1.1922

Problem: In diff mode global operations can be very slow.
Solution: Do not call diff_redraw() many times, call it once when redrawing.
And also don't update folds multiple times.
Files: src/globals.h, src/diff.c, src/proto/diff.pro, src/screen.c,
src/fold.c

Patch 8.1.1923

Problem: Some source files are not in a normal encoding.
Solution: Convert hangulin.c from euc-kr to utf-8 and digraph.c from latin1
to utf-8. (Daniel Hahler, closes #4731)
Files: src/hangulin.c, src/digraph.c, .travis.yml

Patch 8.1.1924

Problem: Using empty string for current buffer is unexpected.
Solution: Make the argument optional for bufname() and bufnr().
Files: src/evalfunc.c, src/testdir/test_arglist.vim, runtime/doc/eval.txt

Patch 8.1.1925

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_bufline.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_cd.vim, src/testdir/test_changelist.vim,
src/testdir/test_cmdline.vim, src/testdir/test_edit.vim,
src/testdir/test_environ.vim, src/testdir/test_file_perm.vim,
src/testdir/test_getvar.vim, src/testdir/test_jumplist.vim,
src/testdir/test_put.vim, src/testdir/test_stat.vim

Patch 8.1.1926

Problem: Cursorline not redrawn when putting a line above the cursor.
Solution: Redraw when the cursor line is below a change. (closes #4862)
Files: src/change.c

Patch 8.1.1927

Problem: Code for dealing with script files is spread out.
Solution: Move the code to scriptfile.c. (Yegappan Lakshmanan, closes #4861)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/cmdexpand.c, src/ex_cmds2.c, src/proto.h,
src/proto/ex_cmds2.pro, src/proto/scriptfile.pro, src/scriptfile.c

Patch 8.1.1928

Problem: Popup windows don't move with the text when making changes.
Solution: Add the **'textprop'** property to the popup window options, position
the popup relative to a text property. (closes #4560)
No tests yet.
Files: runtime/doc/popup.txt, src/textprop.c, src/proto/textprop.pro,
src/structs.h, src/popupwin.c, src/proto/popupwin.pro, src/move.c,
src/proto/move.pro, src/window.c

Patch 8.1.1929

Problem: No tests for text property popup window.
Solution: Add a few tests.
Files: src/testdir/Make_all.mak, src/textprop.c,
src/testdir/test_popupwin_textprop.vim,
src/testdir/dumps/Test_popup_textprop_01.dump,
src/testdir/dumps/Test_popup_textprop_02.dump,
src/testdir/dumps/Test_popup_textprop_03.dump,
src/testdir/dumps/Test_popup_textprop_04.dump,
src/testdir/dumps/Test_popup_textprop_05.dump,
src/testdir/dumps/Test_popup_textprop_06.dump

Patch 8.1.1930

Problem: Cannot recognize .jsx and .tsx files.
Solution: Recognize them as javascriptreact and typescriptreact.
(closes #4830)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim,
runtime/syntax/javascriptreact.vim,
runtime/indent/javascriptreact.vim,
runtime/ftplugin/javascriptreact.vim

Patch 8.1.1931 (after 8.1.1930)

Problem: Syntax test fails.
Solution: Add new javascriptreact type to completions.
Files: src/testdir/test_syntax.vim

Patch 8.1.1932

Problem: Ml_get errors after using append(). (Alex Genco)
Solution: Do not update the cursor twice. (closes #1737)
Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.1.1933

Problem: The eval.c file is too big.
Solution: Move code related to variables to evalvars.c. (Yegappan
Lakshmanan, closes #4868)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/eval.c, src/evalfunc.c, src/evalvars.c, src/globals.h,
src/proto.h, src/proto/eval.pro, src/proto/evalvars.pro, src/vim.h

Patch 8.1.1934

Problem: Not enough tests for text property popup window.
Solution: Add a few more tests.
Files: src/popupwin.c, src/testdir/test_popupwin_textprop.vim,
src/testdir/dumps/Test_popup_textprop_corn_1.dump,
src/testdir/dumps/Test_popup_textprop_corn_2.dump,
src/testdir/dumps/Test_popup_textprop_corn_3.dump,
src/testdir/dumps/Test_popup_textprop_corn_4.dump

Patch 8.1.1935 (after 8.1.1934)

Problem: Test for text property popup window is flaky.
Solution: Remove the undo message
Files: src/testdir/test_popupwin_textprop.vim,
src/testdir/dumps/Test_popup_textprop_corn_4.dump

Patch 8.1.1936

Problem: Not enough tests for text property popup window.
Solution: Add a few more tests. Make negative offset work. Close all popups when window closes.
Files: src/popupwin.c, src/testdir/test_popupwin_textprop.vim,
src/testdir/dumps/Test_popup_textprop_07.dump,
src/testdir/dumps/Test_popup_textprop_off_1.dump,
src/testdir/dumps/Test_popup_textprop_off_2.dump,
src/testdir/dumps/Test_popup_textprop_corn_5.dump,
src/testdir/dumps/Test_popup_textprop_corn_6.dump

Patch 8.1.1937 (after 8.1.1930)

Problem: Errors when using javascriptreact.
Solution: Use ":runtime" instead of ":source". (closes #4875)
Files: runtime/syntax/javascriptreact.vim,
runtime/indent/javascriptreact.vim,
runtime/ftplugin/javascriptreact.vim

Patch 8.1.1938

Problem: May crash when out of memory.
Solution: Initialize v_type to VAR_UNKNOWN. (Dominique Pelle, closes #4871)
Files: src/userfunc.c

Patch 8.1.1939

Problem: Code for handling v: variables in generic eval file.
Solution: Move v: variables to evalvars.c. (Yegappan Lakshmanan,
closes #4872)
Files: src/eval.c, src/evalvars.c, src/proto/eval.pro,
src/proto/evalvars.pro

Patch 8.1.1940 (after 8.1.1939)

Problem: Script tests fail.
Solution: Don't set vimvars type in set_vim_var_nr().
Files: src/eval.c, src/evalvars.c, src/proto/evalvars.pro

Patch 8.1.1941

Problem: getftype() test fails on Mac.
Solution: Skip /dev/fd/.
Files: src/testdir/test_stat.vim

Patch 8.1.1942

Problem: Shadow directory gets outdated when files are added.
Solution: Add the "shadowupdate" target and add a few comments.
Files: src/Makefile

Patch 8.1.1943

Problem: More code can be moved to evalvars.c.
Solution: Move it, clean up comments. Also move some window related functions to window.c. (Yegappan Lakshmanan, closes #4874)
Files: src/eval.c, src/evalfunc.c, src/evalvars.c, src/proto/eval.pro,
src/proto/evalvars.pro, src/proto/window.pro, src/window.c

Patch 8.1.1944

Problem: Leaking memory when using sound callback.

Solution: Free the callback queue item.
Files: src/sound.c

Patch 8.1.1945

Problem: Popup window "firstline" cannot be reset.
Solution: Allow for setting "firstline" to zero. Fix that the text jumps to the top when using win_execute(). (closes #4876)
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_scroll_5.dump, src/testdir/dumps/Test_popupwin_scroll_6.dump

Patch 8.1.1946

Problem: Memory error when profiling a function without a script ID.
Solution: Check for missing script ID. (closes #4877)
Files: src/testdir/test_profile.vim, src/profiler.c

Patch 8.1.1947

Problem: When executing one test the report doesn't show it.
Solution: Adjust the regexp. (Daniel Hahler, closes #4879)
Files: src/testdir/summarize.vim

Patch 8.1.1948

Problem: Mouse doesn't work in Linux console and causes 100% CPU. (James P. Harvey)
Solution: Loop in WaitForCharOrMouse() when gpm_process_wanted is set. (closes #4828)
Files: src/os_unix.c

Patch 8.1.1949

Problem: Cannot scroll a popup window to the very bottom.
Solution: Scroll to the bottom when the "firstline" property was set to -1. (closes #4577) Allow resetting min/max width/height.
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/dict.c, src/proto/dict.pro, src/testdir/dumps/Test_popupwin_firstline.dump, src/testdir/dumps/Test_popupwin_firstline_1.dump, src/testdir/dumps/Test_popupwin_firstline_2.dump, src/testdir/dumps/Test_popupwin_scroll_10.dump

Patch 8.1.1950

Problem: Using NULL pointer after an out-of-memory.
Solution: Check for NULL pointer. (Dominique Pelle, closes #4881)
Files: src/syntax.c

Patch 8.1.1951

Problem: Mouse double click test is a bit flaky.
Solution: Add to list of flaky tests. Update a couple of comments.
Files: src/testdir/runtest.vim, src/testdir/shared.vim, src/testdir/test_substitute.vim

Patch 8.1.1952

Problem: More functions can be used as a method.
Solution: Allow more functions to be used as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,

src/testdir/test_tagjump.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_terminal.vim, src/testdir/test_getvar.vim,
src/testdir/test_escaped_glob.vim,
src/testdir/test_glob2regpat.vim

Patch 8.1.1953

Problem: More functions can be used as a method.
Solution: Allow more functions to be used as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_blob.vim,
src/testdir/test_breakindent.vim, src/testdir/test_delete.vim,
src/testdir/test_functions.vim, src/testdir/test_getcwd.vim,
src/testdir/test_history.vim, src/testdir/test_listdict.vim,
src/testdir/test_syn_attr.vim, src/testdir/test_termcodes.vim,
src/testdir/test_true_false.vim

Patch 8.1.1954

Problem: More functions can be used as a method.
Solution: Allow more functions to be used as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_arglist.vim, src/testdir/test_functions.vim,
src/testdir/test_json.vim, src/testdir/test_lispwords.vim,
src/testdir/test_listener.vim, src/testdir/test_lua.vim,
src/testdir/test_utf8.vim

Patch 8.1.1955

Problem: Tests contain typos.
Solution: Correct the typos. (Dominique Pelle)
Files: src/testdir/popupbounce.vim, src/testdir/runtest.vim,
src/testdir/screendump.vim, src/testdir/test49.vim,
src/testdir/test_autocmd.vim, src/testdir/test_cindent.vim,
src/testdir/test_const.vim, src/testdir/test_popupwin.vim,
src/testdir/test_quickfix.vim, src/testdir/test_search.vim,
src/testdir/test_tabpage.vim, src/testdir/test_tcl.vim

Patch 8.1.1956

Problem: Screenshot tests may use a different encoding. (Dominique Pelle)
Solution: Always set **'encoding'** to "utf-8" when running Vim in a terminal.
(closes #4884)
Files: src/testdir/shared.vim, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_behind.dump

Patch 8.1.1957

Problem: More code can be moved to evalvars.c.
Solution: Move code to where it fits better. (Yegappan Lakshmanan,
closes #4883)
Files: src/eval.c, src/evalvars.c, src/ex_getln.c, src/globals.h,
src/if_py_both.h, src/proto/eval.pro, src/proto/evalvars.pro,
src/proto/ex_getln.pro, src/proto/scriptfile.pro,
src/scriptfile.c, src/session.c, src/viminfo.c

Patch 8.1.1958

Problem: Old style comments taking up space.
Solution: Change to new style comments.
Files: src/vim.h

Patch 8.1.1959

Problem: When using "firstline" in popup window text may jump when redrawing it. (Nick Jensen)

Solution: Set '**scrolloff**' to zero in a popup window. (closes #4882)

Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_scroll_5.dump,
src/testdir/dumps/Test_popupwin_scroll_6.dump

Patch 8.1.1960

Problem: Fold code is spread out.

Solution: Move fold functions to fold.c.

Files: src/evalfunc.c, src/fold.c, src/proto/fold.pro

Patch 8.1.1961

Problem: More functions can be used as a method.

Solution: Allow more functions to be used as a method. Add a test for mapcheck().

Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test70.in,
src/testdir/test_functions.vim, src/testdir/test_getcwd.vim,
src/testdir/test_maparg.vim, src/testdir/test_match.vim

Patch 8.1.1962

Problem: Leaking memory when using tagfunc().

Solution: Free the user_data. (Dominique Pelle, closes #4886)

Files: src/window.c

Patch 8.1.1963

Problem: Popup window filter may be called recursively when using a Normal mode command. (Nick Jensen)

Solution: Prevent recursiveness. (closes #4887) Also restore KeyTyped.

Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.1964

Problem: Crash when using nested map() and filter().

Solution: Do not set the v:key type to string without clearing the pointer. (closes #4888)

Files: src/eval.c, src/testdir/test_filter_map.vim

Patch 8.1.1965

Problem: The search count message is not displayed when using a mapping. (Gary Johnson)

Solution: Ignore cmd_silent for showing the search count. (Christian Brabandt)

Files: src/search.c

Patch 8.1.1966

Problem: Some code in options.c fits better elsewhere.

Solution: Move functions from options.c to other files. (Yegappan Lakshmanan, closes #4889)

Files: src/evalfunc.c, src/globals.h, src/indent.c, src/map.c,
src/option.c, src/proto/map.pro, src/proto/option.pro,
src/proto/quickfix.pro, src/proto/screen.pro, src/proto/spell.pro,
src/proto/window.pro, src/quickfix.c, src/screen.c, src/spell.c,

src/window.c

Patch 8.1.1967

Problem: Line() only works for the current window.
Solution: Add an optional argument for the window to use.
Files: runtime/eval.txt, src/evalfunc.c, src/testdir/test_popupwin.vim

Patch 8.1.1968

Problem: Crash when using nested map().
Solution: Clear the pointer in prepare_vimvar(). (Ozaki Kiichi, closes #4890, closes #4891)
Files: src/evalvars.c, src/testdir/test_filter_map.vim, src/testdir/test_functions.vim

Patch 8.1.1969

Problem: Popup window filter is used in all modes.
Solution: Add the "filtermode" property.
Files: src/popupwin.c, src/vim.h, src/map.c, src/proto/map.pro, src/structs.h, runtime/doc/popup.txt, src/testdir/test_popupwin.vim

Patch 8.1.1970

Problem: Search stat space wrong, no test for 8.1.1965.
Solution: Fix check for cmd_silent. Add a test. (Christian Brabandt)
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.1.1971

Problem: Manually enabling features causes build errors. (John Marriott)
Solution: Adjust #ifdefs.
Files: src/proto.h, src/popupmnu.c, src/buffer.c, src/quickfix.c, src/ui.c

Patch 8.1.1972

Problem: No proper test for getchar().
Solution: Add a test with special characters.
Files: src/testdir/test_functions.vim

Patch 8.1.1973

Problem: Cannot build without the quickfix feature.
Solution: Remove #ifdef for qf_info_T.
Files: src/structs.h

Patch 8.1.1974

Problem: Coverity warns for using pointer as array.
Solution: Call var2fpos() directly instead of using f_line().
Files: src/evalfunc.c

Patch 8.1.1975

Problem: MS-Windows GUI responds slowly to timer.
Solution: Break out of wait loop when timer was added or input is available. (closes #4893)
Files: src/gui_w32.c

Patch 8.1.1976

Problem: Travis log always shows test output.
Solution: Change script to avoid if/else. (Ozaki Kiichi, closes #4892)
Files: .travis.yml

Patch 8.1.1977

Problem: Terminal debugger plugin may hang.
Solution: Wait longer when still reading symbols.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.1978

Problem: The eval.c file is too big.
Solution: Move filter() and map() to list.c.
Files: src/eval.c, src/proto/eval.pro, src/list.c, src/proto/list.pro, src/evalfunc.c

Patch 8.1.1979

Problem: Code for handling file names is spread out.
Solution: Move code to new filepath.c file. Graduate FEAT_MODIFY_FNAME.
Files: src/filepath.c, Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/eval.c, src/evalfunc.c, src/ex_docmd.c, src/feature.h, src/findfile.c, src/if_cscope.c, src/message.c, src/misc1.c, src/proto.h, src/proto/eval.pro, src/proto/evalvars.pro, src/proto/filepath.pro, src/proto/findfile.pro, src/proto/message.pro, src/regexp.c, src/version.c

Patch 8.1.1980

Problem: Fix for search stat not tested.
Solution: Add a screenshot test. (Christian Brabandt)
Files: src/testdir/test_search_stat.vim, src/testdir/dumps/Test_searchstat_1.dump, src/testdir/dumps/Test_searchstat_2.dump

Patch 8.1.1981

Problem: The evalfunc.c file is too big.
Solution: Move undo functions to undo.c. Move cmdline functions to ex_getln.c. Move some container functions to list.c.
Files: src/evalfunc.c, src/undo.c, src/proto/undo.pro, src/ex_getln.c, src/proto/ex_getln.pro, src/list.c, src/proto/list.pro

Patch 8.1.1982

Problem: More functions can be used as methods.
Solution: Make popup functions usable as a method.
Files: runtime/doc/popup.txt, src/evalfunc.c, src/testdir/test_popupwin.vim

Patch 8.1.1983

Problem: Compiler nags for uninitialized variable and unused function.
Solution: Add unnecessary initialization. Move function inside #ifdef.
Files: src/memline.c, src/channel.c

Patch 8.1.1984

Problem: More functions can be used as methods.

Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_functions.vim, src/testdir/test_perl.vim,
src/testdir/test_prompt_buffer.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_pyx2.vim

Patch 8.1.1985

Problem: Code for dealing with paths is spread out.
Solution: Move path related functions from misc1.c to filepath.c.
Remove NO_EXPANDPATH.

Files: src/misc1.c, src/proto/misc1.pro, src/filepath.c,
src/evalfunc.c, src/globals.h, src/misc2.c, src/os_unix.c,
src/os_unix.h, src/proto/filepath.pro, src/scriptfile.c,
src/proto/misc2.pro, src/proto/scriptfile.pro, src/vim.h

Patch 8.1.1986

Problem: More functions can be used as methods.
Solution: Make textprop functions usable as a method.
Files: runtime/doc/textprop.txt, src/evalfunc.c,
src/testdir/test_textprop.vim

Patch 8.1.1987

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_clientserver.vim,
src/testdir/test_eval_stuff.vim, src/testdir/test_functions.vim,
src/testdir/test_reftime.vim, src/testdir/test_rename.vim

Patch 8.1.1988

Problem: :startinsert! does not work the same way as "A".
Solution: Use the same code to move the cursor. (closes #4896)
Files: src/ex_docmd.c, src/normal.c, src/proto/normal.pro,
src/testdir/test_edit.vim

Patch 8.1.1989

Problem: The evalfunc.c file is still too big.
Solution: Move f_pathshorten() to filepath.c. Move f_cscope_connection() to
if_cscope.c. Move diff_ functions to diff.c. Move timer_
functions to ex_cmds2.c. move callback functions to evalvars.c.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/filepath.c,
src/proto/filepath.pro, src/if_cscope.c, src/proto/if_cscope.pro,
src/diff.c, src/proto/diff.pro, src/ex_cmds2.c,
src/proto/ex_cmds2.pro, src/evalvars.c, src/proto/evalvars.pro

Patch 8.1.1990

Problem: Cannot build with eval but without cscope.
Solution: Always include if_cscope.pro.
Files: src/proto.h

Patch 8.1.1991

Problem: Still cannot build with eval but without cscope.
Solution: Move f_cscope_connection() outside of #ifdef.
Files: src/if_cscope.c

Patch 8.1.1992

Problem: The search stat moves when wrapping at the end of the buffer.
Solution: Put the "W" in front instead of at the end.
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.1.1993

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_buflines.vim, src/testdir/test_charsearch.vim,
src/testdir/test_clientserver.vim, src/testdir/test_cmdline.vim,
src/testdir/test_cursor_func.vim, src/testdir/test_diffmode.vim,
src/testdir/test_envron.vim, src/testdir/test_functions.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_popupwin.vim, src/testdir/test_search.vim,
src/testdir/test_searchpos.vim, src/testdir/test_utf8.vim

Patch 8.1.1994

Problem: MS-Windows: cannot build with eval but without cscope
Solution: Adjust the makefiles to always build if_cscope.obj.
Files: src/Make_mvc.mak, src/Make_cyg_ming.mak

Patch 8.1.1995

Problem: More functions can be used as methods.
Solution: Make sign functions usable as a method.
Files: runtime/doc/sign.txt, src/evalfunc.c, src/testdir/test_signs.vim

Patch 8.1.1996

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_bufwintabinfo.vim,
src/testdir/test_cursor_func.vim, src/testdir/test_expr.vim,
src/testdir/test_functions.vim, src/testdir/test_put.vim,
src/testdir/test_quickfix.vim, src/testdir/test_sha256.vim,
src/testdir/test_tabpage.vim, src/testdir/test_tagjump.vim,
src/testdir/test_vartabs.vim

Patch 8.1.1997

Problem: No redraw after a popup window filter is invoked.
Solution: Redraw if needed.
Files: src/popupwin.c, src/testdir/test_popupwin.vim
src/testdir/dumps/Test_popupwin_menu_filter_5.dump

Patch 8.1.1998

Problem: Redraw even when no popup window filter was invoked.
Solution: Only redraw when must_redraw was set to a larger value.
Files: src/popupwin.c

Patch 8.1.1999

Problem: Calling both PlaySoundW() and PlaySoundA().
Solution: Only use PlaySoundW(). (Dan Thompson, closes #4903)
Files: src/sound.c

Patch 8.1.2000

Problem: Plugin cannot get the current IME status.
Solution: Add the getimstatus() function. (closes #4904)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/mbyte.c,
src/proto/mbyte.pro, src/testdir/test_iminsert.vim

Patch 8.1.2001

Problem: Some source files are too big.
Solution: Move buffer and window related functions to evalbuffer.c and evalwindow.c. (Yegappan Lakshmanan, closes #4898)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/buffer.c, src/channel.c, src/evalbuffer.c, src/evalfunc.c, src/evalwindow.c, src/proto.h, src/proto/buffer.pro, src/proto/evalbuffer.pro, src/proto/evalfunc.pro, src/proto/evalwindow.pro, src/proto/window.pro, src/window.c

Patch 8.1.2002

Problem: Version number 2000 missing.
Solution: Add the number in the list of patches.
Files: src/version.c

Patch 8.1.2003

Problem: MS-Windows: code page 65001 is not recognized.
Solution: Use utf-8 for code page 65001. (Dan Thompson, closes #4902)
Files: src/mbyte.c

Patch 8.1.2004

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_breakindent.vim, src/testdir/test_expr.vim, src/testdir/test_functions.vim, src/testdir/test_sound.vim, src/testdir/test_spell.vim, src/testdir/test_substitute.vim, src/testdir/test_swap.vim, src/testdir/test_utf8.vim

Patch 8.1.2005

Problem: The regexp.c file is too big.
Solution: Move the backtracking engine to a separate file. (Yegappan Lakshmanan, closes #4905)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile, src/regexp.c, src/regexp_bt.c

Patch 8.1.2006

Problem: Build failure with huge features but without channel feature.
Solution: Add #ifdef. (Dominique Pelle, closes #4906)
Files: src/ui.c

Patch 8.1.2007

Problem: No test for what 8.1.1926 fixes.
Solution: Add a test case.
Files: src/testdir/test_highlight.vim

Patch 8.1.2008

Problem: Error for invalid range when using listener and undo. (Paul Jolly)
Solution: Do not change the cursor before the lines are restored.
(closes #4908)
Files: src/undo.c, src/testdir/test_listener.vim

Patch 8.1.2009

Problem: Cursorline highlighting not updated in popup window. (Marko Mahnič)
Solution: Check if the cursor position changed. (closes #4912)
Files: src/popupwin.c, src/structs.h, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_cursorline_7.dump

Patch 8.1.2010

Problem: New file uses old style comments.
Solution: Change to new style comments. (Yegappan Lakshmanan, closes #4910)
Files: src/regexp_bt.c

Patch 8.1.2011

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method. Make the window command test faster.
Files: runtime/doc/eval.txt, runtime/doc/testing.txt, src/evalfunc.c, src/testdir/test_assert.vim, src/testdir/test_gui.vim, src/testdir/test_messages.vim, src/testdir/test_options.vim, src/testdir/test_quickfix.vim, src/testdir/test_taglist.vim, src/testdir/test_termcodes.vim, src/testdir/test_timers.vim, src/testdir/test_vimscript.vim, src/testdir/test_viminfo.vim, src/testdir/test_window_cmd.vim

Patch 8.1.2012

Problem: More functions can be used as methods.
Solution: Make terminal functions usable as a method. Fix term_getattr().
Files: runtime/doc/terminal.txt, src/evalfunc.c, src/terminal.c, src/testdir/test_mksession.vim, src/testdir/test_terminal.vim

Patch 8.1.2013

Problem: More functions can be used as methods.
Solution: Make various functions usable as a method.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_cursor_func.vim, src/testdir/test_execute_func.vim, src/testdir/test_functions.vim, src/testdir/test_listchars.vim, src/testdir/test_timers.vim, src/testdir/test_undo.vim, src/testdir/test_window_cmd.vim, src/testdir/test_window_id.vim

Patch 8.1.2014

Problem: Terminal altscreen test fails sometimes.
Solution: Use WaitFor().
Files: src/testdir/test_terminal.vim

Patch 8.1.2015

Problem: Terminal altscreen test still fails sometimes.
Solution: Write the escape sequence in a file.

Files: src/testdir/test_terminal.vim

Patch 8.1.2016

Problem: Terminal altscreen test now fails on MS-Windows.

Solution: Skip the test on MS-Windows

Files: src/testdir/test_terminal.vim

Patch 8.1.2017

Problem: Cannot execute commands after closing the cmdline window.

Solution: Also trigger BufEnter and WinEnter. (closes #4762)

Files: runtime/doc/autocmd.txt, runtime/doc/cmdline.txt, src/ex_getln.c,
src/testdir/test_cmdline.vim

Patch 8.1.2018

Problem: Using freed memory when out of memory and displaying message.

Solution: Make a copy of the message first.

Files: src/main.c, src/message.c, src/normal.c

Patch 8.1.2019

Problem: **'cursorline'** always highlights the whole line.

Solution: Add **'cursorlineopt'** to specify what is highlighted.
(Ozaki Kiichi, closes #4693)

Files: runtime/doc/options.txt, runtime/doc/quickref.txt,
runtime/doc/syntax.txt, runtime/optwin.vim, src/option.c,
src/option.h, src/screen.c, src/structs.h,
src/testdir/Make_all.mak, src/testdir/gen_opt_test.vim,
src/testdir/test_alot.vim, src/testdir/test_cursorline.vim

Patch 8.1.2020

Problem: It is not easy to change the window layout.

Solution: Add win_splitmove(). (Andy Massimino, closes #4561)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/evalwindow.c,
src/proto/evalwindow.pro, src/testdir/test_window_cmd.vim

Patch 8.1.2021

Problem: Some global functions can be local to the file.

Solution: Add "static". (Yegappan Lakshmanan, closes #4917)

Files: src/ex_cmds2.c, src/filepath.c, src/hangulin.c, src/mbyte.c,
src/misc1.c, src/os_unix.c, src/proto/ex_cmds2.pro,
src/proto/filepath.pro, src/proto/hangulin.pro,
src/proto/mbyte.pro, src/proto/misc1.pro, src/proto/os_unix.pro,
src/proto/terminal.pro, src/proto/undo.pro, src/pty.c,
src/terminal.c, src/undo.c

Patch 8.1.2022

Problem: The option.c file is too big.

Solution: Move option definitions to a separate file. (Yegappan Lakshmanan,
closes #4918)

Files: Filelist, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile,
src/option.c, src/optiondefs.h

Patch 8.1.2023

Problem: No test for synIDattr() returning "strikethrough".

Solution: Extend the synIDattr() test. (Jaskaran Singh, closes #4929)

Files: src/testdir/test_syn_attr.vim

Patch 8.1.2024

Problem: Delete call commented out for debugging.
Solution: Restore the delete call. (Christian Brabandt)
Files: src/testdir/test_undo.vim

Patch 8.1.2025

Problem: MS-Windows: Including shlguid.h causes problems for msys2.
Solution: Do not include shlguid.h. (closes #4913)
Files: src/GvimExt/gvimext.h

Patch 8.1.2026

Problem: Possibly using uninitialized memory.
Solution: Check if "dict" is NULL. (closes #4925)
Files: src/ops.c

Patch 8.1.2027

Problem: MS-Windows: problem with ambiwidth characters.
Solution: handle ambiguous width characters in ConPTY on Windows 10 (1903).
(Nobuhiro Takasaki, closes #4411)
Files: src/Make_mvc.mak, src/Make_cyg_ming.mak, src/libvterm/src/parser.c,
src/libvterm/src/state.c, src/libvterm/src/termscreen.c,
src/libvterm/src/unicode.c, src/libvterm/src/vterm_internal.h,
src/misc2.c, src/os_win32.c, src/proto/misc2.pro,
src/proto/os_win32.pro

Patch 8.1.2028

Problem: Options test script does not work.
Solution: Use optiondefs.h for input.
Files: src/testdir/Makefile, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak

Patch 8.1.2029

Problem: Cannot control 'cursorline' highlighting well.
Solution: Add "screenline". (Christian Brabandt, closes #4933)
Files: runtime/doc/options.txt, src/change.c, src/main.c, src/option.c,
src/option.h, src/optiondefs.h, src/screen.c, src/structs.h,
src/highlight.c, src/testdir/dumps/Test_Xcursorline_1.dump,
src/testdir/dumps/Test_Xcursorline_2.dump,
src/testdir/dumps/Test_Xcursorline_3.dump,
src/testdir/dumps/Test_Xcursorline_4.dump,
src/testdir/dumps/Test_Xcursorline_5.dump,
src/testdir/dumps/Test_Xcursorline_6.dump,
src/testdir/dumps/Test_Xcursorline_7.dump,
src/testdir/dumps/Test_Xcursorline_8.dump,
src/testdir/dumps/Test_Xcursorline_9.dump,
src/testdir/dumps/Test_Xcursorline_10.dump,
src/testdir/dumps/Test_Xcursorline_11.dump,
src/testdir/dumps/Test_Xcursorline_12.dump,
src/testdir/dumps/Test_Xcursorline_13.dump,
src/testdir/dumps/Test_Xcursorline_14.dump,
src/testdir/dumps/Test_Xcursorline_15.dump,
src/testdir/dumps/Test_Xcursorline_16.dump,

```
src/testdir/dumps/Test_Xcursorline_17.dump,  
src/testdir/dumps/Test_Xcursorline_18.dump,  
src/testdir/gen_opt_test.vim, src/testdir/test_cursorline.vim,  
src/testdir/dumps/Test_cursorline_yank_01.dump,  
src/testdir/dumps/Test_wincolor_01.dump,  
src/testdir/dumps/Test_textprop_01.dump
```

Patch 8.1.2030

Problem: Tests fail when build with normal features and terminal.
(Dominique Pelle)
Solution: Disable tests that won't work. (closes #4932)
Files: src/testdir/test_popupwin.vim, src/testdir/test_terminal.vim

Patch 8.1.2031

Problem: Cursor position wrong when resizing and using conceal.
Solution: Set the flags that the cursor position is valid when setting the
row and column during redrawing. (closes #4931)
Files: src/screen.c, src/testdir/test_conceal.vim,
src/testdir/dumps/Test_conceal_resize_01.dump,
src/testdir/dumps/Test_conceal_resize_02.dump

Patch 8.1.2032

Problem: Scrollbar thumb wrong in popup window.
Solution: Adjust thumb size and position when scrolled.
Files: src/popupwin.c, src/testdir/dumps/Test_popupwin_scroll_2.dump

Patch 8.1.2033

Problem: Cannot build with tiny features.
Solution: Add #ifdef.
Files: src/screen.c

Patch 8.1.2034

Problem: Dark theme of GTK 3 not supported.
Solution: Add the "d" flag in '**guioptions**'. (Jonathan Conder, closes #4934)
Files: runtime/doc/options.txt, src/feature.h, src/gui.c,
src/gui_gtk_x11.c, src/option.h, src/proto/gui_gtk_x11.pro,
src/testdir/test_gui.vim

Patch 8.1.2035

Problem: Recognizing octal numbers is confusing.
Solution: Introduce scriptversion 4: do not use octal and allow for single
quote inside numbers.
Files: runtime/doc/eval.txt, src/vim.h, src/eval.c, src/scriptfile.c,
src/evalfunc.c, src/testdir/test_eval_stuff.vim,
src/testdir/test_functions.vim

Patch 8.1.2036 (after 8.1.2035)

Problem: The str2nr() tests fail.
Solution: Add missing part of patch.
Files: src/charset.c

Patch 8.1.2037

Problem: Can call win_gotoid() in cmdline window.
Solution: Disallow switching windows. (Yasuhiro Matsumoto, closes #4940)

Files: src/evalwindow.c, src/testdir/test_cmdline.vim

Patch 8.1.2038

Problem: has('vimscrip-4') is always 0.

Solution: Add "vimscrip-4" to the feature table. (Naruhiko Nishino, closes #4941)

Files: src/evalfunc.c, src/testdir/test_eval_stuff.vim

Patch 8.1.2039

Problem: Character from '**showbreak**' does not use '**wincolor**'. (Nick Jensen)

Solution: Mix with '**wincolor**'. (closes #4938)

Files: src/screen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_showbreak.dump

Patch 8.1.2040

Problem: No highlighting of current line in quickfix window.

Solution: Combine with line_attr.

Files: src/screen.c, src/testdir/test_quickfix.vim,
src/testdir/dumps/Test_quickfix_cwindow_1.dump,
src/testdir/dumps/Test_quickfix_cwindow_2.dump

Patch 8.1.2041 (after 8.1.2040)

Problem: No test for diff mode with syntax highlighting.

Solution: Add a test case.

Files: src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_syntax_1.dump

Patch 8.1.2042

Problem: The evalfunc.c file is too big.

Solution: Move getchar() and parse_queued_messages() to getchar.c.

Files: src/getchar.c, src/proto/getchar.pro, src/evalfunc.c, src/misc2.c,
src/proto/misc2.pro

Patch 8.1.2043

Problem: Not sufficient testing for quoted numbers.

Solution: Add a few more test cases.

Files: src/testdir/test_functions.vim, src/testdir/test_eval_stuff.vim

Patch 8.1.2044

Problem: No easy way to process postponed work. (Paul Jolly)

Solution: Add the SafeState autocommand event.

Files: runtime/doc/autocmd.txt, src/main.c, src/proto/main.pro,
src/vim.h, src/autocmd.c, src/channel.c, src/edit.c,
src/ex_getln.c

Patch 8.1.2045

Problem: The option.c file is too big.

Solution: Split off the code dealing with strings. (Yegappan Lakshmanan, closes #4937)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/option.c, src/option.h, src/optiondefs.h, src/optionstr.c,
src/ops.c, src/os_unix.c, src/proto.h, src/proto/option.pro,
src/proto/optionstr.pro

Patch 8.1.2046

Problem: SafeState may be triggered at the wrong moment.
Solution: Move it up higher to after where messages are processed. Add a SafeStateAgain event to trigger there.
Files: runtime/doc/autocmd.txt, src/main.c, src/proto/main.pro, src/getchar.c, src/channel.c, src/autocmd.c, src/vim.h

Patch 8.1.2047

Problem: Cannot check the current state.
Solution: Add the state() function.
Files: runtime/doc/eval.txt, src/misc1.c, src/proto/misc1.pro, src/evalfunc.c, src/proto/evalfunc.pro, src/main.c, src/proto/main.pro, src/channel.c, src/proto/channel.pro, src/userfunc.c, src/proto/userfunc.pro

Patch 8.1.2048

Problem: Not clear why SafeState and SafeStateAgain are not triggered.
Solution: Add log statements.
Files: src/getchar.c, src/main.c, src/proto/main.pro

Patch 8.1.2049 (after 8.1.2048)

Problem: Cannot build tiny version.
Solution: Add #ifdefs.
Files: src/main.c

Patch 8.1.2050

Problem: Popup window test fails in some configurations. (James McCoy)
Solution: Clear the command line.
Files: src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_scroll_10.dump

Patch 8.1.2051

Problem: Double-click test is a bit flaky.
Solution: Correct entry in list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.2052

Problem: Using "x" before a closed fold may delete that fold.
Solution: Do not translate 'x' do "dl". (Christian Brabandt, closes #4927)
Files: src/normal.c, src/testdir/test_fold.vim

Patch 8.1.2053

Problem: SafeStateAgain not triggered if callback uses feedkeys().
Solution: Check for safe state in the input loop. Make log messages easier to find. Add 'S' flag to state().
Files: src/main.c, src/proto/main.pro, src/getchar.c, runtime/doc/eval.txt

Patch 8.1.2054

Problem: Compiler test for Perl may fail.
Solution: Accept any error line number. (James McCoy, closes #4944)
Files: src/testdir/test_compiler.vim

Patch 8.1.2055

Problem: Not easy to jump to function line from profile.
Solution: Use "file:99" instead of "file line 99" so that "gf" works.
(Daniel Hahler, closes #4951)
Files: src/profiler.c, src/testdir/test_profile.vim

Patch 8.1.2056

Problem: "make test" for indent files doesn't cause make to fail.
Solution: Exit the script with ":cquit". (Daniel Hahler, closes #4949)
Files: runtime/indent/testdir/runtest.vim, .gitignore

Patch 8.1.2057

Problem: The screen.c file is much too big.
Solution: Split it in three parts. (Yegappan Lakshmanan, closes #4943)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/drawline.c, src/drawscreen.c, src/globals.h, src/proto.h,
src/proto/drawline.pro, src/proto/drawscreen.pro,
src/proto/screen.pro, src/screen.c, src/vim.h

Patch 8.1.2058

Problem: Function for ex command is named inconsistently.
Solution: Rename do_marks() to ex_marks().
Files: src/mark.c, src/proto/mark.pro, src/ex_cmds.h

Patch 8.1.2059

Problem: Fix for "x" deleting a fold has side effects.
Solution: Fix it where the fold is included.
Files: src/normal.c

Patch 8.1.2060

Problem: "precedes" in '**listchars**' not used properly.
Solution: Correctly handle the "precedes" char in list mode for long lines.
(Zach Wegner, Christian Brabandt, closes #4953)
Files: runtime/doc/options.txt, src/drawline.c,
src/testdir/test_display.vim, src/testdir/view_util.vim

Patch 8.1.2061

Problem: MS-Windows GUI: ":sh" crashes when trying to use a terminal.
Solution: Check for a NULL command. (Yasuhiro Matsumoto, closes #4958)
Files: src/os_win32.c

Patch 8.1.2062

Problem: The mouse code is spread out.
Solution: Move all the mouse code to mouse.c. (Yegappan Lakshmanan,
closes #4959)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/auto/configure, src/configure.ac, src/edit.c, src/ex_cmds.c,
src/ex_docmd.c, src/ex_getln.c, src/insexpand.c,
src/libvterm/src/termmouse.c, src/libvterm/src/mouse.c,
src/main.c, src/message.c, src/misc1.c, src/misc2.c, src/mouse.c,
src/normal.c, src/proto.h, src/proto/edit.pro,
src/proto/misc1.pro, src/proto/misc2.pro, src/proto/mouse.pro,

src/proto/normal.pro, src/proto/term.pro, src/proto/ui.pro,
src/search.c, src/term.c, src/ui.c, src/vim.h, src/window.c

Patch 8.1.2063

Problem: Some tests fail when +balloon_eval_term is missing but
_balloon_eval is present. (Dominique Pelle)
Solution: Check the right feature in the test. (closes #4962)
Files: src/testdir/test_popupwin.vim, src/testdir/test_terminal.vim

Patch 8.1.2064

Problem: MS-Windows: compiler warnings for unused arguments.
Solution: Add UNUSED. (Yegappan Lakshmanan, closes #4963)
Files: src/channel.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c,
src/gui_w32.c, src/main.c, src/memline.c, src/os_mswin.c,
src/os_win32.c, src/terminal.c, src/ui.c, src/undo.c

Patch 8.1.2065

Problem: Compiler warning building non-GUI with MinGW.
Solution: Adjust #ifdefs. (Yegappan Lakshmanan, closes #4964)
Files: sre/mouse.c

Patch 8.1.2066

Problem: No tests for state().
Solution: Add tests. Clean up some feature checks. Make "a" flag work.
Files: src/testdir/test_functions.vim, src/testdir/test_terminal.vim,
src/misc1.c

Patch 8.1.2067

Problem: No tests for SafeState and SafeStateAgain.
Solution: Add tests.
Files: src/testdir/test_autocmd.vim

Patch 8.1.2068 (after 8.1.2067)

Problem: Test for SafeState and SafeStateAgain may fail.
Solution: Accept more possible responses
Files: src/testdir/test_autocmd.vim

Patch 8.1.2069 (after 8.1.2068)

Problem: Test for SafeStateAgain may still fail.
Solution: Send another message to trigger SafeStateAgain.
Files: src/testdir/test_autocmd.vim

Patch 8.1.2070

Problem: Mouse code is spread out.
Solution: Move mouse terminal code parsing to mouse.c. (Yegappan Lakshmanan,
closes #4966)
Files: src/mouse.c, src/proto/mouse.pro, src/proto/term.pro, src/term.c

Patch 8.1.2071

Problem: When '**wincolor**' is set text property changes highlighting. (Andy
Stewart)
Solution: Fix combining colors. (closes #4968)
Files: src/drawline.c, src/testdir/test_highlight.vim,
src/testdir/dumps/Test_wincolor_01.dump

Patch 8.1.2072

Problem: "gk" moves to start of line instead of upwards.
Solution: Fix off-by-one error. (Christian Brabandt, closes #4969)
Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.1.2073

Problem: When editing a buffer '**colorcolumn**' may not work.
Solution: Set the buffer before copying option values. Call check_colorcolumn() after copying window options.
Files: src/buffer.c, src/option.c, src/proto/option.pro, src/proto/indent.pro, src/testdir/test_highlight.vim, src/testdir/dumps/Test_colorcolumn_1.dump

Patch 8.1.2074

Problem: Test for SafeState autocommand is a bit flaky.
Solution: Add to list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.2075

Problem: Get many log messages when waiting for a typed character.
Solution: Do not repeat the repeated messages when nothing happens.
Files: src/globals.h, src/channel.c, src/main.c

Patch 8.1.2076

Problem: Crash when trying to put a terminal buffer in a popup window.
Solution: Check for NULL buffer. Do not allow putting a terminal in a popup window.
Files: src/libvterm/src/termscreen.c, src/terminal.c, src/popupwin.c, runtime/doc/popup.txt, src/testdir/test_popupwin.vim

Patch 8.1.2077

Problem: The ops.c file is too big.
Solution: Move code for dealing with registers to a new file. (Yegappan Lakshmanan, closes #4982)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/ops.c, src/proto.h, src/proto/ops.pro, src/proto/register.pro, src/register.c, src/structs.h

Patch 8.1.2078

Problem: Build error with +textprop but without +terminal. (Tony Mechelynck)
Solution: Add #ifdef.
Files: src/popupwin.c

Patch 8.1.2079

Problem: Popup window test fails without +terminal.
Solution: Check for the +terminal feature.
Files: src/testdir/test_popupwin.vim

Patch 8.1.2080

Problem: The terminal API is limited and can't be disabled.
Solution: Add term_setapi() to set the function prefix. (Ozaki Kiichi, closes #2907)

Files: runtime/doc/eval.txt, runtime/doc/terminal.txt, src/channel.c, src/evalfunc.c, src/proto/terminal.pro, src/structs.h, src/terminal.c, src/testdir/term_util.vim, src/testdir/test_terminal.vim

Patch 8.1.2081

Problem: The spell.c file is too big.

Solution: Move the code for spell suggestions to a separate file. (Yegappan Lakshmanan, closes #4988)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/proto.h, src/proto/spell.pro, src/proto/spellsuggest.pro, src/spell.c, src/spell.h, src/spellsuggest.c

Patch 8.1.2082

Problem: Some files have a weird name to fit in 8.3 characters.

Solution: Use a nicer names.

Files: Filelist, Makefile, src/popupmnu.c, src/popupmenu.c, src/proto/popupmnu.pro, src/proto/popupmenu.pro, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/proto.h, src/README.md, src/uninstal.c, src/uninstall.c, uninstal.txt, uninstall.txt, nsis/gvim.nsi, src/INSTALLpc.txt, src/dosinst.c, src/dosinst.h

Patch 8.1.2083

Problem: Multi-byte chars do not work properly with "%.s" in printf().

Solution: Use mb_ptr2cells(). Daniel Hahler, closes #4989)

Files: src/testdir/test_expr.vim, src/message.c

Patch 8.1.2084

Problem: Amiga: cannot get the user name.

Solution: Use getpwuid() if available. (Ola Söder, closes #4985)

Files: src/os_amiga.c, src/os_amiga.h

Patch 8.1.2085

Problem: MS-Windows: draw error moving cursor over double-cell character.

Solution: Move the cursor to the left edge if needed. (Nobuhiro Takasaki, closes #4986)

Files: src/os_win32.c

Patch 8.1.2086 (after 8.1.2082)

Problem: Missing a few changes for the renamed files.

Solution: Rename in a few more places. (Ken Takata)

Files: nsis/README.txt, runtime/doc/gui_w32.txt, runtime/doc/usr_90.txt, src/GvimExt/GvimExt.reg, src/GvimExt/README.txt, src/proto/popupmenu.pro, src/proto/popupmnu.pro

Patch 8.1.2087

Problem: Cannot easily select one test function to execute.

Solution: Support the \$TEST_FILTER environment variable. (Ozaki Kiichi, closes #2695)

Files: src/Makefile, src/testdir/runtest.vim, src/testdir/summarize.vim

Patch 8.1.2088

Problem: Renamed libvterm mouse.c file not in distributed file list.
Solution: Rename the file in the file list.
Files: Filelist

Patch 8.1.2089 (after 8.1.2087)

Problem: Do not get a hint that \$TEST_FILTER was active.
Solution: Mention \$TEST_FILTER if no functions were executed.
Files: src/testdir/runtest.vim

Patch 8.1.2090

Problem: Not clear why channel log file ends.
Solution: Add a "closing" line.
Files: src/channel.c

Patch 8.1.2091

Problem: Double free when memory allocation fails. (Zu-Ming Jiang)
Solution: Use VIM_CLEAR() instead of vim_free(). (closes #4991)
Files: src/getchar.c

Patch 8.1.2092

Problem: MS-Windows: redirect in system() does not work.
Solution: Handle '**shellxescape**' and '**shellxquote**' better. (Yasuhiro Matsumoto, closes #2054)
Files: src/ex_cmds.c, src/misc2.c, src/testdir/test_system.vim

Patch 8.1.2093

Problem: MS-Windows: system() test fails.
Solution: Expect CR when using systemlist().
Files: src/testdir/test_system.vim

Patch 8.1.2094

Problem: The fileio.c file is too big.
Solution: Move buf_write() to bufwrite.c. (Yegappan Lakshmanan, closes #4990)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/bufwrite.c, src/fileio.c, src/option.c, src/proto.h, src/proto/bufwrite.pro, src/proto/fileio.pro, src/structs.h

Patch 8.1.2095

Problem: Leaking memory when getting item from dict.
Solution: Also free the key when not evaluating.
Files: src/dict.c

Patch 8.1.2096

Problem: Too many #ifdefs.
Solution: Graduate FEAT_COMMENTS.
Files: src/feature.h, src/buffer.c, src/change.c, src/edit.c, src/evalfunc.c, src/fold.c, src/insexpand.c, src/misc1.c, src/normal.c, src/ops.c, src/option.c, src/optionstr.c, src/search.c, src/version.c, src/globals.h, src/option.h, src/optiondefs.h, src/structs.h, runtime/doc/change.txt, runtime/doc/options.txt, runtime/doc/varsious.txt

Patch 8.1.2097

Problem: :mksession is not sufficiently tested.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #4992)
Files: src/testdir/test_mksession.vim

Patch 8.1.2098 (after 8.1.2097)

Problem: mksession test fails on MS-Windows.
Solution: Skip testing with backslashes on MS-Windows.
Files: src/testdir/test_mksession.vim

Patch 8.1.2099

Problem: state() test fails on some Mac systems.
Solution: Increase the wait time. (closes #4983)
Files: src/testdir/test_functions.vim

Patch 8.1.2100

Problem: :mksession is not sufficiently tested.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #4993)
Files: src/testdir/test_mksession.vim

Patch 8.1.2101

Problem: write_session_file() often defined but not used.
Solution: Adjust the #ifdef. (Yegappan Lakshmanan, closes #4998)
Files: src/session.c

Patch 8.1.2102

Problem: Can't build with GTK and FEAT_GUI_GNOME. (Tony Mechelynck)
Solution: Adjust the #ifdef. (Yegappan Lakshmanan)
Files: src/session.c

Patch 8.1.2103

Problem: wrong error message if "termdebugger" is not executable.
Solution: Check if "termdebugger" is executable and give a clear error message. (Ozaki Kiichi, closes #5000) Fix indents.
Files: runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.1.2104

Problem: The normal.c file is too big.
Solution: Move do_pending_operator() to ops.c. (Yegappan Lakshmanan, closes #4999).
Files: src/normal.c, src/ops.c, src/proto/normal.pro, src/proto/ops.pro, src/globals.h

Patch 8.1.2105

Problem: MS-Windows: system() may crash.
Solution: Do not use "itmp" when it is NULL. (Yasuhiro Matsumoto, closes #5005)
Files: src/ex_cmds.c

Patch 8.1.2106

Problem: No tests for dragging the mouse beyond the window.
Solution: Add a test. (Dominique Pelle, closes #5004)
Files: src/testdir/test_termcodes.vim

Patch 8.1.2107

Problem: Various memory leaks reported by asan.
Solution: Free the memory. (Ozaki Kiichi, closes #5003)
Files: src/buffer.c, src/change.c, src/eval.c, src/evalfunc.c,
src/option.c, src/popupwin.c, src/proto/change.pro,
src/scriptfile.c, src/terminal.c, src/testdir/test_method.vim

Patch 8.1.2108

Problem: Cannot close the cmdline window from CmdWinEnter. (George Brown)
Solution: Reset cmdwin_result earlier. (Christian Brabandt, closes #4980)
Files: src/ex_getln.c, src/testdir/test_autocmd.vim

Patch 8.1.2109

Problem: popup_getoptions() hangs with tab-local popup.
Solution: Correct pointer name. (Marko Mahnič, closes #5006)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.2110

Problem: **CTRL-C** closes two popups instead of one.
Solution: Reset got_int when the filter consumed the key.
Files: src/getchar.c, src/testdir/test_popupwin.vim

Patch 8.1.2111

Problem: Viminfo file not sufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5009)
Files: src/testdir/test_viminfo.vim

Patch 8.1.2112

Problem: Build number for ConPTY is outdated.
Solution: Update to new build number. (Nobuhiro Takasaki, closes #5014)
Files: src/os_win32.c

Patch 8.1.2113

Problem: ":help expr-!~?" only works after searching.
Solution: Escape "~" after "expr-". (closes #5015)
Files: src/ex_cmds.c, src/testdir/test_help.vim

Patch 8.1.2114

Problem: When a popup is closed with **CTRL-C** the callback aborts.
Solution: Reset got_int when invoking the callback. (closes #5008)
Files: src/popupwin.c

Patch 8.1.2115

Problem: MS-Windows: shell commands fail if &shell contains a space.
Solution: Use quotes instead of escaping. (closes #4920)
Files: src/option.c, src/os_win32.c, src/testdir/test_startup.vim,
src/testdir/test_system.vim, src/vimrun.c,

Patch 8.1.2116

Problem: No check for out of memory.
Solution: Check for NULL pointer.
Files: src/option.c

Patch 8.1.2117

Problem: CursorLine highlight used while 'cursorline' is off.
Solution: Check 'cursorline' is set. (closes #5017)
Files: src/drawline.c, src/testdir/test_cursorline.vim

Patch 8.1.2118

Problem: Termcodes test fails when \$TERM is "dumb".
Solution: Skip the test. (James McCoy, closes #5019)
Files: src/testdir/test_termcodes.vim

Patch 8.1.2119

Problem: memory access error for empty string when 'encoding' is a single byte encoding.
Solution: Check for empty string when getting the length. (Dominique Pelle, closes #5021, closes #5007)
Files: src/macros.h

Patch 8.1.2120

Problem: Some MB_ macros are more complicated than necessary. (Dominique Pelle)
Solution: Simplify the macros. Expand inline.
Files: src/macros.h, src/beval.c, src/diff.c src/eval.c src/evalfunc.c src/ex_getln.c, src/filepath.c, src/findfile.c, src/getchar.c, src/highlight.c, src/ops.c, src/os_mswin.c, src/popupmenu.c, src/search.c, src/spell.c, src/spellsuggest.c, src/terminal.c

Patch 8.1.2121

Problem: Mode is not updated when switching to terminal in Insert mode.
Solution: Redraw the mode when entering a terminal window. (Jason Franklin)
Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.1.2122 (after 8.1.2121)

Problem: Cannot build without terminal feature.
Solution: Add #ifdef.
Files: src/window.c

Patch 8.1.2123

Problem: Parsing CSI sequence is messy.
Solution: Generalize parsing a CSI sequence.
Files: src/term.c

Patch 8.1.2124

Problem: Ruler is not updated if win_execute() moves cursor.
Solution: Update the status line. (closes #5022)
Files: src/evalwindow.c, src/testdir/test_execute_func.vim

Patch 8.1.2125

Problem: Fnamemodify() fails when repeating :e.
Solution: Do not go before the tail. (Rob Pilling, closes #5024)
Files: src/filepath.c, src/testdir/test_fnamemodify.vim

Patch 8.1.2126

Problem: Viminfo not sufficiently tested.
Solution: Add more test cases. Clean up comments. (Yegappan Lakshmanan, closes #5032)

Files: src/search.c, src/structs.h, src/testdir/test_viminfo.vim,
src/viminfo.c

Patch 8.1.2127

Problem: The indent.c file is a bit big.

Solution: Move C-indent code to a new cindent.c file. Move other
indent-related code to indent.c. (Yegappan Lakshmanan,
closes #5031)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/change.c, src/cindent.c, src/edit.c, src/evalfunc.c,
src/ex_cmds.c, src/globals.h, src/indent.c, src/misc1.c,
src/ops.c, src/proto.h, src/proto/cindent.pro, src/proto/edit.pro,
src/proto/ex_cmds.pro, src/proto/indent.pro, src/proto/misc1.pro,
src/proto/ops.pro, src/userfunc.c

Patch 8.1.2128

Problem: Renamed libvterm sources makes merging difficult.

Solution: Rename back to the original name and only rename the .o files.
Also clean the libvterm build artifacts. (James McCoy,
closes #5027)

Files: src/libvterm/src/termmouse.c, src/libvterm/src/mouse.c,
src/libvterm/src/termscreen.c, src/libvterm/src/screen.c,
src/Makefile, src/configure.ac, src/auto/configure,
src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.1.2129

Problem: Using hard coded executable path in test.

Solution: Use v:proppath. Use \$VIMRUNTIME instead of "runtime". (James
McCoy, closes #5025)

Files: src/testdir/test49.vim, src/testdir/test_compiler.vim,
src/testdir/test_spell.vim

Patch 8.1.2130 (after 8.1.2128)

Problem: MSVC build fails.

Solution: Add the source file name explicitly.

Files: src/Make_mvc.mak

Patch 8.1.2131 (after 8.1.2129)

Problem: MSVC tests fail.

Solution: Replace backslashes with slashes.

Files: src/testdir/test_compiler.vim, src/testdir/test_spell.vim

Patch 8.1.2132

Problem: MS-Windows: screen mess when not recognizing insider build.

Solution: Always move the cursor to the first column first. (Nobuhiro
Takasaki, closes #5036)

Files: src/os_win32.c

Patch 8.1.2133

Problem: Some tests fail when run as root.

Solution: Add CheckNotRoot and use it. (James McCoy, closes #5020)

Files: src/testdir/check.vim, src/testdir/shared.vim,
src/testdir/test_rename.vim, src/testdir/test_swap.vim,

src/testdir/test_terminal.vim, src/testdir/test_viminfo.vim

Patch 8.1.2134

Problem: Modifier keys are not always recognized.
Solution: Handle key codes generated by xterm with modifyOtherKeys set. Add this to libvterm so we can debug it.
Files: src/term.c, src/getchar.c, src/libvterm/src/vterm_internal.h, src/libvterm/src/state.c, src/libvterm/src/keyboard.c, src/libvterm/include/vterm.h, src/globals.h, src/terminal.c

Patch 8.1.2135

Problem: With modifyOtherKeys Alt-a does not work properly.
Solution: Remove the ALT modifier. Get multibyte after applying ALT.
Files: src/getchar.c

Patch 8.1.2136

Problem: using freed memory with autocmd from fuzzer. (Dhiraj Mishra, Dominique Pelle)
Solution: Avoid using "wp" after autocommands. (closes #5041)
Files: src/window.c, src/testdir/test_autocmd.vim

Patch 8.1.2137

Problem: Parsing the termresponse is not tested.
Solution: Add a first test. (related to #5042)
Files: src/testdir/test_termcodes.vim

Patch 8.1.2138

Problem: Including the build number in the Win32 binary is confusing.
Solution: Only use the patchlevel.
Files: src/vim.rc

Patch 8.1.2139

Problem: The modifyOtherKeys codes are not tested.
Solution: Add a test case.
Files: src/testdir/test_termcodes.vim

Patch 8.1.2140

Problem: "gk" and "gj" do not work correctly in number column.
Solution: Allow for a negative "curswant". (Zach Wegner, closes #4969)
Files: src/testdir/test_normal.vim, src/misc2.c, src/normal.c

Patch 8.1.2141

Problem: :tselect has an extra hit-enter prompt.
Solution: Do not set need_wait_return when only moving the cursor. (closes #5040)
Files: src/message.c, src/testdir/test_tagjump.vim, src/testdir/dumps/Test_tselect_1.dump

Patch 8.1.2142

Problem: Some key mappings do not work with modifyOtherKeys.
Solution: Remove the Shift modifier if it is already included in the key.
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.1.2143

Problem: Cannot see each command even when **'verbose'** is set.
Solution: List each command when **'verbose'** is at least 16.
Files: src/ex_docmd.c src/testdir/test_tagjump.vim,
src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_verbose_option_1.dump

Patch 8.1.2144

Problem: Side effects when using t_ti to enable modifyOtherKeys.
Solution: Add t_TI and t_TE.
Files: runtime/doc/term.txt, src/term.c, src/optiondefs.h, src/term.h,

Patch 8.1.2145

Problem: Cannot map <C-H> when modifyOtherKeys is enabled.
Solution: Add the <C-H> mapping twice, both with modifier and as 0x08. Use only the first one when modifyOtherKeys has been detected.
Files: src/term.c, src/eval.c, src/getchar.c, src/globals.h,
src/gui_mac.c, src/gui_w32.c, src/highlight.c, src/if_ole.cpp,
src/main.c, src/map.c, src/menu.c, src/misc2.c, src/option.c,
src/proto/misc2.pro, src/proto/term.pro,
src/testdir/test_termcodes.vim, src/structs.h, src/terminal.c,
src/usercmd.c, src/vim.h

Patch 8.1.2146 (after 8.1.2145)

Problem: Build failure.
Solution: Include omitted changed file.
Files: src/optionstr.c

Patch 8.1.2147

Problem: Crash when allocating memory fails. (Zu-Ming Jiang)
Solution: Check that **'spellcapcheck'** is not NULL. (closes #5048)
Files: src/spell.c

Patch 8.1.2148

Problem: No test for right click extending Visual area.
Solution: Add a test. (Dominique Pelle, closes #5018)
Files: src/testdir/test_termcodes.vim

Patch 8.1.2149

Problem: Crash when running out of memory very early.
Solution: Do not use IObuff when it's NULL. (closes #5052)
Files: src/message.c

Patch 8.1.2150

Problem: No test for **'ttymouse'** set from xterm version response.
Solution: Test the three possible values.
Files: src/testdir/test_termcodes.vim

Patch 8.1.2151

Problem: State test is a bit flaky.
Solution: Add to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.2152

Problem: Problems navigating tags file on macOS Catalina.

Solution: Use fseek instead of lseek. (John Lamb, fixes #5061)
Files: src/tag.c

Patch 8.1.2153

Problem: Combining text property and syntax highlight is wrong. (Nick Jensen)

Solution: Compute the syntax highlight attribute much earlier. (closes #5057)

Files: src/drawline.c, src/testdir/test_textprop.vim, src/testdir/dumps/Test_textprop_syn_1.dump

Patch 8.1.2154

Problem: Quickfix window height wrong when there is a tabline. (Daniel Hahler)

Solution: Take the tabline height into account. (closes #5058)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.2155

Problem: In a terminal window '**cursorlineopt**' does not work properly.

Solution: Check the '**cursorlineopt**' value. (closes #5055)

Files: src/drawline.c, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_normal_1.dump, src/testdir/dumps/Test_terminal_normal_2.dump, src/testdir/dumps/Test_terminal_normal_3.dump

Patch 8.1.2156

Problem: First character after Tab is not highlighted.

Solution: Remember the syntax attribute for a column.

Files: src/drawline.c, src/testdir/test_syntax.vim, src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.1.2157

Problem: Libvterm source files missing from distribution.

Solution: Rename source files. (closes #5065)

Files: Filelist

Patch 8.1.2158

Problem: Terminal attributes missing in Terminal-normal mode.

Solution: Use "syntax_attr".

Files: src/drawline.c, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_dumpload.dump

Patch 8.1.2159

Problem: Some mappings are listed twice.

Solution: Skip mappings duplicated for modifyOtherKeys. (closes #5064)

Files: src/map.c, src/testdir/test_mapping.vim

Patch 8.1.2160

Problem: Cannot build with +syntax but without +terminal.

Solution: Add #ifdef.

Files: src/drawline.c

Patch 8.1.2161

Problem: Mapping test fails.

Solution: Run the test separately.
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 8.1.2162

Problem: Popup resize test is flaky. (Christian Brabandt)
Solution: Add the function to the list of flaky tests.
Files: src/testdir/runtest.vim

Patch 8.1.2163

Problem: Cannot build with +spell but without +syntax.
Solution: Add #ifdef. (John Marriott)
Files: src/drawline.c

Patch 8.1.2164

Problem: Stuck when using "j" in a popupwin with popup_filter_menu if a line wraps.
Solution: Check the cursor line is visible. (closes #4577)
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_wrap_1.dump, src/testdir/dumps/Test_popupwin_wrap_2.dump

Patch 8.1.2165

Problem: Mapping test fails on Mac.
Solution: Remove the default Mac mapping.
Files: src/testdir/test_mapping.vim

Patch 8.1.2166

Problem: Rubyeval() not tested as a method.
Solution: Change a test case.
Files: src/testdir/test_ruby.vim

Patch 8.1.2167

Problem: Mapping test fails on MS-Windows.
Solution: Remove all the existing Insert-mode mappings.
Files: src/testdir/test_mapping.vim

Patch 8.1.2168

Problem: Heredoc assignment not skipped in if block.
Solution: Check if "skip" is set. (closes #5063)
Files: src/evalvars.c, src/testdir/test_let.vim

Patch 8.1.2169

Problem: Terminal flags are never reset.
Solution: Reset the flags when setting '**term**'.
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.1.2170 (after 8.1.2169)

Problem: Cannot build without the +termresponse feature.
Solution: Add #ifdef.
Files: src/term.c

Patch 8.1.2171

Problem: Mouse support not always available.
Solution: Enable mouse support also in tiny version. Do not define

FEAT_MOUSE_XTERM on MS-Windows (didn't really work).
Files: src/feature.h, src/edit.c, src/evalfunc.c, src/ex_getln.c,
src/getchar.c, src/message.c, src/misc1.c, src/mouse.c,
src/move.c, src/normal.c, src/ops.c, src/option.c,
src/optionstr.c, src/os_unix.c, src/os_win32.c, src/register.c,
src/term.c, src/testing.c, src/window.c, src/globals.h,
src/option.h, src/optiondefs.h, src/os_win32.h, src/vim.h,
src/version.c

Patch 8.1.2172

Problem: Spell highlight is wrong at start of the line.
Solution: Fix setting the "v" variable. (closes #5078)
Files: src/drawline.c, src/testdir/test_spell.vim,
src/testdir/dumps/Test_spell_1.dump

Patch 8.1.2173

Problem: Searchit() has too many arguments.
Solution: Move optional arguments to a struct. Add the "wrapped" argument.
Files: src/search.c, src/proto/search.pro, src/structs.h, src/evalfunc.c,
src/ex_docmd.c, src/gui.c, src/quickfix.c, src/spell.c, src/tag.c,
src/ex_getln.c, src/insexpand.c, src/normal.c

Patch 8.1.2174

Problem: Screen not recognized as supporting "sgr" mouse codes.
Solution: Recognize screen 4.7. (Jordan Christiansen, closes #5042)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.1.2175

Problem: Meson files are not recognized.
Solution: Add the meson filetype. (Liam Beguin, Nirbheek Chauhan,
closes #5056) Also recognize hollywood.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.2176

Problem: Syntax attributes not combined with Visual highlighting. (Arseny
Nasokin)
Solution: Combine the attributes. (closes #5083)
Files: src/drawline.c, src/testdir/test_syntax.vim,
src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.1.2177

Problem: Dart files are not recognized.
Solution: Add a filetype rule. (Eugene Ciurana, closes #5087)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.2178

Problem: Accessing uninitialized memory in test.
Solution: Check if there was a match before using the match position.
(Dominique Pelle, closes #5088)
Files: src/search.c

Patch 8.1.2179

Problem: Pressing "q" at the more prompt doesn't stop Python output. (Daniel
Hahler)

Solution: Check for `got_int` in `writer()`. (closes #5053)
Also do this for Lua.
Files: `src/if_py_both.h`, `src/if_lua.c`

Patch 8.1.2180

Problem: Error E303 is not useful when **'directory'** is empty.
Solution: Skip the error message. (Daniel Hahler, #5067)
Files: `src/memline.c`, `src/testdir/test_recover.vim`,
`runtime/doc/options.txt`, `runtime/doc/message.txt`

Patch 8.1.2181

Problem: Highlighting wrong when item follows tab.
Solution: Don't use syntax attribute when `n_extra` is non-zero.
(Christian Brabandt, closes #5076)
Files: `src/drawline.c`, `src/feature.h`,
`src/testdir/dumps/Test_syntax_c_01.dump`

Patch 8.1.2182

Problem: Test42 seen as binary by git diff.
Solution: Add `.gitattributes` file. Make explicit that **'cpo'** does not
contain 'S'. (Daniel Hahler, closes #5072)
Files: `.gitattributes`, `Filelist`, `src/testdir/test42.in`

Patch 8.1.2183

Problem: Running a test is a bit verbose.
Solution: Silence some messages. (Daniel Hahler, closes #5070)
Files: `src/testdir/Makefile`

Patch 8.1.2184

Problem: Option context is not copied when splitting a window. (Daniel
Hahler)
Solution: Copy the option context, so that `":verbose set"` works.
(closes #5066)
Files: `src/option.c`, `src/testdir/test_options.vim`

Patch 8.1.2185 (after 8.1.2181)

Problem: Syntax test fails.
Solution: Add missing file patch.
Files: `src/testdir/test_syntax.vim`

Patch 8.1.2186 (after 8.1.2184)

Problem: Cannot build without the `+eval` feature.
Solution: Move line inside `#ifdef`.
Files: `src/option.c`

Patch 8.1.2187

Problem: Error for bad regexp even though regexp is not used when writing
a file. (Arseny Nasokin)
Solution: Ignore regexp errors. (closes #5059)
Files: `src/cmdexpand.c`, `src/ex_docmd.c`, `src/testdir/test_writefile.vim`

Patch 8.1.2188 (after 8.1.2187)

Problem: Build error for missing define.
Solution: Add missing change.

Files: src/vim.h

Patch 8.1.2189

Problem: Syntax highlighting wrong for tab.

Solution: Don't clear syntax attribute n_extra is non-zero.

Files: src/drawline.c, src/testdir/test_syntax.vim,
src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.1.2190

Problem: Syntax test fails on Mac.

Solution: Limit the window size to 20 rows.

Files: src/testdir/test_syntax.vim,
src/testdir/dumps/Test_syntax_c_01.dump

Patch 8.1.2191

Problem: When using modifyOtherKeys **CTRL-X** mode may not work.

Solution: Recognize a control character also in the form with a modifier.

Files: src/getchar.c

Patch 8.1.2192

Problem: Cannot easily fill the info popup asynchronously.

Solution: Add the "popuphidden" value to '**completeopt**'. (closes #4924)

Files: src/popupmenu.c, src/proto/popupmenu.pro, src/popupwin.c,
src/proto/popupwin.pro, src/vim.h, runtime/doc/options.txt,
runtime/doc/insert.txt, src/ex_cmds.c, src/proto/ex_cmds.pro,
src/optionstr.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_infopopup_hidden_1.dump,
src/testdir/dumps/Test_popupwin_infopopup_hidden_2.dump,
src/testdir/dumps/Test_popupwin_infopopup_hidden_3.dump

Patch 8.1.2193

Problem: Popup_setoptions(popup_getoptions()) does not work.

Solution: Also accept a list with three entries for "moved" and
"mousemoved". (closes #5081)

Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.2194

Problem: ModifyOtherKeys is not enabled by default.

Solution: Add t_TI and t_TE to the builtin xterm termcap.

Files: runtime/doc/map.txt, src/term.c

Patch 8.1.2195

Problem: Vim does not exit when closing a terminal window and it is the
last window.

Solution: Exit Vim if the closed terminal window is the last one.
(closes #4539)

Files: runtime/doc/terminal.txt, src/terminal.c, src/ex_docmd.c,
src/proto/ex_docmd.pro, src/testdir/test_terminal.vim

Patch 8.1.2196

Problem: MS-Windows: running tests with MSVC lacks updates.

Solution: Improve running individual tests on MS-Windows. (closes #4922)

Files: src/Make_mvc.mak, src/testdir/Make_dos.mak

Patch 8.1.2197

Problem: ExitPre autocommand may cause accessing freed memory.
Solution: Check the window pointer is still valid. (closes #5093)
Files: src/testdir/test_exit.vim, src/ex_docmd.c

Patch 8.1.2198

Problem: Crash when using :center in autocommand.
Solution: Bail out early for an empty line. (Dominique Pelle, closes #5095)
Files: src/ex_cmds.c, src/testdir/test_textformat.vim

Patch 8.1.2199

Problem: Build failure when using normal features without GUI and EXITFREE defined.
Solution: Add #ifdef. (Dominique Pelle, closes #5106)
Files: src/scriptfile.c

Patch 8.1.2200

Problem: Crash when memory allocation fails.
Solution: Check for NULL curwin and curbuf. (Christian Brabandt, closes #4839)
Files: src/getchar.c

Patch 8.1.2201

Problem: Cannot build with dynamically linked Python 3.8.
Solution: Implement py3__Py_DECREF() and py3__Py_XDECREF(). (Ken Takata, closes #4080)
Files: src/if_python3.c

Patch 8.1.2202

Problem: MS-Windows: build failure with GUI and small features.
Solution: Add #ifdef. (Michael Soyka, closes #5097)
Files: src/gui_w32.c

Patch 8.1.2203

Problem: Running libvterm tests without the +terminal feature.
Solution: Only add the libvterm test target when building libvterm.
Files: src/configure.ac, src/auto/configure, src/config.mk.in, src/Makefile

Patch 8.1.2204

Problem: Crash on exit when closing terminals. (Corey Hickey)
Solution: Actually wait for the job to stop. (closes #5100)
Files: src/terminal.c

Patch 8.1.2205

Problem: Sign entry structure has confusing name.
Solution: Rename signlist_T to sign_entry_T and prefix se_ to the fields.
Files: src/structs.h, src/netbeans.c, src/sign.c, src/globals.h, src/drawline.c

Patch 8.1.2206

Problem: No test for fixed issue #3893.
Solution: Add a test. (Christian Brabandt, #3893)

Files: src/testdir/test_display.vim,
src/testdir/dumps/Test_winline_rnu.dump

Patch 8.1.2207

Problem: "gn" doesn't work quite right. (Jaehwang Jerry Jung)
Solution: Improve and simplify the search logic. (Christian Brabandt,
closes #5103, closes #5075)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.1.2208

Problem: Unix: Tabs in output might be expanded to spaces.
Solution: Reset the XTABS flag. (closes #5108)
Files: src/os_unix.c

Patch 8.1.2209

Problem: LF in escape codes may be expanded to CR-LF.
Solution: Do not expand LF in escape codes to CR-LF. (closes #5107)
Files: src/term.c

Patch 8.1.2210

Problem: Using negative offset for popup_create() does not work.
Solution: Use -1 instead of zero. (closes #5111)
Files: src/popupwin.c, src/popupwin.vim, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_corners.dump

Patch 8.1.2211

Problem: Listener callback "added" argument is not the total. (Andy
Massimino)
Solution: Compute the total. (closes #5105)
Files: src/change.c, src/testdir/test_listener.vim

Patch 8.1.2212

Problem: Cannot see the selection type in :reg output. (Ayberk Aydın)
Solution: Add c/l/b. (Christian Brabandt, closes #5110, closes #4546)
Files: runtime/doc/change.txt, src/register.c,
src/testdir/test_registers.vim

Patch 8.1.2213

Problem: Popup_textprop tests fail.
Solution: Adjust the column and line positioning.
Files: src/popupwin.c

Patch 8.1.2214

Problem: Too much is redrawn when 'cursorline' is set.
Solution: Don't do a complete redraw. (closes #5079)
Files: src/main.c, src/change.c, src/drawscreen.c,
src/testdir/dumps/Test_Xcursorline_13.dump,
src/testdir/dumps/Test_Xcursorline_14.dump,
src/testdir/dumps/Test_Xcursorline_15.dump,
src/testdir/dumps/Test_Xcursorline_16.dump,
src/testdir/dumps/Test_Xcursorline_17.dump,
src/testdir/dumps/Test_Xcursorline_18.dump

Patch 8.1.2215

Problem: Unreachable code in adjusting text prop columns.
Solution: Remove the code. (Christian Brabandt)
Files: src/textprop.c

Patch 8.1.2216

Problem: Text property in wrong place after :substitute.
Solution: Pass the new column instead of the old one. (Christian Brabandt, closes #4427)
Files: src/ex_cmds.c, src/testdir/test_textprop.vim

Patch 8.1.2217

Problem: Compiler warning for unused variable.
Solution: Move variable inside #ifdef. (John Marriott)
Files: src/ex_cmds.c

Patch 8.1.2218

Problem: "gN" is off by one in Visual mode.
Solution: Check moving forward. (Christian Brabandt, #5075)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.1.2219

Problem: No autocommand for open window with terminal.
Solution: Add TerminalWinOpen. (Christian Brabandt)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/terminal.c, src/testdir/test_terminal.vim, src/vim.h

Patch 8.1.2220

Problem: :cfile does not abort like other quickfix commands.
Solution: Abort when desired. Add tests for aborting. (Yegappan Lakshmanan, closes #5121)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.2221

Problem: Cannot filter :disp output.
Solution: Support filtering :disp output. (Andy Massimino, closes #5117)
Files: runtime/doc/various.txt, src/register.c, src/testdir/test_filter_cmd.vim

Patch 8.1.2222

Problem: Accessing invalid memory. (Dominique Pelle)
Solution: Reset highlight_match every time. (closes #5125)
Files: src/ex_getln.c

Patch 8.1.2223

Problem: Cannot see what buffer an ml_get error is for.
Solution: Add the buffer number and name in the message
Files: src/memline.c

Patch 8.1.2224

Problem: Cannot build Amiga version.
Solution: Add dummy mch_setmouse(). (Ola Söder, closes #5126)
Files: src/os_amiga.c, src/proto/os_amiga.pro

Patch 8.1.2225

Problem: The "last used" info of a buffer is under used.
Solution: Add "lastused" to getbufinfo(). List buffers sorted by last-used field. (Andy Massimino, closes #4722)
Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/windows.txt, src/buffer.c, src/evalbuffer.c, src/ex_getln.c, src/misc1.c, src/option.c, src/option.h, src/proto/misc1.pro, src/proto/viminfo.pro, src/testdir/test_bufwintabinfo.vim, src/testdir/test_cmdline.vim, src/testdir/test_excmod.vim, src/undo.c, src/vim.h, src/viminfo.c

Patch 8.1.2226

Problem: Cannot use system copy/paste in non-xterm terminals.
Solution: Instead of setting 'mouse' to "a" set it to "nvi" in defaults.vim.
Files: runtime/defaults.vim, runtime/doc/term.txt, runtime/doc/options.txt

Patch 8.1.2227

Problem: Layout wrong if 'lines' changes while cmdline window is open.
Solution: Do not restore the window layout if 'lines' changed. (closes #5130)
Files: src/window.c, src/testdir/test_cmdline.vim, src/testdir/dumps/Test_cmdwin_restore_1.dump, src/testdir/dumps/Test_cmdwin_restore_2.dump, src/testdir/dumps/Test_cmdwin_restore_3.dump

Patch 8.1.2228

Problem: screenpos() returns wrong values when 'number' is set. (Ben Jackson)
Solution: Compare the column with the window width. (closes #5133)
Files: src/move.c, src/testdir/test_cursor_func.vim

Patch 8.1.2229

Problem: Cannot color number column above/below cursor differently.
Solution: Add LineNrAbove and LineNrBelow. (Shaun Brady, closes #624)
Files: runtime/doc/syntax.txt, runtime/doc/options.txt, src/optiondefs.h, src/drawline.c, src/vim.h, src/testdir/test_number.vim, src/testdir/dumps/Test_relnr_colors_1.dump, src/testdir/dumps/Test_relnr_colors_2.dump, src/testdir/dumps/Test_relnr_colors_3.dump, src/testdir/dumps/Test_relnr_colors_4.dump

Patch 8.1.2230

Problem: MS-Windows: testing external commands can be improved.
Solution: Adjust tests, remove duplicate test. (closes #4928)
Files: src/testdir/test_normal.vim, src/testdir/test_system.vim, src/testdir/test_terminal.vim, src/testdir/test_undo.vim

Patch 8.1.2231

Problem: Not easy to move to the middle of a text line.
Solution: Add the gM command. (Yasuhiro Matsumoto, closes #2070)
Files: runtime/doc/index.txt, runtime/doc/motion.txt, runtime/doc/quickref.txt, runtime/doc/usr_25.txt, src/normal.c, src/testdir/test_normal.vim

Patch 8.1.2232

Problem: MS-Windows: compiler warning for int size.
Solution: Add type cast. (Mike Williams)
Files: src/normal.c

Patch 8.1.2233

Problem: Cannot get the Vim command line arguments.
Solution: Add v:argv. (Dmitri Vereshchagin, closes #1322)
Files: runtime/doc/eval.txt, src/evalvars.c, src/vim.h,
src/proto/evalvars.pro, src/main.c, src/testdir/test_startup.vim

Patch 8.1.2234

Problem: get_short_pathname() fails depending on encoding.
Solution: Use the wide version of the library function. (closes #5129)
Files: src/filepath.c, src/testdir/test_shortpathname.vim

Patch 8.1.2235

Problem: "C" with '**virtualedit**' set does not include multibyte char.
Solution: Include the whole multibyte char. (Nobuhiro Takasaki,
closes #5152)
Files: src/ops.c, src/testdir/test_virtualedit.vim

Patch 8.1.2236

Problem: Ml_get error if pattern matches beyond last line.
Solution: Adjust position if needed. (Christian Brabandt, closes #5139)
Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.1.2237

Problem: Mode() result after using "r" depends on whether CURSOR_SHAPE is
defined. (Christian Brabandt)
Solution: Move the #ifdef to only skip ui_cursor_shape().
Files: src/normal.c

Patch 8.1.2238

Problem: Error in docs tags goes unnoticed.
Solution: Adjust tags build command. (Ken Takata, closes #5158)
Files: Filelist, .travis.yml, runtime/doc/Makefile,
runtime/doc/doctags.vim

Patch 8.1.2239

Problem: CI fails when running tests without building Vim.
Solution: Skip creating doc tags if the execute does not exist.
Files: runtime/doc/Makefile

Patch 8.1.2240

Problem: Popup window width changes when scrolling.
Solution: Also adjust maxwidth when applying minwidth and there is a
scrollbar. Fix off-by-one error. (closes #5162)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_scroll_11.dump,
src/testdir/dumps/Test_popupwin_scroll_12.dump,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump,
src/testdir/dumps/Test_popupwin_previewpopup_5.dump,
src/testdir/dumps/Test_popupwin_previewpopup_7.dump,

src/testdir/dumps/Test_popupwin_previewpopup_8.dump,

Patch 8.1.2241

Problem: Match highlight does not combine with **'wincolor'**.
Solution: Apply **'wincolor'** last on top of any other attribute. (closes #5159)
Files: src/drawline.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_matches.dump
src/testdir/dumps/Test_popupwin_menu_01.dump
src/testdir/dumps/Test_popupwin_menu_02.dump
src/testdir/dumps/Test_popupwin_menu_04.dump

Patch 8.1.2242

Problem: Creating docs tags uses user preferences. (Tony Mechelynck)
Solution: Add "--clean".
Files: runtime/doc/Makefile

Patch 8.1.2243

Problem: Typos in comments.
Solution: Fix the typos. (Dominique Pelle, closes #5160) Also adjust formatting a bit.
Files: src/autocmd.c, src/buffer.c, src/cindent.c, src/crypt.c,
src/diff.c, src/getchar.c, src/globals.h, src/gui_gtk_x11.c,
src/highlight.c, src/insexpand.c, src/macros.h, src/map.c,
src/memline.c, src/message.c, src/option.c, src/os_unix.c,
src/pty.c, src/quickfix.c, src/regexp_nfa.c, src/register.c,
src/spellsuggest.c, src/structs.h, src/textprop.c, src/ui.c,
src/undo.c, src/vim.h, src/viminfo.c

Patch 8.1.2244

Problem: **'wrapscan'** is not used for "gn".
Solution: Only reset **'wrapscan'** for the first search round. (closes #5164)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.1.2245

Problem: Third character of **'listchars'** tab shows in wrong place when **'breakindent'** is set.
Solution: Set c_final to NUL. (Naruhiko Nishino, closes #5165)
Files: src/drawline.c, src/testdir/test_breakindent.vim

Patch 8.1.2246

Problem: Some tests are still in old style.
Solution: Change a few tests to new style. (Yegappan Lakshmanan)
Files: src/testdir/Make_all.mak, src/testdir/test49.ok,
src/testdir/test49.vim, src/testdir/test_trycatch.vim,
src/testdir/test_vimscript.vim

Patch 8.1.2247

Problem: "make vimtags" does not work in runtime/doc.
Solution: Test existence with "which" instead of "test -x". (Ken Takata)
Files: runtime/doc/Makefile

Patch 8.1.2248

Problem: **CTRL-W** dot does not work in a terminal when modifyOtherKeys is enabled.

Solution: Use the modifier when needed. Pass the modifier along with the key to avoid mistakes.
Files: src/terminal.c, src/proto/terminal.pro, src/mouse.c

Patch 8.1.2249

Problem: "make vimtags" does not print any message.
Solution: Add a message that the tags have been updated.
Files: runtime/doc/Makefile

Patch 8.1.2250

Problem: **CTRL-U** and **CTRL-D** don't work in popup window.
Solution: Initialize '**scroll**'. Add "lastline" in popup_getpos(). (closes #5170)
Files: src/popupwin.c, src/testdir/test_popupwin.vim, runtime/doc/popup.txt

Patch 8.1.2251

Problem: ":term command" may not work without a shell.
Solution: Add the ++shell option to :term. (closes #3340)
Files: runtime/doc/terminal.txt, src/terminal.c, src/os_unix.c, src/proto/os_unix.pro, src/testdir/test_terminal.vim

Patch 8.1.2252

Problem: Compiler warning for int size.
Solution: Add type cast. (Mike Williams)
Files: src/filepath.c

Patch 8.1.2253

Problem: Using "which" to check for an executable is not reliable.
Solution: Use "command -v" instead. Also exit with error code when generating tags has an error. (closes #5174)
Files: runtime/doc/Makefile

Patch 8.1.2254

Problem: MS-Windows: mouse scroll wheel doesn't work in popup.
Solution: Handle mouse wheel events separately. (closes #5138)
Files: src/gui_w32.c, src/gui.c, src/proto/gui.pro

Patch 8.1.2255

Problem: ":term ++shell" does not work on MS-Windows.
Solution: Add MS-Windows support.
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.1.2256 (after 8.1.2255)

Problem: Test for ":term ++shell" fails on MS-Windows.
Solution: Accept failure of "dir" executable.
Files: src/testdir/test_terminal.vim

Patch 8.1.2257

Problem: MS-Windows GUI: scroll wheel always uses current window.
Solution: Add the '**scrollfocus**' option for MS-Windows.
Files: runtime/doc/options.txt, src/gui_w32.c, src/optiondefs.h, src/option.h

Patch 8.1.2258

Problem: May get hit-enter prompt after entering a number. (Malcolm Rowe)
Solution: Put back accidentally deleted lines. (closes #5176)
Files: src/misc1.c

Patch 8.1.2259

Problem: Running tests may leave XfakeHOME behind.
Solution: Source summarize.vim without using setup.vim. (closes #5177)
Also fix that on MS-Windows the test log isn't echoed.
Files: src/testdir/Makefile, src/testdir/Make_dos.mak

Patch 8.1.2260

Problem: Terminal test may fail on MS-Windows.
Solution: Catch the situation that "term dir" fails with a CreateProcess error.
Files: src/testdir/test_terminal.vim

Patch 8.1.2261

Problem: With modifyOtherKeys set '**noesckey**s' doesn't work. (James McCoy)
Solution: Disable modifyOtherKeys while in Insert mode when '**noesckey**s' is set. (closes #5180)
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.1.2262

Problem: Unpack assignment in function not recognized.
Solution: Skip over "[a, b]". (closes #5051)
Files: src/userfunc.c, src/testdir/test_let.vim

Patch 8.1.2263

Problem: '**noesckey**s' test fails in GUI.
Solution: Skip the test in the GUI.
Files: src/testdir/test_edit.vim

Patch 8.1.2264

Problem: There are two test files for :let.
Solution: Merge the two files.
Files: src/testdir/test_assign.vim, src/testdir/test_let.vim,
src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 8.1.2265

Problem: When popup with "botleft" does not fit it flips incorrectly.
Solution: Only flip when there is more space on the other side. Add the "posinvert" option to disable flipping and do it in both directions if enabled. (closes #5151)
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/vim.h,
src/testdir/dumps/Test_popupwin_nospace.dump

Patch 8.1.2266

Problem: Position unknown for a mouse click in a popup window.
Solution: Set v:mouse_col and v:mouse_lnum. (closes #5171)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.2267

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Rearrange the code.
Files: src/buffer.c

Patch 8.1.2268

Problem: Spell file flag zero is not recognized.
Solution: Use -1 as an error value, so that zero can be used as a valid flag number.
Files: src/spellfile.c, src/testdir/test_spell.vim

Patch 8.1.2269

Problem: Tags file with very long line stops using binary search.
Solution: Reallocate the buffer if needed.
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.1.2270

Problem: "gf" is not tested in Visual mode.
Solution: Add Visual mode test and test errors. (Dominique Pelle, closes #5197)
Files: src/testdir/test_gf.vim

Patch 8.1.2271

Problem: Build error if FEAT_TAG_BINS is not defined. (John Marriott)
Solution: Add #ifdef.
Files: src/tag.c

Patch 8.1.2272

Problem: Test may hang at more prompt.
Solution: Reset 'more' after resetting 'compatible'. (Michael Soyka)
Files: src/testdir/test_vimscript.vim

Patch 8.1.2273

Problem: Wrong default when "pos" is changed with popup_atcursor().
Solution: Adjust the default line and col when "pos" is not the default value. (#5151)
Files: runtime/doc/popup.txt, src/structs.h, src/popupwin.c, src/proto/popupwin.pro, src/ex_cmds.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_atcursor_pos.dump

Patch 8.1.2274

Problem: Newlines in 'balloonexpr' result only work in the GUI.
Solution: Also recognize newlines in the terminal. (closes #5193)
Files: src/popupmenu.c, src/testdir/test_balloon.vim, src/testdir/dumps/Test_balloon_eval_term_01.dump, src/testdir/dumps/Test_balloon_eval_term_01a.dump, src/testdir/dumps/Test_balloon_eval_term_02.dump

Patch 8.1.2275

Problem: Using "seesion" looks like a mistake.
Solution: Use an underscore to make the function sort first.
Files: src/testdir/test_mksession.vim

Patch 8.1.2276

Problem: MS-Windows: session test leaves files behind.
Solution: Wipe out buffers before deleting the directory. (closes #5187)
Files: src/testdir/test_mksession.vim

Patch 8.1.2277

Problem: Terminal window is not updated when info popup changes.
Solution: Redraw windows when re-using an info popup. (closes #5192)
Files: src/ex_cmds.c

Patch 8.1.2278

Problem: Using "cd" with "exe" may fail.
Solution: Use chdir() instead.
Files: src/testdir/test_autochdir.vim, src/testdir/test_autocmd.vim,
src/testdir/test_cd.vim, src/testdir/test_expand.vim,
src/testdir/test_find_complete.vim, src/testdir/test_findfile.vim,
src/testdir/test_getcwd.vim, src/testdir/test_shortpathname.vim

Patch 8.1.2279

Problem: Computation of highlight attributes is too complicated.
Solution: Simplify the attribute computation and make it more consistent.
(closes #5190) Fix that 'combine' set to zero doesn't work.
Files: src/drawline.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.1.2280

Problem: Crash when passing partial to substitute().
Solution: Take extra arguments into account. (closes #5186)
Files: src/userfunc.c, src/structs.h, src/regexp.c, src/proto/regexp.pro,
src/testdir/test_substitute.vim

Patch 8.1.2281

Problem: 'showbreak' cannot be set for one window.
Solution: Make 'showbreak' global-local.
Files: src/optiondefs.h, src/option.c, src/option.h,
src/proto/option.pro, src/structs.h, src/charset.c,
src/drawline.c, src/edit.c, src/move.c, src/normal.c, src/ops.c,
src/optionstr.c, src/testdir/test_highlight.vim,
src/testdir/test_breakindent.vim, runtime/doc/options.txt

Patch 8.1.2282

Problem: Crash when passing many arguments through a partial. (Andy Massimino)
Solution: Check the number of arguments. (closes #5186)
Files: src/userfunc.c, src/proto/userfunc.pro, src/evalfunc.c,
src/regexp.c, src/testdir/test_expr.vim,
src/testdir/test_substitute.vim

Patch 8.1.2283

Problem: Missed one use of p_sbr.
Solution: Add missing p_sbr change.
Files: src/indent.c

Patch 8.1.2284

Problem: Compiler warning for unused variable. (Tony Mechelynck)

Solution: Add #ifdef.
Files: src/move.c

Patch 8.1.2285

Problem: Padding in structures wastes memory.
Solution: Move fields to avoid padding. (Dominique Pelle, closes #5202)
Files: src/structs.h

Patch 8.1.2286

Problem: Using border highlight in popup window leaks memory.
Solution: Free memory before overwriting. (Dominique Pelle, closes #5203)
Files: src/popupwin.c

Patch 8.1.2287

Problem: Using EndOfBuffer highlight in popup does not look good.
Solution: Do not EndOfBuffer highlight. (closes #5204)
Files: src/drawscreen.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_02.dump,
src/testdir/dumps/Test_popupwin_04.dump,
src/testdir/dumps/Test_popupwin_04a.dump,
src/testdir/dumps/Test_popupwin_05.dump,
src/testdir/dumps/Test_popupwin_06.dump,
src/testdir/dumps/Test_popupwin_07.dump,
src/testdir/dumps/Test_popupwin_08.dump

Patch 8.1.2288

Problem: Not using all space when popup with "topleft" flips to above.
Solution: Recompute the height when a popup flips from below to above.
(closes #5151)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_nospace.dump

Patch 8.1.2289

Problem: After :diffsplit closing the window does not disable diff.
Solution: Add "closeoff" to 'diffopt' and add it to the default.
Files: runtime/doc/options.txt, src/optiondefs.h, src/diff.c,
src/proto/diff.pro, src/window.c, src/testdir/test_diffmode.vim

Patch 8.1.2290

Problem: Autocommand test fails.
Solution: Remove 'closeoff' from 'diffopt'.
Files: src/testdir/test_autocmd.vim

Patch 8.1.2291

Problem: Memory leak when executing command in a terminal.
Solution: Free "argv". (Dominique Pelle, closes #5208)
Files: src/terminal.c

Patch 8.1.2292

Problem: v:mouse_winid not set on click in popup window.
Solution: Set v:mouse_winid. (closes #5171)
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.1.2293

Problem: Join adds trailing space when second line is empty. (Brennan Vincent)
Solution: Do not add a trailing space.
Files: src/ops.c, src/testdir/test_join.vim

Patch 8.1.2294

Problem: Cursor position wrong when characters are concealed and a search causes a scroll.
Solution: Fix the cursor column in a concealed line after window scroll. (closes #5215, closes #5012)
Files: src/drawscreen.c, src/testdir/test_matchadd_conceal.vim

Patch 8.1.2295

Problem: If buffer of popup is in another window cursorline sign shows.
Solution: Check the group of the sign.
Files: src/option.c, src/proto/option.pro, src/sign.c, src/proto/sign.pro, src/screen.c, src/drawline.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_cursorline_8.dump

Patch 8.1.2296

Problem: Text properties are not combined with syntax by default.
Solution: Make it work as documented. (closes #5190)
Files: src/testprop.c, src/testdir/test_textprop.vim

Patch 8.1.2297

Problem: The ex_vimgrep() function is too long.
Solution: Split it in three parts. (Yegappan Lakshmanan, closes #5211)
Files: src/quickfix.c

Patch 8.1.2298 (after 8.1.2296)

Problem: Missing part of 8.1.2296.
Solution: s/test/text/
Files: src/textprop.c

Patch 8.1.2299

Problem: ConPTY in MS-Windows 1909 is still wrong.
Solution: Use same solution as for 1903. (Nobuhiro Takasaki, closes #5217)
Files: src/misc2.c, src/os_win32.c

Patch 8.1.2300

Problem: Redraw breaks going through list of popup windows.
Solution: Use different flags for popup_reset_handled(). (closes #5216)
Files: src/popupwin.c, src/proto/popupwin.pro, src/structs.h, src/vim.h, src/mouse.c, src/testdir/test_popupwin.vim

Patch 8.1.2301

Problem: MS-Windows GUI: drawing error when background color changes.
Solution: Implement gui_mch_new_colors(). (Simon Sadler)
Files: src/gui_w32.c

Patch 8.1.2302

Problem: :lockmarks does not work for '[' and '].

Solution: save and restore '[' and ']' marks. (James McCoy, closes #5222)
Files: runtime/doc/motion.txt, src/bufwrite.c, src/diff.c, src/ex_cmds.c,
src/fileio.c, src/indent.c, src/ops.c, src/register.c,
src/testdir/test_autocmd.vim, src/testdir/test_diffmode.vim

Patch 8.1.2303

Problem: Cursor in wrong position after horizontal scroll.
Solution: Set w_valid_leftcol. (closes #5214, closes #5224)
Files: src/move.c, src/testdir/test_matchadd_conceal.vim

Patch 8.1.2304

Problem: Cannot get the mouse position when getting a mouse click.
Solution: Add getmousepos().
Files: runtime/doc/eval.txt, runtime/doc/popup.txt, src/mouse.c
src/proto/mouse.pro, src/evalfunc.c, src/popupwin.c,
src/popupwin.pro, src/testdir/test_popupwin.vim,
src/testdir/test_functions.vim

Patch 8.1.2305

Problem: No warning for wrong entry in translations.
Solution: Check semicolons in keywords entry of desktop file.
Files: src/po/check.vim

Patch 8.1.2306

Problem: Double and triple clicks are not tested.
Solution: Test mouse clicks to select text. (closes #5226)
Files: src/testdir/test_termcodes.vim

Patch 8.1.2307

Problem: Positioning popup doesn't work for buffer-local textprop.
Solution: Make it work. (closes #5225)
Files: src/popupwin.c, src/testdir/test_popupwin_textprop.vim

Patch 8.1.2308

Problem: Deleting text before zero-width textprop removes it.
Solution: Keep zero-width textprop when deleting text.
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.1.2309

Problem: Compiler warning for argument type.
Solution: Use linenr_T and cast to varnumber_T. (John Marriott)
Files: src/mouse.c

Patch 8.1.2310

Problem: No proper test for directory changes in quickfix.
Solution: Add a test that uses multiple directories. (Yegappan Lakshmanan,
closes #5230)
Files: src/testdir/test_quickfix.vim

Patch 8.1.2311

Problem: Warning for missing function prototype.
Solution: Add the proto. (Dominique Pelle, closes #5233)
Files: src/proto/popupwin.pro

Patch 8.1.2312

Problem: "line:" field in tags file not used.
Solution: Recognize the field and use the value. (Andy Massimino, Daniel Hahler, closes #5232, closes #2546, closes #1057)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.1.2313

Problem: Debugging where a delay comes from is not easy.
Solution: Use different values when calling ui_delay().
Files: src/buffer.c, src/change.c, src/fileio.c, src/gui.c,
src/if_xcmdsrv.c, src/insexpand.c, src/main.c, src/normal.c,
src/screen.c, src/search.c, src/tag.c, src/term.c, src/ui.c

Patch 8.1.2314

Problem: vi' sometimes does not select anything.
Solution: Recognize an empty selection. (Christian Brabandt, closes #5183)
Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.1.2315

Problem: Not always using the right window when jumping to an error.
Solution: Add the "uselast" flag in '**switchbuf**'. (closes #1652)
Files: runtime/doc/options.txt, src/option.h, src/optionstr.c,
src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.2316

Problem: FORTIFY_SOURCE can also be present in CPPFLAGS.
Solution: Remove it in configure. (Benedikt Morbach, closes #2786)
Files: src/configure.ac, src/auto/configure

Patch 8.1.2317

Problem: No test for spell affix file with flag on suffix.
Solution: Add a test case.
Files: src/testdir/test_spell.vim

Patch 8.1.2318 (after 8.1.2301)

Problem: MS-Windows GUI: main background shows in toolbar.
Solution: Remove transparency from the toolbar. (Simon Sadler)
Files: src/gui_w32.c

Patch 8.1.2319

Problem: Compiler warning for int size.
Solution: Add typecast. (Mike Williams)
Files: src/mouse.c

Patch 8.1.2320

Problem: Insufficient test coverage for quickfix.
Solution: Add more tests. Fix uncovered problem. (Yegappan Lakshmanan, closes #5238)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.2321

Problem: Cannot select all text with the mouse. (John Marriott)
Solution: Move limiting the mouse column to f_getmousepos(). (closes #5242)
Files: src/mouse.c

Patch 8.1.2322 (after 8.1.2320)

Problem: Quickfix test fails in very big terminal.

Solution: Adjust the expected result for the width. (Masato Nishihata, closes #5244)

Files: src/testdir/test_quickfix.vim

Patch 8.1.2323

Problem: Old MSVC version no longer tested.

Solution: Drop support for MSVC 2008 and older. (Ken Takata, closes #5248)

Files: src/INSTALLpc.txt, src/Make_mvc.mak, src/gui_w32.c, src/os_win32.c

Patch 8.1.2324

Problem: Width of scrollbar in popup menu not taken into account.

Solution: Add the width of the scrollbar.

Files: src/popupmenu.c, src/testdir/dumps/Test_popupwin_infopopup_6.dump, src/testdir/test_popupwin.vim

Patch 8.1.2325

Problem: Crash when using balloon with empty line.

Solution: Handle empty lines. (Markus Braun)

Files: src/popupmenu.c, src/testdir/test_popup.vim

Patch 8.1.2326

Problem: Cannot parse a date/time string.

Solution: Add strtptime(). (Stephen Wall, closes #5250)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/auto/configure, src/config.h.in, src/configure.ac, src/evalfunc.c, src/os_unix.h, src/testdir/test_functions.vim

Patch 8.1.2327

Problem: Cannot build with Hangul input.

Solution: Remove Hangul input support.

Files: Filelist, src/Makefile, runtime/doc/hangulin.txt, src/feature.h, src/gui_gtk_x11.c, src/gui_x11.c, src/gui.c, src/edit.c, src/mbyte.c, src/screen.c, src/ui.c, src/hangulin.c, src/globals.h, src/proto/hangulin.pro, src/proto.h, src/evalfunc.c, src/version.c, src/configure.ac, src/auto/configure, src/config.h.in, src/config.mk.in

Patch 8.1.2328

Problem: A few hangul input pieces remain.

Solution: Update VMS makefile.

Files: src/Make_vms.mms

Patch 8.1.2329

Problem: Mouse multiple click test is a bit flaky.

Solution: Add it to the list of flaky tests.

Files: src/testdir/runtest.vim

Patch 8.1.2330 (after 8.1.2314)

Problem: vi' does not always work when **'selection'** is exclusive.

Solution: Adjust start position.

Files: src/search.c, src/testdir/test_textobjects.vim

Patch 8.1.2331

Problem: The option.c file is still very big.

Solution: Move a few functions to where they fit better. (Yegappan Lakshmanan, closes #4895)

Files: src/option.c, src/proto/option.pro, src/change.c,
src/proto/change.pro, src/ex_getln.c, src/proto/ex_getln.pro,
src/globals.h, src/gui.c, src/proto/gui.pro, src/ui.c,
src/proto/ui.pro, src/term.c, src/proto/term.pro, src/indent.c,
src/proto/indent.pro

Patch 8.1.2332 (after 8.1.2331)

Problem: Missing file in refactoring.

Solution: Update missing file.

Files: src/search.c

Patch 8.1.2333

Problem: With modifyOtherKeys **CTRL-^** doesn't work.

Solution: Handle the exception.

Files: src/getchar.c, src/testdir/test_termcodes.vim

Patch 8.1.2334

Problem: Possible NULL pointer dereference in popup_locate(). (Coverity)

Solution: Check for NULL pointer.

Files: src/popupwin.c

Patch 8.1.2335

Problem: Error message for function arguments may use NULL pointer.
(Coverity)

Solution: Use the original function name.

Files: src/evalfunc.c

Patch 8.1.2336

Problem: When an expr mapping moves the cursor it is not restored.

Solution: Position the cursor after an expr mapping. (closes #5256)

Files: src/getchar.c, src/testdir/test_mapping.vim,
src/testdir/dumps/Test_map_expr_1.dump

Patch 8.1.2337

Problem: Double-click time sometimes miscomputed.

Solution: Correct time computation. (Dominique Pelle, closes #5259)

Files: src/mouse.c, src/testdir/runtest.vim

Patch 8.1.2338

Problem: Using Visual mark with :s gives E20 if not set.

Solution: Ignore errors when handling '**incsearch**'. (closes #3837)

Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_substitute_14.dump

Patch 8.1.2339

Problem: Insufficient testing for quickfix.

Solution: Add a few more tests. (Yegappan Lakshmanan, closes #5261)

Files: src/testdir/test_quickfix.vim

Patch 8.1.2340

Problem: Quickfix test fails under valgrind and asan.
Solution: Make sure long line does not overflow IObuff. (Dominique Pelle, closes #5263) Put back fix for large terminals. (Yegappan Lakshmanan, closes #5264)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.1.2341

Problem: Not so easy to interrupt a script programmatically.
Solution: Add the interrupt() function. (Yasuhiro Matsumoto, closes #2834)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/ex_eval.c, src/testdir/Make_all.mak, src/testdir/test_interrupt.vim

Patch 8.1.2342

Problem: Random number generator in Vim script is slow.
Solution: Add rand() and srand(). (Yasuhiro Matsumoto, closes #1277)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/Make_all.mak, src/testdir/test_random.vim

Patch 8.1.2343

Problem: Using time() for srand() is not very random.
Solution: use /dev/urandom if available
Files: src/evalfunc.c, src/testdir/test_random.vim

Patch 8.1.2344

Problem: Cygwin: warning for using strtptime().
Solution: Move defining _XOPEN_SOURCE and __USE_XOPEN to vim.h. (Ken Takata, closes #5265) Use 700 for _XOPEN_SOURCE for mkdtemp().
Files: src/os_unix.h, src/vim.h

Patch 8.1.2345

Problem: .cjs files are not recognized as Javascript.
Solution: Add the *.cjs pattern. (closes #5268)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.1.2346

Problem: **CTRL-R CTRL-R** doesn't work with modifyOtherKeys.
Solution: Allow key codes when fetching argument for **CTRL-R**. (closes #5266) Also fix **CTRL-G** in Insert mode.
Files: src/edit.c, src/ex_getln.c, src/testdir/test_termcodes.vim

Patch 8.1.2347 (after 8.1.2344)

Problem: macOS: build fails.
Solution: Don't define _XOPEN_SOURCE for Mac.
Files: src/vim.h

Patch 8.1.2348

Problem: :const cannot be followed by "| endif".
Solution: Check following command for :const. (closes #5269) Also fix completion after :const.
Files: src/testdir/test_let.vim, src/testdir/test_const.vim, src/ex_docmd.c, src/cmdexpand.c, src/eval.c, src/testdir/test_cmdline.vim

Patch 8.1.2349

Problem: :lockvar and :unlockvar cannot be followed by "| endif".
Solution: Check for following commands. (closes #5269)
Files: src/testdir/test_const.vim, src/ex_docmd.c

Patch 8.1.2350

Problem: Other text for **CTRL-V** in Insert mode with modifyOtherKeys.
Solution: Convert the Escape sequence back to key as if modifyOtherKeys is not set, and use **CTRL-SHIFT-V** to get the Escape sequence itself. (closes #5254)
Files: runtime/doc/insert.txt, runtime/doc/cmdline.txt, src/edit.c, src/proto/edit.pro, src/term.c, src/proto/term.pro, src/getchar.c, src/proto/getchar.pro, src/testdir/test_termcodes.vim, src/ex_getln.c

Patch 8.1.2351

Problem: **'wincolor'** not used for > for not fitting double width char. Also: popup drawn on right half of double width character looks wrong.
Solution: Adjust color for > character. Clear left half of double width character if right half is being overwritten.
Files: src/drawline.c, src/screen.c, src/testdir/dumps/Test_popupwin_doublewidth_1.dump

Patch 8.1.2352

Problem: CI doesn't cover FreeBSD.
Solution: Configure Cirrus-CI. (Christian Brabandt, closes #5273)
Files: .cirrus.yml, README.md

Patch 8.1.2353

Problem: Build failure on FreeBSD.
Solution: Change #ifdef to only check for Linux-like systems.
Files: src/vim.h

Patch 8.1.2354

Problem: Cirrus CI runs on another repository.
Solution: Run Cirrus CI on vim/vim.
Files: .cirrus.yml, README.md

Patch 8.1.2355

Problem: Test with "man" fails on FreeBSD.
Solution: Use "-P" instead of "--pager".
Files: src/testdir/test_normal.vim

Patch 8.1.2356

Problem: rand() does not use the best algorithm.
Solution: use xoshiro128** instead of xorshift. (Kaito Udagawa, closes #5279)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_random.vim

Patch 8.1.2357

Problem: No test with wrong argument for rand().
Solution: Add a test case.
Files: src/testdir/test_random.vim

Patch 8.1.2358

Problem: Tests fail on Cirrus CI for FreeBSD.
Solution: Fix a test and skip some. (Christian Brabandt, closes #5281)
Files: Filelist, .cirrus.yml, src/testdir/check.vim,
src/testdir/test_normal.vim, src/testdir/test_quickfix.vim,
src/testdir/test_source_utf8.vim, src/testdir/test_terminal.vim,
src/testdir/test_utf8_comparisons.vim

Patch 8.1.2359

Problem: Cannot build without FEAT_FLOAT. (John Marriott)
Solution: Fix #ifdefs around f_srand().
Files: src/evalfunc.c

Patch 8.1.2360

Problem: Quickfix test coverage can still be improved.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5276)
Files: src/testdir/test_quickfix.vim

Patch 8.1.2361

Problem: MS-Windows: test failures related to VIMDLL.
Solution: Adjust code and tests. (Ken Takata, closes #5283)
Files: src/evalfunc.c, src/ex_cmds.c, src/gui_w32.c, src/mbyte.c,
src/menu.c, src/proto.h, src/testdir/test_highlight.vim

Patch 8.1.2362

Problem: Cannot place signs in a popup window. (Maxim Kim)
Solution: Use the group prefix "PopUp" to specify which signs should show up
in a popup window. (closes #5277)
Files: runtime/doc/sign.txt, src/popupwin.c, src/sign.c,
src/testdir/dumps/Test_popupwin_sign_1.dump

Patch 8.1.2363

Problem: ml_get error when accessing Visual area in **'statusline'**.
Solution: Disable Visual mode when using another window. (closes #5278)
Files: src/testdir/test_statusline.vim, src/buffer.c

Patch 8.1.2364

Problem: Termwincscroll test is flaky on FreeBSD.
Solution: Add to list of flaky tests. Rename function.
Files: src/testdir/runtest.vim, src/testdir/test_terminal.vim

Patch 8.1.2365

Problem: Missing tests for recent popupwin changes.
Solution: Add test cases.
Files: src/testdir/test_popupwin.vim

Patch 8.1.2366

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/ascii.h, src/beval.h, src/dosinst.h, src/feature.h,
src/glbl_ime.h, src/globals.h, src/gui_at_sb.h, src/gui_gtk_f.h,
src/gui_gtk_vms.h, src/gui.h, src/gui_x11_pm.h, src/gui_xmewbp.h,
src/if_cscope.h, src/if_mzsch.h, src/if_ole.h, src/if_py_both.h,

src/iscygpity.h, src/keymap.h, src/macros.h, src/nbdebug.h,
src/option.h, src/os_amiga.h, src/os_beos.h, src/os_dos.h,
src/os_mac.h, src/os_qnx.h, src/os_unix.h, src/os_unixx.h,
src/os_vms_conf.h, src/os_win32.h, src/proto.h, src/regexp.h,
src/spell.h, src/structs.h, src/term.h, src/version.h, src/vimio.h

Patch 8.1.2367

Problem: Registers are not sufficiently tested.
Solution: Add a few more test cases. (Yegappan Lakshmanan, closes #5288)
Files: src/testdir/test_registers.vim

Patch 8.1.2368

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/autocmd.c, src/beval.c, src/blob.c, src/blowfish.c,
src/buffer.c, src/change.c, src/channel.c, src/charset.c,
src/cindent.c, src/encrypt.c, src/encrypt_zip.c

Patch 8.1.2369

Problem: Cannot build with quickfix and without text properties.
Solution: Fix typo. (Naruhiko Nishino)
Files: src/popupmenu.c

Patch 8.1.2370

Problem: Build problems on VMS.
Solution: Adjust the build file. (Zoltan Arpadffy)
Files: src/Make_vms.mms, src/os_unix.c, src/os_vms.c

Patch 8.1.2371

Problem: FEAT_TEXT_PROP is a confusing name.
Solution: Use FEAT_PROP_POPUP. (Naruhiko Nishino, closes #5291)
Files: runtime/doc/popup.txt, src/beval.c, src/buffer.c, src/change.c,
src/drawline.c, src/drawscreen.c, src/edit.c, src/eval.c,
src/evalbuffer.c, src/evalfunc.c, src/evalwindow.c, src/ex_cmds.c,
src/ex_docmd.c, src/feature.h, src/fileio.c, src/getchar.c,
src/globals.h, src/gui.c, src/gui_w32.c, src/indent.c,
src/insexpand.c, src/macros.h, src/main.c, src/memline.c,
src/misc2.c, src/mouse.c, src/move.c, src/ops.c, src/option.h,
src/optiondefs.h, src/optionstr.c, src/popupmenu.c,
src/popupwin.c, src/proto.h, src/screen.c, src/search.c,
src/sign.c, src/structs.h, src/tag.c, src/testdir/runtest.vim,
src/testdir/test_execute_func.vim, src/testdir/test_popupwin.vim,
src/testdir/test_popupwin_textprop.vim, src/textprop.c, src/ui.c,
src/version.c, src/vim.h, src/window.c

Patch 8.1.2372

Problem: VMS: failing realloc leaks memory. (Chakshu Gupta)
Solution: Free the memory. (partly fixes #5292)
Files: src/os_vms.c

Patch 8.1.2373

Problem: Cannot build with +popupwin but without +quickfix. (John Marriott)
Solution: Adjust #ifdefs.
Files: src/ex_cmds.c, src/popupmenu.c, src/popupwin.c, src/fileio.c,

src/testdir/test_compiler.vim, src/testdir/test_tagjump.vim,
src/testdir/test86.in, src/testdir/test87.in,
src/testdir/test_autocmd.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_channel.vim, src/testdir/test_edit.vim,
src/testdir/test_execute_func.vim,
src/testdir/test_filter_cmd.vim, src/testdir/test_gui.vim,
src/testdir/test_makeencoding.vim, src/testdir/test_mksession.vim,
src/testdir/test_normal.vim, src/testdir/test_popup.vim,
src/testdir/test_popupwin.vim, src/testdir/test_preview.vim,
src/testdir/test_startup.vim, src/testdir/test_statusline.vim,
src/testdir/test_tabpage.vim, src/testdir/test_window_cmd.vim,
src/testdir/test_window_id.vim

Patch 8.1.2374

Problem: Unused parts of libvterm are included.
Solution: Delete the unused files.
Files: Filelist, src/libvterm/bin/vterm-ctrl.c,
src/libvterm/bin/unterm.c, src/libvterm/bin/vterm-dump.c

Patch 8.1.2375

Problem: No sufficient testing for registers.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5296)
Fix that "p" on last virtual column of tab inserts spaces.
Files: src/register.c, src/testdir/test_registers.vim,
src/testdir/test_virtualedit.vim, src/testdir/test_visual.vim

Patch 8.1.2376

Problem: Preprocessor indents are incorrect.
Solution: Fix the indents. (Ken Takata, closes #5298)
Files: src/drawline.c, src/gui_w32.c, src/os_mswin.c, src/os_win32.c,
src/proto.h

Patch 8.1.2377

Problem: GUI: when losing focus a pending operator is executed.
Solution: Do not execute an operator when getting K_IGNORE. (closes #5300)
Files: src/normal.c

Patch 8.1.2378

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/dict.c, src/diff.c, src/digraph.c, src/dosinst.c, src/edit.c,
src/eval.c, src/evalbuffer.c, src/evalfunc.c

Patch 8.1.2379

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/ex_getln.c, src/fileio.c, src/filepath.c, src/findfile.c,
src/fold.c

Patch 8.1.2380

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/getchar.c, src/gui.c, src/gui_at_fs.c, src/gui_at_sb.c,

src/gui_athena.c, src/gui_beval.c, src/gui_gtk.c, src/gui_gtk_f.c,
src/gui_gtk_x11.c

Patch 8.1.2381

Problem: Not all register related code is covered by tests.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5301)
Files: src/testdir/test_marks.vim, src/testdir/test_registers.vim,
src/testdir/test_virtualedit.vim

Patch 8.1.2382

Problem: MS-Windows: When using VTP bold+inverse doesn't work.
Solution: Compare with the default colors. (Nobuhiro Takasaki, closes #5303)
Files: src/os_win32.c, src/proto/os_win32.pro, src/screen.c

Patch 8.1.2383

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_w32.c,
src/gui_x11.c, src/gui_xmdlg.c, src/gui_xmebw.c

Patch 8.1.2384

Problem: Test 48 is old style.
Solution: Merge test cases into new style test. (Yegappan Lakshmanan,
closes #5307)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test48.in, src/testdir/test48.ok,
src/testdir/test_virtualedit.vim

Patch 8.1.2385

Problem: Opening cmdline window with feedkeys() does not work. (Yegappan
Lakshmanan)
Solution: Recognize K_CMDWIN also when ex_normal_busy is set.
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.1.2386

Problem: **'wincolor'** is not used for **'listchars'**.
Solution: Combine the colors. (closes #5308)
Files: src/drawline.c, src/testdir/test_highlight.vim,
src/testdir/dumps/Test_wincolor_lcs.dump

Patch 8.1.2387

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/hardcopy.c, src/hashtab.c, src/if_cscope.c, src/if_lua.c,
src/if_mzsch.c, src/if_perlsfio.c, src/if_python.c,
src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/if_xcmdsrv.c

Patch 8.1.2388

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/json.c, src/json_test.c, src/kword_test.c, src/list.c,
src/main.c, src/mark.c, src/mbyte.c, src/memfile.c,
src/memfile_test.c, src/memline.c, src/menu.c

Patch 8.1.2389

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/libvterm/src/screen.c, src/libvterm/src/unicode.c,
src/libvterm/src/vterm.c, src/libvterm/t/harness.c,
src/libvterm/include/vterm.h, src/xdiff/xdiffi.c,
src/xdiff/xemit.c, src/xdiff/xhistogram.c, src/xdiff/xpatience.c,
src/xdiff/xutils.c, src/xdiff/xdiff.h, src/xdiff/xdiffi.h,
src/xdiff/xemit.h, src/xdiff/xinclude.h, src/xdiff/xmacros.h,
src/xdiff/xprepare.h, src/xdiff/xtypes.h, src/xdiff/xutils.h

Patch 8.1.2390

Problem: Test94 is old style, fix 7.4.441 not tested.
Solution: Turn test94 into a new style test. Add tests for the fix in patch
7.4.441. (Yegappan Lakshmanan, closes #5316)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test94.in, src/testdir/test94.ok,
src/testdir/test_cmdline.vim, src/testdir/test_visual.vim

Patch 8.1.2391

Problem: Cannot build when `__QNXNTO__` is defined. (Ian Wayne Larson)
Solution: Move the check for "qansi". (Ken Takata, closes #5317)
Files: src/highlight.c

Patch 8.1.2392

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/nbdebug.c, src/netbeans.c, src/normal.c, src/ops.c,
src/option.c

Patch 8.1.2393

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/os_amiga.c, src/os_beos.c, src/os_mac_conv.c, src/os_mswin.c,
src/os_qnx.c, src/os_unix.c, src/os_vms.c, src/os_win32.c

Patch 8.1.2394

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/popupmenu.c, src/pty.c, src/quickfix.c, src/regexp.c,
src/regexp_nfa.c, src/screen.c, src/search.c, src/sha256.c,
src/sign.c

Patch 8.1.2395

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/spell.c, src/spellfile.c, src/syntax.c, src/tag.c, src/term.c,
src/terminal.c, src/termlib.c, src/testing.c

Patch 8.1.2396

Problem: Using old C style comments.
Solution: Use // comments where appropriate.
Files: src/ui.c, src/undo.c, src/uninstall.c, src/usercmd.c,
src/userfunc.c, src/winclip.c, src/window.c, src/xpm_w32.c

Patch 8.1.2397

Problem: Should not define `__USE_XOPEN`. `_XOPEN_SOURCE` is not needed for Android.

Solution: Remove `__USE_XOPEN` and adjust `#ifdefs`. (Ozaki Kiichi, closes #5322)

Files: `src/vim.h`

Patch 8.1.2398

Problem: `strptime()` test fails on Japanese Mac.

Solution: Use `%T` instead of `%X`.

Files: `src/testdir/test_functions.vim`

Patch 8.1.2399

Problem: Info popup on top of cursor if it doesn't fit.

Solution: Hide the popup if it doesn't fit.

Files: `src/popupmenu.c`, `src/testdir/test_popupwin.vim`,
`src/testdir/dumps/Test_popupwin_infopopup_wide_1.dump`

Patch 8.1.2400

Problem: Test39 is old style.

Solution: Convert the test cases into new style. (Yegappan Lakshmanan, closes #5324)

Files: `src/Makefile`, `src/testdir/Make_all.mak`, `src/testdir/Make_vms.mms`,
`src/testdir/test39.in`, `src/testdir/test39.ok`,
`src/testdir/test_blockedit.vim`, `src/testdir/test_visual.vim`

Patch 8.1.2401

Problem: `:cexpr` does not handle `|` in expression.

Solution: Remove `EX_TRLBAR` and set `nextcmd` pointer.

Files: `src/testdir/test_quickfix.vim`, `src/ex_cmds.h`, `src/quickfix.c`

Patch 8.1.2402

Problem: Typos and other small things.

Solution: Small fixes.

Files: `.gitignore`, `src/testdir/shared.vim`, `src/testdir/test49.vim`,
`src/message.c`, `src/Makefile`

Patch 8.1.2403

Problem: `Autocmd` test fails under `valgrind`.

Solution: Wait a bit longer.

Files: `src/testdir/test_autocmd.vim`

Patch 8.1.2404

Problem: `Channel` test fails under `valgrind`.

Solution: Sleep a bit longer.

Files: `src/testdir/test_channel.vim`

Patch 8.1.2405

Problem: `matchadd_conceal` test fails under `valgrind`.

Solution: Use `WaitForAssert()` and wait a bit longer.

Files: `src/testdir/test_matchadd_conceal.vim`

Patch 8.1.2406

Problem: Leaking memory in test_paste and test_registers.
Solution: Free the old title. Don't copy expr_line.
Files: src/term.c, src/os_unix.c, src/register.c

Patch 8.1.2407

Problem: proto file and dependencies outdated.
Solution: Update proto files and dependencies.
Files: src/Makefile, src/proto/bufwrite.pro, src/proto/cmdhist.pro,
src/proto/optionstr.pro, src/proto/popupwin.pro,
src/proto/viminfo.pro, src/proto/if_cscope.pro

Patch 8.1.2408

Problem: Syntax menu and build instructions outdated.
Solution: Update build instructions and syntax menu.
Files: Makefile, runtime/makemenu.vim, runtime/synmenu.vim

Patch 8.1.2409

Problem: Creating the distribution doesn't work as documented.
Solution: Adjust name of uninstall binary. Create src/auto directory if
needed.
Files: tools/rename.bat, src/Make_mvc.mak

Patch 8.1.2410

Problem: MS-Windows: test_iminsert fails without IME support.
Solution: Skip the test when imgetstatus() doesn't work.
Files: src/testdir/test_iminsert.vim

Patch 8.1.2411

Problem: Function argument copied unnecessarily.
Solution: Use the argument directly.
Files: src/ex_docmd.c

Patch 8.1.2412

Problem: Crash when evaluating expression with error. (Dhiraj Mishra)
Solution: Check parsing failed. (closes #5329)
Files: src/eval.c, src/testdir/test_lambda.vim

Patch 8.1.2413

Problem: Cannot update ex_cmdidxs.h on MS-Windows.
Solution: Add build rules and dependencies. (Ken Takata, closes #5337)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms

Patch 8.1.2414

Problem: MS-Windows: properties dialog box shows wrong character.
Solution: Explicitly specify encoding. (Ken Takata, closes #5338)
Files: src/vim.rc

Patch 8.1.2415

Problem: Popup menu flickers if an info popup is used. (Nick Jensen)
Solution: Set the pum_skip_redraw flag.
Files: src/popupmenu.c

Patch 8.1.2416

Problem: Loading menus sets v:errmsg.

Solution: Avoid setting v:errmsg and add a test for that. (Jason Franklin)
Files: runtime/delmenu.vim, runtime/menu.vim, src/testdir/test_menu.vim

Patch 8.1.2417

Problem: MinGW/Cygwin build does not clean up all files.
Solution: Delete *.map files. (Michael Soyka)
Files: src/Make_cyg_ming.mak

Patch 8.1.2418

Problem: bufnr('\$') is wrong after recycling popup buffer.
Solution: Sort the buffer list by buffer number. (closes #5335)
Files: src/buffer.c, src/testdir/test_popupwin.vim

Patch 8.1.2419

Problem: With a long file name the hit-enter prompt appears. (J. Lewis Muir)
Solution: When checking for text to wrap don't do this when outputting a CR.
Files: src/message.c, src/testdir/test_display.vim, src/testdir/dumps/Test_long_file_name_1.dump

Patch 8.1.2420

Problem: Crash when calling popup_close() in win_execute().
Solution: Disallow popup_close() in popup window. (Yasuhiro Matsumoto, closes #5345)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.1.2421

Problem: Test88 is old style.
Solution: Turn into a new style test. (Yegappan Lakshmanan, closes #5347)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms, src/testdir/test88.in, src/testdir/test88.ok, src/testdir/test_conceal.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.1.2422

Problem: "make depend" does not work correctly for libvterm.
Solution: Fix build dependencies. And a few minor improvements.
Files: src/Makefile, src/filepath.c, src/insexpand.c, src/main.c

Patch 8.1.2423

Problem: MS-Windows properties shows version as "8, 1, 0".
Solution: Use "8.1.0". (Ken Takata, closes #5342)
Files: src/vim.rc

Patch 8.1.2424

Problem: MS-Windows: console buffer is resized unnecessarily.
Solution: Only call ResizeConBuf() when the size differs. (Nobuhiro Takasaki, closes #5343)
Files: src/os_win32.c

=====

Patch 8.2.0001 and later can be found at [patches-after-8.2](#) .

[version9.txt](#) For [Vim version 9.1](#). Last change: 2024 Mar 03

VIM REFERENCE MANUAL by Bram Moolenaar

[vim-9.0](#) [vim-9](#) [version-9.0](#) [version9.0](#)

Welcome to Vim 9! Several years have passed since the previous release. A large number of bugs have been fixed, many nice features have been added and the Vim9 script syntax is here! This file mentions all the new things and changes to existing features since Vim 8.2.0. The patches up to Vim 8.2 can be found here: [vim-8.2](#) .

Use this command to see the full version and features information of the Vim program you are using:

`:version`

NEW FEATURES	new-9
Vim script enhancements	new-vim-script-9
Command line completion in a popup menu	new-popup-compl
Updated colorschemes	new-colorschemes-9
Various new items	new-items-9
INCOMPATIBLE CHANGES	incompatible-9
IMPROVEMENTS	improvements-9
COMPILE TIME CHANGES	compile-changes-9
PATCHES	patches-9
VERSION 9.1	version-9.1
Changed	changed-9.1
Added	added-9.1
Patches	patches-9.1

See [vi_diff.txt](#) for an overview of differences between Vi and Vim 9.0. See [version4.txt](#) , [version5.txt](#) , [version6.txt](#) , [version7.txt](#) and [version8.txt](#) for differences between Vim versions.

You can find an overview of the most important changes (according to Martin Tournoij) on this site: <https://www.arp242.net/vimlog/>

[Sven-Guckes](#)

[Vim version 9.0](#) is dedicated to Sven Guckes, who passed away in February 2022 when the release was being prepared. Sven was a long time supporter of Vim. He registered the vim.org domain and created the first Vim website. We will remember him!

[Bram Moolenaar](#) [Bram-Moolenaar](#)

[Vim version 9.1](#) is dedicated to Bram Moolenaar, who passed away on August 3rd 2023 while still working full-time on Vim. The Vim project would not exist without his ongoing passion to lead and develop Vim and the community for more

than 30 years. Bram was also passionate about his [ICCF](#) foundation to help children in Uganda. If you enjoy using Vim, please consider donating! We will miss his guidance, passion and leadership.

Obituary Articles: <https://github.com/vim/vim/discussions/12742>
Say Farewell: <https://github.com/vim/vim/discussions/12737>

NEW FEATURES

new-9

First an overview of the more interesting new features. A comprehensive list is below.

Vim9 script

new-vim-script-9

The Vim script language has been changed step by step over many years, preserving backwards compatibility. Several choices made in the early days got in the way of making it work better. At the same time, Vim script is being used much more often, since there are so many plugins being used.

Vim9 script provides a syntax that is much more similar to other languages. In other words: "less weird". Compiled functions are introduced which allow for a large speed improvement. You can expect around ten times faster execution, or even more. The price to pay is that Vim9 script is not backwards compatible. But don't worry, you can still use your old scripts, the new script language is added, it does not replace the legacy script.

Information about Vim9 script can be found in the [Vim9](#) help file.

Command line completion in a popup menu

new-popup-compl

Before there was the **'wildmenu'** option, which uses the space of one line above the statusline to show matches. Only a few matches fit there.

Now a popup menu can be used by setting **'wildoptions'** to "pum". This allows for showing many more matches. This requires redrawing more of the display, but since computers are fast enough that is not a problem.

Updated colorschemes

new-colorschemes-9

Colorschemes from <https://github.com/vim/colorschemes> have been included. They were made to work consistently across many types of terminals. Although generally an improvement, a lot of personal preference is involved. You can always get the old version if you prefer it, look here: https://github.com/vim/colorschemes/blob/master/legacy_colors/

Various new items

new-items-9

Options:

'autosHELLdir'	change directory to the shell's current directory
'cdhome'	change directory to the home directory by ":cd"
'cinscopedecls'	words that are recognized by 'cino-g'
'guiligatures'	GTK GUI: ASCII characters that can form shapes
'mousemoveevent'	report mouse moves with <MouseMove>
'quickfixtextfunc'	function for the text in the quickfix window
'spelloptions'	options for spell checking
'thesaurusfunc'	function to be used for thesaurus completion
'xtermcodes'	request terminal codes from an xterm

Ex commands:

:abstract	(reserved for future use)
:argdedupe	remove duplicates from the argument list
:balt	like ":badd" but also set the alternate file
:class	(reserved for future use)
:def	define a Vim9 user function
:defcompile	compile Vim9 user functions in current script
:disassemble	disassemble Vim9 user function
:echoconsole	like :echormsg but write to stdout
:endinterface	(reserved for future use)
:endclass	(reserved for future use)
:enddef	end of a user function started with :def
:endenum	(reserved for future use)
:enum	(reserved for future use)
:eval	evaluate an expression and discard the result
:export	Vim9: export an item from a script
:final	declare an immutable variable in Vim9
:import	Vim9: import an item from another script
:interface	(reserved for future use)
:static	(reserved for future use)
:type	(reserved for future use)
:var	variable declaration in Vim9
:vim9script	indicates Vim9 script file

Ex command modifiers:

:legacy	make following command use legacy script syntax
:vim9cmd	make following command use Vim9 script syntax

New and extended functions:

assert_nobeep()	assert that a command does not cause a beep
autocmd_add()	add a list of autocmds and groups
autocmd_delete()	delete a list of autocmds and groups
autocmd_get()	return a list of autocmds
blob2list()	get a list of numbers from a blob
charclass()	class of a character
charcol()	character number of the cursor or a mark
charidx()	character index of a byte in a string
digraph_get()	get digraph

digraph_getlist()	get all digraphs
digraph_set()	register digraph
digraph_setlist()	register multiple digraphs
echoraw()	output characters as-is
exists_compiled()	like exists() but check at compile time
extendnew()	make a new Dictionary and append items
flatten()	flatten a List
flattennew()	flatten a copy of a List
fullcommand()	get full command name
getcharpos()	get character position of cursor, mark, etc.
getcharstr()	get a character from the user as a string
getcndcompltpe()	return current cmdline completion type
getcndscreenpos()	return the current cursor position in the cmdline
getcursorcharpos()	get character position of the cursor
getmarklist()	list of global/local marks
getreginfo()	get information about a register
gettext()	lookup message translation
hlget()	get highlight group attributes
hlset()	set highlight group attributes
isabsoluteptath()	check if a path is absolute
list2blob()	get a blob from a list of numbers
maplist()	list of all mappings, a dict for each
mapnew()	make a new List with changed items
mapset()	restore a mapping
matchfuzzy()	fuzzy matches a string in a list of strings
matchfuzzypos()	fuzzy matches a string in a list of strings
menu_info()	get information about a menu item
popup_list()	get list of all popup window IDs
prompt_getprompt()	get the effective prompt text for a buffer
prop_add_list()	attach a property at multiple positions
prop_find()	search for a property
readblob()	read a file into a Blob
readdirer()	get a List of file information in a directory
reduce()	reduce a List to a value
searchcount()	get number of matches before/after the cursor
setcellwidths()	set character cell width overrides
setcharpos()	set character position of cursor, mark, etc.
setcursorcharpos()	set character position of the cursor
slice()	take a slice of a List
strcharlen()	length of a string in characters
terminalprops()	properties of the terminal
test_gui_event()	generate a GUI event for testing
test_null_function()	return a null Funcref
test_srand_seed()	set the seed value for srand()
test_unknown()	return a value with unknown type
test_void()	return a value with void type
typename()	type of a variable as text
virtcol2col()	byte index of a character on screen
win_gettype()	get type of window
win_move_separator()	move window vertical separator
win_move_statusline()	move window status line
windowsversion()	get MS-Windows version

New Vim variables:

<code>v:numbermax</code>	maximum value of a number
<code>v:numbermin</code>	minimum value of a number (negative)
<code>v:numbersize</code>	number of bits in a Number
<code>v:collate</code>	current locale setting for collation order
<code>v:exiting</code>	vim exit code
<code>v:colornames</code>	dictionary that maps color names to hex color strings
<code>v:sizeofint</code>	number of bytes in an int
<code>v:sizeoflong</code>	number of bytes in a long
<code>v:sizeofpointer</code>	number of bytes in a pointer
<code>v:maxcol</code>	maximum line length

New autocommand events:

<code>CompleteDonePre</code>	after Insert mode completion done, before clearing info
<code>DirChangedPre</code>	before the working directory will change
<code>InsertLeavePre</code>	just before leaving Insert mode
<code>ModeChanged</code>	after changing the mode
<code>SigUSR1</code>	after the SIGUSR1 signal has been detected
<code>WinClosed</code>	after closing a window
<code>WinScrolled</code>	after scrolling or resizing a window
<code>VimSuspend</code>	when suspending Vim
<code>VimResume</code>	when Vim is resumed after being suspended

New operators:

<code>>></code>	bitwise right shift
<code><<</code>	bitwise left shift
<code>??</code>	falsy operator

New runtime files:

Too many to list here.

INCOMPATIBLE CHANGES

[incompatible-9](#)

There is only one change that is incompatible with previous releases:

- Lua arrays are now one-based, they used to be zero-based.

Note that when using `Vim9` script several things work differently, see [vim9-differences](#) .

IMPROVEMENTS

[improvements-9](#)

Various small and useful improvements have been made since Vim 8.2, here is a summary.

Many memory leaks, invalid memory accesses and crashes have been fixed.

See the list of patches below: [bug-fixes-9](#) .

Support for Vim expression evaluation in a string. [interpolated-string](#)
Support for evaluating Vim expressions in a heredoc. [:let-heredoc](#)

Support for fuzzy matching:

- a string in a List of strings. [fuzzy-matching](#)
- completion support for command line completion using ['wildoptions'](#).
- for [:vimgrep](#) .

Added support for the [Haiku](#) OS.

Support for "lsp" channel mode to simplify LSP server RPC communication
[language-server-protocol](#) . Support for using a Unix domain socket with a
[channel](#) . IPv6 support in channels [channel-address](#) .

Support for sourcing lines from the current buffer. [:source-range](#)

Terminal window improvements:

- Support for opening a terminal in a popup window. [popup-terminal](#)
- Allow setting underline color in terminal.
- Detect focus events in terminal ([FocusGained](#) and [FocusLost](#)).
- Add bell support for the terminal window. ('belloff')
- Support mouse left-right scrolling in a terminal window.

Support for stopping profiling a Vim script: `:profile stop` and dumping the report to a file: `:profile dump` . [:profile](#)

Completion improvements:

- Argument completion support for the [:breakadd](#) , [:breakdel](#) , [:diffget](#) , [:diffput](#) , [:profile](#) , [:profdel](#) and [:scriptnames](#) commands.
- Support using any Vim type for user_data with the completion functions ([complete-items](#)).
- Stop insert mode completion without changing text ([i_CTRL-X_CTRL-Z](#)).
- Add the "cmdline" option to [getcompletion\(\)](#) to return the command line arguments.

Support for setting the ['foldtext'](#) , ['completefunc'](#) , ['omnifunc'](#) , ['operatorfunc'](#) , ['thesaurusfunc'](#) , ['quickfixtextfunc'](#) , ['tagfunc'](#) , ['imactivatefunc'](#) and ['imstatusfunc'](#) options to a function reference or a lambda function or a script-local function.

Support directly setting the ['balloonexpr'](#) , ['charconvert'](#) , ['foldexpr'](#) , ['formatexpr'](#) , ['includeexpr'](#) , ['printexpr'](#) , ['patchexpr'](#) , ['indentexpr'](#) , ['modelineexpr'](#) , ['diffexpr'](#) and ['printexpr'](#) options to a script-local function.

Improvements in ['fillchars'](#):

- Support for configuring the character used to mark the beginning of a fold, show a closed fold and show a fold separator using "foldopen", "foldclose" and "foldsep" respectively in ['fillchars'](#).
- Support for configuring the character displayed in non existing lines using "eob" in ['fillchars'](#).
- Support for using multibyte items with the "stl", "stlnc", "foldopen", "foldclose" and "foldsep" items in the ['fillchars'](#) option.

Support for the XChaCha20 encryption method. **'cryptmethod'**

Spell checking:

- Spell check current word with `z=` even when **'spell'** is off.
- Add "timeout" to **'spellsuggest'** to limit the searching time for spell suggestions.
- Add support for spell checking CamelCased words by adding "camel" to **'spelloptions'**.

Support for executing Ex commands in a map without changing the current mode `<Cmd>` and `<ScriptCmd>` .

Add optional error code to `:quit` .

Recognize numbers as unsigned when "unsigned" is set in **'nrformats'**.

Expand script ID using `expand('<SID>')` and script name using `expand('<script>')`. `expand()`

Jump to the last accessed tab page using `g<Tab>` and support using the last accessed tab page in `:tabnext` et al.

Locale aware sorting using `:sort` and `sort()` .

Hide cursor when sleeping using `:sleep!` .

Add "multispace" to **'listchars'** to show two or more spaces no matter where they appear. Add "leadmultispace" to **'listchars'** to show two or more leading spaces. Add "lead" to **'listchars'** to set the character used to show leading spaces. Support specifying a character using the hexadecimal notation in **'listchars'** (`\x`, `\u` and `\U`).

Make **'listchars'**, **'virtualedit'** and **'thesaurusfunc'** global-local options.

Support for looping over a string using ``:for``.

Don't reset **'wrap'** for diff windows when "followwrap" is set in **'diffopt'**.

Support for re-evaluating the **'statusline'** expression as a statusline format string (`%{expr %}`)

Add `zp` and `zP` to paste in block mode without adding trailing white space. Add `zy` to yank without trailing white space in block mode.

Add the 'P' command in visual mode to paste text in Visual mode without yanking the deleted text to the unnamed register. `put-Visual-mode`

Add `\%.l`, `\%<.l` and `\%>.l` atoms to match the line the cursor is currently on. See `/\%l` for more information.

Add "list" to **'breakindentopt'** to add additional indent for lines that match a numbered or bulleted list. Add "column" to **'breakindentopt'** to indent soft-wrapped lines at a specific column.

Add the `hl-CursorLineSign` and `hl-CursorLineFold` default highlight groups to adjust sign highlighting for `'cursorline'`.

Add the `hl-CurSearch` default highlight group for the current search match.

Add support for logging on Vim startup (`--log`).

Add support for customizing the quickfix buffer contents using `'quickfixtextfunc'`. Support for the "note" error type (%t) in `errorformat`. Add support for parsing the end line number (%e) and end column number (%k) using `'errorformat'`.

Support truncating the tag stack using `settagstack()`.

Display every option in a separate line when "!" is used with `:set`.

Add "nostop" to `'backspace'` to allow backspacing over the start of insert for `CTRL-W` and `CTRL-U` also.

Sync the undo file if `'fsync'` is set.

Support excluding the `'runtimepath'` and `'packpath'` options from a session file using "skiprtp" in `'sessionoptions'`.

Support for getting the number of lines (line count) in a buffer using `getbufinfo()`.

Support `filter()` and `map()` for blob and string types.

Support for using a multi-byte character for the tag kind. `tags-file-format`

Add support for checking whether a function name is valid using `exists()`.

Update xdiff to version 2.33. Update libvterm to revision 789.

Support `'trim'` for Python/Lua/Perl/Tcl/Ruby/MzScheme interface heredoc.

Add the `t_AU` and `t_8u` termcap codes for underline and undercurl. Add the `t_fd` and `t_fe` termcap codes for detecting focus events.

Support for indenting C pragmas like normal code. (`cinop`)

Add support for defining the syntax fold level. (`:syn-foldlevel`)

Add support for using `\<xxx>` in a string to prepend a modifier to a character. (`expr-quote`)

Add support trimming characters at the beginning or end of a string using `trim()`.

Make `":verbose pwd"` show the scope of the directory. `:pwd-verbose`

Add the "0o" notation for specifying octal numbers. `scriptversion-4`

Support for changing to the previous tab-local and window-local directories using the "tcd -" and "lcd -" commands. (:tcd- and :lcd-)

Add support for skipping an expression using search() .

Add support for sorting the directory contents returned by the readdir() and readdirlex() functions by case.

Add support for executing (:@) a register containing line continuation.

Lua support:

- Call Vim functions from Lua (vim.call() and vim.fn()).
- Convert a Lua function and a closure to a Vim funcref so that it can be accessed in a Vim script (lua-funcref).
- Not backwards compatible: Make Lua arrays one based.
- Add support for using table.insert() and table.remove() functions with Vim lists.
- Support for running multiple Ex-mode commands using vim.command().
- Add vim.lua_version to get the Lua version.
- Add support for accessing Vim namespace dictionaries from Lua (lua-vim-variables).

Support for new UTF-8 characters from Unicode release 13.

Support for using a command block (:command-repl) when defining a :command or an :autocmd .

Support for using :z! to use the Vim display height instead of the current window height.

Support for deleting a buffer-local command using ":delcommand -buffer {cmd}".

When formatting a // comment after a statement, find the start of the line comment, insert the comment leader and indent the comment properly (fo-/).

Add the "numhl" argument to `:sign define` to use a separate highlight group for the line number on a line where a sign is placed. :sign-define

When \$SHELL ends in "nologin" or "false", start Vim in restricted mode.

TermDebug enhancements:

- Support for showing the disassembled code in a separate window.
- Support for the GDB until command.
- Use a separate group for the signs.

xxd: Support for showing offset as a decimal number (-d).

The C omni-complete plugin (ft-c-omni), the file type detection script (ft.vim) and the syntax menu generation script (makemenu.vim) have been rewritten using the Vim9 script syntax.

A large number of tests have been added to verify the Vim functionality. Most of the old style tests have been converted to new style tests using the new

style assert_* functions.

Many Coverity static analysis warnings have been fixed.

COMPILE TIME CHANGES

[compile-changes-9](#)

The following features are now enabled in all the builds:

```
+cindent
+jumplist
+lispindent
+num64
+smartindent
+tag_binary
+title
```

The following features have been removed. They are either obsolete or didn't work properly:

- Athena and neXTaw GUI support (use Motif instead)
- EBCDIC support
- Atari MiNT and BeOS
- Mac Carbon GUI (use MacVim instead)

The rgb.txt file is no longer included, use colors/lists/default.vim instead.

Several large source files were split, mainly to make it easier to inspect code coverage information. Source files have also been refactored for maintainability.

Support for building Vim with Mingw64 clang compiler on MS-Windows.

Support for building Vim with Python 3.10, Lua 5.4.4, Perl 5.34 and Ruby 3.1.0.

PATCHES

[patches-9](#) [bug-fixes-9](#)
[patches-after-8.2](#)

The list of patches that got included since 8.2.0. This includes all the new features, but does not include runtime file changes (syntax, indent, help, etc.)

Patch 8.2.0001

Problem: #endif comments do not reflect corresponding #ifdef.
Solution: Update the comments. (Rene Nyffenegger, closes #5351)
Files: src/ui.c

Patch 8.2.0002

Problem: "dj" only deletes first line of closed fold.
Solution: Adjust last line of operator for linewise motion. (closes #5354)
Files: src/ops.c, src/testdir/test_fold.vim

Patch 8.2.0003

Problem: Build file dependencies are incomplete.

Solution: Fix the dependencies. (Ken Takata, closes #5356)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms,
src/Makefile

Patch 8.2.0004

Problem: Get E685 and E931 if buffer reload is interrupted.
Solution: Do not abort deleting a dummy buffer. (closes #5361)
Files: src/buffer.c, src/proto/buffer.pro, src/testdir/test_trycatch.vim,
src/ex_cmds.c, src/ex_getln.c, src/misc2.c, src/quickfix.c,
src/window.c, src/vim.h

Patch 8.2.0005

Problem: Duplication in version info.
Solution: Use preprocessor string concatenation. (Ken Takata, closes #5357)
Files: src/version.h

Patch 8.2.0006

Problem: Test using long file name may fail. (Vladimir Lomov)
Solution: Limit the name length. (Christian Brabandt, closes #5358)
Files: src/testdir/test_display.vim

Patch 8.2.0007

Problem: Popup menu positioned wrong with folding in two tabs.
Solution: Update the cursor line height. (closes #5353)
Files: src/move.c, src/proto/move.pro, src/popupmenu.c,
src/testdir/test_ins_complete.vim,
src/testdir/dumps/Test_pum_with_folds_two_tabs.dump

Patch 8.2.0008

Problem: Test72 is old style.
Solution: Convert to new style test. (Yegappan Lakshmanan, closes #5362)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test72.in, src/testdir/test72.ok,
src/testdir/test_undo.vim

Patch 8.2.0009

Problem: VMS: terminal version doesn't build.
Solution: Move MIN definition. Adjust #ifdefs. (Zoltan Arpadffy)
Files: src/bufwrite.c, src/fileio.c, src/ui.c, src/xxd/Make_vms.mms

Patch 8.2.0010

Problem: Test64 is old style.
Solution: Convert to new style test. (Yegappan Lakshmanan, closes #5363)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test64.in, src/testdir/test64.ok,
src/testdir/test95.in, src/testdir/test_regexp_latin.vim

Patch 8.2.0011

Problem: Screen updating wrong when opening preview window.
Solution: Redraw the window when the preview window opens.
Files: src/popupmenu.c, src/testdir/test_ins_complete.vim,
src/testdir/dumps/Test_pum_with_preview_win.dump

Patch 8.2.0012

Problem: Some undo functionality is not tested.
Solution: Add a few more test cases. (Dominique Pellé, closes #5364)
Files: src/testdir/test_undo.vim

Patch 8.2.0013

Problem: Not using a typedef for condstack.
Solution: Add a typedef.
Files: src/structs.h, src/ex_docmd.c, src/ex_eval.c, src/userfunc.c,
src/ex_cmds.h, src/proto/ex_eval.pro

Patch 8.2.0014

Problem: Test69 and test95 are old style.
Solution: Convert to new style tests. (Yegappan Lakshmanan, closes #5365)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test69.in, src/testdir/test69.ok,
src/testdir/test95.in, src/testdir/test95.ok,
src/testdir/test_regexp_utf8.vim, src/testdir/test_textformat.vim

Patch 8.2.0015

Problem: Not all modeline variants are tested.
Solution: Add modeline tests. (Dominique Pellé, closes #5369)
Files: src/testdir/test_modeline.vim

Patch 8.2.0016

Problem: Test name used twice, option not restored properly.
Solution: Rename function, restore option with "&".
Files: src/testdir/test_textformat.vim

Patch 8.2.0017

Problem: OS/2 and MS-DOS are still mentioned, even though support was removed long ago.
Solution: Update documentation. (Yegappan Lakshmanan, closes #5368)
Files: runtime/doc/autocmd.txt, runtime/doc/change.txt,
runtime/doc/cmdline.txt, runtime/doc/editing.txt,
runtime/doc/eval.txt, runtime/doc/gui.txt, runtime/doc/insert.txt,
runtime/doc/options.txt, runtime/doc/print.txt,
runtime/doc/quickfix.txt, runtime/doc/repeat.txt,
runtime/doc/starting.txt, runtime/doc/usr_01.txt,
runtime/doc/usr_05.txt, runtime/doc/usr_41.txt,
runtime/doc/vi_diff.txt, runtime/gvimrc_example.vim,
runtime/tools/README.txt, runtime/vimrc_example.vim, src/feature.h

Patch 8.2.0018

Problem: :join does not add white space where it should. (Zdenek Dohnal)
Solution: Handle joining multiple lines properly.
Files: src/ops.c, src/testdir/test_join.vim

Patch 8.2.0019

Problem: Cannot get number of lines of another buffer.
Solution: Add "linecount" to getbufinfo(). (Yasuhiro Matsumoto, closes #5370)
Files: src/evalbuffer.c, src/testdir/test_bufwintabinfo.vim,
runtime/doc/eval.txt

Patch 8.2.0020

Problem: Mouse clicks in the command line not tested.
Solution: Add tests. (Dominique Pellé, closes #5366)
Files: src/testdir/test_termcodes.vim

Patch 8.2.0021

Problem: Timer test fails too often on Travis with macOS.
Solution: Be less strict with the time.
Files: src/testdir/test_timers.vim

Patch 8.2.0022

Problem: Click in popup window doesn't close it in the GUI. (Sergey Vlasov)
Solution: When processing the selection also send a button release event.
(closes #5367)
Files: src/gui.c

Patch 8.2.0023

Problem: Command line editing not sufficiently tested.
Solution: Add more tests. (Dominique Pellé, closes #5374)
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_cmdline.vim, src/testdir/test_ex_mode.vim

Patch 8.2.0024

Problem: Filetype Rego not recognized.
Solution: Add *.rego. (Matt Dunford, closes #5376)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0025

Problem: Repeated word in comment.
Solution: Remove one. (Rene Nyffenegger, closes #5384)
Files: src/structs.h

Patch 8.2.0026

Problem: Still some /* */ comments.
Solution: Convert to // comments.
Files: src/message.c, src/message_test.c, src/misc1.c, src/misc2.c,
src/move.c

Patch 8.2.0027

Problem: Still some /* */ comments.
Solution: Convert to // comments.
Files: src/iid_ole.c, src/indent.c, src/insexpand.c, src/iscygpty.c,
src/version.c

Patch 8.2.0028

Problem: Searchpairpos() is not tested.
Solution: Add tests. Also improve searchpair() testing. (Dominique Pellé,
closes #5388)
Files: src/testdir/test_search.vim

Patch 8.2.0029

Problem: MS-Windows: crash with empty job command.
Solution: Check for NULL result. (Yasuhiro Matsumoto, closes #5390)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.2.0030

Problem: "gF" does not work on output of "verbose command".
Solution: Recognize " line " and translations. (closes #5391)
Files: src/globals.h, src/eval.c, src/findfile.c, src/testdir/test_gf.vim

Patch 8.2.0031 (after 8.2.0029)

Problem: MS-Windows: test for empty job fails
Solution: Check for error message, make it also fail on Unix.
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.2.0032 (after 8.2.0031)

Problem: MS-Windows: test for blank job fails
Solution: Check before escaping.
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.2.0033

Problem: Crash when make_extmatch() runs out of memory.
Solution: Check for NULL. (Dominique Pellé, closes #5392)
Files: src/regexp_bt.c, src/regexp_nfa.c

Patch 8.2.0034

Problem: Missing check for out of memory.
Solution: Check for NULL after vim_strsave(). (Dominique Pellé,
closes #5393)
Files: src/filepath.c

Patch 8.2.0035

Problem: Saving and restoring called_emsg is clumsy.
Solution: Count the number of error messages.
Files: src/message.c, src/buffer.c, src/channel.c, src/drawscreen.c,
src/ex_cmds2.c, src/gui.c, src/highlight.c, src/main.c,
src/regexp.c, src/search.c, src/testing.c, src/globals.h

Patch 8.2.0036

Problem: Not enough test coverage for match functions.
Solution: Add a few more test cases. (Dominique Pellé, closes #5394)
Add error number.
Files: src/testdir/test_match.vim

Patch 8.2.0037

Problem: Missing renamed message.
Solution: Now really add the error number.
Files: src/highlight.c

Patch 8.2.0038

Problem: Spell suggestions insufficiently tested.
Solution: Add spell suggestion tests. (Dominique Pellé, closes #5398)
Files: src/testdir/test_spell.vim

Patch 8.2.0039

Problem: Memory access error when "z=" has no suggestions.
Solution: Check for negative index.
Files: src/testdir/test_spell.vim, src/spellsuggest.c

Patch 8.2.0040

Problem: Timers test is still flaky on Travis for Mac.
Solution: Run separately instead of as part of test_alot.
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 8.2.0041

Problem: Leaking memory when selecting spell suggestion.
Solution: Free previous value at the right time.
Files: src/spellsuggest.c

Patch 8.2.0042

Problem: Clearing funccal values twice.
Solution: Remove clearing individual fields.
Files: src/userfunc.c

Patch 8.2.0043

Problem: Timers test is still flaky on Travis for Mac.
Solution: Increase maximum expected time.
Files: src/testdir/test_timers.vim

Patch 8.2.0044

Problem: Expression type is used inconsistently.
Solution: Add "ETYPE_IS" and "ETYPE_ISNOT" as separate enum values. Rename "TYPE_" to "ETYPE_" to avoid confusion.
Files: src/structs.h, src/eval.c, src/proto/eval.pro, src/debugger.c

Patch 8.2.0045 (after 8.2.0044)

Problem: Script test fails.
Solution: For numbers "is" and "isnot" work like "==" and "!=".
Files: src/eval.c

Patch 8.2.0046

Problem: Tests for spell suggestions are slow.
Solution: Use shorter words. Test with latin1 and utf-8 to cover more code. (Dominique Pellé, closes #5399)
Files: src/testdir/test_spell.vim

Patch 8.2.0047

Problem: Cannot skip tests for specific MS-Windows platform.
Solution: Add windowsversion().
Files: src/os_win32.c, src/globals.h, src/evalfunc.c, runtime/doc/eval.txt, src/testdir/gen_opt_test.vim, src/testdir/test_options.vim

Patch 8.2.0048

Problem: Another timers test is flaky on Travis for Mac.
Solution: Increase maximum expected time.
Files: src/testdir/test_timers.vim

Patch 8.2.0049

Problem: Command line completion not fully tested.
Solution: Add more test cases. Make help sorting stable. (Dominique Pellé, closes #5402)

Files: src/ex_cmds.c, src/testdir/test_cd.vim,
src/testdir/test_cmdline.vim, src/testdir/test_help.vim,
src/testdir/test_menu.vim, src/testdir/test_options.vim,
src/testdir/test_syntax.vim

Patch 8.2.0050

Problem: After deleting a file mark it is still in viminfo.
Solution: When a file mark was deleted more recently than the mark in the merged viminfo file was updated, do not store the mark. (Pavol Juhas, closes #5401, closes #1339)

Files: src/mark.c, src/testdir/test_marks.vim,
src/testdir/test_viminfo.vim, src/viminfo.c

Patch 8.2.0051 (after 8.2.0049)

Problem: Command line completion test skipped. (Christian Brabandt)

Solution: Invert condition.

Files: src/testdir/test_cmdline.vim

Patch 8.2.0052

Problem: More-prompt not properly tested.

Solution: Add a test case. (Dominique Pellé, closes #5404)

Files: src/testdir/test_messages.vim

Patch 8.2.0053

Problem: windowsversion() does not always return the right value.

Solution: Add a compatibility section in the manifest. (Ken Takata, closes #5407)

Files: src/gvim.exe.mnf

Patch 8.2.0054

Problem: :diffget and :diffput don't have good completion.

Solution: Add proper completion. (Dominique Pellé, closes #5409)

Files: runtime/doc/eval.txt, src/buffer.c, src/cmdexpand.c,
src/testdir/test_diffmode.vim, src/usercmd.c, src/vim.h

Patch 8.2.0055

Problem: Cannot use ":gui" in vimrc with VIMDLL enabled.

Solution: Change the logic, check "gui.starting". (Ken Takata, closes #5408)

Files: src/gui.c

Patch 8.2.0056

Problem: Execution stack is incomplete and inefficient.

Solution: Introduce a proper execution stack and use it instead of sourcing_name/sourcing_lnum. Create a string only when used.

Files: src/structs.h, src/globals.h, src/autocmd.c, src/buffer.c
src/debugger.c, src/ex_docmd.c, src/ex_eval.c, src/highlight.c,
src/main.c, src/map.c, src/message.c, src/proto/scriptfile.pro,
src/scriptfile.c, src/option.c, src/profiler.c, src/spellfile.c,
src/term.c, src/testing.c, src/usercmd.c, src/userfunc.c,
src/kword_test.c, src/testdir/test_debugger.vim

Patch 8.2.0057 (after 8.2.0056)

Problem: Cannot build with small features.

Solution: Add #ifdefs.

Files: src/scriptfile.c

Patch 8.2.0058

Problem: Running tests changes ~/.viminfo.

Solution: Make '**viminfo**' empty when summarizing tests results. (closes #5414)

Files: src/testdir/summarize.vim

Patch 8.2.0059

Problem: Compiler warnings for unused variables in small build. (Tony Mechelynck)

Solution: Add #ifdef.

Files: src/scriptfile.c

Patch 8.2.0060

Problem: Message test only runs with one encoding. (Dominique Pellé)

Solution: Run the test with "utf-8" and "latin1". Fix underflow. (related to #5410)

Files: src/message_test.c, src/message.c

Patch 8.2.0061

Problem: The execute stack can grow big and never shrinks.

Solution: Reduce the size in garbage collect.

Files: src/eval.c

Patch 8.2.0062

Problem: Memory test is flaky on FreeBSD.

Solution: Add a short sleep before getting the first size.

Files: src/testdir/test_memory_usage.vim

Patch 8.2.0063

Problem: Wrong size argument to vim_snprintf(). (Dominique Pellé)

Solution: Reduce the size by the length. (related to #5410)

Files: src/ops.c

Patch 8.2.0064

Problem: Diffmode completion doesn't use per-window setting.

Solution: Check if a window is in diff mode. (Dominique Pellé, closes #5419)

Files: src/buffer.c, src/testdir/test_diffmode.vim

Patch 8.2.0065

Problem: Amiga and alike: autoopen only used on Amiga OS4.

Solution: Adjust #ifdefs. (Ola Söder, closes #5413)

Files: src/os_amiga.c

Patch 8.2.0066

Problem: Some corners of vim_snprintf() are not tested.

Solution: Add a test in C. (Dominique Pellé, closes #5422)

Files: src/message_test.c

Patch 8.2.0067

Problem: ERROR_UNKNOWN clashes on some systems.

Solution: Rename ERROR_ to FCERR_. (Ola Söder, closes #5415)

Files: src/evalfunc.c, src/userfunc.c, src/vim.h

Patch 8.2.0068

Problem: Crash when using Python 3 with "utf32" encoding. (Dominique Pellé)
Solution: Use "utf-8" whenever enc_utf8 is set. (closes #5423)
Files: src/testdir/test_python3.vim, src/if_py_both.h

Patch 8.2.0069

Problem: ETYPE_ is used for two different enums.
Solution: Rename one to use EXPR_.
Files: src/structs.h, src/eval.c, src/debugger.c

Patch 8.2.0070

Problem: Crash when using Python 3 with "debug" encoding. (Dominique Pellé)
Solution: Use "euc-jp" whenever enc_dbcs is set.
Files: src/testdir/test_python3.vim, src/if_py_both.h

Patch 8.2.0071

Problem: Memory test often fails on Cirrus CI.
Solution: Allow for more tolerance in the upper limit. Remove sleep.
Files: src/testdir/test_memory_usage.vim

Patch 8.2.0072 (after 8.2.0071)

Problem: Memory test still fails on Cirrus CI.
Solution: Allow for a tiny bit more tolerance in the upper limit.
Files: src/testdir/test_memory_usage.vim

Patch 8.2.0073

Problem: Initializing globals with COMMA is clumsy.
Solution: Use INIT2(), INIT3(), etc.
Files: src/vim.h, src/globals.h

Patch 8.2.0074

Problem: Python 3 unicode test sometimes fails.
Solution: Make 'termencoding' empty. Correct number of error message.
Files: src/change.c, runtime/doc/options.txt, runtime/doc/message.txt, src/testdir/test_python3.vim

Patch 8.2.0075

Problem: Python 3 unicode test still sometimes fails.
Solution: Skip the test when 'termencoding' is not empty.
Files: src/testdir/test_python3.vim

Patch 8.2.0076

Problem: Python 3 unicode test fails on MS-Windows.
Solution: Do not set 'encoding' to "debug" on MS-Windows.
Files: src/testdir/test_python3.vim

Patch 8.2.0077

Problem: settagstack() cannot truncate at current index.
Solution: Add the "t" action. (Yegappan Lakshmanan, closes #5417)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/tag.c, src/testdir/test_tagjump.vim

Patch 8.2.0078

Problem: Expanding <sfiler> works differently the second time.

Solution: Keep the expanded name when redefining a function. (closes #5425)
Files: src/testdir/test_vimscript.vim, src/userfunc.c

Patch 8.2.0079

Problem: Python 3 unicode test still fails on MS-Windows.
Solution: Do not set **'encoding'** to "euc-tw" on MS-Windows.
Files: src/testdir/test_python3.vim

Patch 8.2.0080

Problem: Globals using INIT4() are not in the tags file.
Solution: Adjust the tags command.
Files: src/configure.ac, src/auto/configure

Patch 8.2.0081

Problem: MS-Windows also need the change to support INIT4().
Solution: Add the ctags arguments. (Ken Takata)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.2.0082

Problem: When reusing a buffer listeners are not cleared. (Axel Forsman)
Solution: Clear listeners when reusing a buffer. (closes #5431)
Files: src/testdir/test_listener.vim, src/buffer.c

Patch 8.2.0083

Problem: Text properties wrong when tabs and spaces are exchanged.
Solution: Take text properties into account. (Nobuhiro Takasaki,
closes #5427)
Files: src/edit.c, src/testdir/test_textprop.vim

Patch 8.2.0084

Problem: Complete item "user_data" can only be a string.
Solution: Accept any type of variable. (closes #5412)
Files: src/testdir/test_ins_complete.vim, src/insexpand.c, src/dict.c,
src/proto/dict.pro, src/eval.c, runtime/doc/insert.txt

Patch 8.2.0085

Problem: Dead code in builtin functions.
Solution: Clean up the code.
Files: src/evalvars.c, src/sound.c, src/textprop.c

Patch 8.2.0086 (after 8.2.0084)

Problem: Build error for small version. (Tony Mechelynck)
Solution: Only use "user_data" with the +eval feature. Remove unused
variable.
Files: src/insexpand.c, src/dict.c

Patch 8.2.0087

Problem: Crash in command line expansion when out of memory.
Solution: Check for NULL pointer. Also make ExpandGeneric() static.
(Dominique Pellé, closes #5437)
Files: src/cmdexpand.c, src/proto/cmdexpand.pro

Patch 8.2.0088

Problem: Insufficient tests for tags; bug in using extra tag field when

using an ex command to position the cursor.
Solution: Fix the bug, add more tests. (Yegappan Lakshmanan, closes #5439)
Files: runtime/doc/tagsrch.txt, src/tag.c,
src/testdir/test_ins_complete.vim, src/testdir/test_tagfunc.vim,
src/testdir/test_tagjump.vim, src/testdir/test_taglist.vim

Patch 8.2.0089

Problem: Crash when running out of memory in :setfiletype completion.
Solution: Do not allocate memory. (Dominique Pellé, closes #5438)
Files: src/cmdexpand.c

Patch 8.2.0090

Problem: Generated files show up in git status.
Solution: Ignore a few more files.
Files: .gitignore

Patch 8.2.0091

Problem: Compiler warnings for size_t / int types.
Solution: Change type to size_t. (Mike Williams)
Files: src/scriptfile.c

Patch 8.2.0092

Problem: Tags functionality insufficiently tested.
Solution: Add more tags tests. (Yegappan Lakshmanan, closes #5446)
Files: src/testdir/test_tagjump.vim

Patch 8.2.0093

Problem: win_splitmove() can make Vim hang.
Solution: Check windows exists in the current tab page. (closes #5444)
Files: src/testdir/test_window_cmd.vim, src/evalwindow.c

Patch 8.2.0094

Problem: MS-Windows: cannot build with Strawberry Perl 5.30.
Solution: Define __builtin_expect() as a workaround. (Ken Takata,
closes #5267)
Files: src/if_perl.xs

Patch 8.2.0095

Problem: Cannot specify exit code for :cquit.
Solution: Add optional argument. (Thinca, Yegappan Lakshmanan, closes #5442)
Files: runtime/doc/quickfix.txt, src/ex_cmds.h, src/ex_docmd.c,
src/testdir/test_quickfix.vim

Patch 8.2.0096

Problem: Cannot create tiny popup window in last column. (Daniel Steinberg)
Solution: Remove position limit. (closes #5447)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_20.dump,
src/testdir/dumps/Test_popupwin_21.dump

Patch 8.2.0097

Problem: Crash with autocommand and spellfile. (Tim Pope)
Solution: Do not pop exestack when not pushed. (closes #5450)
Files: src/testdir/test_autocmd.vim, src/spellfile.c

Patch 8.2.0098

Problem: Exe stack length can be wrong without being detected.
Solution: Add a check when ABORT_ON_INTERNAL_ERROR is defined.
Files: src/macros.h, src/autocmd.c, src/buffer.c, src/ex_docmd.c,
src/main.c, src/map.c, src/scriptfile.c, src/spellfile.c,
src/userfunc.c

Patch 8.2.0099

Problem: Use of NULL pointer when out of memory.
Solution: Check for NULL pointer. (Dominique Pellé, closes #5449)
Files: src/cmdexpand.c

Patch 8.2.0100

Problem: Macros for Ruby are too complicated.
Solution: Do not use DYNAMIC_RUBY_VER, use RUBY_VERSION. (Ken Takata,
closes #5452)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/auto/configure,
src/configure.ac, src/if_ruby.c

Patch 8.2.0101

Problem: Crash when passing null object to ":echomsg".
Solution: Check for NULL pointer. (Yasuhiro Matsumoto, closes #5460)
Files: src/eval.c, src/testdir/test_messages.vim

Patch 8.2.0102

Problem: Messages test fails in small version.
Solution: Only use test_null_job() when available.
Files: src/testdir/test_messages.vim

Patch 8.2.0103

Problem: Using null object with execute() has strange effects.
Solution: Give an error message for Job and Channel.
Files: src/testdir/test_execute_func.vim, src/globals.h, src/eval.c,
src/evalfunc.c

Patch 8.2.0104

Problem: Using channel or job with ":execute" has strange effects.
Solution: Give an error message for Job and Channel.
Files: src/testdir/test_eval_stuff.vim, src/eval.c

Patch 8.2.0105

Problem: Vim license not easy to find on github.
Solution: Add a separate LICENCE file. (closes #5458)
Files: LICENSE, Filelist

Patch 8.2.0106

Problem: Printf formats are not exactly right.
Solution: Adjust signed/unsigned conversions. (Frazer Clews, closes #5456)
Files: runtime/tools/ccfilter.c, src/libvterm/src/parser.c,
src/libvterm/src/pen.c, src/ui.c

Patch 8.2.0107

Problem: Hgignore is out of sync from gitignore.

Solution: Add lines to hgignore. (Ken Takata)
Files: .hgignore

Patch 8.2.0108

Problem: When sign text is changed a manual redraw is needed. (Pontus Lietzler)

Solution: Redraw automatically. (closes #5455)
Files: src/testdir/test_signs.vim, src/sign.c,
src/testdir/dumps/Test_sign_cursor_1.dump,
src/testdir/dumps/Test_sign_cursor_2.dump,
src/testdir/dumps/Test_sign_cursor_3.dump,
src/testdir/dumps/Test_sign_cursor_01.dump,
src/testdir/dumps/Test_sign_cursor_02.dump

Patch 8.2.0109

Problem: Corrupted text properties when expanding spaces.

Solution: Reallocate the line. (Nobuhiro Takasaki, closes #5457)
Files: src/edit.c, src/testdir/test_textprop.vim

Patch 8.2.0110

Problem: prop_find() is not implemented.

Solution: Implement prop_find(). (Ryan Hackett, closes #5421, closes #4970)
Files: src/evalfunc.c, src/proto/textprop.pro,
src/testdir/test_textprop.vim, src/textprop.c,
runtime/doc/textprop.txt

Patch 8.2.0111

Problem: VAR_SPECIAL is also used for booleans.

Solution: Add VAR_BOOL for better type checking.
Files: src/structs.h, src/dict.c, src/eval.c, src/evalfunc.c,
src/evalvars.c, src/if_lua.c, src/if_mzsch.c, src/if_py_both.h,
src/if_ruby.c, src/json.c, src/popupmenu.c, src/proto/dict.pro,
src/testing.c, src/vim.h, src/viminfo.c

Patch 8.2.0112

Problem: Illegal memory access when using '**cindent**'.

Solution: Check for NUL byte. (Dominique Pellé, closes #5470)
Files: src/cindent.c, src/testdir/test_cindent.vim

Patch 8.2.0113 (after 8.2.0095)

Problem: "make cmdidxs" fails.

Solution: Allow address for ":cquit". Add --not-a-term to avoid a delay.
Files: src/ex_cmds.h, src/Makefile, src/Make_cyg_ming.mak,
src/Make_mvc.mak

Patch 8.2.0114

Problem: Info about sourced scripts is scattered.

Solution: Use scriptitem_T for info about a script, including s: variables.
Drop ga_scripts.
Files: src/structs.h, src/evalvars.c, src/scriptfile.c, src/eval.c

Patch 8.2.0115

Problem: Byte2line() does not work correctly with text properties. (Billie Cleek)

Solution: Take the bytes of the text properties into account.
(closes #5334)

Files: src/testdir/test_textprop.vim, src/memline.c

Patch 8.2.0116

Problem: BufEnter autocmd not triggered on ":tab drop". (Andy Stewart)

Solution: Decrement autocmd_no_enter for the last file. (closes #1660,
closes #5473)

Files: src/arglist.c, src/testdir/test_tabpage.vim

Patch 8.2.0117

Problem: Crash when using gettabwinvar() with invalid arguments. (Yilin Yang)

Solution: Use "curtab" if "tp" is NULL. (closes #5475)

Files: src/evalwindow.c, src/testdir/test_getvar.vim

Patch 8.2.0118

Problem: Crash when cycling to buffers involving popup window.

Solution: Do not decrement buffer reference count.

Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_infopopup_7.dump

Patch 8.2.0119

Problem: Message test fails on some platforms. (Elimar Riesebieter)

Solution: Add type cast to vim_snprintf() argument. (Dominique Pellé)

Files: src/message_test.c

Patch 8.2.0120

Problem: virtcol() does not check arguments to be valid, which may lead to
a crash.

Solution: Check the column to be valid. Do not decrement MAXCOL.
(closes #5480)

Files: src/evalfunc.c, src/testdir/test_marks.vim

Patch 8.2.0121

Problem: filter() and map() on blob don't work.

Solution: Correct the code. (closes #5483)

Files: src/list.c, src/testdir/test_blob.vim

Patch 8.2.0122

Problem: Readme files still mention MS-DOS.

Solution: Update readme files. (Ken Takata, closes #5486)

Files: README.md, README.txt, READMEdir/README_dos.txt,
READMEdir/README_srcdos.txt, READMEdir/README_w32s.txt,
runtime/doc/os_win32.txt

Patch 8.2.0123

Problem: complete_info() does not work when CompleteDone is triggered.

Solution: Trigger CompleteDone before clearing the info.

Files: src/insexpand.c, runtime/doc/autocmd.txt,
src/testdir/test_ins_complete.vim

Patch 8.2.0124

Problem: Compiler warnings for variable types.

Solution: Change type, add type cast. (Mike Williams)
Files: src/memline.c

Patch 8.2.0125

Problem: :mode no longer works for any system.
Solution: Always give an error message.
Files: src/ex_docmd.c, runtime/doc/quickref.txt, src/os_amiga.c,
src/proto/os_amiga.pro, src/os_mswin.c, src/proto/os_mswin.pro,
src/os_unix.c, src/proto/os_unix.pro

Patch 8.2.0126 (after 8.2.0124)

Problem: Textprop test fails.
Solution: Fix sign in computation.
Files: src/memline.c

Patch 8.2.0127

Problem: Some buffer commands work in a popup window.
Solution: Disallow :bnext, :bprev, etc. (Naruhiko Nishino, closes #5494)
Files: src/ex_docmd.c, src/testdir/test_popupwin.vim

Patch 8.2.0128

Problem: Cannot list options one per line.
Solution: Use ":set!" to list one option per line.
Files: src/ex_docmd.c, src/option.c, src/proto/option.pro, src/vim.h,
src/ex_cmds.h, src/optiondefs.h, src/testdir/test_options.vim,
runtime/doc/options.txt

Patch 8.2.0129

Problem: MS-Windows installer doesn't use Turkish translations.
Solution: Enable the Turkish translations and fix a few. (Emir Sarı,
closes #5493)
Files: nsis/gvim.nsi, nsis/lang/turkish.nsi

Patch 8.2.0130

Problem: Python3 ranges are not tested.
Solution: Add test. (Dominique Pellé, closes #5498)
Files: src/testdir/test_python3.vim

Patch 8.2.0131

Problem: Command line is not cleared when switching tabs and the command
line height differs.
Solution: Set the "clear_cmdline" flag when needed. (Naruhiko Nishino,
closes #5495)
Files: src/testdir/dumps/Test_cmdlineclear_tabenter.dump,
src/testdir/test_cmdline.vim, src/window.c

Patch 8.2.0132

Problem: Script may be re-used when deleting and creating a new one.
Solution: When the inode matches, also check the file name.
Files: src/scriptfile.c, src/testdir/test_source.vim

Patch 8.2.0133

Problem: Invalid memory access with search command.
Solution: When :normal runs out of characters in bracketed paste mode break

Files: out of the loop.(closes #5511)
src/testdir/test_search.vim, src/edit.c

Patch 8.2.0134

Problem: Some map functionality not covered by tests.
Solution: Add tests. (Yegappan Lakshmanan, closes #5504)
Files: src/testdir/test_maparg.vim, src/testdir/test_mapping.vim

Patch 8.2.0135 (after 8.2.0133)

Problem: Bracketed paste can still cause invalid memory access. (Dominique Pellé)
Solution: Check for NULL pointer.
Files: src/edit.c, src/testdir/test_search.vim

Patch 8.2.0136

Problem: Stray ch_logfile() call.
Solution: Remove it. (closes #5503)
Files: src/testdir/test_source.vim

Patch 8.2.0137

Problem: Crash when using win_execute() from a new tab.
Solution: Set the tp_*win pointers. (Ozaki Kiichi, closes #5512)
Files: src/testdir/test_winbuf_close.vim, src/window.c

Patch 8.2.0138

Problem: Memory leak when starting a job fails.
Solution: Free the list of arguments. (Ozaki Kiichi, closes #5510)
Files: src/channel.c, src/testdir/test_channel.vim

Patch 8.2.0139

Problem: MS-Windows: default for IME is inconsistent.
Solution: Also make IME default enabled with MVC. (Ken Takata, closes #5508)
Files: src/Make_mvc.mak

Patch 8.2.0140

Problem: CI does not test building doc tags.
Solution: Add the vimtags/gcc build. Cleanup showing version. (Ozaki Kiichi, closes #5513)
Files: .travis.yml, Filelist, ci/if_ver-1.vim, ci/if_ver-2.vim, ci/if_ver-cmd.vim, runtime/doc/Makefile, runtime/doc/doctags.vim, src/testdir/if_ver-1.vim, src/testdir/if_ver-2.vim

Patch 8.2.0141

Problem: No swift filetype detection.
Solution: Add swift, swiftgyb and sil. (Emir Sarı, closes #5517)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0142

Problem: Possible to enter popup window with **CTRL-W** p. (John Devin)
Solution: Check entered window is not a popup window. (closes #5515)
Files: src/window.c, src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_previewpopup_9.dump, src/testdir/dumps/Test_popupwin_previewpopup_10.dump

Patch 8.2.0143

Problem: Coverity warning for possible use of NULL pointer.
Solution: Check argv is not NULL.
Files: src/channel.c

Patch 8.2.0144

Problem: Some mapping code is not fully tested.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5519)
Files: src/testdir/test_langmap.vim, src/testdir/test_maparg.vim,
src/testdir/test_mapping.vim

Patch 8.2.0145

Problem: Using #error for compilation errors should be OK now.
Solution: Use #error. (Ken Takata, closes #5299)
Files: src/blowfish.c, src/vim.h

Patch 8.2.0146

Problem: Wrong indent when **'showbreak'** and **'breakindent'** are set and **'briopt'** includes "sbr".
Solution: Reset "need_showbreak" where needed. (Ken Takata, closes #5523)
Files: src/drawline.c, src/testdir/test_breakindent.vim

Patch 8.2.0147

Problem: Block Visual mode operators not correct when **'linebreak'** set.
Solution: Set w_p_lbr to lbr_saved more often. (Ken Takata, closes #5524)
Files: src/ops.c, src/testdir/test_listlbr.vim

Patch 8.2.0148

Problem: Mapping related function in wrong source file.
Solution: Move the function. Add a few more test cases. (Yegappan Lakshmanan, closes #5528)
Files: src/map.c, src/proto/term.pro, src/term.c,
src/testdir/test_mapping.vim

Patch 8.2.0149

Problem: Maintaining a Vim9 branch separately is more work.
Solution: Merge the Vim9 script changes.
Files: README.md, README_VIM9.md, runtime/doc/Makefile,
runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/tags,
runtime/doc/vim9.txt, runtime/ftplugin/vim.vim,
runtime/indent/vim.vim, runtime/syntax/vim.vim,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile, src/blob.c,
src/channel.c, src/dict.c, src/eval.c, src/evalbuffer.c,
src/evalfunc.c, src/evalvars.c, src/ex_cmdidxs.h, src/ex_cmds.h,
src/ex_docmd.c, src/ex_eval.c, src/filepath.c, src/globals.h,
src/gui.c, src/if_lua.c, src/if_py_both.h, src/insexpand.c,
src/json.c, src/list.c, src/macros.h, src/main.c, src/message.c,
src/misc1.c, src/proto.h, src/proto/blob.pro, src/proto/eval.pro,
src/proto/evalfunc.pro, src/proto/evalvars.pro,
src/proto/ex_docmd.pro, src/proto/ex_eval.pro, src/proto/list.pro,
src/proto/message.pro, src/proto/scriptfile.pro,
src/proto/userfunc.pro, src/proto/vim9compile.pro,
src/proto/vim9execute.pro, src/proto/vim9script.pro,
src/scriptfile.c, src/session.c, src/structs.h, src/syntax.c,

src/testdir/Make_all.mak, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim, src/testing.c, src/userfunc.c,
src/vim.h, src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/vim9script.c, src/viminfo.c

Patch 8.2.0150

Problem: Cannot define python function when using :execute. (Yasuhiro Matsumoto)

Solution: Do not recognize "def" inside "function."

Files: src/testdir/test_vim9_script.vim, src/userfunc.c

Patch 8.2.0151

Problem: Detecting a script was already sourced is unreliable.

Solution: Do not use the inode number.

Files: src/scriptfile.c, src/structs.h, src/testdir/test_vim9_script.vim

Patch 8.2.0152

Problem: Restoring ctrl_x_mode is not needed.

Solution: Remove restoring the old value, it's changed again soon.

Files: src/insexpand.c

Patch 8.2.0153

Problem: Warning shows when listing version info.

Solution: Use "-u NONE". (Ozaki Kiichi, closes #5534)

Files: .travis.yml

Patch 8.2.0154

Problem: Reallocating the list of scripts is inefficient.

Solution: Instead of using a growarray of scriptitem_T, store pointers and allocate each scriptitem_T separately. Also avoids that the growarray pointers change when sourcing a new script.

Files: src/globals.h, src/eval.c, src/evalvars.c, src/ex_docmd.c,
src/profiler.c, src/scriptfile.c, src/vim9compile.c,
src/vim9execute.c, src/vim9script.c

Patch 8.2.0155

Problem: Warnings from MinGW compiler. (John Marriott) Json test fails when building without +float feature.

Solution: Init variables. Fix Json parsing. Skip a few tests that require the +float feature.

Files: src/vim9script.c, src/vim9compile.c, src/vim9execute.c,
src/if_py_both.h, src/json.c, src/testdir/test_method.vim

Patch 8.2.0156

Problem: Various typos in source files and tests.

Solution: Fix the typos. (Emir Sari, closes #5532)

Files: Makefile, src/INSTALLvms.txt, src/Make_vms.mms, src/beval.h,
src/buffer.c, src/charset.c, src/evalvars.c, src/ex_cmds.c,
src/ex_docmd.c, src/getchar.c, src/gui.c, src/gui_mac.c,
src/gui_photon.c, src/if_perl.xs,
src/libvterm/t/11state_movecursor.test,
src/libvterm/t/41screen_unicode.test, src/mbyte.c, src/memline.c,
src/normal.c, src/ops.c, src/option.c, src/option.h,
src/os_unix.c, src/os_win32.c, src/quickfix.c, src/register.c,

src/spell.c, src/tag.c, src/term.c,
src/testdir/test_breakindent.vim, src/testdir/test_channel.vim,
src/testdir/test_cindent.vim, src/testdir/test_digraph.vim,
src/testdir/test_edit.vim, src/testdir/test_netbeans.vim,
src/testdir/test_quickfix.vim, src/testdir/test_registers.vim,
src/testdir/test_stat.vim, src/ui.c, src/xd/xd.c

Patch 8.2.0157

Problem: Vim9 script files not in list of distributed files.
Solution: Add the entries.
Files: Filelist

Patch 8.2.0158 (after 8.2.0123)

Problem: Triggering CompleteDone earlier is not backwards compatible.
(Daniel Hahler)
Solution: Add CompleteDonePre instead.
Files: src/insexpand.c, runtime/doc/autocmd.txt, src/autocmd.c,
src/vim.h, src/testdir/test_ins_complete.vim

Patch 8.2.0159

Problem: Non-materialized range() list causes problems. (Fujiwara Takuya)
Solution: Materialize the list where needed.
Files: src/testdir/test_functions.vim, src/testdir/test_python3.vim,
src/userfunc.c, src/evalfunc.c, src/highlight.c, src/evalvars.c,
src/popupmenu.c, src/insexpand.c, src/json.c, src/channel.c,
src/eval.c

Patch 8.2.0160 (after 8.2.0159)

Problem: Range test fails.
Solution: Include change in list code. (#5541)
Files: src/list.c

Patch 8.2.0161

Problem: Not recognizing .gv file as dot filetype.
Solution: Add *.gv to dot pattern. (closes #5544)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0162

Problem: Balloon test fails in the GUI.
Solution: Skip test in the GUI.
Files: src/testdir/test_functions.vim

Patch 8.2.0163

Problem: Test hangs on MS-Windows console.
Solution: use feedkeys() instead of test_feedinput(). (Ken Takata)
Files: src/testdir/test_functions.vim, src/testing.c

Patch 8.2.0164

Problem: Test_alot takes too long.
Solution: Run several tests individually.
Files: src/testdir/test_alot.vim, src/testdir/Make_all.mak

Patch 8.2.0165

Problem: Coverity warning for using NULL pointer.

Solution: Add missing "else".
Files: src/vim9compile.c

Patch 8.2.0166

Problem: Coverity warning for using uninitialized variable.
Solution: Check for failure.
Files: src/vim9execute.c

Patch 8.2.0167

Problem: Coverity warning for ignoring return value.
Solution: Check the return value and jump if failed.
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0168

Problem: Coverity warning for assigning NULL to an option.
Solution: Use empty string instead of NULL.
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0169

Problem: Coverity warning for dead code.
Solution: Check if inside try-finally.
Files: src/vim9execute.c

Patch 8.2.0170

Problem: Coverity warning for ignoring return value.
Solution: Check the return value and return if failed.
Files: src/vim9compile.c

Patch 8.2.0171

Problem: Coverity warning for using uninitialized buffer.
Solution: Check the skip flag.
Files: src/userfunc.c

Patch 8.2.0172

Problem: Coverity warning for not restoring character.
Solution: Restore the character also in case of failure.
Files: src/vim9script.c

Patch 8.2.0173

Problem: Build fails with old compiler.
Solution: Do not use anonymous unions. (John Marriott)
Files: src/vim9compile.c, src/evalvars.c, src/list.c, src/structs.h,
src/evalfunc.c, src/channel.c, src/if_mzsch.c, src/if_py_both.h

Patch 8.2.0174

Problem: Various commands not completely tested.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5551)
Files: src/testdir/test_excmod.vim, src/testdir/test_fnameescape.vim,
src/testdir/test_ga.vim, src/testdir/test_global.vim,
src/testdir/test_move.vim, src/testdir/test_options.vim,
src/testdir/test_packadd.vim, src/testdir/test_sort.vim,
src/testdir/test_substitute.vim, src/testdir/test_textformat.vim,
src/testdir/test_writefile.vim

Patch 8.2.0175

Problem: Crash when removing list element in map().
Solution: Lock the list. (closes #2652)
Files: src/testdir/test_filter_map.vim, src/list.c

Patch 8.2.0176

Problem: Generating os headers does not work for Swedish.
Solution: Set the locale to C. (Christian Brabandt, closes #5258)
Files: src/osdef.sh

Patch 8.2.0177

Problem: Memory leak in get_tags().
Solution: Free matches when finding a pseudo-tag line. (Dominique Pellé, closes #5553)
Files: src/tag.c

Patch 8.2.0178

Problem: With VTP the screen may not be restored properly.
Solution: Add another set of saved RGB values. (Nobuhiro Takasaki, closes #5548)
Files: src/os_win32.c

Patch 8.2.0179

Problem: Still a few places where range() does not work.
Solution: Fix using range() causing problems.
Files: src/terminal.c, src/testdir/test_functions.vim, src/testdir/test_popupwin.vim, src/popupwin.c, src/tag.c, src/testdir/dumps/Test_popupwin_20.dump, src/testdir/dumps/Test_popupwin_21.dump, src/testdir/dumps/Test_popup_settext_07.dump, src/globals.h

Patch 8.2.0180

Problem: Test for wrapmargin fails if terminal is not 80 columns.
Solution: Vertical split the window. (Ken Takata, closes #5554)
Files: src/testdir/test_textformat.vim

Patch 8.2.0181

Problem: Problems parsing :term arguments.
Solution: Improve parsing, fix memory leak, add tests. (Ozaki Kiichi, closes #5536)
Files: src/channel.c, src/proto/channel.pro, src/structs.h, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0182

Problem: Min() and max() materialize a range() list.
Solution: Compute the result without materializing the list. (#5541)
Files: src/evalfunc.c

Patch 8.2.0183

Problem: Tests fail when the float feature is disabled.
Solution: Skip tests that don't work without float support.
Files: src/testdir/shared.vim, src/testdir/test_blob.vim, src/testdir/test_channel.vim, src/testdir/test_cscope.vim, src/testdir/test_execute_func.vim, src/testdir/test_expr.vim,

src/testdir/test_functions.vim, src/testdir/test_lambda.vim,
src/testdir/test_listdict.vim, src/testdir/test_lua.vim,
src/testdir/test_options.vim, src/testdir/test_partial.vim,
src/testdir/test_ruby.vim, src/testdir/test_sort.vim,
src/testdir/test_timers.vim, src/testdir/test_true_false.vim,
src/testdir/test_user_func.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vimscript.vim, src/testdir/test_regexp_latin.vim,
src/testdir/test_glob2regpat.vim

Patch 8.2.0184

Problem: Blob test fails.
Solution: Check for different error when float feature is missing.
Files: src/testdir/test_blob.vim

Patch 8.2.0185

Problem: Vim9 script: cannot use "if has()" to skip lines.
Solution: Evaluate constant expression at runtime.
Files: src/vim9compile.c, src/evalfunc.c, src/proto/evalfunc.pro,
src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.0186

Problem: A couple of tests may fail when features are missing.
Solution: Check for features. (Dominique Pellé, closes #5561)
Files: src/testdir/test_functions.vim, src/testdir/test_highlight.vim

Patch 8.2.0187

Problem: Redundant code.
Solution: Remove unused assignments. (Dominique Pellé, closes #5557)
Files: src/vim9compile.c

Patch 8.2.0188

Problem: Check commands don't work well with Vim9 script.
Solution: Improve constant expression handling.
Files: src/vim9compile.c, src/testdir/check.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.0189

Problem: cd() with NULL argument crashes.
Solution: Check for NULL. (Ken Takata, closes #5558)
Files: src/testdir/test_cd.vim, src/ex_docmd.c

Patch 8.2.0190

Problem: Kotlin files are not recognized.
Solution: Detect Kotlin files. (Alkeryn, closes #5560)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0191

Problem: Cannot put a terminal in a popup window.
Solution: Allow opening a terminal in a popup window. It will always have
keyboard focus until closed.
Files: src/popupwin.c, src/proto/popupwin.pro, src/terminal.c,
src/proto/terminal.pro, src/macros.h, src/mouse.c,
src/highlight.c, src/drawline.c, src/optionstr.c, src/window.c,
src/testdir/test_terminal.vim,

src/testdir/dumps/Test_terminal_popup_1.dump,
src/testdir/dumps/Test_terminal_popup_2.dump,
src/testdir/dumps/Test_terminal_popup_3.dump

Patch 8.2.0192 (after 8.2.0191)

Problem: Build failure without +terminal feature.

Solution: Add #ifdefs.

Files: src/popupwin.c

Patch 8.2.0193 (after 8.2.0191)

Problem: Still build failure without +terminal feature.

Solution: Add more #ifdefs.

Files: src/macros.h

Patch 8.2.0194 (after 8.2.0193)

Problem: Some commands can cause problems in terminal popup.

Solution: Disallow more commands.

Files: src/macros.h, src/popupwin.c, src/proto/popupwin.pro,
src/arglist.c, src/ex_docmd.c, src/window.c,
src/testdir/test_terminal.vim

Patch 8.2.0195

Problem: Some tests fail when run in the GUI.

Solution: Make sure the window width is enough. In the GUI run terminal Vim
in the terminal, if possible.

Files: src/testdir/test_highlight.vim, src/testdir/check.vim,
src/testdir/test_terminal.vim

Patch 8.2.0196

Problem: Blocking commands for a finished job in a popup window.

Solution: Do not block commands if the job has finished. Adjust test.

Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/window.c,
src/terminal.c, src/proto/terminal.pro

Patch 8.2.0197

Problem: Some Ex commands not sufficiently tested.

Solution: Add more tests. (Yegappan Lakshmanan, closes #5565)

Files: src/testdir/test_global.vim, src/testdir/test_help.vim,
src/testdir/test_help_tagjump.vim, src/testdir/test_options.vim,
src/testdir/test_substitute.vim, src/testdir/test_textformat.vim,
src/testdir/test_writefile.vim

Patch 8.2.0198

Problem: No tests for y/n prompt.

Solution: Add tests. (Dominique Pellé, closes #5564)

Files: src/testdir/test_messages.vim

Patch 8.2.0199

Problem: Vim9 script commands not sufficiently tested.

Solution: Add more tests. Fix script-local function use.

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim,
src/userfunc.c

Patch 8.2.0200

Problem: Vim9 script commands not sufficiently tested.
Solution: Add more tests. Fix storing global variable. Make script variables work.
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/evalvars.c, src/proto/evalvars.pro, src/testdir/test_vim9_script.vim, src/misc1.c, src/proto/misc1.pro

Patch 8.2.0201

Problem: Cannot assign to an imported variable.
Solution: Make it work.
Files: src/evalvars.c, src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.0202

Problem: When '**lazyredraw**' is set the window title may not be updated.
Solution: Set "do_redraw" before entering the main loop. (Jason Franklin)
Files: src/main.c

Patch 8.2.0203

Problem: :helptags and some other functionality not tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5567)
Files: src/testdir/test_compiler.vim, src/testdir/test_ex_mode.vim, src/testdir/test_excmd.vim, src/testdir/test_filechanged.vim, src/testdir/test_help.vim, src/testdir/test_help_tagjump.vim, src/testdir/test_timers.vim, src/testdir/test_window_cmd.vim

Patch 8.2.0204

Problem: Crash when using winnr('j') in a popup window.
Solution: Do not search for neighbors in a popup window. (closes #5568)
Files: src/window.c, src/testdir/test_popupwin.vim, src/evalwindow.c

Patch 8.2.0205

Problem: Error code E899 used twice.
Solution: Use E863 for the terminal in popup error.
Files: src/popupwin.c

Patch 8.2.0206

Problem: Calling Vim9 function using default argument fails.
Solution: Give an appropriate error. (closes #5572)
Files: src/testdir/test_vim9_script.vim, src/vim9compile.c, src/vim9execute.c

Patch 8.2.0207

Problem: Crash when missing member type on list argument.
Solution: Check for invalid type. (closes #5572)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.0208

Problem: Fnamemodify() does not apply ":~" when followed by "::.".
Solution: Don't let a failing "::-" cause the ":~" to be skipped. (Yasuhiro Matsumoto, closes #5577)
Files: runtime/doc/cmdline.txt, src/filepath.c, src/testdir/test_fnamemodify.vim

Patch 8.2.0209

Problem: Function a bit far away from where it's used.
Solution: Move function close to where it's used. (Ken Takata, closes #5569)
Files: src/fileio.c, src/filepath.c

Patch 8.2.0210

Problem: Coverity complains about uninitialized field.
Solution: Initialize the field.
Files: src/vim9compile.c

Patch 8.2.0211

Problem: Test for ANSI colors fails without an "ls" command.
Solution: Use "dir". (Ken Takata, closes #5582)
Files: src/testdir/test_functions.vim

Patch 8.2.0212

Problem: Missing search/substitute pattern hardly tested.
Solution: Add test_clear_search_pat() and tests. (Yegappan Lakshmanan, closes #5579)
Files: runtime/doc/eval.txt, runtime/doc/testing.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/regexp.pro,
src/proto/search.pro, src/proto/testing.pro, src/regexp.c,
src/search.c, src/testdir/test_quickfix.vim,
src/testdir/test_search.vim, src/testdir/test_sort.vim,
src/testdir/test_substitute.vim, src/testing.c

Patch 8.2.0213

Problem: Configure does not recognize gcc 10.0 and later.
Solution: Adjust the pattern matching the version number. (Sergei Trofimovich, closes #5580)
Files: src/configure.ac, src/auto/configure

Patch 8.2.0214

Problem: A popup window with a terminal can be made hidden.
Solution: Disallow hiding a terminal popup.
Files: src/testdir/test_terminal.vim, src/popupwin.c,
src/testdir/dumps/Test_terminal_popup_4.dump

Patch 8.2.0215 (after 8.2.0208)

Problem: Wrong file name shortening. (Ingo Karkat)
Solution: Better check for path separator. (Yasuhiro Matsumoto, closes #5583, closes #5584)
Files: src/filepath.c, src/testdir/test_fnamemodify.vim

Patch 8.2.0216

Problem: Several Vim9 instructions are not tested.
Solution: Add more tests. Fix :disassemble output. Make catch with pattern work.
Files: src/testdir/test_vim9_script.vim, src/vim9execute.c,
src/vim9compile.c

Patch 8.2.0217 (after 8.2.0214)

Problem: Terminal test fails on Mac.
Solution: Add a short wait.

Files: src/testdir/test_terminal.vim

Patch 8.2.0218

Problem: Several Vim9 instructions are not tested.

Solution: Add more tests.

Files: src/testdir/test_vim9_script.vim

Patch 8.2.0219 (after 8.2.0217)

Problem: Terminal test still fails on Mac.

Solution: Skip part of the test on Mac.

Files: src/testdir/test_terminal.vim

Patch 8.2.0220

Problem: Terminal test did pass on Mac.

Solution: Remove the skip again.

Files: src/testdir/test_terminal.vim

Patch 8.2.0221

Problem: No test for Vim9 += and ..=.

Solution: Add tests.

Files: src/testdir/test_vim9_script.vim

Patch 8.2.0222

Problem: Vim9: optional function arguments don't work yet.

Solution: Implement optional function arguments.

Files: src/userfunc.c, src/vim9compile.c, src/vim9execute.c,
src/structs.h, src/testdir/test_vim9_script.vim

Patch 8.2.0223

Problem: Some instructions not yet tested.

Solution: Disassemble more instructions. Move tests to a new file. Compile call to s:function().

Files: src/testdir/test_vim9_script.vim, src/testdir/Make_all.mak,
src/testdir/test_vim9_disassemble.vim, src/vim9compile.c,
src/userfunc.c, src/proto/userfunc.pro, src/vim.h

Patch 8.2.0224

Problem: compiling :elseif not tested yet.

Solution: Add test for :elseif. Fix generating jumps.

Files: src/testdir/test_vim9_script.vim, src/vim9compile.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0225

Problem: compiling lambda not tested yet.

Solution: Add test for lambda and funcref. Drop unused instruction arg.

Files: src/testdir/test_vim9_disassemble.vim, src/vim9.h,
src/vim9execute.c

Patch 8.2.0226

Problem: Compiling for loop not tested.

Solution: Add a test. Make variable initialization work for more types.

Files: src/testdir/test_vim9_disassemble.vim, src/vim9compile.c

Patch 8.2.0227

Problem: Compiling a few instructions not tested.
Solution: Add more test cases.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0228

Problem: Configure does not recognize gcc version on BSD.
Solution: Do not use "\+" in the pattern matching the version number. (Ozaki Kiichi, closes #5590)
Files: src/configure.ac, src/auto/configure

Patch 8.2.0229

Problem: Compare instructions not tested.
Solution: Add test cases. Fix disassemble with line continuation.
Files: src/testdir/test_vim9_disassemble.vim, src/vim9execute.c, src/vim9compile.c

Patch 8.2.0230

Problem: Terminal popup test is flaky.
Solution: Increase wait time a bit.
Files: src/testdir/test_terminal.vim

Patch 8.2.0231

Problem: Silent system command may clear the screen.
Solution: Do not clear the screen in t_te.
Files: src/term.c

Patch 8.2.0232

Problem: The :compiler command causes a crash. (Daniel Steinberg)
Solution: Do not use the script index if it isn't set.
Files: src/ex_docmd.c, src/testdir/test_compiler.vim

Patch 8.2.0233

Problem: Crash when using garbagecollect() in between rand().
Solution: Redesign the rand() and srand() implementation. (Yasuhiro Matsumoto, closes #5587, closes #5588)
Files: src/evalfunc.c, src/testdir/test_random.vim, runtime/doc/testing.txt, runtime/doc/eval.txt

Patch 8.2.0234

Problem: Message test fails on SunOS.
Solution: Adjust expectation for printf "%p". (Ozaki Kiichi, closes #5595)
Files: src/message_test.c

Patch 8.2.0235

Problem: Draw error when an empty group is removed from **'statusline'**.
Solution: Do not use highlighting from a removed group.
Files: src/buffer.c, src/testdir/test_statusline.vim, src/testdir/dumps/Test_statusline_1.dump

Patch 8.2.0236

Problem: MS-Windows uninstall doesn't delete vimtutor.bat.
Solution: Change directory before deletion. (Ken Takata, closes #5603)
Files: src/uninstall.c

Patch 8.2.0237

Problem: Crash when setting **'wincolor'** on finished terminal window.
(Bakudankun)
Solution: Check that the vterm is not NULL. (Yasuhiro Matsumoto, closes #5607, closes #5610)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0238

Problem: MS-Windows: job_stop() results in exit value zero.
Solution: Call TerminateJobObject() with -1 instead of 0. (Yasuhiro Matsumoto, closes #5150, closes #5614)
Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 8.2.0239

Problem: MS-Windows: **'env'** job option does not override existing environment variables. (Tim Pope)
Solution: Set the environment variables later. (Yasuhiro Matsumoto, closes #5485, closes #5608)
Files: src/os_win32.c, src/testdir/test_channel.vim

Patch 8.2.0240

Problem: Using memory after it was freed. (Dominique Pellé)
Solution: Do not mix conversion buffer with other buffer.
Files: src/viminfo.c, src/vim.h

Patch 8.2.0241

Problem: Crash when setting **'buftype'** to "quickfix".
Solution: Check that error list is not NULL. (closes #5613)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.0242

Problem: Preview popup window test fails with long directory name. (Jakub Kądziołka)
Solution: Use "silent cd". (closes #5615)
Files: src/testdir/test_popupwin.vim

Patch 8.2.0243

Problem: Insufficient code coverage for ex_docmd.c functions.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5618)
Files: src/testdir/Make_all.mak, src/testdir/test_arglist.vim, src/testdir/test_buffer.vim, src/testdir/test_cd.vim, src/testdir/test_cmdline.vim, src/testdir/test_ex_mode.vim, src/testdir/test_excmd.vim, src/testdir/test_mapping.vim, src/testdir/test_quickfix.vim, src/testdir/test_search.vim, src/testdir/test_sort.vim, src/testdir/test_source.vim, src/testdir/test_substitute.vim, src/testdir/test_undo.vim, src/testdir/test_vimscript.vim, src/testdir/test_window_cmd.vim, src/testdir/test_writefile.vim

Patch 8.2.0244

Problem: Compiler warning in Lua interface.
Solution: Add type cast. (Ken Takata, closes #5621)
Files: src/if_lua.c

Patch 8.2.0245

Problem: MSVC: error message if the auto directory already exists.
Solution: Add "if not exists". (Ken Takata, closes #5620)
Files: src/Make_mvc.mak

Patch 8.2.0246

Problem: MSVC: deprecation warnings with Ruby.
Solution: Move _CRT_SECURE_NO_DEPRECATED to build file. (Ken Takata, closes #5622)
Files: src/Make_mvc.mak, src/if_ruby.c, src/os_win32.h, src/vim.h, src/vimio.h

Patch 8.2.0247

Problem: Misleading comment in NSIS installer script.
Solution: Negate the meaning of the comment. (Ken Takata, closes #5627)
Files: nsis/gvim.nsi

Patch 8.2.0248

Problem: MS-Windows: dealing with deprecation is too complicated.
Solution: Use io.h directly. Move _CRT_SECURE_NO_DEPRECATED to the build file. Suppress C4091 warning by setting "_WIN32_WINNT". (Ken Takata, closes #5626)
Files: src/Make_mvc.mak, src/dosinst.h, src/vim.h, src/vimio.h, src/winclip.c, Filelist

Patch 8.2.0249

Problem: MS-Windows: various warnings.
Solution: Set the charset to utf-8. Add _WIN32_WINNT and _USING_V110_SDK71_. (Ken Takata, closes #5625)
Files: src/GvimExt/Makefile, src/Make_mvc.mak

Patch 8.2.0250

Problem: test_clear_search_pat() is unused.
Solution: Remove the function. (Yegappan Lakshmanan, closes #5624)
Files: runtime/doc/eval.txt, runtime/doc/testing.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/regexp.pro, src/proto/search.pro, src/proto/testing.pro, src/regexp.c, src/search.c, src/testdir/test_writefile.vim, src/testing.c

Patch 8.2.0251

Problem: A couple of function return types can be more specific.
Solution: Use a better return type. (Ken Takata, closes #5629)
Files: src/evalfunc.c, src/globals.h

Patch 8.2.0252

Problem: Windows compiler warns for using size_t.
Solution: Change to int. (Mike Williams)
Files: src/vim9compile.c

Patch 8.2.0253

Problem: Crash when using :disassemble without argument. (Dhiraj Mishra)
Solution: Check for missing argument. (Dominique Pellé, closes #5635, closes #5637)
Files: src/vim9execute.c, src/testdir/test_vim9_disassemble.vim,

src/ex_cmds.h

Patch 8.2.0254

Problem: Compiler warning for checking size_t to be negative.
Solution: Only check for zero. (Zoltan Arpadffy)
Files: src/vim9compile.c

Patch 8.2.0255

Problem: VMS: missing files in build.
Solution: Add the files. (Zoltan Arpadffy)
Files: src/Make_vms.mms

Patch 8.2.0256

Problem: Time and timer related code is spread out.
Solution: Move time and timer related code to a new file. (Yegappan Lakshmanan, closes #5604)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/evalfunc.c, src/ex_cmds.c, src/ex_cmds2.c, src/main.c, src/memline.c, src/misc1.c, src/misc2.c, src/proto.h, src/proto/ex_cmds.pro, src/proto/ex_cmds2.pro, src/proto/main.pro, src/proto/memline.pro, src/proto/misc1.pro, src/proto/misc2.pro, src/proto/time.pro, src/time.c

Patch 8.2.0257

Problem: Cannot recognize a terminal in a popup window.
Solution: Add the win_gettype() function.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/evalwindow.c, src/proto/evalwindow.pro, src/testdir/test_cmdline.vim, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_popup_1.dump

Patch 8.2.0258

Problem: ModifyOtherKeys cannot be temporarily disabled.
Solution: Add echoraw() with an example for modifyOtherKeys.
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_functions.vim, src/testdir/dumps/Test_functions_echoraw.dump

Patch 8.2.0259

Problem: Terminal in popup test sometimes fails.
Solution: Clear the command line.
Files: src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_popup_1.dump

Patch 8.2.0260

Problem: Several lines of code are duplicated.
Solution: Move duplicated code to a function. (Yegappan Lakshmanan, closes #5330)
Files: src/option.c, src/os_unix.c, src/os_win32.c, src/proto/term.pro, src/quickfix.c, src/regexp.c, src/regexp_bt.c, src/regexp_nfa.c, src/term.c

Patch 8.2.0261

Problem: Some code not covered by tests.
Solution: Add test cases. (Yegappan Lakshmanan, closes #5645)
Files: src/testdir/test_buffer.vim, src/testdir/test_cmdline.vim,
src/testdir/test_exists.vim, src/testdir/test_filechanged.vim,
src/testdir/test_fileformat.vim, src/testdir/test_mapping.vim,
src/testdir/test_marks.vim, src/testdir/test_normal.vim,
src/testdir/test_plus_arg_edit.vim, src/testdir/test_quickfix.vim,
src/testdir/test_tabpage.vim, src/testdir/test_visual.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_writefile.vim

Patch 8.2.0262 (after 8.2.0261)

Problem: Fileformat test fails on MS-Windows.
Solution: Set fileformat of buffer.
Files: src/testdir/test_fileformat.vim

Patch 8.2.0263

Problem: A few new Vim9 messages are not localized.
Solution: Add the gettext wrapper. (Dominique Pellé, closes #5647)
Files: src/vim9compile.c, src/vim9execute.c

Patch 8.2.0264 (after 8.2.0262)

Problem: Fileformat test still fails on MS-Windows.
Solution: Set fileformat of buffer in the right place.
Files: src/testdir/test_fileformat.vim

Patch 8.2.0265

Problem: "eval" after "if 0" doesn't check for following command.
Solution: Add "eval" to list of commands that check for a following command.
(closes #5640)
Files: src/ex_docmd.c, src/testdir/test_expr.vim

Patch 8.2.0266

Problem: Terminal in popup test sometimes fails on Mac.
Solution: Add a short delay.
Files: src/testdir/test_terminal.vim

Patch 8.2.0267

Problem: No check for a following command when calling a function fails.
Solution: Also check for a following command when inside a try block.
(closes #5642)
Files: src/userfunc.c, src/testdir/test_user_func.vim

Patch 8.2.0268 (after 8.2.0267)

Problem: Trycatch test fails.
Solution: When calling function fails only check for following command, do
not give another error.
Files: src/userfunc.c

Patch 8.2.0269

Problem: Vim9: operator after list index does not work. (Yasuhiro
Matsumoto)
Solution: After indexing a list change the type to the list member type.
(closes #5651)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0270

Problem: Some code not covered by tests.
Solution: Add test cases. (Yegappan Lakshmanan, closes #5649)
Files: src/testdir/test_autocmd.vim, src/testdir/test_buffer.vim,
src/testdir/test_edit.vim, src/testdir/test_ex_mode.vim,
src/testdir/test_excmd.vim, src/testdir/test_expand.vim,
src/testdir/test_filetype.vim, src/testdir/test_findfile.vim,
src/testdir/test_join.vim, src/testdir/test_move.vim,
src/testdir/test_normal.vim, src/testdir/test_registers.vim,
src/testdir/test_source.vim, src/testdir/test_tabpage.vim,
src/testdir/test_tagjump.vim, src/testdir/test_vimscript.vim,
src/testdir/test_visual.vim, src/testdir/test_window_cmd.vim,
src/testdir/test_writefile.vim

Patch 8.2.0271

Problem: The "num64" feature is available everywhere and building without it causes problems.
Solution: Graduate the "num64" feature. (James McCoy, closes #5650)
Files: src/evalfunc.c, src/feature.h, src/message.c, src/structs.h,
src/testdir/test_expr.vim, src/testdir/test_largefile.vim,
src/testdir/test_sort.vim, src/testdir/test_vimscript.vim,
src/version.c

Patch 8.2.0272

Problem: ":helptags ALL" gives error for directories without write permission. (Matěj Cepl)
Solution: Ignore errors for ":helptags ALL". (Ken Takata, closes #5026, closes #5652)
Files: src/ex_cmds.c, src/testdir/test_help.vim

Patch 8.2.0273

Problem: MS-Windows uninstall may delete wrong batch file.
Solution: Add specific marker in the generated batch file. (Ken Takata, closes #5654)
Files: src/Make_mvc.mak, src/dosinst.c, src/dosinst.h, src/uninstall.c

Patch 8.2.0274

Problem: Hang with combination of feedkeys(), Ex mode and :global. (Yegappan Lakshmanan)
Solution: Add the pending_exmode_active flag.
Files: src/ex_docmd.c, src/globals.h, src/getchar.c,
src/testdir/test_ex_mode.vim

Patch 8.2.0275

Problem: Some Ex code not covered by tests.
Solution: Add test cases. (Yegappan Lakshmanan, closes #5659)
Files: src/testdir/test_arglist.vim, src/testdir/test_autocmd.vim,
src/testdir/test_excmd.vim, src/testdir/test_quickfix.vim,
src/testdir/test_search.vim, src/testdir/test_swap.vim,
src/testdir/test_window_cmd.vim

Patch 8.2.0276

Problem: Vim9: not allowing space before ")" in function call is too

restrictive. (Ben Jackson)
Solution: Skip space before the ")". Adjust other space checks.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0277

Problem: Vim9: not all instructions covered by tests.
Solution: Add more test cases.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0278

Problem: Channel test is flaky on Mac.
Solution: Reset variable before sending message.
Files: src/testdir/test_channel.vim

Patch 8.2.0279

Problem: Vim9: no test for deleted :def function.
Solution: Add a test. Clear uf_cleared flag when redefining a function.
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.0280

Problem: Vim9: throw in :def function not caught higher up.
Solution: Set "need_rethrow".
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.0281

Problem: Two placed signs in the same line are not combined. E.g. in the terminal debugger a breakpoint and the PC cannot be both be displayed.
Solution: Combine the sign column and line highlight attributes.
Files: src/sign.c, src/testdir/test_signs.vim,
src/testdir/dumps/Test_sign_cursor_3.dump,
src/testdir/dumps/Test_sign_cursor_4.dump

Patch 8.2.0282

Problem: Vim9: setting number option not tested.
Solution: Add more tests. Fix assigning to global variable.
Files: src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim,
src/vim9execute.c

Patch 8.2.0283

Problem: Vim9: failing to load script var not tested.
Solution: Add more tests. Fix using s: in old script.
Files: src/testdir/test_vim9_expr.vim, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0284

Problem: Vim9: assignment test fails.
Solution: Avoid duplicating "s:".
Files: src/vim9compile.c

Patch 8.2.0285

Problem: Unused error message. Cannot create s:var.
Solution: Remove the error message. Make assignment to s:var work.
Files: src/vim9compile.c, src/vim9execute.c,

src/testdir/test_vim9_script.vim

Patch 8.2.0286

Problem: Cannot use popup_close() for a terminal popup.
Solution: Allow using popup_close(). (closes #5666)
Files: src/popupwin.c, runtime/doc/popup.txt,
src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_popup_5.dump,
src/testdir/dumps/Test_terminal_popup_6.dump

Patch 8.2.0287

Problem: Vim9: return in try block not tested; catch with pattern not tested.
Solution: Add tests. Make it work.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.0288

Problem: Vim9: some float and blob operators not tested.
Solution: Add float and blob tests. Fix addition.
Files: src/testdir/test_vim9_expr.vim, src/vim9compile.c

Patch 8.2.0289

Problem: Vim9: :echo did not clear the rest of the line.
Solution: Call msg_clr_eos(). (Ken Takata, closes #5668)
Files: src/vim9execute.c

Patch 8.2.0290

Problem: Running individual test differs from all tests.
Solution: Pass on environment variables. (Yee Cheng Chin, closes #5672)
Files: src/testdir/Makefile, src/testdir/README.txt

Patch 8.2.0291

Problem: Vim9: assigning [] to list<string> doesn't work.
Solution: Use void for empty list and dict. (Ken Takata, closes #5669)
Files: src/vim9compile.c, src/globals.h, src/testdir/test_vim9_script.vim

Patch 8.2.0292

Problem: Vim9: CHECKNR and CHECKTYPE instructions not tested.
Solution: Add tests.
Files: src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0293

Problem: Various Ex commands not sufficiently tested.
Solution: Add more test cases. (Yegappan Lakshmanan, closes #5673)
Files: src/testdir/test_arglist.vim, src/testdir/test_cmdline.vim,
src/testdir/test_ex_mode.vim, src/testdir/test_excmd.vim,
src/testdir/test_expand.vim, src/testdir/test_filetype.vim,
src/testdir/test_filter_cmd.vim, src/testdir/test_global.vim,
src/testdir/test_normal.vim, src/testdir/test_plus_arg_edit.vim,
src/testdir/test_quickfix.vim, src/testdir/test_trycatch.vim,
src/testdir/test_vimscript.vim

Patch 8.2.0294

Problem: Cannot use Ex command that is also a function name.

Solution: Recognize an Ex command by a colon prefix.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
runtime/doc/vim9.txt

Patch 8.2.0295

Problem: Highlighting for :s wrong when using different separator.
Solution: Use separate argument for search direction and separator. (Rob Pilling, closes #5665)
Files: src/ex_docmd.c, src/ex_getln.c, src/gui.c, src/normal.c,
src/proto/search.pro, src/quickfix.c, src/search.c, src/spell.c,
src/tag.c, src/testdir/dumps/Test_incsearch_substitute_15.dump,
src/testdir/test_search.vim

Patch 8.2.0296

Problem: Mixing up "long long" and __int64 may cause problems. (John Marriott)
Solution: Pass varnumber_T to vim_snprintf(). Add v:numbersize.
Files: src/message.c, src/eval.c, src/fileio.c, src/json.c, src/ops.c,
src/vim.h, src/structs.h, src/evalvars.c, runtime/doc/eval.txt,
runtime/doc/various.txt, src/testdir/test_eval_stuff.vim

Patch 8.2.0297

Problem: Compiler warnings for the Ruby interface.
Solution: Undefine a few macros, fix initialization. (Ozaki Kiichi, closes #5677)
Files: src/if_ruby.c

Patch 8.2.0298

Problem: Vim9 script: cannot start command with a string constant.
Solution: Recognize expression starting with '('.
Files: src/ex_docmd.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim, runtime/doc/vim9.txt

Patch 8.2.0299

Problem: Vim9: ISN_STORE with argument not tested. Some cases in tv2bool() not tested.
Solution: Add tests. Add test_unknown() and test_void().
Files: src/testing.c, src/proto/testing.pro, src/evalfunc.c,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, runtime/doc/eval.txt,
runtime/doc/testing.txt

Patch 8.2.0300

Problem: Vim9: expression test fails without channel support.
Solution: Add has('channel') check.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.0301

Problem: Insufficient testing for exception handling and the "attention" prompt.
Solution: Add test cases. (Yegappan Lakshmanan, closes #5681)
Files: src/testdir/test_swap.vim, src/testdir/test_trycatch.vim

Patch 8.2.0302

Problem: Setting **'term'** may cause error in TermChanged autocommand.
Solution: Use aucmd_prepbuf() to switch to the buffer where the autocommand is to be executed. (closes #5682)
Files: src/term.c, src/testdir/test_aucmd.vim

Patch 8.2.0303

Problem: TermChanged test fails in the GUI.
Solution: Skip the test when running the GUI.
Files: src/testdir/test_aucmd.vim

Patch 8.2.0304

Problem: Terminal test if failing on some systems.
Solution: Wait for the job to finish. (James McCoy)
Files: src/testdir/test_terminal.vim

Patch 8.2.0305

Problem: Relativenumber test fails on some systems. (James McCoy)
Solution: Clear the command line.
Files: src/testdir/test_number.vim,
src/testdir/dumps/Test_relnr_colors_2.dump,
src/testdir/dumps/Test_relnr_colors_3.dump

Patch 8.2.0306

Problem: Vim9: :substitute(pat(repl does not work in Vim9 script.
Solution: Remember starting with a colon. (closes #5676)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.0307

Problem: Python 3 vim.eval not well tested.
Solution: Add a test. (Dominique Pellé, closes #5680)
Files: src/testdir/test_python3.vim

Patch 8.2.0308

Problem: **'showbreak'** does not work for a very long line. (John Little)
Solution: Check whether **'briopt'** contains "sbr". (Ken Takata, closes #5523, closes #5684)
Files: src/drawline.c, src/testdir/test_breakindent.vim

Patch 8.2.0309

Problem: Window-local values have confusing name.
Solution: Rename w_p_bri* to w_briopt*.
Files: src/structs.h, src/indent.c, src/drawline.c

Patch 8.2.0310

Problem: Autocmd test fails on a slow system.
Solution: Adjust the expectations. (James McCoy, closes #5685)
Files: src/testdir/test_aucmd.vim

Patch 8.2.0311

Problem: Vim9: insufficient script tests.
Solution: Add tests. Free imports when re-using a script.
Files: src/testdir/test_vim9_script.vim, src/scriptfile.c

Patch 8.2.0312

Problem: Vim9: insufficient script tests.
Solution: Add more tests. Make "import * as Name" work.
Files: src/testdir/test_vim9_script.vim, src/vim9script.c,
src/proto/vim9script.pro, src/vim9compile.c

Patch 8.2.0313

Problem: Vim9: insufficient script tests.
Solution: Add tests. Make import of alphanumeric name work.
Files: src/testdir/test_vim9_script.vim, src/vim9script.c

Patch 8.2.0314

Problem: Short name not set for terminal buffer.
Solution: Set the short name. (closes #5687)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0315

Problem: Build failure on HP-UX system.
Solution: Use LONG_LONG_MIN instead of LLONG_MIN. Add type casts for switch statement. (John Marriott)
Files: src/structs.h, src/json.c

Patch 8.2.0316

Problem: ex_getln.c code has insufficient test coverage.
Solution: Add more tests. Fix a problem. (Yegappan Lakshmanan, closes #5693)
Files: src/cmdhist.c, src/testdir/test_cmdline.vim,
src/testdir/test_functions.vim, src/testdir/test_history.vim,
src/testdir/test_menu.vim

Patch 8.2.0317

Problem: MSVC: _CRT_SECURE_NO_DEPRECATED not defined on DEBUG build.
Solution: Move where CFLAGS is updated. (Ken Takata, closes #5692)
Files: src/Make_mvc.mak

Patch 8.2.0318

Problem: Vim9: types not sufficiently tested.
Solution: Add tests with more types.
Files: src/globals.h, src/vim9compile.c,
src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.0319

Problem: File missing in distribution, comments outdated.
Solution: Correct path of README file. Update comments.
Files: Filelist, src/evalvars.c, src/register.c, src/if_python3.c

Patch 8.2.0320

Problem: No Haiku support.
Solution: Add support for Haiku. (Emir Sari, closes #5605)
Files: Filelist, runtime/doc/Makefile, runtime/doc/eval.txt,
runtime/doc/gui.txt, runtime/doc/help.txt,
runtime/doc/options.txt, runtime/doc/os_haiku.txt,
runtime/doc/starting.txt, runtime/doc/tags,
runtime/gvimrc_example.vim, runtime/vimrc_example.vim,
src/INSTALL, src/Makefile, src/auto/configure, src/configure.ac,
src/evalfunc.c, src/feature.h, src/fileio.c, src/globals.h,

src/gui.c, src/gui.h, src/gui_haiku.cc, src/gui_haiku.h,
src/mbyte.c, src/menu.c, src/misc1.c, src/mouse.c, src/option.h,
src/os_haiku.h, src/os_haiku.rdef, src/os_unix.c, src/os_unix.h,
src/osdef1.h.in, src/proto.h, src/proto/gui_haiku.pro, src/pty.c,
src/screen.c, src/structs.h, src/term.c, src/version.c, src/vim.h

Patch 8.2.0321

Problem: Vim9: ":execute" does not work yet.
Solution: Add ISN_EXECUTE. (closes #5699) Also make :echo work with more than one argument.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0322

Problem: Vim9: error checks not tested.
Solution: Add more test cases. Avoid error for function loaded later.
Files: src/vim9compile.c, src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.0323

Problem: Vim9: calling a function that is defined later is slow.
Solution: Once the function is found update the instruction so it can be called directly.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0324

Problem: Text property not updated correctly when inserting/deleting.
Solution: Use the right column when deleting. Make zero-width text properties respect start_incl and end_incl. (Axel Forsman, closes #5696, closes #5679)
Files: src/change.c, src/textprop.c, src/testdir/test_listener.vim,
src/testdir/test_textprop.vim

Patch 8.2.0325

Problem: Ex_getln.c code not covered by tests.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #5702)
Files: src/testdir/test_cmdline.vim, src/testdir/test_ex_mode.vim,
src/testdir/test_functions.vim, src/testdir/test_history.vim,
src/testdir/test_options.vim

Patch 8.2.0326

Problem: Compiler warning for using uninitialized variable. (Yegappan Lakshmanan)
Solution: Do not jump to failed but return.
Files: src/vim9execute.c

Patch 8.2.0327

Problem: Crash when opening and closing two popup terminal windows.
Solution: Check that prevwin is valid. (closes #5707)
Files: src/popupwin.c, src/testdir/test_terminal.vim

Patch 8.2.0328

Problem: No redraw when leaving terminal-normal mode in a terminal popup

window.
Solution: Redraw the popup window. (closes #5708)
Files: src/macros.h, src/vim.h, src/terminal.c, src/drawscreen.c,
src/move.c, src/popupwin.c, src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_popup_7.dump,
src/testdir/dumps/Test_terminal_popup_8.dump

Patch 8.2.0329

Problem: Popup filter converts 0x80 bytes.
Solution: Keep 0x80 bytes as-is. (Ozaki Kiichi, closes #5706)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.0330

Problem: Build error with popup window but without terminal.
Solution: Add #ifdef.
Files: src/popupwin.c

Patch 8.2.0331

Problem: Internal error when using test_void() and test_unknown().
(Dominique Pellé)
Solution: Give a normal error.
Files: src/evalfunc.c, src/testdir/test_functions.vim,
src/testdir/test_vimscript.vim

Patch 8.2.0332

Problem: Some code in ex_getln.c not covered by tests.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #5710)
Files: src/testdir/test_arabic.vim, src/testdir/test_cmdline.vim

Patch 8.2.0333

Problem: Terminal in popup test is flaky.
Solution: Make sure redraw is done before opening the popup.
Files: src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_popup_1.dump

Patch 8.2.0334

Problem: Abort called when using test_void(). (Dominique Pellé)
Solution: Only give an error, don't abort.
Files: src/message.c, src/proto/message.pro, src/evalfunc.c,
src/eval.c, src/json.c, src/testdir/test_functions.vim

Patch 8.2.0335

Problem: No completion for :disassemble.
Solution: Make completion work. Also complete script-local functions if the
name starts with "s:".
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim,
runtime/doc/vim9.txt

Patch 8.2.0336

Problem: Vim9: insufficient test coverage for compiling.
Solution: Add more tests.
Files: src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim,
src/vim9.h, src/vim9compile.c, src/vim9execute.c

Patch 8.2.0337

Problem: Build fails on a few systems.
Solution: Use vim_snprintf() instead of snprintf().
Files: src/cmdexpand.c

Patch 8.2.0338

Problem: Build failure without the channel feature.
Solution: Add #ifdef
Files: src/vim9compile.c

Patch 8.2.0339

Problem: Vim9: function return type may depend on arguments.
Solution: Instead of a fixed return type use a function to figure out the return type.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9compile.c, src/evalbuffer.c, src/proto/evalbuffer.pro, src/testdir/test_vim9_script.vim

Patch 8.2.0340

Problem: Vim9: function and partial types not tested.
Solution: Support more for partial, add tests.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.0341

Problem: Using ":for" in Vim9 script gives an error.
Solution: Pass the LET_NO_COMMAND flag. (closes #5715)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.0342

Problem: Some code in ex_getln.c not covered by tests.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5717)
Files: src/testdir/test_cmdline.vim, src/testdir/test_ex_mode.vim, src/testdir/test_history.vim, src/testdir/test_iminsert.vim

Patch 8.2.0343

Problem: Vim9: using wrong instruction, limited test coverage.
Solution: Use ISN_PUSHJOB. Add a few more tests.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.0344

Problem: ":def" not skipped properly.
Solution: Add CMD_def to list of commands the require evaluation even when not being executed.
Files: src/ex_docmd.c

Patch 8.2.0345

Problem: Compiler warning when building without the float feature.
Solution: Add #ifdef. (John Marriott)
Files: src/evalfunc.c

Patch 8.2.0346

Problem: Vim9: finding common list type not tested.
Solution: Add more tests. Fix listing function. Fix overwriting type.
Files: src/vim9compile.c, src/userfunc.c,
src/testdir/test_vim9_script.vim, src/testdir/runtest.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0347

Problem: Various code not covered by tests.
Solution: Add more test coverage. (Yegappan Lakshmanan, closes #5720)
Files: src/testdir/gen_opt_test.vim, src/testdir/test86.in,
src/testdir/test_cmdline.vim, src/testdir/test_digraph.vim,
src/testdir/test_ex_mode.vim, src/testdir/test_history.vim

Patch 8.2.0348

Problem: Vim9: not all code tested.
Solution: Add a few more tests. fix using "b:" in literal dictionary.
Files: src/testdir/test_vim9_expr.vim, src/vim9compile.c,
src/proto/vim9compile.pro, src/testdir/test_vim9_script.vim

Patch 8.2.0349

Problem: Vim9: constant expression not well tested.
Solution: Add tests for "if" with constant expression.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0350

Problem: Vim9: expression tests don't use recognized constants.
Solution: Recognize "true" and "false" as constants. Make skipping work for
assignment and expression evaluation.
Files: src/vim9compile.c

Patch 8.2.0351

Problem: Terminal in popup test is still a bit flaky.
Solution: Clear and redraw before opening the popup.
Files: src/testdir/test_terminal.vim

Patch 8.2.0352

Problem: FreeBSD: test for sourcing utf-8 is skipped.
Solution: Run the matchadd_conceal test separately to avoid that setting
'term' to "ansi" causes problems for other tests. (Ozaki Kiichi,
closes #5721)
Files: src/testdir/Make_all.mak, src/testdir/test_alot_utf8.vim,
src/testdir/test_source_utf8.vim

Patch 8.2.0353

Problem: Vim9: while loop not tested.
Solution: Add test with "while", "break" and "continue"
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0354

Problem: Python 3.9 does not define _Py_DEC_REFTOTAL. (Zdenek Dohnal)
Solution: Remove it, it was only for debugging.
Files: src/if_python3.c

Patch 8.2.0355

Problem: Vim9: str_val is confusing, it's a number
Solution: Rename to stnr_val.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c

Patch 8.2.0356

Problem: MS-Windows: feedkeys() with VIMDLL cannot handle CSI correctly.
Solution: Modify mch_inchar() to encode CSI bytes. (Ozaki Kiichi, Ken Takata, closes #5726)
Files: src/getchar.c, src/os_win32.c, src/testdir/test_popupwin.vim

Patch 8.2.0357

Problem: Cannot delete a text property matching both id and type. (Axel Forsman)
Solution: Add the "both" argument.
Files: src/textprop.c, runtime/doc/textprop.txt, src/testdir/test_textprop.vim

Patch 8.2.0358

Problem: Insufficient testing for indent.c.
Solution: Add indent tests. (Yegappan Lakshmanan, closes #5736)
Files: src/testdir/Make_all.mak, src/testdir/test_ex_mode.vim, src/testdir/test_expand_func.vim, src/testdir/test_indent.vim, src/testdir/test_lispwords.vim, src/testdir/test_smartindent.vim, src/testdir/test_vartabs.vim

Patch 8.2.0359

Problem: popup_atcursor() may hang. (Yasuhiro Matsumoto)
Solution: Take the decoration into account. (closes #5728)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.0360

Problem: Yaml files are only recognized by the file extension.
Solution: Check for a line starting with "%YAML". (Jason Franklin)
Files: runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.0361

Problem: Internal error when using "0" for a callback.
Solution: Give a normal error. (closes #5743)
Files: src/evalvars.c, src/testdir/test_timers.vim

Patch 8.2.0362

Problem: MS-Windows: channel test fails if grep is not available.
Solution: Use another command. (Ken Takata, closes #5739)
Files: src/testdir/test_channel.vim

Patch 8.2.0363

Problem: Some Normal mode commands not tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5746)
Files: src/testdir/test_cindent.vim, src/testdir/test_cmdline.vim, src/testdir/test_edit.vim, src/testdir/test_indent.vim, src/testdir/test_normal.vim, src/testdir/test_prompt_buffer.vim, src/testdir/test_virtualedit.vim, src/testdir/test_visual.vim

Patch 8.2.0364

Problem: Printf test failing on Haiku.
Solution: Make a difference between int and short. (Dominique Pellé, closes #5749)
Files: src/message.c

Patch 8.2.0365

Problem: Tag kind can't be a multibyte character. (Marcin Szamotulski)
Solution: Recognize multibyte character. (closes #5724)
Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.2.0366

Problem: Hardcopy command not tested enough.
Solution: Add tests for printing. (Dominique Pellé, closes #5748)
Files: src/testdir/test_hardcopy.vim

Patch 8.2.0367

Problem: Can use :pedit in a popup window.
Solution: Disallow it.
Files: src/ex_docmd.c, src/testdir/test_popupwin.vim

Patch 8.2.0368

Problem: Vim9: import that redefines local variable does not fail.
Solution: Check for already defined symbols.
Files: src/vim9script.c, src/proto/vim9script.pro, src/vim9compile.c, src/proto/vim9compile.pro, src/testdir/test_vim9_script.vim

Patch 8.2.0369

Problem: Various Normal mode commands not fully tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5751)
Files: src/testdir/test_arglist.vim, src/testdir/test_changelist.vim, src/testdir/test_charsearch.vim, src/testdir/test_cmdline.vim, src/testdir/test_edit.vim, src/testdir/test_ex_mode.vim, src/testdir/test_excmd.vim, src/testdir/test_gf.vim, src/testdir/test_iminsert.vim, src/testdir/test_increment.vim, src/testdir/test_marks.vim, src/testdir/test_normal.vim, src/testdir/test_prompt_buffer.vim, src/testdir/test_put.vim, src/testdir/test_registers.vim, src/testdir/test_tagjump.vim, src/testdir/test_visual.vim

Patch 8.2.0370

Problem: The typebuf_was_filled flag is sometimes not reset, which may cause a hang.
Solution: Make sure typebuf_was_filled is reset when the typeahead buffer is empty.
Files: src/edit.c, src/getchar.c,

Patch 8.2.0371

Problem: Crash with combination of terminal popup and autocmd.
Solution: Disallow closing a popup that is the current window. Add a check that the current buffer is valid. (closes #5754)
Files: src/macros.h, src/buffer.c, src/popupwin.c, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0372

Problem: Prop_find() may not find text property at start of the line.
Solution: Adjust the loop to find properties. (Axel Forsman, closes #5761, closes #5663)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.0373

Problem: Type of term_sendkeys() is unknown.
Solution: Just return zero. (closes #5762)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0374

Problem: Using wrong printf directive for jump location.
Solution: Change "%lld" to "%d". (James McCoy, closes #5773)
Files: src/vim9execute.c

Patch 8.2.0375

Problem: Coverity warning for not using return value.
Solution: Move error message to separate function.
Files: src/popupwin.c

Patch 8.2.0376

Problem: Nasty callback test fails on some systems.
Solution: Increase the sleep time.
Files: src/testdir/test_terminal.vim

Patch 8.2.0377

Problem: No CI test for a big-endian system.
Solution: Test with s390x. (James McCoy, closes #5772)
Files: .travis.yml

Patch 8.2.0378

Problem: prop_find() does not find all props.
Solution: Check being in the start line. (Axel Forsman, closes #5776)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.0379

Problem: Gcc warns for ambiguous else.
Solution: Add braces. (Dominique Pellé, closes #5778)
Files: src/textprop.c

Patch 8.2.0380

Problem: Tiny popup when creating a terminal popup without minwidth.
Solution: Use a default minimum size of 5 lines of 20 characters.
Files: src/popupwin.c, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_popup_m1.dump

Patch 8.2.0381

Problem: Using freed memory with :lvimgrep and autocommand. (extracted from POC by Dominique Pellé)
Solution: Avoid deleting a dummy buffer used in a window. (closes #5777)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.0382

Problem: Some tests fail when run under valgrind.

Solution: Increase timeouts.
Files: src/testdir/test_autocmd.vim, src/testdir/test_debugger.vim,
src/testdir/test_channel.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_popup_1.dump,
src/testdir/dumps/Test_terminal_popup_2.dump,
src/testdir/dumps/Test_terminal_popup_3.dump,
src/testdir/dumps/Test_terminal_popup_5.dump,
src/testdir/dumps/Test_terminal_popup_6.dump,
src/testdir/dumps/Test_terminal_popup_7.dump,
src/testdir/dumps/Test_terminal_popup_8.dump,
src/testdir/dumps/Test_terminal_popup_m1.dump

Patch 8.2.0383

Problem: Wrong feature check causes test not to be run.
Solution: Use CheckFunction instead of CheckFeature. (Ozaki Kiichi,
closes #5781)
Files: src/testdir/test_channel.vim

Patch 8.2.0384

Problem: Travis CI has warnings.
Solution: Avoid warnings, clean up the config. (Ozaki Kiichi, closes #5779)
Files: .travis.yml

Patch 8.2.0385

Problem: Menu functionality insufficiently tested.
Solution: Add tests. Add menu_info(). (Yegappan Lakshmanan, closes #5760)
Files: runtime/doc/eval.txt, runtime/doc/gui.txt, runtime/doc/usr_41.txt,
src/evalfunc.c, src/menu.c, src/proto/menu.pro,
src/testdir/test_menu.vim, src/testdir/test_popup.vim,
src/testdir/test_termcodes.vim

Patch 8.2.0386 (after 8.2.0385)

Problem: Part from unfinished patch got included.
Solution: Undo that part.
Files: src/evalfunc.c

Patch 8.2.0387

Problem: Error for possible NULL argument to qsort().
Solution: Don't call qsort() when there is nothing to sort. (Dominique
Pellé, closes #5780)
Files: src/spellsuggest.c

Patch 8.2.0388

Problem: Printmbcharset option not tested.
Solution: Add a test. Enable PostScript for AppVeyor build. (Dominique
Pellé, closes #5783)
Files: appveyor.yml, src/testdir/test_hardcopy.vim

Patch 8.2.0389

Problem: Delayed redraw when shifting text from Insert mode.
Solution: Use msg_attr_keep() instead of msg(). (closes #5782)
Files: src/ops.c

Patch 8.2.0390

Problem: Terminal postponed scrollbar test is flaky.
Solution: Add delay in between sending keys. Rename dump files.
Files: src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_01.dump,
src/testdir/dumps/Test_terminal_02.dump,
src/testdir/dumps/Test_terminal_03.dump,
src/testdir/dumps/Test_terminal_scrollback_1.dump,
src/testdir/dumps/Test_terminal_scrollback_2.dump,
src/testdir/dumps/Test_terminal_scrollback_3.dump

Patch 8.2.0391 (after 8.2.0377)

Problem: CI test coverage dropped.
Solution: Set \$DISPLAY also for non-GUI builds. (James McCoy, closes #5788)
Files: .travis.yml

Patch 8.2.0392

Problem: Coverity warns for using array index out of range.
Solution: Add extra "if" to avoid warning.
Files: src/menu.c

Patch 8.2.0393

Problem: Coverity warns for not using return value.
Solution: Add (void).
Files: src/popupmenu.c

Patch 8.2.0394

Problem: Coverity complains about using NULL pointer.
Solution: Use empty string when option value is NULL.
Files: src/optionstr.c

Patch 8.2.0395

Problem: Build fails with FEAT_EVAL but without FEAT_MENU.
Solution: Add #ifdef. (John Marriott)
Files: src/evalfunc.c

Patch 8.2.0396

Problem: Cmdexpand.c insufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5789)
Files: src/testdir/test_cmdline.vim, src/testdir/test_taglist.vim,
src/testdir/test_terminal.vim, src/testdir/test_usercommands.vim

Patch 8.2.0397

Problem: Delayed screen update when using undo from Insert mode.
Solution: Update w_topleft and cursor shape before sleeping. (closes #5790)
Files: src/normal.c

Patch 8.2.0398

Problem: Profile test fails when two functions take same time.
Solution: Add a short sleep in once function. (closes #5797)
Files: src/testdir/test_profile.vim

Patch 8.2.0399

Problem: Various memory leaks.

Solution: Avoid the leaks. (Ozaki Kiichi, closes #5803)
Files: src/ex_docmd.c, src/ex_getln.c, src/menu.c, src/message.c,
src/scriptfile.c, src/userfunc.c

Patch 8.2.0400

Problem: Not all tests using a terminal are in the list of flaky tests.
Solution: Introduce the test_is_flaky flag.
Files: src/testdir/runtest.vim, src/testdir/term_util.vim,
src/testdir/screendump.vim, src/testdir/test_autocmd.vim

Patch 8.2.0401

Problem: Not enough test coverage for evalvars.c.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5804)
Files: src/testdir/test_cmdline.vim, src/testdir/test_const.vim,
src/testdir/test_diffmode.vim, src/testdir/test_excmd.vim,
src/testdir/test_functions.vim, src/testdir/test_let.vim,
src/testdir/test_listdict.vim, src/testdir/test_spell.vim,
src/testdir/test_unlet.vim, src/testdir/test_user_func.vim,
src/testdir/test_vimscript.vim

Patch 8.2.0402 (after 8.2.0401)

Problem: Setting local instead of global flag.
Solution: Prepend "g:" to "test_is_flaky".
Files: src/testdir/term_util.vim, src/testdir/screendump.vim,
src/testdir/test_autocmd.vim

Patch 8.2.0403

Problem: When **'buftype'** is "nofile" there is no overwrite check.
Solution: Also check for existing file when **'buftype'** is set.
(closes #5807)
Files: src/ex_cmds.c, src/testdir/test_options.vim

Patch 8.2.0404

Problem: Writefile() error does not give a hint.
Solution: Add remark about first argument.
Files: src/filepath.c, src/testdir/test_writefile.vim

Patch 8.2.0405

Problem: MSVC: build fails with some combination of features.
Solution: Enable CHANNEL if TERMINAL is enabled. (Mike Williams)
Files: src/Make_mvc.mak

Patch 8.2.0406

Problem: FileReadCmd event not well tested.
Solution: Add a test.
Files: src/testdir/test_autocmd.vim

Patch 8.2.0407

Problem: No early check if :find and :sfind have an argument.
Solution: Add EX_NEEDARG.
Files: src/ex_cmds.h, src/testdir/test_findfile.vim,
src/testdir/test_find_complete.vim

Patch 8.2.0408

Problem: Delete() commented out for testing.
Solution: Undo commenting-out.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0409

Problem: Search test leaves file behind.
Solution: Delete the file. Also use Check commands.
Files: src/testdir/test_search.vim

Patch 8.2.0410

Problem: Channel test fails too often on slow Mac.
Solution: Increase waiting time to 10 seconds.
Files: src/testdir/test_channel.vim

Patch 8.2.0411

Problem: Mac: breakcheck is using a value from the stone ages.
Solution: Delete BREAKCHECK_SKIP from the Mac header file. (Ben Jackson)
Files: src/os_mac.h

Patch 8.2.0412

Problem: MS-Windows: cannot use vimtutor from the start menu.
Solution: Better check for writable directory. Use the right path for the executable. (Wu Yongwei, closes #5774, closes #5756)
Files: vimtutor.bat

Patch 8.2.0413

Problem: Buffer menu does not handle special buffers properly.
Solution: Keep a dictionary with buffer names to reliably keep track of entries.
Also trigger BufFilePre and BufFilePost for command-line and terminal buffers when the name changes.
Files: src/testdir/test_alot.vim, src/testdir/Make_all.mak, runtime/menu.vim, src/ex_getln.c, src/terminal.c, src/testdir/test_menu.vim

Patch 8.2.0414

Problem: Channel connect_waittime() test is flaky.
Solution: Set the test_is_flaky flag. Use test_is_flaky for more tests.
Files: src/testdir/test_channel.vim, src/testdir/test_terminal.vim, src/testdir/runtest.vim

Patch 8.2.0415

Problem: BsdL filetype is not detected.
Solution: Add an entry in the filetype list. (Daniel Kho, closes #5810)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0416

Problem: Test leaves file behind.
Solution: Delete the file.
Files: src/testdir/test_indent.vim

Patch 8.2.0417

Problem: Travis CI config can be improved.
Solution: Remove COVERAGE variable. Add load-snd-dummy script. add "-i NONE"

to avoid messages about viminfo. (Ozaki Kiichi, closes #5813)
Files: .travis.yml, ci/load-snd-dummy.sh

Patch 8.2.0418

Problem: Code in eval.c not sufficiently covered by tests.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5815)
Files: src/testdir/test_blob.vim, src/testdir/test_channel.vim,
src/testdir/test_cmdline.vim, src/testdir/test_eval_stuff.vim,
src/testdir/test_expr.vim, src/testdir/test_functions.vim,
src/testdir/test_job_fails.vim, src/testdir/test_lambda.vim,
src/testdir/test_let.vim, src/testdir/test_listdict.vim,
src/testdir/test_marks.vim, src/testdir/test_method.vim,
src/testdir/test_normal.vim, src/testdir/test_unlet.vim,
src/testdir/test_usercommands.vim, src/testdir/test_vimscript.vim,
src/testdir/test_window_cmd.vim

Patch 8.2.0419

Problem: Various memory leaks in Vim9 script code.
Solution: Fix the leaks. (Ozaki Kiichi, closes #5814)
Files: src/proto/vim9compile.pro, src/scriptfile.c, src/structs.h,
src/testdir/test_vim9_script.vim, src/vim9.h, src/vim9compile.c,
src/vim9execute.c, src/vim9script.c

Patch 8.2.0420

Problem: Vim9: cannot interrupt a loop with **CTRL-C**.
Solution: Check for **CTRL-C** once in a while. Doesn't fully work yet.
Files: src/misc1.c, src/proto/misc1.pro,
src/testdir/test_vim9_script.vim

Patch 8.2.0421

Problem: Interrupting with **CTRL-C** does not always work.
Solution: Recognize **CTRL-C** while modifyOtherKeys is set.
Files: src/ui.c, src/testdir/test_vim9_script.vim, src/evalfunc.c

Patch 8.2.0422

Problem: Crash when passing popup window to win_splitmove(). (john Devin)
Solution: Disallow moving a popup window. (closes #5816)
Files: src/testdir/test_popupwin.vim, src/evalwindow.c

Patch 8.2.0423

Problem: In some environments a few tests are expected to fail.
Solution: Add \$TEST_MAY_FAIL to list tests that should not cause make to fail.
Files: src/testdir/runtest.vim

Patch 8.2.0424

Problem: Checking for wrong return value. (Tom)
Solution: Invert the check and fix the test.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.0425

Problem: Code for modeless selection not sufficiently tested.
Solution: Add tests. Move mouse code functionality to a common script file.
(Yegappan Lakshmanan, closes #5821)

Files: src/testdir/Make_all.mak, src/testdir/gen_opt_test.vim,
src/testdir/mouse.vim, src/testdir/test_edit.vim,
src/testdir/test_global.vim, src/testdir/test_modeless.vim,
src/testdir/test_normal.vim, src/testdir/test_selectmode.vim,
src/testdir/test_termcodes.vim, src/testdir/test_visual.vim,
src/ui.c

Patch 8.2.0426

Problem: Some errors were not tested for.
Solution: Add tests. (Dominique Pellé, closes #5824)
Files: src/testdir/test_buffer.vim, src/testdir/test_options.vim,
src/testdir/test_tcl.vim, src/testdir/test_terminal.vim,
src/testdir/test_window_cmd.vim

Patch 8.2.0427

Problem: It is not possible to check for a typo in a feature name.
Solution: Add an extra argument to has().
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/check.vim,
src/testdir/test_functions.vim

Patch 8.2.0428

Problem: Buffer name may leak.
Solution: Free the buffer name before overwriting it.
Files: src/terminal.c

Patch 8.2.0429

Problem: No warning when test checks for option that never exists.
Solution: In tests check that the option can exist.
Files: src/testdir/check.vim

Patch 8.2.0430

Problem: Window creation failure not properly tested.
Solution: Improve the test. (Yegappan Lakshmanan, closes #5826)
Files: src/testdir/test_cmdline.vim, src/testdir/test_window_cmd.vim

Patch 8.2.0431

Problem: Some compilers don't support using \e for Esc. (Yegappan
Lakshmanan)
Solution: use \033 instead.
Files: src/ui.c

Patch 8.2.0432

Problem: A few tests fail in a huge terminal.
Solution: Make the tests pass. (Dominique Pellé, closes #5829)
Files: src/testdir/test_autocmd.vim, src/testdir/test_options.vim,
src/testdir/test_termcodes.vim, src/testdir/test_terminal.vim,
src/testdir/test_window_cmd.vim

Patch 8.2.0433

Problem: INT signal not properly tested.
Solution: Add a test. Also clean up some unnecessary lines. (Dominique
Pellé, closes #5828)
Files: src/testdir/test_display.vim, src/testdir/test_ex_mode.vim,
src/testdir/test_excmd.vim, src/testdir/test_messages.vim,

src/testdir/test_signals.vim

Patch 8.2.0434

Problem: MS-Windows with VTP: Normal color not working.
Solution: After changing the Normal color update the VTP console color.
(Nobuhiro Takasaki, closes #5836)
Files: src/highlight.c

Patch 8.2.0435

Problem: Channel contents might be freed twice.
Solution: Call either channel_free_channel() or channel_free(), not both.
(Nobuhiro Takasaki, closes #5835)
Files: src/channel.c

Patch 8.2.0436

Problem: No warnings for incorrect printf arguments.
Solution: Fix attribute in declaration. Fix uncovered mistakes. (Dominique Pellé, closes #5834)
Files: src/proto.h, src/eval.c, src/ops.c, src/spellfile.c,
src/vim9compile.c, src/vim9execute.c, src/viminfo.c, src/gui.c

Patch 8.2.0437

Problem: MS-Windows installer contains old stuff.
Solution: Rely on Windows NT. (Ken Takata, closes #5832)
Files: src/dosinst.c

Patch 8.2.0438

Problem: Terminal noblock test is very flaky on BSD.
Solution: Change WaitFor() to WaitForAssert() to be able to see why it failed. Add a short wait in between sending keys.
Files: src/testdir/test_terminal.vim

Patch 8.2.0439

Problem: :disassemble has minor flaws.
Solution: Format the code. Use (int) instead of (char) for %c.
(also by James McCoy, closes #5831)
Files: src/vim9execute.c

Patch 8.2.0440

Problem: Terminal noblock test is still very flaky on BSD.
Solution: Increase the waiting time.
Files: src/testdir/test_terminal.vim

Patch 8.2.0441

Problem: Terminal noblock test is still failing on BSD.
Solution: Reduce the amount of text.
Files: src/testdir/test_terminal.vim

Patch 8.2.0442

Problem: Channel contents might be used after being freed.
Solution: Reset the job channel before freeing the channel.
Files: src/channel.c

Patch 8.2.0443

Problem: Clipboard code is spread out.
Solution: Move clipboard code to its own file. (Yegappan Lakshmanan, closes #5827)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/clipboard.c, src/ops.c, src/proto.h, src/proto/clipboard.pro, src/proto/ops.pro, src/proto/register.pro, src/proto/ui.pro, src/register.c, src/ui.c

Patch 8.2.0444

Problem: Swap file test fails on some systems.
Solution: Preserve the swap file. Send NL terminated keys.
Files: src/testdir/test_swap.vim

Patch 8.2.0445

Problem: Png and xpm files not in MS-Windows zip file.
Solution: Move files to shared between Unix and Windows target.
Files: Filelist

Patch 8.2.0446

Problem: Listener with undo of deleting all lines not tested.
Solution: Add a test.
Files: src/testdir/test_listener.vim

Patch 8.2.0447

Problem: Terminal scroll tests fails on some systems.
Solution: Remove the fixed 100msec wait for Win32. Add a loop to wait until scrolling has finished. (James McCoy, closes #5842)
Files: src/testdir/test_terminal.vim

Patch 8.2.0448

Problem: Various functions not properly tested.
Solution: Add more tests, especially for failures. (Yegappan Lakshmanan, closes #5843)
Files: runtime/doc/eval.txt, src/testdir/test_blob.vim, src/testdir/test_breakindent.vim, src/testdir/test_charsearch.vim, src/testdir/test_clientserver.vim, src/testdir/test_cmdline.vim, src/testdir/test_exists.vim, src/testdir/test_expand_func.vim, src/testdir/test_expr.vim, src/testdir/test_file_perm.vim, src/testdir/test_functions.vim, src/testdir/test_gui.vim, src/testdir/test_listdict.vim, src/testdir/test_marks.vim, src/testdir/test_partial.vim, src/testdir/test_registers.vim, src/testdir/test_search.vim, src/testdir/test_spell.vim, src/testdir/test_substitute.vim, src/testdir/test_syn_attr.vim, src/testdir/test_syntax.vim, src/testdir/test_taglist.vim, src/testdir/test_utf8.vim, src/testdir/test_vartabs.vim, src/testdir/test_window_cmd.vim

Patch 8.2.0449

Problem: Vim9: crash if return type is invalid. (Yegappan Lakshmanan)
Solution: Always return some type, not NULL.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0450

Problem: Not enough testing for restricted mode and function calls.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5847)
Files: src/testdir/test_method.vim, src/testdir/test_restricted.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0451

Problem: Win32: double-width character displayed incorrectly.
Solution: First move the cursor to the first column. (Nobuhiro Takasaki,
closes #5848)
Files: src/os_win32.c

Patch 8.2.0452

Problem: channel_parse_messages() fails when called recursively.
Solution: Return for a recursive call. (closes #5835)
Files: src/channel.c

Patch 8.2.0453

Problem: Trailing space in job_start() command causes empty argument.
Solution: Ignore trailing space. (closes #5851)
Files: src/misc2.c, src/testdir/test_channel.vim

Patch 8.2.0454

Problem: Some tests fail when the system is slow.
Solution: Make the run number global, use in the test to increase the
waiting time. (closes #5841)
Files: src/testdir/runtest.vim, src/testdir/test_functions.vim

Patch 8.2.0455

Problem: Cannot set the highlight group for a specific terminal.
Solution: Add the "highlight" option to term_start(). (closes #5818)
Files: src/terminal.c, src/structs.h, src/channel.c,
src/testdir/test_terminal.vim, runtime/doc/terminal.txt,
src/testdir/dumps/Test_terminal_popup_Terminal.dump,
src/testdir/dumps/Test_terminal_popup_MyTermCol.dump

Patch 8.2.0456

Problem: Test_confirm_cmd is flaky.
Solution: Add a term_wait() call. (closes #5854)
Files: src/testdir/test_excmd.vim

Patch 8.2.0457

Problem: Test_quotestar() often fails when run under valgrind.
Solution: Wait longer for the GUI to start.
Files: src/testdir/test_quotestar.vim

Patch 8.2.0458

Problem: Missing feature check in test function.
Solution: Add check commands.
Files: src/testdir/test_excmd.vim

Patch 8.2.0459

Problem: Cannot check if a function name is correct.
Solution: Add "?funcname" to exists().
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_exists.vim,

src/testdir/check.vim

Patch 8.2.0460 (after 8.2.0459)

Problem: Build failure because of wrong feature name.
Solution: Correct feature name.
Files: src/evalfunc.c

Patch 8.2.0461

Problem: Confirm test fails on amd64 system. (Alimar Riesebieter)
Solution: Add an extra WaitForAssert(). (Dominique Pellé)
Files: src/testdir/test_excmod.vim

Patch 8.2.0462

Problem: Previewwindow test fails on some systems. (James McCoy)
Solution: Wait a bit after sending the "o". (closes #5849)
Files: src/testdir/test_popup.vim,
src/testdir/dumps/Test_popup_and_previewwindow_01.dump

Patch 8.2.0463

Problem: Build error without float and channel feature. (John Marriott)
Solution: Define return types always.
Files: src/globals.h, src/evalfunc.c

Patch 8.2.0464

Problem: Typos and other small problems.
Solution: Fix the typos. Add missing files to the distribution.
Files: Filelist, src/buffer.c, src/drawline.c, src/gui_gtk_x11.c,
src/os_unixx.h, src/proto/popupwin.pro

Patch 8.2.0465

Problem: Vim9: dead code and wrong return type.
Solution: Remove dead code. Fix return type. Add more tests.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0466 (after 8.2.0452)

Problem: Not parsing messages recursively breaks the govim plugin.
Solution: When called recursively do handle messages but do not close channels.
Files: src/channel.c

Patch 8.2.0467

Problem: Vim9: some errors are not tested
Solution: Add more tests. Fix that Vim9 script flag is not reset.
Files: src/vim9compile.c, src/scriptfile.c, src/dict.c,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0468

Problem: GUI: pixel dust with some fonts and characters.
Solution: Always redraw the character before the cursor. (Nir Lichtman,
closes #5549, closes #5856)
Files: src/gui.c, src/proto/gui.pro, src/screen.c

Patch 8.2.0469

Problem: Vim9: no error for missing] after list.

Solution: Add error message. Add more tests.
Files: src/globals.h, src/list.c, src/userfunc.c,
src/testdir/test_vim9_expr.vim, src/testdir/test_lambda.vim

Patch 8.2.0470

Problem: Test_confirm_cmd_cancel() can fail on a slow system.
Solution: Use WaitForAssert(). (Ozaki Kiichi, closes #5861)
Files: src/testdir/test_excmod.vim

Patch 8.2.0471

Problem: Missing change to compile_list().
Solution: Add error message.
Files: src/vim9compile.c

Patch 8.2.0472

Problem: Terminal highlight name is set twice, leaking memory.
Solution: Delete one.
Files: src/terminal.c

Patch 8.2.0473

Problem: Variables declared in an outer scope.
Solution: Declare variables only in the scope where they are used.
Files: src/evalvars.c

Patch 8.2.0474 (after 8.2.0403)

Problem: Cannot use :write when using a plugin with BufWriteCmd.
Solution: Reset BF_NOTEDITED after BufWriteCmd. (closes #5807)
Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 8.2.0475

Problem: Channel out_cb test still fails sometimes on Mac.
Solution: Use an even longer timeout.
Files: src/testdir/test_channel.vim

Patch 8.2.0476

Problem: Terminal nasty callback test fails sometimes.
Solution: use term_wait() instead of a sleep. (Yee Cheng Chin, closes #5865)
Files: src/testdir/test_terminal.vim

Patch 8.2.0477

Problem: Vim9: error messages not tested.
Solution: Add more tests.
Files: src/testdir/test_vim9_expr.vim, src/vim9execute.c

Patch 8.2.0478

Problem: New buffers are not added to the Buffers menu.
Solution: Turn number into string. (Yee Cheng Chin, closes #5864)
Files: runtime/menu.vim, src/testdir/test_menu.vim

Patch 8.2.0479

Problem: Unloading shared libraries on exit has no purpose.
Solution: Do not unload shared libraries on exit.
Files: src/if_lua.c, src/if_perl.xs, src/if_python.c, src/if_python3.c,
src/if_ruby.c, src/if_tcl.c

Patch 8.2.0480

Problem: Vim9: some code is not tested.
Solution: Add more tests.
Files: src/testdir/test_vim9_expr.vim, src/vim9compile.c

Patch 8.2.0481

Problem: Travis is still using trusty.
Solution: Adjust config to use bionic. (Ozaki Kiichi, closes #5868)
Files: .travis.yml, src/testdir/lsan-suppress.txt

Patch 8.2.0482

Problem: Channel and sandbox code not sufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5855)
Files: src/option.h, src/testdir/test_channel.vim,
src/testdir/test_clientserver.vim, src/testdir/test_cmdline.vim,
src/testdir/test_edit.vim, src/testdir/test_excmd.vim,
src/testdir/test_normal.vim, src/testdir/test_prompt_buffer.vim,
src/testdir/test_restricted.vim, src/testdir/test_smartindent.vim,
src/testdir/test_substitute.vim, src/testdir/test_terminal.vim,
src/testdir/test_textformat.vim, src/testdir/test_visual.vim

Patch 8.2.0483

Problem: Vim9: "let x = x + 1" does not give an error.
Solution: Hide the variable when compiling the expression.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0484

Problem: Vim9: some error messages not tested.
Solution: Add more tests.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.0485 (after 8.2.0483)

Problem: Vim9 script test fails.
Solution: Stricter condition for adding new local variable.
Files: Stricter condition for adding new local variable.

Patch 8.2.0486

Problem: Vim9: some code and error messages not tested.
Solution: Add more tests.
Files: src/vim9compile.c, src/evalvars.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0487

Problem: Vim9: compiling not sufficiently tested.
Solution: Add more tests. Fix bug with PCALL.
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0488

Problem: Vim9: Compiling can break when using a lambda inside :def.
Solution: Do not keep a pointer to the dfunc_T for longer time.
Files: src/vim9compile.c, src/vim9.h

Patch 8.2.0489

Problem: Vim9: memory leaks.
Solution: Free memory in the right place. Add hints for using asan.
Files: src/vim9compile.c, src/testdir/lsan-suppress.txt, src/Makefile

Patch 8.2.0490

Problem: Win32: VTP doesn't respect **'restorescreen'**.
Solution: Use escape codes to switch to alternate screen. (Nobuhiro Takasaki, closes #5872)
Files: src/os_win32.c

Patch 8.2.0491

Problem: Cannot recognize a `<script>` mapping using maparg().
Solution: Add the "script" key. (closes #5873)
Files: src/map.c, runtime/doc/eval.txt, src/testdir/test_maparg.vim

Patch 8.2.0492

Problem: Vim9: some error messages not tested.
Solution: Add more tests. Remove dead code. Fix uncovered bugs.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0493

Problem: Vim9: some error messages not tested.
Solution: Add more tests. Fix uncovered bugs.
Files: src/vim9compile.c, src/vim9execute.c, src/testing.c, src/eval.c, src/proto/testing.pro, src/evalfunc.c, runtime/doc/eval.txt, runtime/doc/testing.txt, src/testdir/test_vim9_script.vim

Patch 8.2.0494

Problem: Vim9: asan error.
Solution: Only get the type when there is one.
Files: src/vim9compile.c

Patch 8.2.0495

Problem: Vim9: some code not tested.
Solution: Add more tests. Support more const expressions.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0496

Problem: Vim9: disassemble test fails.
Solution: Separate test cases with recognized constant expressions.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0497

Problem: Too verbose output from the asan build in Travis.
Solution: Filter out suppression messages. (Ozaki Kiichi, closes #5874)
Files: .travis.yml

Patch 8.2.0498

Problem: Coverity complains about uninitialized field.
Solution: Initialize the whole typval_T.
Files: src/vim9compile.c

Patch 8.2.0499

Problem: Calling a lambda is slower than evaluating a string.
Solution: Make calling a lambda faster. (Ken Takata, closes #5727)
Files: src/userfunc.c

Patch 8.2.0500

Problem: Using the same loop in many places.
Solution: Define more FOR_ALL macros. (Yegappan Lakshmanan, closes #5339)
Files: src/arglist.c, src/autocmd.c, src/buffer.c, src/change.c,
src/channel.c, src/cmdexpand.c, src/diff.c, src/eval.c,
src/evalbuffer.c, src/evalfunc.c, src/evalvars.c,
src/evalwindow.c, src/ex_cmds2.c, src/filepath.c, src/globals.h,
src/gui.c, src/if_py_both.h, src/if_ruby.c, src/insexpand.c,
src/list.c, src/misc2.c, src/netbeans.c, src/popupwin.c,
src/quickfix.c, src/screen.c, src/sign.c, src/spell.c,
src/spellfile.c, src/spellsuggest.c, src/tag.c, src/terminal.c,
src/userfunc.c, src/window.c

Patch 8.2.0501

Problem: Vim9: script test fails when channel feature is missing.
Solution: Add a has() condition.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0502

Problem: Vim9: some code is not tested.
Solution: Add more tests. Fix uncovered problems.
Files: src/vim9compile.c, src/regexp.c, src/proto/regexp.pro,
src/cmdexpand.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_eval.c,
src/ex_getln.c, src/highlight.c, src/search.c, src/syntax.c,
src/tag.c, src/userfunc.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0503

Problem: Vim9: some code is not tested.
Solution: Add tests. Fix uncovered problems.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0504

Problem: Vim9: leaking scope memory when compilation fails.
Solution: Cleanup the scope list.
Files: src/vim9compile.c

Patch 8.2.0505

Problem: term_gettty() not sufficiently tested.
Solution: Add more asserts. (Dominique Pellé, closes #5877)
Files: src/testdir/test_terminal.vim

Patch 8.2.0506

Problem: Coverity complains about ignoring return value.
Solution: Add (void).
Files: src/userfunc.c

Patch 8.2.0507 (after 8.2.0472)

Problem: Getbufvar() may get the wrong dictionary. (David le Blanc)
Solution: Check for empty name. (closes #5878)
Files: src/evalvars.c, src/testdir/test_functions.vim

Patch 8.2.0508

Problem: Vim9: func and partial types not done yet
Solution: Fill in details about func declaration, drop a separate partial declaration.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/globals.h, src/structs.h, src/evalfunc.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.0509

Problem: various code is not properly tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5871)
Files: src/main.c, src/testdir/check.vim, src/testdir/shared.vim, src/testdir/term_util.vim, src/testdir/test_clientserver.vim, src/testdir/test_ex_mode.vim, src/testdir/test_expand.vim, src/testdir/test_functions.vim, src/testdir/test_options.vim, src/testdir/test_startup.vim, src/testdir/test_textformat.vim, src/testdir/test_trycatch.vim, src/testdir/test_viminfo.vim

Patch 8.2.0510

Problem: Coverity complains about using uninitialized variable.
Solution: Assign a value to "scol". Move code inside NULL check.
Files: src/beval.c, src/popupwin.c

Patch 8.2.0511

Problem: Cscope code not fully tested.
Solution: Add more test cases. (Dominique Pellé, closes #5886)
Files: src/testdir/test_cscope.vim

Patch 8.2.0512

Problem: Vim9: no optional arguments in func type.
Solution: Check for question mark after type. Find function reference without function().
Files: src/vim9compile.c, src/vim9execute.c, src/structs.h, src/globals.h, src/vim.h, src/vim9.h, src/userfunc.c, src/testdir/Make_all.mak, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.0513

Problem: Reading past allocated memory when using varargs.
Solution: Fix copying function argument types.
Files: src/vim9compile.c

Patch 8.2.0514

Problem: Several global functions are used in only one file.
Solution: Make the functions static. (Yegappan Lakshmanan, closes #5884)
Files: src/drawscreen.c, src/evalvars.c, src/getchar.c, src/list.c, src/proto/drawscreen.pro, src/proto/evalvars.pro, src/proto/getchar.pro, src/proto/list.pro, src/proto/version.pro,

src/version.c

Patch 8.2.0515

Problem: Some compilers cannot add to "void *".
Solution: Cast to "char *".
Files: src/vim9compile.c

Patch 8.2.0516

Problem: Client-server code is spread out.
Solution: Move client-server code to a new file. (Yegappan Lakshmanan, closes #5885)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/clientserver.c, src/evalfunc.c, src/main.c, src/proto.h, src/proto/clientserver.pro, src/proto/main.pro

Patch 8.2.0517

Problem: Vim9: cannot separate "func" and "func(): void".
Solution: Use VAR_ANY for "any" and VAR_UNKNOWN for "no type".
Files: src/structs.h, src/globals.h, src/eval.c, src/evalfunc.c, src/evalvars.c, src/testing.c, src/vim9compile.c, src/vim9execute.c, src/viminfo.c, src/if_py_both.h, src/json.c, src/testdir/test_vim9_func.vim

Patch 8.2.0518

Problem: A terminal falls back to setting \$TERM to "xterm".
Solution: Use "xterm-color" if more than 16 colors are supported and "xterm-256color" if at least 256 colors are supported. (closes #5887)
Files: src/os_unix.c

Patch 8.2.0519

Problem: Vim9: return type not properly checked.
Solution: Check type properly, also at runtime.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0520

Problem: Tests are not listed in sorted order.
Solution: Move test_ex_mode. (Doug Richardson, closes #5889)
Files: src/testdir/Make_all.mak

Patch 8.2.0521

Problem: Crash when reading a blob fails.
Solution: Avoid keeping a pointer to a freed blob object. (Dominique Pellé, closes #5890) Adjust error messages.
Files: src/filepath.c, src/testdir/test_blob.vim

Patch 8.2.0522

Problem: Several errors are not tested for.
Solution: Add tests. (Yegappan Lakshmanan, closes #5892)
Files: src/testdir/test_autocmd.vim, src/testdir/test_clientserver.vim, src/testdir/test_digraph.vim, src/testdir/test_expand.vim, src/testdir/test_expr.vim, src/testdir/test_functions.vim, src/testdir/test_gui.vim, src/testdir/test_highlight.vim,

src/testdir/test_ins_complete.vim, src/testdir/test_lambda.vim,
src/testdir/test_listdict.vim, src/testdir/test_normal.vim,
src/testdir/test_options.vim, src/testdir/test_preview.vim,
src/testdir/test_user_func.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_viminfo.vim,
src/testdir/test_vimscript.vim, src/testdir/test_window_cmd.vim

Patch 8.2.0523

Problem: Loops are repeated.
Solution: Use FOR_ALL_ macros. (Yegappan Lakshmanan, closes #5882)
Files: src/buffer.c, src/drawscreen.c, src/evalfunc.c, src/evalwindow.c,
src/globals.h, src/gui_athena.c, src/gui_gtk.c, src/gui_motif.c,
src/gui_w32.c, src/list.c, src/menu.c, src/popupmenu.c,
src/popupwin.c, src/quickfix.c, src/syntax.c, src/time.c,
src/userfunc.c, src/vim9compile.c

Patch 8.2.0524

Problem: Win32: searching for file matches is slow.
Solution: Instead of making another round to find any short filename, check
for the short name right away. Avoid using an ordinary file like a
directory. (Nir Lichtman, closes #5883)
Files: src/filepath.c

Patch 8.2.0525 (after 8.2.0524)

Problem: Win32: typo in assignment and misplaced paren.
Solution: Fix the syntax.
Files: src/filepath.c

Patch 8.2.0526

Problem: Gcc 9 complains about empty statement.
Solution: Add {}. (Dominique Pellé, closes #5894)
Files: src/evalfunc.c

Patch 8.2.0527

Problem: Vim9: function types insufficiently tested.
Solution: Add more tests. Fix white space check. Add "test_vim9" target.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim, src/Makefile,
src/testdir/Makefile, src/testdir/Make_all.mak

Patch 8.2.0528

Problem: Vim9: function arguments insufficiently tested.
Solution: Check types. Add more tests. Fix function with varargs only.
Files: src/vim9compile.c, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.0529

Problem: Vim9: function argument with default not checked.
Solution: Check type of argument with default value.
Files: src/vim9compile.c, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.0530

Problem: Test crashes on s390. (James McCoy)
Solution: Explicitly define an 8 big signed type. (closes #5897)
Files: src/structs.h

Patch 8.2.0531

Problem: Various errors not tested.
Solution: Add tests. (Yegappan Lakshmanan, closes #5895)
Files: src/testdir/test_search.vim, src/testdir/test_source.vim,
src/testdir/test_syntax.vim, src/testdir/test_user_func.vim,
src/testdir/test_vimscript.vim

Patch 8.2.0532

Problem: Cannot use simplify() as a method.
Solution: Add FEARG_1. (closes #5896)
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_functions.vim

Patch 8.2.0533

Problem: Tests using term_wait() can still be flaky.
Solution: Increase the wait time when rerunning a test. (James McCoy,
closes #5899) Halve the initial times to make tests run faster
when there is no rerun.
Files: src/testdir/term_util.vim, src/testdir/test_arglist.vim,
src/testdir/test_autocmd.vim, src/testdir/test_balloon.vim,
src/testdir/test_buflines.vim, src/testdir/test_channel.vim,
src/testdir/test_cmdline.vim, src/testdir/test_conceal.vim,
src/testdir/test_cursorline.vim, src/testdir/test_debugger.vim,
src/testdir/test_diffmode.vim, src/testdir/test_display.vim,
src/testdir/test_functions.vim, src/testdir/test_highlight.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_mapping.vim,
src/testdir/test_match.vim, src/testdir/test_matchadd_conceal.vim,
src/testdir/test_messages.vim, src/testdir/test_number.vim,
src/testdir/test_popup.vim, src/testdir/test_popupwin.vim,
src/testdir/test_profile.vim, src/testdir/test_search.vim,
src/testdir/test_search_stat.vim, src/testdir/test_startup.vim,
src/testdir/test_startup_utf8.vim,
src/testdir/test_statusline.vim, src/testdir/test_suspend.vim,
src/testdir/test_swap.vim, src/testdir/test_tagjump.vim,
src/testdir/test_terminal.vim, src/testdir/test_terminal_fail.vim,
src/testdir/test_timers.vim, src/testdir/test_vimscript.vim

Patch 8.2.0534

Problem: Client-server test fails under valgrind.
Solution: Use WaitForAssert().
Files: src/testdir/test_clientserver.vim

Patch 8.2.0535

Problem: Regexp patterns not fully tested.
Solution: Add more regexp tests and others. (Yegappan Lakshmanan,
closes #5901)
Files: src/testdir/test_marks.vim, src/testdir/test_options.vim,
src/testdir/test_regexp_latin.vim, src/testdir/test_search.vim

Patch 8.2.0536

Problem: Vim9: some compilation code not tested.
Solution: Add more test cases.
Files: src/evalvars.c, src/proto/evalvars.pro, src/vim9compile.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.0537

Problem: Vim9: no check for sandbox when setting v:var.
Solution: Check for sandbox.
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.0538

Problem: Vim9: VAR_PARTIAL is not used during compilation.
Solution: Remove VAR_PARTIAL.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c

Patch 8.2.0539

Problem: Comparing two NULL list fails.
Solution: Change the order of comparing two lists.
Files: src/list.c, src/testdir/test_assert.vim

Patch 8.2.0540

Problem: Regexp and other code not tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5904)
Files: src/testdir/test_backspace_opt.vim, src/testdir/test_expr.vim,
src/testdir/test_increment.vim, src/testdir/test_normal.vim,
src/testdir/test_options.vim, src/testdir/test_regexp_latin.vim,
src/testdir/test_search.vim, src/testdir/test_substitute.vim,
src/testdir/test_terminal.vim, src/testdir/test_virtualedit.vim

Patch 8.2.0541

Problem: Travis CI does not give compiler warnings.
Solution: Add flags for warnings. Fix uncovered problems. (Ozaki Kiichi,
closes #5898)
Files: .travis.yml, ci/config.mk.clang.sed, ci/config.mk.gcc.sed,
ci/config.mk.sed, src/if_perl.xs, src/if_ruby.c,
src/libvterm/t/harness.c

Patch 8.2.0542

Problem: No test for E386.
Solution: Add a test. (Dominique Pellé, closes #5911)
Files: src/testdir/test_search.vim

Patch 8.2.0543

Problem: Vim9: function with varargs does not work properly.
Solution: Improve function type spec and add tests. Fix bugs.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/vim9execute.c,
src/structs.h, src/testdir/test_vim9_func.vim

Patch 8.2.0544

Problem: Memory leak in search test.
Solution: Free msgbuf. (Dominique Pellé, closes #5912)
Files: src/search.c

Patch 8.2.0545

Problem: Unused arguments ignored in non-standard way.
Solution: Add UNUSED instead of (void).
Files: src/libvterm/t/harness.c

Patch 8.2.0546

Problem: Vim9: varargs implementation is inefficient.
Solution: Create list without moving the arguments.
Files: src/vim9compile.c, src/vim9execute.c

Patch 8.2.0547

Problem: Win32: restoring screen not always done right.
Solution: Use a more appropriate method. (Nobuhiro Takasaki, closes #5909)
Files: src/os_win32.c

Patch 8.2.0548

Problem: Vim9: not all possible func type errors tested.
Solution: Add more tests.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0549

Problem: User systemd files not recognized.
Solution: Add filetype patterns. (Kevin Locke, closes #5914)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0550

Problem: Some changes in the libvterm upstream code.
Solution: Include some changes.
Files: src/libvterm/t/harness.c

Patch 8.2.0551

Problem: Not all code for options is tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5913)
Files: src/testdir/test_options.vim, src/testdir/test_python3.vim,
src/testdir/test_undo.vim, src/testdir/test_vimscript.vim

Patch 8.2.0552

Problem: Vim9: some errors not covered by tests.
Solution: Add more tests. Check Funcref argument types.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0553 (after 8.2.0550)

Problem: Error for unused argument.
Solution: Add UNUSED.
Files: src/libvterm/t/harness.c

Patch 8.2.0554

Problem: The GUI doesn't set t_Co.
Solution: In the GUI set t_Co to 256 * 256 * 256. (closes #5903)
Files: src/term.c, src/proto/term.pro, src/gui.c,
src/testdir/test_gui.vim

Patch 8.2.0555

Problem: Vim9: line continuation is not always needed.
Solution: Recognize continuation lines automatically in list and dict.
Files: runtime/doc/vim9.txt, src/vim9compile.c,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0556

Problem: Vim9: memory leak when finding common type.
Solution: Store allocated memory in type growarray.
Files: src/vim9compile.c

Patch 8.2.0557

Problem: No IPv6 support for channels.
Solution: Add IPv6 support. (Ozaki Kiichi, closes #5893)
Files: .travis.yml, runtime/doc/channel.txt, runtime/doc/various.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/auto/configure, src/channel.c, src/config.h.in, src/configure.ac, src/evalfunc.c, src/proto/channel.pro, src/testdir/check.vim, src/testdir/runtest.vim, src/testdir/test_cdo.vim, src/testdir/test_channel.py, src/testdir/test_channel.vim, src/testdir/test_channel_6.py, src/testdir/test_escaped_glob.vim, src/testdir/test_getcwd.vim, src/testdir/test_hide.vim

Patch 8.2.0558

Problem: Vim9: dict code not covered by tests.
Solution: Remove dead code, adjust test case.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0559

Problem: Clearing a struct is verbose.
Solution: Define and use CLEAR_FIELD() and CLEAR_POINTER().
Files: src/vim.h, src/blowfish.c, src/channel.c, src/charset.c, src/clipboard.c, src/diff.c, src/eval.c, src/evalfunc.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c, src/findfile.c, src/gui_gtk_f.c, src/gui_mac.c, src/gui_motif.c, src/gui_w32.c, src/gui_x11.c, src/hardcopy.c, src/hashtab.c, src/highlight.c, src/if_mzsch.c, src/insexpand.c, src/kword_test.c, src/list.c, src/main.c, src/map.c, src/memfile.c, src/message_test.c, src/misc1.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_mswin.c, src/os_win32.c, src/popupmenu.c, src/quickfix.c, src/regexp.c, src/regexp_bt.c, src/regexp_nfa.c, src/search.c, src/sign.c, src/spell.c, src/spellfile.c, src/spellsuggest.c, src/syntax.c, src/tag.c, src/terminal.c, src/time.c, src/undo.c, src/userfunc.c, src/vim9compile.c, src/vim9execute.c, src/if_py_both.h

Patch 8.2.0560

Problem: Compiler warning in tiny build.
Solution: Move declaration inside #ifdef. (Dominique Pellé, closes #5915)
Files: src/ex_docmd.c

Patch 8.2.0561

Problem: Vim9: cannot split function call in multiple lines.
Solution: Find more arguments in following lines.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0562

Problem: Vim9: cannot split an expression into multiple lines.
Solution: Continue in next line after an operator.
Files: runtime/doc/vim9.txt, src/macros.h, src/vim9compile.c,

src/testdir/test_vim9_expr.vim

Patch 8.2.0563

Problem: Vim9: cannot split a function line.
Solution: Continue in next line so long as the function isn't done.
Files: runtime/doc/vim9.txt, src/userfunc.c, src/proto/userfunc.pro,
src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0564

Problem: Vim9: calling a def function from non-vim9 may fail.
Solution: Convert varargs to a list.
Files: src/testdir/test_vim9_func.vim, src/vim9execute.c

Patch 8.2.0565

Problem: Vim9: tests contain superfluous line continuation.
Solution: Remove line continuation no longer needed. Skip empty lines.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0566

Problem: Vim9: variable can be used uninitialized.
Solution: Jump to after where variable is used.
Files: src/vim9execute.c

Patch 8.2.0567

Problem: Vim9: cannot put comments halfway expressions.
Solution: Support # comments in many places.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/userfunc.c,
src/ex_docmd.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0568

Problem: The man filetype plugin overwrites the unnamed register.
Solution: Use the black hole register. (Jason Franklin)
Files: runtime/ftplugin/man.vim, src/testdir/test_man.vim

Patch 8.2.0569

Problem: Build failure with tiny version.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.0570

Problem: Vim9: no error when omitting type from argument.
Solution: Enforce specifying argument types.
Files: src/userfunc.c, src/ex_eval.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0571

Problem: Double free when passing invalid argument to job_start().
Solution: Clear the argument when freed. (Masato Nishihata, closes #5926)
Files: src/misc2.c, src/testdir/test_channel.vim

Patch 8.2.0572 (after 8.2.0571)

Problem: Using two lines for free and reset.
Solution: Use VIM_CLEAR() instead. (Yegappan Lakshmanan)
Files: src/misc2.c

Patch 8.2.0573

Problem: using :version twice leaks memory
Solution: Only initialize variables once. (Dominique Pellé, closes #5917)
Files: src/testdir/Make_all.mak, src/testdir/test_alot.vim,
src/testdir/test_version.vim, src/version.c, src/globals.h

Patch 8.2.0574

Problem: Ipv6 feature not shown in :version output.
Solution: Add ipv6 in :version output. (Ozaki Kiichi, closes #5924)
Files: runtime/doc/eval.txt, src/version.c

Patch 8.2.0575

Problem: :digraph! not tested.
Solution: Add a test. (Dominique Pellé, closes #5925)
Files: src/testdir/test_digraph.vim

Patch 8.2.0576

Problem: Some errors are not covered by tests.
Solution: Add a few more tests. (Dominique Pellé, closes #5920)
Files: src/testdir/test_buffer.vim, src/testdir/test_digraph.vim,
src/testdir/test_expr.vim, src/testdir/test_messages.vim

Patch 8.2.0577

Problem: Not all modifiers supported for :options.
Solution: Use all cmdmod.split flags. (closes #4401)
Files: src/usercmd.c, src/proto/usercmd.pro, src/scriptfile.c,
src/testdir/test_options.vim, src/testdir/test_usercommands.vim

Patch 8.2.0578

Problem: Here doc for interfaces does not support "trim".
Solution: Update the script heredoc support to be same as the :let command.
(Yegappan Lakshmanan, closes #5916)
Files: runtime/doc/if_lua.txt, runtime/doc/if_mzsch.txt,
runtime/doc/if_perl.txt, runtime/doc/if_pyth.txt,
runtime/doc/if_ruby.txt, runtime/doc/if_tcl.txt, src/evalvars.c,
src/ex_getln.c, src/proto/evalvars.pro, src/testdir/test86.in,
src/testdir/test87.in, src/testdir/test_lua.vim,
src/testdir/test_perl.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_pyx2.vim,
src/testdir/test_pyx3.vim, src/testdir/test_ruby.vim,
src/testdir/test_tcl.vim, src/userfunc.c, src/vim9compile.c

Patch 8.2.0579

Problem: Coverity warns for unused value.
Solution: Change order and use "else if".
Files: src/os_unix.c

Patch 8.2.0580

Problem: Window size wrong if 'ea' is off and 'splitright' is on and
splitting then closing a window.

Solution: Put abandoned window space in the right place. (Mark Waggoner)
Files: src/testdir/test_winbuf_close.vim, src/window.c

Patch 8.2.0581 (after 8.2.0547)

Problem: Win32 console: the cursor position is always top-left.
Solution: Revert the patch for restoring screen.
Files: src/os_win32.c

Patch 8.2.0582

Problem: Color ramp test does not show text colors.
Solution: Add a row of 16 text colors and 16 bold text colors.
Files: src/testdir/color_ramp.vim

Patch 8.2.0583

Problem: Vim9: # comment not recognized in :def function.
Solution: Recognize and skip # comment.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0584

Problem: Viminfo file uses obsolete function file_readable().
Solution: Use filereadable(). (closes #5934)
Files: src/session.c

Patch 8.2.0585

Problem: Vim9: # comment not recognized after :vim9script.
Solution: Check script type. Make comment after ":echo" work. And in several other places.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/eval.c, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0586

Problem: Vim9: # comment not sufficiently tested
Solution: Check for preceding white space.
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.0587

Problem: Compiler warning for unused variable.
Solution: Add UNUSED.
Files: src/ex_docmd.c

Patch 8.2.0588

Problem: Putty does not use "sgr" **'ttermouse'** by default.
Solution: Make "sgr" the default for Putty. (Christian Brabandt, closes #5942)
Files: src/term.c

Patch 8.2.0589

Problem: .bsd file type not recognized.
Solution: Recognize .bsd as BSDL. (Daniel Kho, closes #5945)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0590

Problem: No **'backspace'** value allows ignoring the insertion point.
Solution: Add the "nostop" and 3 values. (Christian Brabandt, closes #5940)

Files: runtime/doc/options.txt, src/edit.c, src/option.c, src/option.h,
src/optionstr.c, src/testdir/gen_opt_test.vim,
src/testdir/test_backspace_opt.vim

Patch 8.2.0591

Problem: MS-Windows: should always support IPv6
Solution: Add build flag. (Ozaki Kiichi, closes #5944)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.2.0592

Problem: MS-Windows with VTP: cursor is not made invisible.
Solution: Output the code to make the cursor visible or invisible. (Nobuhiro
Takasaki, closes #5941)
Files: src/os_win32.c

Patch 8.2.0593

Problem: Finding a user command is not optimal.
Solution: Start further down in the list of commands.
Files: src/ex_cmds.h, src/ex_docmd.c

Patch 8.2.0594

Problem: MS-Windows: cannot build with WINVER set to 0x0501.
Solution: Only use inet_ntop() when available. (Ozaki Kiichi, closes #5946)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/auto/configure,
src/channel.c, src/config.h.in, src/configure.ac

Patch 8.2.0595

Problem: Vim9: not all commands using ends_excmd() tested.
Solution: Find # comment after regular commands. Add more tests. Report
error for where it was caused.
Files: src/ex_docmd.c, src/vim9compile.c, src/vim9execute.c, src/usercmd.c,
src/evalfunc.c, src/userfunc.c, src/proto/userfunc.pro,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0596

Problem: Crash in test49.
Solution: Check the right pointer.
Files: src/userfunc.c, src/testdir/test_eval.ok

Patch 8.2.0597

Problem: Test_eval is old style.
Solution: Change some tests to a new style test.
Files: src/testdir/test_eval.in, src/testdir/test_eval.ok,
src/testdir/test_eval_stuff.vim

Patch 8.2.0598

Problem: Test_eval_stuff fails in normal terminal.
Solution: Close the new window.
Files: src/testdir/test_eval_stuff.vim

Patch 8.2.0599

Problem: Netbeans interface insufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5921)

Files: runtime/doc/netbeans.txt, src/netbeans.c, src/os_win32.c,
src/testdir/runtest.vim, src/testdir/test_netbeans.py,
src/testdir/test_netbeans.vim

Patch 8.2.0600

Problem: Vim9: cannot read or write w:, t: and b: variables.
Solution: Implement load and store for w:, t: and b: variables.
(closes #5950)

Files: src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim,
src/vim9.h, src/vim9compile.c, src/vim9execute.c

Patch 8.2.0601

Problem: Vim9: :unlet is not compiled.
Solution: Implement :unlet instruction and check for errors.

Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9.h,
src/vim9execute.c, src/evalvars.c, src/proto/evalvars.pro,
src/eval.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0602

Problem: :unlet \$VAR does not work properly.
Solution: Make ":lockvar \$VAR" fail. Check the "skip" flag.

Files: src/evalvars.c, src/globals.h, src/testdir/test_vimscript.vim

Patch 8.2.0603

Problem: Configure does not detect moonjit.
Solution: Add check for moonjit. (Shlomi Fish, closes #5947)

Files: src/configure.ac, src/auto/configure

Patch 8.2.0604

Problem: :startinsert in a terminal window used later.
Solution: Ignore :startinsert in a terminal window. (closes #5952)

Files: src/ex_docmd.c, src/testdir/test_terminal.vim

Patch 8.2.0605

Problem: Vim9: cannot unlet an environment variable.
Solution: Implement unlet for \$VAR.

Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0606

Problem: Several syntax HL errors not checked.
Solution: Add tests. (Yegappan Lakshmanan, closes #5954)

Files: src/testdir/test_syntax.vim

Patch 8.2.0607

Problem: Gcc warns for using uninitialized variable. (John Marriott)
Solution: Set name_end also for environment variables.

Files: src/evalvars.c

Patch 8.2.0608

Problem: Warning from clang when building message test.

Solution: Use a void pointer. (Dominique Pellé, closes #5958)
Files: src/message_test.c

Patch 8.2.0609

Problem: Configure does not detect moonjit correctly.
Solution: Double the brackets. (Ozaki Kiichi)
Files: src/configure.ac, src/auto/configure

Patch 8.2.0610

Problem: Some tests are still old style.
Solution: Convert to new style tests. (Yegappan Lakshmanan, closes #5957)
Files: src/testdir/test_blob.vim, src/testdir/test_cursor_func.vim,
src/testdir/test_eval.in, src/testdir/test_eval.ok,
src/testdir/test_eval_func.vim, src/testdir/test_eval_stuff.vim,
src/testdir/test_expr.vim, src/testdir/test_filter_map.vim,
src/testdir/test_functions.vim, src/testdir/test_listdict.vim,
src/testdir/test_sort.vim, src/testdir/test_syntax.vim,
src/testdir/test_utf8.vim, src/testdir/test_vimscript.vim

Patch 8.2.0611

Problem: Vim9: no check for space before #comment.
Solution: Add space checks.
Files: src/eval.c, src/evalvars.c, src/ex_docmd.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0612

Problem: Vim9: no check for space before #comment.
Solution: Add space checks.
Files: src/ex_eval.c, src/ex_cmds.c, src/regexp.c, src/proto/regexp.pro,
src/gui.c, src/highlight.c, src/testdir/test_vim9_script.vim,
src/testdir/test_sort.vim

Patch 8.2.0613

Problem: Vim9: no check for space before #comment.
Solution: Add space checks.
Files: src/highlight.c, src/menu.c, src/syntax.c,
src/testdir/test_vim9_script.vim,
runtime/lang/menu_de_de.latin1.vim

Patch 8.2.0614

Problem: Get ml_get error when deleting a line in '**complefunc**'. (Yegappan Lakshmanan)
Solution: Lock the text while evaluating '**complefunc**'.
Files: src/insexpand.c, src/globals.h, src/edit.c, src/ex_getln.c,
src/undo.c, src/testdir/test_edit.vim, src/testdir/test_excmd.vim,
src/testdir/test_gf.vim, src/testdir/test_popup.vim,
src/testdir/test_ex_mode.vim, runtime/doc/insert.txt

Patch 8.2.0615

Problem: Regexp benchmark test is old style.
Solution: Make it a new style test. Fix using a NULL list. Add more tests. (Yegappan Lakshmanan, closes #5963)
Files: src/evalbuffer.c, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Makefile,

src/testdir/bench_re_freeze.in, src/testdir/bench_re_freeze.vim,
src/testdir/test_autocmd.vim, src/testdir/test_bench_regexp.vim,
src/testdir/test_blob.vim, src/testdir/test_bufline.vim,
src/testdir/test_channel.vim, src/testdir/test_cmdline.vim,
src/testdir/test_functions.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_popupwin.vim, src/testdir/test_prompt_buffer.vim,
src/testdir/test_tagjump.vim, src/testdir/test_window_cmd.vim

Patch 8.2.0616

Problem: Build error when disabling the diff feature.
Solution: Move parenthesis outside of #ifdef. (Tom Ryder)
Files: src/drawline.c

Patch 8.2.0617

Problem: New error check triggers in Swedish menu.
Solution: Insert backslash. (Mats Tegner, closes #5966)
Files: runtime/lang/menu_sv_se.latin1.vim

Patch 8.2.0618

Problem: Echoing a null list results in no output. (Yegappan Lakshmanan)
Solution: Return "[]" instead of NULL in echo_string_core().
Files: src/eval.c, src/testdir/test_messages.vim

Patch 8.2.0619

Problem: Null dict is not handled like an empty dict.
Solution: Fix the code and add tests. (Yegappan Lakshmanan, closes #5968)
Files: src/dict.c, src/eval.c, src/testdir/test_blob.vim,
src/testdir/test_expr.vim, src/testdir/test_filter_map.vim,
src/testdir/test_let.vim, src/testdir/test_listdict.vim,
src/testdir/test_search.vim, src/testdir/test_unlet.vim,
src/testdir/test_usercommands.vim, src/testdir/test_vimscript.vim

Patch 8.2.0620

Problem: Error in menu translations.
Solution: Insert a backslash before a space.
Files: runtime/lang/menu_it_it.latin1.vim,
runtime/lang/menu_chinese_gb.936.vim

Patch 8.2.0621

Problem: After running tests asan files may remain.
Solution: Clean up asan files with "make testclean".
Files: src/testdir/Makefile, src/Makefile

Patch 8.2.0622

Problem: Haiku: GUI does not compile.
Solution: Various fixes. (Emir Sari, closes #5961)
Files: Filelist, README.md, READMEdir/README_haiku.txt,
runtime/doc/os_haiku.txt, src/Makefile, src/beval.h,
src/gui_haiku.cc, src/proto/gui_haiku.pro

Patch 8.2.0623

Problem: Typo in test comment. (Christ van Willegen)
Solution: Avoid mixing up a data structure with a body part.
Files: src/testdir/test_listdict.vim

Patch 8.2.0624

Problem: Vim9: no check for space before #comment.
Solution: Add space checks. Fix :throw with double quoted string.
Files: src/usercmd.c, src/userfunc.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0625

Problem: Vim9: confusing error when calling unknown function.
Solution: Give error while compiling.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.0626

Problem: Vim9: wrong syntax of function in Vim9 script.
Solution: Give error for missing space. Implement :echomsg and :echoerr.
(closes #5670)
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/userfunc.c,
src/eval.c, src/globals.h, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0627

Problem: Vim9: error message does not work. (Yegappan Lakshmanan)
Solution: Swap lines.
Files: src/userfunc.c

Patch 8.2.0628

Problem: Error in menu translations.
Solution: Insert a backslash before a space in one more file. (Shun Bai,
Emir Sarı)
Files: runtime/lang/menu_zh_cn.utf-8.vim,
runtime/lang/menu_ca_es.latin1.vim,
runtime/lang/menu_cs_cz.iso_8859-2.vim,
runtime/lang/menu_cs_cz.utf-8.vim,
runtime/lang/menu_czech_czech_republic.1250.vim,
runtime/lang/menu_czech_czech_republic.ascii.vim,
runtime/lang/menu_da.utf-8.vim,
runtime/lang/menu_fi_fi.latin1.vim,
runtime/lang/menu_hu_hu.iso_8859-2.vim,
runtime/lang/menu_hu_hu.utf-8.vim,
runtime/lang/menu_is_is.latin1.vim,
runtime/lang/menu_no_no.latin1.vim, runtime/lang/menu_pt_br.vim,
runtime/lang/menu_pt_pt.vim,
runtime/lang/menu_sk_sk.iso_8859-2.vim,
runtime/lang/menu_sl_si.latin2.vim,
runtime/lang/menu_slovak_slovak_republic.1250.vim,
runtime/lang/menu_tr_tr.cp1254.vim,
runtime/lang/menu_tr_tr.iso_8859-9.vim,
runtime/lang/menu_tr_tr.utf-8.vim, runtime/lang/menu_vi_vn.vim

Patch 8.2.0629

Problem: Setting a boolean option to v:false does not work.
Solution: Do not use the string representation of the value. (Christian

Brabandt, closes #5974)
Files: src/evalvars.c, src/testdir/test_options.vim

Patch 8.2.0630

Problem: "make tags" does not cover Haiku GUI file.
Solution: Add *.cc files.
Files: src/Make_all.mak

Patch 8.2.0631

Problem: Haiku file formatted with wrong tabstop.
Solution: Use normal tabstop. Fix white space.
Files: src/gui_haiku.cc

Patch 8.2.0632

Problem: Crash when using Haiku.
Solution: Lock the screen. (closes #5975, closes #5973)
Files: src/screen.c

Patch 8.2.0633

Problem: Crash when using null partial in filter().
Solution: Fix crash. Add more tests. (Yegappan Lakshmanan, closes #5976)
Files: src/eval.c, src/testdir/test_blob.vim,
src/testdir/test_channel.vim, src/testdir/test_eval_stuff.vim,
src/testdir/test_execute_func.vim, src/testdir/test_expr.vim,
src/testdir/test_filter_map.vim, src/testdir/test_fold.vim,
src/testdir/test_functions.vim, src/testdir/test_let.vim,
src/testdir/test_listdict.vim, src/testdir/test_partial.vim,
src/testdir/test_usercommands.vim

Patch 8.2.0634

Problem: Crash with null partial and blob.
Solution: Check for NULL pointer. Add more tests. (Yegappan Lakshmanan, closes #5984)
Files: src/eval.c, src/list.c, src/testdir/test_blob.vim,
src/testdir/test_bufwintabinfo.vim, src/testdir/test_cd.vim,
src/testdir/test_channel.vim, src/testdir/test_cursor_func.vim,
src/testdir/test_eval_stuff.vim, src/testdir/test_expr.vim,
src/testdir/test_filter_map.vim, src/testdir/test_fnamemodify.vim,
src/testdir/test_functions.vim, src/testdir/test_getvar.vim,
src/testdir/test_listdict.vim, src/testdir/test_messages.vim,
src/testdir/test_partial.vim, src/testdir/test_quickfix.vim,
src/testdir/test_tabpage.vim, src/testdir/test_vimscript.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_window_id.vim,
src/testdir/test_writefile.vim

Patch 8.2.0635

Problem: When using 256 colors DarkYellow does not show expected color.
Solution: Use color 3 instead of 130. (Romain Lafourcade, closes #5985)
Files: src/highlight.c

Patch 8.2.0636

Problem: :messages does not show the maintainer when \$LANG is unset.
Solution: Call get_mess_lang() if available. (closes #5978)
Files: src/message.c

Patch 8.2.0637

Problem: Incsearch highlighting does not work for ":sort!".
Solution: Skip over the exclamation point. (closes #5983)
Files: src/ex_getln.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_incsearch_sort_02.dump

Patch 8.2.0638

Problem: MS-Windows: messages test fails.
Solution: Clear environment variables.
Files: src/testdir/test_messages.vim

Patch 8.2.0639

Problem: MS-Windows: messages test still fails.
Solution: Filter out the maintainer message.
Files: src/testdir/test_messages.vim

Patch 8.2.0640

Problem: Vim9: expanding `=expr` does not work.
Solution: Find wildcards in not compiled commands. Reorganize test files.
Files: Filelist, src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/vim9.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim,
src/testdir/Make_all.mak

Patch 8.2.0641

Problem: Vim9: `=expr` not expanded in :hardcopy and "syntax include".
Solution: Add the EX_EXPAND flag. Expend "syntax include".
Files: src/ex_cmds.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.0642

Problem: Vim9: using invalid index.
Solution: Check index for being valid. Fix memory leak.
Files: src/vim9compile.c, src/clientserver.c

Patch 8.2.0643 (after 8.2.0635)

Problem: Terminal uses brown instead of dark yellow. (Romain Lafourcade)
Solution: Use color index 3 instead of 130. (closes #5993)
Files: src/terminal.c

Patch 8.2.0644

Problem: Insufficient testing for invalid function arguments.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5988)
Files: runtime/doc/eval.txt, src/testdir/test_buflines.vim,
src/testdir/test_channel.vim, src/testdir/test_clientserver.vim,
src/testdir/test_expr.vim, src/testdir/test_functions.vim,
src/testdir/test_listener.vim, src/testdir/test_match.vim,
src/testdir/test_menu.vim, src/testdir/test_quickfix.vim,
src/testdir/test_registers.vim, src/testdir/test_reftime.vim,
src/testdir/test_terminal.vim, src/testdir/test_textprop.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_window_id.vim,
src/testdir/test_writefile.vim

Patch 8.2.0645

Problem: MS-Windows terminal: **CTRL-C** does not get to child job.
Solution: Remove CREATE_NEW_PROCESS_GROUP from CreateProcessW(). (Nobuhiro Takasaki, closes #5987)
Files: src/terminal.c

Patch 8.2.0646

Problem: t_Co uses the value of \$COLORS in the GUI. (Masato Nishihata)
Solution: Ignore \$COLORS for the GUI. (closes #5992)
Files: src/os_unix.c, src/term.c

Patch 8.2.0647

Problem: MS-Windows: repeat count for events was not used.
Solution: Check the repeat count. (Nobuhiro Takasaki, closes #5989)
Files: src/os_win32.c

Patch 8.2.0648

Problem: Semicolon search does not work in first line.
Solution: Allow the cursor to be in line zero. (Christian Brabandt, closes #5996)
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.0649

Problem: Undo problem when an InsertLeave autocommand resets undo. (Kutsan Kaplan)
Solution: Do not create a new undo block when leaving Insert mode.
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.2.0650

Problem: Vim9: script function can be deleted.
Solution: Disallow deleting script function. Delete functions when sourcing a script again.
Files: src/userfunc.c, src/proto/userfunc.pro, src/evalfunc.c, src/vim9compile.c, src/vim9execute.c, src/vim9script.c, src/scriptfile.c, src/testing.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0651

Problem: Old style benchmark test still in list of distributed files.
Solution: Remove the files from the list.
Files: Filelist

Patch 8.2.0652 (after 8.2.0650)

Problem: Compiler warning for char conversion.
Solution: Use unsigned char buffer.
Files: src/userfunc.c

Patch 8.2.0653 (after 8.2.0650)

Problem: using uninitialized pointer.
Solution: Move assignment up. (John Marriott)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.0654

Problem: Building with Python fails.
Solution: Add missing argument.
Files: src/if_py_both.h

Patch 8.2.0655

Problem: Search code not sufficiently tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #5999)
Files: src/testdir/test_charsearch.vim, src/testdir/test_gn.vim,
src/testdir/test_goto.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_normal.vim, src/testdir/test_search.vim,
src/testdir/test_textformat.vim, src/testdir/test_textobjects.vim,
src/testdir/test_visual.vim

Patch 8.2.0656

Problem: MS-Windows: redrawing right screen edge may not be needed.
Solution: Check the build version. (Nobuhiro Takasaki, closes #6002)
Files: src/drawscreen.c, src/os_win32.c, src/proto/os_win32.pro

Patch 8.2.0657

Problem: Vim9: no check if called variable is a FuncRef.
Solution: Add a type check.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.0658 (after 8.2.0646)

Problem: HP-UX build fails when setenv() is not defined.
Solution: Change "colors" to "t_colors". (John Marriott)
Files: src/os_unix.c

Patch 8.2.0659

Problem: Vim9: no test for equal func type.
Solution: Add a test. Improve type check.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0660

Problem: The search.c file is a bit big.
Solution: Split off the text object code to a separate file. (Yegappan Lakshmanan, closes #6007)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/proto.h, src/proto/search.pro, src/proto/textobject.pro,
src/search.c, src/textobject.c

Patch 8.2.0661

Problem: Eval test is still old style.
Solution: Change into new style tests. (Yegappan Lakshmanan, closes #6009)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test_eval.in, src/testdir/test_eval.ok,
src/testdir/test_eval_stuff.vim

Patch 8.2.0662

Problem: Cannot use input() in a channel callback.
Solution: Reset vgetc_busy. (closes #6010)
Files: src/globals.h, src/ex_getln.c, src/evalfunc.c,

src/testdir/test_channel.vim

Patch 8.2.0663

Problem: Not all systemd temp files are recognized.
Solution: Add two more patterns. (Jamie Macdonald, closes #6003)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0664

Problem: Included undesired changes in Makefile.
Solution: Revert the changes.
Files: src/Makefile

Patch 8.2.0665

Problem: Wrongly assuming Python executable is called "python".
Solution: Use detected python command. (Ken Takata, closes #6016)
Also use CheckFunction if possible.
Files: src/testdir/test_terminal.vim, src/testdir/check.vim

Patch 8.2.0666

Problem: Ruby test fails on MS-Windows.
Solution: Remove the "maintainer" line. (Ken Takata, closes #6015)
Files: src/testdir/shared.vim, src/testdir/test_messages.vim,
src/testdir/test_ruby.vim

Patch 8.2.0667

Problem: Cannot install Haiku version from source.
Solution: Update Makefile and rdef file. (Emir Sarı, closes #6013)
Files: Filelist, READMEdir/README_haiku.txt, runtime/doc/os_haiku.txt,
src/Makefile, src/os_haiku.rdef.in, src/os_haiku.rdef

Patch 8.2.0668

Problem: Compiler warning for int/size_t usage.
Solution: Change "int" to "size_t". (Mike Williams)
Files: src/vim9execute.c

Patch 8.2.0669

Problem: MS-Windows: display in VTP is a bit slow.
Solution: Optimize the code. (Nobuhiro Takasaki, closes #6014)
Files: src/os_win32.c, src/screen.c

Patch 8.2.0670

Problem: Cannot change window when evaluating **'completefunc'**.
Solution: Make a difference between not changing text or buffers and also not changing window.
Files: src/ex_getln.c, src/beval.c, src/change.c, src/edit.c, src/eval.c,
src/ex_docmd.c, src/insexpand.c, src/globals.h, src/indent.c,
src/map.c, src/window.c, src/proto/ex_getln.pro, src/register.c,
src/undo.c, src/testdir/test_edit.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_popup.vim

Patch 8.2.0671

Problem: Haiku: compiler warnings.
Solution: Avoid the warnings. Drop display_errors() copy. (Emir Sarı, closes #6018)

Files: .gitignore, src/gui.c, src/gui_haiku.cc

Patch 8.2.0672

Problem: Heredoc in scripts does not accept lower case marker.
Solution: Allow lower case only in non-Vim scripts. (Ken Takata, closes #6019)

Files: src/evalvars.c, src/testdir/test_lua.vim,
src/testdir/test_perl.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_pyx2.vim,
src/testdir/test_pyx3.vim, src/testdir/test_ruby.vim

Patch 8.2.0673

Problem: Cannot build Haiku in shadow directory.
Solution: Add symlink. (Ozaki Kiichi, closes #6023)
Files: src/Makefile

Patch 8.2.0674

Problem: Some source files are too big.
Solution: Move text formatting functions to a new file. (Yegappan Lakshmanan, closes #6021)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/edit.c, src/getchar.c, src/ops.c, src/option.c, src/proto.h,
src/proto/edit.pro, src/proto/getchar.pro, src/proto/ops.pro,
src/proto/option.pro, src/proto/textformat.pro, src/textformat.c

Patch 8.2.0675

Problem: Vim9: no support for closures.
Solution: Do not re-use stack entries.
Files: src/vim9compile.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/evalvars.c, src/proto/evalvars.pro

Patch 8.2.0676

Problem: Pattern in list of distributed files does not match.
Solution: Drop "testdir/test_[a-z]*.ok". Add CI sed files.
Files: Filelist

Patch 8.2.0677

Problem: Vim9: no support for closures.
Solution: Find variables in the outer function scope, so long as the scope exists.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c,
src/vim9execute.c, src/structs.h, src/vim9.h,
src/testdir/test_vim9_func.vim

Patch 8.2.0678

Problem: Rare crash for popup menu.
Solution: Check for NULL pointer. (Nobuhiro Takasaki, closes #6027)
Files: src/popupmenu.c

Patch 8.2.0679

Problem: Vim9: incomplete support for closures.
Solution: At the end of a function copy arguments and local variables if they are still used by a referenced closure.

Files: src/structs.h, src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.0680

Problem: PTYGROUP and PTYMODE are unused.
Solution: Remove from autoconf. (closes #6024)
Files: src/configure.ac, src/auto/configure, src/config.h.in

Patch 8.2.0681

Problem: Pattern for **'hlsearch'** highlighting may leak. (Dominique Pellé)
Solution: Call end_search_hl() to make sure the previous pattern is freed.
(closes #6028)
Files: src/screen.c

Patch 8.2.0682

Problem: Vim9: parsing function argument type can get stuck.
Solution: Bail out when not making progress.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0683

Problem: Vim9: parsing type does not always work.
Solution: Handle func type without return value. Test more closures.
Fix type check offset. Fix garbage collection.
Files: src/vim9compile.c, src/vim9execute.c, src/proto/vim9execute.pro,
src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.0684

Problem: Vim9: memory leak when using lambda.
Solution: Move the funccal context to the partial. Free the function when
exiting.
Files: src/vim9.h, src/structs.h, src/vim9execute.c, src/userfunc.c,
src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.0685 (after 8.2.0684)

Problem: Build failure.
Solution: Include missing changes.
Files: src/vim9compile.c

Patch 8.2.0686

Problem: Formatoptions not sufficiently tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #6031)
Files: src/testdir/test_normal.vim, src/testdir/test_textformat.vim

Patch 8.2.0687

Problem: Some tests do not work on FreeBSD.
Solution: Enable modeline. Use WaitFor() in more cases. (Ozaki Kiichi,
closes #6036)
Files: src/testdir/test_quickfix.vim, src/testdir/test_terminal.vim

Patch 8.2.0688

Problem: Output clobbered if setting **'verbose'** to see shell commands.
Solution: Only output "Searching for" when **'verbose'** is 11 or higher.
Files: src/scriptfile.c, runtime/doc/options.txt

Patch 8.2.0689

Problem: When using getaddrinfo() the error message is unclear.
Solution: Use gai_strerror() to get the message. (Ozaki Kiichi, closes #6034)
Files: src/channel.c

Patch 8.2.0690

Problem: Line number of option set by modeline is wrong.
Solution: Do not double the line number. (Ozaki Kiichi, closes #6035)
Files: src/option.c, src/testdir/test_modeline.vim

Patch 8.2.0691

Problem: Startup test fails.
Solution: Adjust expected output from -V2 argument.
Files: src/testdir/test_startup.vim

Patch 8.2.0692

Problem: Startup test fails on MS-Windows.
Solution: Allow for any path.
Files: src/testdir/test_startup.vim

Patch 8.2.0693

Problem: Closure using argument not tested.
Solution: Add a test, make it work.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0694

Problem: Haiku: channel and terminal do not work.
Solution: Close files when the job has finished. (Ozaki Kiichi, closes #6039)
Files: src/channel.c, src/getchar.c, src/gui_haiku.cc, src/misc1.c

Patch 8.2.0695

Problem: Vim9: cannot define a function inside a function.
Solution: Initial support for :def inside :def.
Files: src/userfunc.c, src/proto/userfunc.pro, src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.0696

Problem: Vim9: nested function does not work properly
Solution: Create a function reference. Check argument count.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.0697

Problem: Vim9: memory leak when using nested function.
Solution: Unreference function when deleting instructions. Adjust reference count for local variables.
Files: src/vim9compile.c, src/vim9execute.c

Patch 8.2.0698

Problem: Insert mode completion not fully tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #6041)
Files: src/testdir/test_edit.vim, src/testdir/test_ins_complete.vim,

src/testdir/test_textformat.vim

Patch 8.2.0699

Problem: Vim9: not all errors tested.
Solution: Add test for deleted function. Bail out on first error.
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_expr.vim, src/testdir/vim9.vim

Patch 8.2.0700

Problem: Vim9: converting error message to exception not tested.
Solution: Test exception from error. Do not continue after :echoerr.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.0701

Problem: Vim9 test fails without job feature.
Solution: Add feature check.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0702

Problem: Running channel tests may leave running process behind.
Solution: Make Python client exit when running into EOF. (Kurtis Rader,
part of #6046)
Files: src/testdir/test_channel_pipe.py

Patch 8.2.0703

Problem: Vim9: closure cannot store value in outer context.
Solution: Make storing value in outer context work. Make :disassemble
accept a function reference.
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/eval.c,
src/structs.h, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.0704

Problem: Vim9: memory leak in disassemble test.
Solution: Decrement refcount when creating funccal.
Files: src/vim9execute.c

Patch 8.2.0705

Problem: Indent tests don't run on CI for FreeBSD.
Solution: Set modeline. (Ozaki Kiichi, closes #6048)
Files: .cirrus.yml, runtime/indent/testdir/runtest.vim

Patch 8.2.0706

Problem: Vim9: using assert_fails() causes function to finish.
Solution: Check did_emsg instead of called_emsg.
Files: src/vim9execute.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0707

Problem: Vim9 function test fails.
Solution: Adjust expected error code.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.0708

Problem: Vim9: constant expressions are not simplified.
Solution: Simplify string concatenation.
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.0709

Problem: MS-Windows: compiler warning for int vs size_t.
Solution: Add type cast. (Mike Williams)
Files: src/channel.c

Patch 8.2.0710

Problem: Netbeans test sometimes fails.
Solution: Mark any test using an external command as flaky.
Files: src/testdir/shared.vim

Patch 8.2.0711

Problem: With a long running Vim the temp directory might be cleared on some systems.
Solution: Lock the temp directory. (closes #6044)
Files: src/config.h.in, src/configure.ac, src/auto/configure,
src/fileio.c, src/globals.h, src/os_unix.h

Patch 8.2.0712

Problem: Various code not fully tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #6049)
Files: src/testdir/test_functions.vim, src/testdir/test_options.vim,
src/testdir/test_system.vim, src/testdir/test_termcodes.vim

Patch 8.2.0713

Problem: The pam_environment file is not recognized.
Solution: Add a filetype pattern for pamenv. (closes #6051)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0714

Problem: Vim9: handling constant expression does not scale.
Solution: Use another solution, passing typval_T.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0715

Problem: Vim9: leaking memory.
Solution: Free strings after concatenating them.
Files: src/vim9compile.c

Patch 8.2.0716

Problem: Vim9: another memory leak.
Solution: Clear typval when failing.
Files: src/vim9compile.c

Patch 8.2.0717

Problem: Vim9: postponed constant expressions does not scale.
Solution: Add a structure to pass around postponed constants.
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.0718

Problem: Gcc warning for returning pointer to local variable. (John
Marriott)
Solution: Return another pointer.
Files: src/evalvars.c

Patch 8.2.0719

Problem: Vim9: more expressions can be evaluated at compile time
Solution: Recognize has('name').
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.0720

Problem: Occasional exit when encountering an X error. (Manfred Lotz)
Solution: On an X error do not exit, do preserve files.
Files: src/os_unix.c

Patch 8.2.0721

Problem: Vim9: leaking memory when skipping.
Solution: Disable skipping in generate_ppconst().
Files: src/vim9compile.c

Patch 8.2.0722

Problem: Vim9: not handling constant expression for elseif.
Solution: Use postponed constants. Delete the code for evaluating a
constant expression.
Files: src/vim9compile.c

Patch 8.2.0723

Problem: Vim9: nested constant expression not evaluated compile time.
Solution: Use compile_expr1() for parenthesis.
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.0724

Problem: Vim9: appending to buffer/window/tab variable not tested
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0725

Problem: Vim9: cannot call a function declared later in Vim9 script.
Solution: Make two passes through the script file.
Files: src/scriptfile.c, src/proto/scriptfile.pro, src/vim9script.c,
src/vim9compile.c, src/vim9execute.c, src/proto/vim9compile.pro,
src/userfunc.c, src/proto/userfunc.pro, src/evalvars.c,
src/proto/evalvars.pro, src/vim.h,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0726

Problem: Vim9: leaking memory when calling not compiled :def function.
Solution: Check if function is compiled earlier.
Files: src/vim9execute.c

Patch 8.2.0727

Problem: MS-Windows: new gcc compiler does not support scanf format.
Solution: Use "%ll" instead of "%I". (Ken Takata)

Files: src/vim.h

Patch 8.2.0728

Problem: Messages about a deadly signal are not left aligned.
Solution: Output a CR before the NL. (Dominique Pellé, #6055)
Files: src/misc1.c, src/os_unix.c

Patch 8.2.0729

Problem: Vim9: When reloading a script variables are not cleared.
Solution: When sourcing a script again clear all script-local variables.
Files: src/dict.c, src/proto/dict.pro, src/scriptfile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0730

Problem: Vim9: Assignment to dict member does not work.
Solution: Parse dict assignment. Implement getting dict member.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/globals.h,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.0731

Problem: Vim9: parsing declarations continues after :finish.
Solution: Bail out when encountering :finish.
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.0732

Problem: Vim9: storing value in dict messes up stack.
Solution: Correct item count of stack.
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.0733

Problem: Vim9: assigning to dict or list argument does not work.
Solution: Recognize an argument as assignment target.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.0734

Problem: Vim9: leaking memory when using :finish.
Solution: Do not check for next line in third pass.
Files: src/scriptfile.c

Patch 8.2.0735

Problem: Vim9: using uninitialized memory.
Solution: Clear the arg_lvar field.
Files: src/vim9compile.c

Patch 8.2.0736

Problem: Some files not recognized as pamenv.
Solution: Add pam_inv.conf. (closes #6065)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0737

Problem: When shell doesn't support **CTRL-Z** Vim still handles it.
Solution: Ignore the STOP signal if it was ignored on startup.
(Kurtis Rader, closes #5990, closes #6058)

Files: src/os_unix.c

Patch 8.2.0738

Problem: Mouse handling in a terminal window not well tested.
Solution: Add tests. (Yegappan Lakshmanan, closes #6052)
Files: src/testdir/term_util.vim, src/testdir/test_gui.vim,
src/testdir/test_modeless.vim, src/testdir/test_terminal.vim

Patch 8.2.0739

Problem: Incomplete profiling when exiting because of a deadly signal.
Solution: Call __gcov_flush() if available.
Files: src/os_unix.c, src/Makefile, .travis.yml

Patch 8.2.0740

Problem: Minor message mistakes.
Solution: Change vim to Vim and other fixes.
Files: src/if_py_both.h, src/if_tcl.c, src/main.c

Patch 8.2.0741

Problem: Python tests fail because of changed message.
Solution: Adjust the expected messages (Dominique Pellé, closes #6066)
Files: src/testdir/test86.ok, src/testdir/test87.ok

Patch 8.2.0742

Problem: Handling of a TERM signal not tested.
Solution: Add a test for SIGTERM. (Dominique Pellé, closes #6055)
Files: src/testdir/test_signals.vim

Patch 8.2.0743

Problem: Can move to another buffer from a terminal in popup window.
Solution: Do not allow "gf" or editing a file. (closes #6072)
Files: src/normal.c, src/ex_cmds.c, src/testdir/test_popupwin.vim

Patch 8.2.0744

Problem: The name vim is not capitalized in a message.
Solution: Use "Vim" instead of "vim".
Files: src/main.c

Patch 8.2.0745

Problem: Crash on exit when not all popups are closed.
Solution: Close popups when freeing all memory. Disable checking for popup
when editing a file for now.
Files: src/misc2.c, src/ex_cmds.c

Patch 8.2.0746

Problem: popup_clear() hangs when a popup can't be closed.
Solution: Bail out when a popup can't be closed.
Files: src/popupwin.c, src/proto/popupwin.pro

Patch 8.2.0747

Problem: Cannot forcefully close all popups.
Solution: Add the "force" argument to popup_clear(). Use it after running a
test. Put back the check for a popup when editing a file.
Files: runtime/doc/popup.txt, src/evalfunc.c, src/popupwin.c,

src/proto/popupwin.pro, src/tag.c, src/window.c, src/misc2.c,
src/ex_cmds.c, src/channel.c, src/testdir/runtest.vim,
src/testdir/test_terminal.vim

Patch 8.2.0748

Problem: Cannot get a list of all popups.
Solution: Add popup_list(). Use it in the test runner.
Files: runtime/doc/eval.txt, runtime/doc/popup.txt, src/popupwin.c,
src/proto/popupwin.pro, src/evalfunc.c,
src/testdir/test_popupwin.vim, src/testdir/runtest.vim

Patch 8.2.0749

Problem: TERM signal test fails on FreeBSD.
Solution: Do not check the messages, they may appear anywhere. (Dominique Pellé, closes #6075)
Files: src/testdir/test_signals.vim

Patch 8.2.0750

Problem: Netbeans test is a bit flaky.
Solution: Allow for standard sign to be defined. Use WaitForAssert().
Files: src/testdir/test_netbeans.vim

Patch 8.2.0751

Problem: Vim9: performance can be improved.
Solution: Don't call break. Inline check for list materialize. Make an inline version of ga_grow().
Files: src/macros.h, src/evalfunc.c, src/misc2.c,
src/proto/misc2.pro, src/channel.c, src/eval.c, src/evalbuffer.c,
src/evalvars.c, src/filepath.c, src/highlight.c, src/insexpand.c,
src/json.c, src/list.c, src/popupmenu.c, src/popupwin.c,
src/userfunc.c, src/if_py_both.h

Patch 8.2.0752

Problem: Terminal in popup window test is a bit flaky.
Solution: Wait for shell job status to be "run". Mark as flaky test.
Files: src/testdir/test_popupwin.vim

Patch 8.2.0753

Problem: Vim9: expressions are evaluated in the discovery phase.
Solution: Bail out if an expression is not a constant. Require a type for declared constants.
Files: src/vim.h, src/evalvars.c, src/eval.c, src/ex_eval.c,
src/evalfunc.c, src/userfunc.c, src/dict.c, src/list.c,
src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.0754

Problem: Vim9: No test for forward declaration.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0755

Problem: Vim9: No error when variable initializer is not a constant.
Solution: Return FAIL when trying to get a variable value. Do not execute a

script when an error is detected in the first or second phase.
Files: src/eval.c, src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.0756 (after 8.2.0249)

Problem: MS-Windows: still a compiler warning.
Solution: Move flag to another place in the Makefile. (Ken Takata,
closes #6083)
Files: src/Make_mvc.mak

Patch 8.2.0757

Problem: Vim9: no test for MEMBER instruction.
Solution: Add a test. Make matches stricter.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0758

Problem: Vim9: no test for STORELIST and STOREDICT.
Solution: Add a test. Make matches stricter.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.0759 (after 8.2.0751)

Problem: Vim9: missing changes for performance improvements
Solution: Use GA_GROW(). Don't call breakcheck so often.
Files: src/vim9execute.c

Patch 8.2.0760

Problem: Vim9: dict member errors not tested.
Solution: Delete unreachable error. Add tests.
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.0761

Problem: Vim9: instructions not tested
Solution: Use a variable instead of a constant.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.0762

Problem: Buffer is not considered modified after setting crypt key.
Solution: Set the modified flag. (Christian Brabandt, closes #6082)
Files: src/optionstr.c, src/testdir/test_crypt.vim

Patch 8.2.0763

Problem: GUI test fails without the terminal feature.
Solution: Check the terminal feature is supported. (Ken Takata,
closes #6084)
Files: src/testdir/test_gui.vim

Patch 8.2.0764

Problem: Vim9: assigning to option not fully tested.
Solution: Add more test cases. Allow using any type for assignment.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0765

Problem: In the GUI can't use all the modifiers. (Andri Möll)
Solution: Do not apply Alt/Meta early, do it later like with the terminal.

Avoid the Motif test from crashing.
Files: src/gui_gtk_x11.c, src/gui_x11.c, src/gui_mac.c, src/gui_w32.c, src/gui_motif.c

Patch 8.2.0766

Problem: Display error when using **'number'** and **'breakindent'**.
Solution: Adjust extra spaces in the first row. (Ken Takata, closes #6089, closes #5986)
Files: src/drawline.c, src/testdir/test_breakindent.vim

Patch 8.2.0767

Problem: ModifyOtherKeys active when using a shell command in autocmd.
Solution: Output T_CTE when going to cooked mode. (closes 5617)
Files: src/term.c

Patch 8.2.0768

Problem: Vim9: memory leak in script test.
Solution: Clear typval before giving an error message.
Files: src/vim9execute.c

Patch 8.2.0769

Problem: VimLeavePre not triggered when Vim is terminated.
Solution: Unblock autocommands.
Files: src/main.c, src/testdir/test_signals.vim

Patch 8.2.0770

Problem: Cannot map **CTRL-B** when using the GUI.
Solution: Reset the CTRL modifier when used. (closes #6092)
Files: src/gui_gtk_x11.c

Patch 8.2.0771

Problem: Vim9: cannot call a compiled closure from not compiled code.
Solution: Pass funcexe to call_user_func().
Files: src/userfunc.c, src/vim9execute.c, src/proto/vim9execute.pro, src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.0772

Problem: Vim9: some variable initializations not tested.
Solution: Add a few more tests
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0773

Problem: Switching to raw mode every time ":" is used.
Solution: When executing a shell set cur_tmde to TMODE_UNKNOWN, so that the next time TMODE_RAW is used it is set, but not every time.
Files: src/term.h, src/os_unix.c, src/term.c, src/os_amiga.c, src/os_win32.c

Patch 8.2.0774

Problem: t_TI and t_TE are output when using **'visualbell'**. (Dominique Pellé)
Solution: Do not change the terminal mode for a short sleep. Do not output t_TI and t_TE when switching to/from TMODE_SLEEP. Make tmde an enum.

Files: src/os_unix.c, src/proto/os_unix.pro, src/os_amiga.c,
src/proto/os_amiga.pro, src/os_mswin.c, src/proto/os_mswin.pro,
src/os_vms.c, src/proto/os_vms.pro, src/os_win32.c,
src/proto/os_win32.pro, src/term.c, src/term.h, src/globals.h

Patch 8.2.0775

Problem: Not easy to call a Vim function from Lua.
Solution: Add vim.call() and vim.fn(). (Prabir Shrestha, closes #6063)
Files: runtime/doc/if_lua.txt, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.0776

Problem: Libvterm code lags behind the upstream version.
Solution: Include revision 719.
Files: Filelist, src/libvterm/README, src/libvterm/Makefile,
src/libvterm/find-wide-chars.pl, src/libvterm/src/fullwidth.inc,
src/libvterm/src/unicode.c

Patch 8.2.0777 (after 8.2.0776)

Problem: Terminal test fails.
Solution: Adjust character position for double-wide characters.
Files: src/testdir/test_terminal.vim

Patch 8.2.0778

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 720 - 723.
Files: src/libvterm/t/10state_putglyph.test, src/libvterm/Makefile,
src/libvterm/t/run-test.pl, src/libvterm/src/state.c,
src/libvterm/t/92lp1805050.test

Patch 8.2.0779

Problem: Tmode_T not used everywhere.
Solution: Also use tmode_T for settmode().
Files: src/term.c, src/proto/term.pro

Patch 8.2.0780

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 724 - 726.
Files: Filelist, src/libvterm/t/40screen_ascii.test,
src/libvterm/t/60screen_ascii.test,
src/libvterm/t/41screen_unicode.test,
src/libvterm/t/61screen_unicode.test,
src/libvterm/t/42screen_damage.test,
src/libvterm/t/62screen_damage.test,
src/libvterm/t/43screen_resize.test,
src/libvterm/t/63screen_resize.test,
src/libvterm/t/44screen_pen.test,
src/libvterm/t/64screen_pen.test,
src/libvterm/t/45screen_protect.test,
src/libvterm/t/65screen_protect.test,
src/libvterm/t/46screen_extent.test,
src/libvterm/t/66screen_extent.test,
src/libvterm/t/47screen_dbl_wh.test,
src/libvterm/t/67screen_dbl_wh.test,
src/libvterm/t/48screen_termprops.test,

src/libvterm/t/68screen_termprops.test, src/libvterm/t/30pen.test,
src/libvterm/t/30state_pen.test, src/libvterm/t/92lp1805050.test,
src/libvterm/t/31state_rep.test, src/libvterm/doc/seqs.txt

Patch 8.2.0781 (after 8.2.0775)

Problem: Compiler warning for not using value in Lua.

Solution: Add "(void)".

Files: src/if_lua.c

Patch 8.2.0782

Problem: Cannot build with Lua on MS-Windows.

Solution: Add DLL symbol for luaL_Loadstring. (Ken Takata)

Files: src/if_lua.c

Patch 8.2.0783

Problem: Libvterm code lags behind the upstream version.

Solution: Include revisions 728 - 729.

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile,
src/libvterm/src/keyboard.c, src/libvterm/t/25state_input.test,
src/libvterm/t/harness.c, src/libvterm/src/vterm.c,
src/libvterm/src/vterm_internal.h,
src/libvterm/t/26state_query.test

Patch 8.2.0784

Problem: Libvterm code lags behind the upstream version.

Solution: Include revisions 730 - 733.

Files: src/libvterm/src/vterm.c, src/libvterm/src/state.c,
src/libvterm/include/vterm.h, src/libvterm/src/vterm_internal.h,
src/libvterm/t/harness.c

Patch 8.2.0785

Problem: Libvterm code lags behind the upstream version.

Solution: Include revisions 734 - 740.

Files: src/libvterm/include/vterm.h, src/libvterm/src/pen.c,
src/libvterm/src/vterm.c, src/libvterm/doc/seqs.txt,
src/libvterm/t/30state_pen.test, src/libvterm/t/run-test.pl,
src/libvterm/Makefile, src/libvterm/CONTRIBUTING

Patch 8.2.0786

Problem: Channel test is flaky on FreeBSD.

Solution: Set the socket TCP_NODELAY option. Adjust expected line count in netbeans test. (Ozaki Kiichi, closes #6097)

Files: src/testdir/test_channel.py, src/testdir/test_netbeans.vim

Patch 8.2.0787

Problem: Libvterm code lags behind the upstream version.

Solution: Include revisions 741 - 742.

Files: Filelist, src/libvterm/src/screen.c

Patch 8.2.0788

Problem: Memory leak in libvterm.

Solution: free tmpbuffer.

Files: src/libvterm/src/vterm.c

Patch 8.2.0789

Problem: Vim9: expression testing lost coverage using constants.
Solution: Use a few variables instead of constants.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.0790

Problem: Vim9: list index not well tested.
Solution: Add a few more tests.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0791

Problem: A second popup window with terminal causes trouble.
Solution: Disallow opening a second terminal-popup window. (closes #6101, closes #6103) Avoid defaulting to an invalid line number.
Files: runtime/doc/popup.txt, src/popupwin.c, src/ex_docmd.c, src/testdir/test_popupwin.vim, src/testdir/test_terminal.vim

Patch 8.2.0792

Problem: Build failure with small features.
Solution: Add #ifdef.
Files: src/popupwin.c

Patch 8.2.0793

Problem: MS-Windows: cannot build GUI with small features. (Michael Soyka)
Solution: Add #ifdef around use of windowsVersion. (Ken Takata)
Files: src/os_win32.c

Patch 8.2.0794

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 743 - 747.
Files: src/libvterm/src/state.c, src/libvterm/src/screen.c, src/libvterm/src/vterm_internal.h, src/libvterm/include/vterm.h, src/libvterm/t/67screen_dbl_wh.test, src/libvterm/t/run-test.pl

Patch 8.2.0795

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 748 - 754.
Files: src/libvterm/include/vterm.h, src/libvterm/src/screen.c, src/libvterm/src/state.c, src/libvterm/t/32state_flow.test, src/libvterm/t/60screen_ascii.test, src/libvterm/t/62screen_damage.test, src/libvterm/t/63screen_resize.test, src/libvterm/t/harness.c, src/libvterm/t/run-test.pl

Patch 8.2.0796

Problem: MS-Windows: compiler can't handle C99 construct in libvterm.
Solution: Change to C90 construct.
Files: src/libvterm/src/state.c

Patch 8.2.0797

Problem: MS-Windows: compiler still can't handle C99 construct.
Solution: Change to C90 construct. (Dominique Pellé, closes #6106)
Files: src/libvterm/src/state.c

Patch 8.2.0798

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 755 - 758.
Files: src/libvterm/t/run-test.pl, src/libvterm/src/screen.c,
src/libvterm/t/harness.c, src/libvterm/include/vterm.h,
src/libvterm/src/parser.c, src/libvterm/src/state.c,
src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h,
src/libvterm/t/02parser.test,
src/libvterm/t/18state_termprops.test,
src/libvterm/t/29state_fallback.test,
src/libvterm/t/68screen_termprops.test, src/terminal.c

Patch 8.2.0799

Problem: Build fails if snprintf is not available.
Solution: Use vim_snprintf().
Files: src/libvterm/src/state.c

Patch 8.2.0800

Problem: Errors from failing test are unclear.
Solution: Include text where parsing failed.
Files: src/json.c, src/testdir/test_json.vim

Patch 8.2.0801

Problem: Terminal test fails on Mac.
Solution: Concatenate OSC pieces.
Files: src/terminal.c

Patch 8.2.0802

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 759 - 762.
Files: src/terminal.c, src/libvterm/doc/seqs.txt,
src/libvterm/include/vterm.h, src/libvterm/src/pen.c,
src/libvterm/src/screen.c, src/libvterm/src/state.c,
src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h,
src/libvterm/t/harness.c, src/libvterm/t/12state_scroll.test

Patch 8.2.0803

Problem: Libvterm code lags behind the upstream version.
Solution: Include revisions 764 - 767
Files: src/Makefile, src/libvterm/src/parser.c,
src/libvterm/src/vterm_internal.h, src/libvterm/t/02parser.test,
src/libvterm/t/run-test.pl, src/libvterm/find-wide-chars.pl,
src/libvterm/src/fullwidth.inc

Patch 8.2.0804

Problem: Libvterm code lags behind the upstream version.
Solution: Include revision 727, but add the index instead of switching
between RGB and indexed.
Files: src/terminal.c, src/term.c, src/libvterm/include/vterm.h,
src/libvterm/src/pen.c, src/libvterm/src/screen.c,
src/libvterm/src/vterm_internal.h,
src/libvterm/t/30state_pen.test,
src/libvterm/t/harness.c, src/libvterm/src/state.c,
src/libvterm/t/26state_query.test,

src/libvterm/t/64screen_pen.test

Patch 8.2.0805

Problem: Terminal key codes test fails on some systems.
Solution: Skip keypad 3 and 9. (Yegappan Lakshmanan, closes #6070)
Files: src/testdir/test_terminal.vim

Patch 8.2.0806

Problem: using "func!" after vim9script gives confusing error.
Solution: Give E477. (closes #6107)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.0807

Problem: Cannot easily restore a mapping.
Solution: Add mapset().
Files: runtime/doc/eval.txt, src/map.c, src/proto/map.pro, src/evalfunc.c
src/testdir/test_maparg.vim

Patch 8.2.0808

Problem: Not enough testing for the terminal window.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6069) Fix memory leak.
Files: src/testdir/test_gui.vim, src/testdir/test_terminal.vim, src/terminal.c

Patch 8.2.0809

Problem: Build failure with small features. (Tony Mechelynck)
Solution: Move "expr" inside #ifdef.
Files: src/map.c

Patch 8.2.0810

Problem: Error when appending "tagfile" to **'wildoptions'**.
Solution: use flags P_ONECOMMA and P_NODUP. (Dmitri Vereshchagin, closes #6105)
Files: src/optiondefs.h, src/testdir/test_options.vim

Patch 8.2.0811

Problem: Terminal keycode test is flaky.
Solution: Use WaitForAssert()
Files: src/testdir/test_terminal.vim

Patch 8.2.0812

Problem: mapset() does not properly handle <> notation.
Solution: Convert <> codes. (closes #6116)
Files: src/map.c, src/testdir/test_maparg.vim

Patch 8.2.0813

Problem: libvterm code is slightly different from upstream.
Solution: Use upstream text to avoid future merge problems. Mainly comment style changes.
Files: src/libvterm/include/vterm.h, src/libvterm/src/rect.h, src/libvterm/src/utf8.h, src/libvterm/src/vterm_internal.h, src/libvterm/src/encoding.c, src/libvterm/src/keyboard.c, src/libvterm/src/mouse.c, src/libvterm/src/parser.c,

src/libvterm/src/pen.c, src/libvterm/src/screen.c,
src/libvterm/src/state.c, src/libvterm/src/unicode.c,
src/libvterm/src/vterm.c

Patch 8.2.0814

Problem: Clang warning for implicit conversion.
Solution: Add type cast. (Dominique Pellé, closes #6124)
Files: src/evalfunc.c

Patch 8.2.0815

Problem: maparg() does not provide enough information for mapset().
Solution: Add "lhsraw" and "lhsrawalt" items. Drop "simplified"
Files: src/map.c, runtime/doc/eval.txt, src/testdir/test_maparg.vim

Patch 8.2.0816

Problem: Terminal test fails when compiled with Athena.
Solution: Do give an error when the GUI is not running. (hint by Dominique Pellé, closes #5928, closes #6132)
Files: src/globals.h, src/gui.c, src/term.c, src/channel.c,
src/testdir/test_terminal.vim

Patch 8.2.0817

Problem: Not enough memory allocated when converting string with special character.
Solution: Reserve space for modifier code. (closes #6130)
Files: src/eval.c, src/testdir/test_functions.vim

Patch 8.2.0818

Problem: Vim9: using a discovery phase doesn't work well.
Solution: Remove the discovery phase, instead compile a function only when it is used. Add :defcompile to compile def functions earlier.
Files: runtime/doc/vim9.txt, src/vim9script.c, src/structs.h,
src/userfunc.c, src/proto/userfunc.pro, src/eval.c,
src/evalvars.c, src/proto/evalvars.pro, src/vim9compile.c,
src/proto/vim9compile.pro, src/vim9execute.c, src/ex_cmds.h,
src/ex_docmd.c, src/ex_cmdidxs.h, src/vim.h, src/testdir/vim9.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.0819

Problem: Compiler warning for unused variable.
Solution: Remove the variable.
Files: src/evalvars.c

Patch 8.2.0820

Problem: Vim9: function type isn't set until compiled.
Solution: Set function type early.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c,
src/testdir/test_vim9_func.vim

Patch 8.2.0821

Problem: Vim9: memory leak in expr test.
Solution: Do not decrement the length of the list of functions if the current function is not at the end.

Files: src/vim9compile.c

Patch 8.2.0822

Problem: Vim9: code left over from discovery phase.

Solution: Remove the dead code.

Files: src/scriptfile.c, src/proto/scriptfile.pro, src/ex_cmds.h,
src/evalvars.c, src/proto/evalvars.pro, src/ex_docmd.c

Patch 8.2.0823

Problem: Vim9: script reload test is disabled.

Solution: Compile a function in the context of the script where it was defined. Set execution stack for compiled function. Add a test that an error is reported for the right file/function.

Files: src/vim9compile.c, src/vim9execute.c, src/scriptfile.c,
src/proto/scriptfile.pro, src/userfunc.c, src/globals.h,
src/structs.h, src/ex_docmd.c, src/ex_eval.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0824 (after 8.2.0817)

Problem: Still not enough memory allocated when converting string with special character.

Solution: Reserve space for expanding K_SPECIAL. (closes #6130)

Files: src/eval.c, src/testdir/test_functions.vim

Patch 8.2.0825

Problem: def_function() may return pointer that was freed.

Solution: Set "fp" to NULL after freeing it.

Files: src/userfunc.c

Patch 8.2.0826

Problem: Vim9: crash in :defcompile.

Solution: Restart the loop after a call to compile_def_function() caused the hash table to resize.

Files: src/userfunc.c

Patch 8.2.0827

Problem: Vim9: crash in :defcompile.

Solution: Fix off-by-one error.

Files: src/userfunc.c

Patch 8.2.0828

Problem: Travis: regexp pattern doesn't work everywhere.

Solution: Use [:blank:] instead of \b. (Ozaki Kiichi, closes #6146)

Files: .travis.yml, ci/config.mk.clang.sed, ci/config.mk.gcc.sed,
ci/config.mk.sed, src/if_ruby.c

Patch 8.2.0829

Problem: filter() may give misleading error message.

Solution: Also mention Blob as an allowed argument.

Files: src/list.c, src/testdir/test_filter_map.vim

Patch 8.2.0830

Problem: Motif: can't map "!". (Ben Jackson)

Solution: Remove the shift modifier if it's already included in the key.

(closes #6147)
Files: src/gui_x11.c

Patch 8.2.0831

Problem: Compiler warnings for integer sizes.
Solution: Add type casts. (Mike Williams)
Files: src/libvterm/src/pen.c, src/terminal.c

Patch 8.2.0832

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Add initial value.
Files: src/map.c

Patch 8.2.0833

Problem: Mapping <C-bslash> doesn't work in the GUI.
Solution: Reset seenModifyOtherKeys when starting the GUI. (closes #6150)
Files: src/gui.c

Patch 8.2.0834

Problem: :drop command in terminal popup causes problems.
Solution: Check for using a popup window. (closes #6151)
Files: src/ex_cmds.c, src/testdir/test_popupwin.vim

Patch 8.2.0835

Problem: Motif: mapping <C-bslash> still doesn't work.
Solution: Accept CSI for K_SPECIAL. Do not apply CTRL to the character early. (closes #6150)
Files: src/getchar.c, src/gui_x11.c

Patch 8.2.0836

Problem: Not all :cdo output is visible.
Solution: Reset 'shortmess' temporarily. (Yegappan Lakshmanan, closes #6155)
Files: src/ex_cmds2.c, src/testdir/test_cdo.vim

Patch 8.2.0837

Problem: Compiler warning for value set but not used.
Solution: Move variable inside #ifdef.
Files: src/channel.c

Patch 8.2.0838

Problem: MS-Windows: compiler warning for uninitialized variables.
Solution: Initialize variables.
Files: src/screen.c

Patch 8.2.0839

Problem: Dropping modifier when putting a character back in typeahead.
Solution: Add modifier to ins_char_typebuf(). (closes #6158)
Files: src/getchar.c, src/proto/getchar.pro, src/message.c, src/normal.c, src/terminal.c, src/globals.h, src/testdir/test_messages.vim

Patch 8.2.0840

Problem: Search match count wrong when only match is in fold.
Solution: Update search stats when in a closed fold. (Christian Brabandt, closes #6160, closes #6152)

Files: src/search.c, src/testdir/dumps/Test_searchstat_3.dump,
src/testdir/test_search_stat.vim

Patch 8.2.0841

Problem: **'verbose'** value 16 causes duplicate output.

Solution: Combine levels 15 and 16 into one message. (Christian Brabandt,
closes #6153)

Files: runtime/doc/options.txt, src/ex_docmd.c

Patch 8.2.0842 (after 8.2.0837)

Problem: MS-Windows: channel tests fail.

Solution: Adjust #ifdefs. (closes #6162)

Files: src/channel.c

Patch 8.2.0843

Problem: Filetype elm not detected.

Solution: Recognize *.elm files. (closes #6157)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0844

Problem: Text properties crossing lines not handled correctly.

Solution: When saving for undo include an extra line when needed and do not
adjust properties when undoing. (Axel Forsman, closes #5875)

Files: src/memline.c, src/proto/memline.pro, src/undo.c, src/structs.h

Patch 8.2.0845

Problem: Text properties crossing lines not handled correctly.

Solution: When joining lines merge text properties if possible.
(Axel Forsman, closes #5839, closes #5683)

Files: src/testdir/test_textprop.vim, src/memline.c, src/ops.c,
src/proto/textprop.pro, src/textprop.c,
src/testdir/dumps/Test_textprop_01.dump

Patch 8.2.0846

Problem: Build failure with small features.

Solution: Add #ifdef.

Files: src/undo.c

Patch 8.2.0847

Problem: Typval related code is spread out.

Solution: Move code to new typval.c file. (Yegappan Lakshmanan, closes #6093)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak,
src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md,
src/eval.c, src/evalfunc.c, src/globals.h, src/proto.h,
src/proto/eval.pro, src/proto/evalfunc.pro, src/proto/typval.pro,
src/typval.c

Patch 8.2.0848

Problem: MS-Windows: the Windows terminal code has some flaws.

Solution: Do not redraw the right edge of the screen. Remove the background
color trick. Flush the screen output buffer often. (Nobuhiro
Takasaki, #5546)

Files: src/os_win32.c, src/proto/os_win32.pro, src/term.c

Patch 8.2.0849

Problem: BeOS code is not maintained and probably unused.
Solution: Remove the BeOS code. (Emir Sarı, closes #5817)
Files: Filelist, src/Makefile, src/configure.ac, src/auto/configure, src/evalfunc.c, src/normal.c, src/os_beos.c, src/os_beos.h, src/os_beos.rsrc, src/os_unix.c, src/proto.h, src/proto/os_beos.pro, src/pty.c, src/screen.c, src/term.c, src/testdir/test_functions.vim, src/ui.c, src/vim.h

Patch 8.2.0850

Problem: MS-Windows: exepath() works differently from cmd.exe.
Solution: Make exepath() work better on MS-Windows. (closes #6115)
Files: runtime/doc/eval.txt, src/os_win32.c, src/testdir/test_functions.vim

Patch 8.2.0851 (after 8.2.0833)

Problem: Can't distinguish <M-a> from accented "a" in the GUI.
Solution: Use another way to make mapping <C-bslash> work. (closes #6163)
Files: src/gui.c, src/gui_gtk_x11.c, src/getchar.c

Patch 8.2.0852

Problem: Cannot map CTRL-S on some systems.
Solution: Do not use CTRL-S for flow control.
Files: src/os_unix.c

Patch 8.2.0853

Problem: ml_delete() often called with FALSE argument.
Solution: Use ml_delete_flags(x, ML_DEL_MESSAGE) when argument is TRUE.
Files: src/buffer.c, src/change.c, src/diff.c, src/evalbuffer.c, src/ex_cmds.c, src/ex_docmd.c, src/fileio.c, src/if_lua.c, src/if_mzsch.c, src/if_ruby.c, src/if_tcl.c, src/normal.c, src/popupmenu.c, src/popupwin.c, src/quickfix.c, src/spell.c, src/terminal.c, src/if_perl.xs, src/if_py_both.h, src/memline.c, src/proto/memline.pro

Patch 8.2.0854

Problem: Xxd cannot show offset as a decimal number.
Solution: Add the "-d" flag. (Aapo Rantalainen, closes #5616)
Files: src/testdir/test_xxd.vim, src/xxd/xxd.c

Patch 8.2.0855

Problem: GUI tests fail because the test doesn't use a modifier.
Solution: Add "\{xxx}" to be able to encode a modifier.
Files: runtime/doc/eval.txt, src/typval.c, src/misc2.c, src/vim.h, src/proto/misc2.pro, src/gui_mac.c, src/option.c, src/highlight.c, src/term.c, src/testdir/test_backspace_opt.vim, src/testdir/test_mapping.vim, src/testdir/test_messages.vim

Patch 8.2.0856 (after 8.2.0852)

Problem: CTRL-S stops output.
Solution: Invert the IXON flag. (closes #6166)
Files: src/os_unix.c

Patch 8.2.0857

Problem: GTK cell height can be a pixel too much.
Solution: Subtract 3 instead of 1 when rounding. (closes #6168)
Files: src/gui_gtk_x11.c

Patch 8.2.0858

Problem: Not easy to require Lua modules.
Solution: Improve use of Lua path. (Prabir Shrestha, closes #6098)
Files: Filelist, src/if_lua.c, src/optionstr.c, src/proto/if_lua.pro,
src/testdir/test_lua.vim,
src/testdir/testluaplugin/luaplugin/hello.lua,
src/testdir/testluaplugin/luaplugin/init.lua

Patch 8.2.0859

Problem: No Turkish translation of the manual.
Solution: Add Turkish translations. (Emir Sarı, closes #5641)
Files: Filelist, runtime/doc/Makefile, runtime/doc/evim-tr.1,
runtime/doc/evim-tr.UTF-8.1, runtime/doc/vim-tr.1,
runtime/doc/vim-tr.UTF-8.1, runtime/doc/vimdiff-tr.1,
runtime/doc/vimdiff-tr.UTF-8.1, runtime/doc/vimtutor-tr.1,
runtime/doc/vimtutor-tr.UTF-8.1, src/Makefile

Patch 8.2.0860

Problem: Cannot use **CTRL-A** and **CTRL-X** on unsigned numbers.
Solution: Add "unsigned" to '**nrformats**'. (Naruhiko Nishino, closes #6144)
Files: runtime/doc/options.txt, src/ops.c, src/optionstr.c,
src/testdir/test_increment.vim

Patch 8.2.0861

Problem: Cannot easily get all the current marks.
Solution: Add getmarklist(). (Yegappan Lakshmanan, closes #6032)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/mark.c, src/proto/mark.pro, src/testdir/test_marks.vim

Patch 8.2.0862

Problem: ":term ++curwin" makes the current buffer hidden. (Harm te Hennepe)
Solution: Do not hide the current buffer. (closes #6170)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.0863

Problem: Cannot set a separate color for underline/undercurl.
Solution: Add the t_AU and t_8u termcap codes. (Timur Celik, closes #6011)
Files: runtime/doc/syntax.txt, runtime/doc/term.txt, src/globals.h,
src/highlight.c, src/optiondefs.h, src/proto/term.pro,
src/screen.c, src/structs.h, src/term.c, src/term.h,
src/testdir/test_options.vim

Patch 8.2.0864

Problem: Pragmas are indented all the way to the left.
Solution: Add an option to indent pragmas like normal code. (Max Rumpf, closes #5468)
Files: runtime/doc/indent.txt, src/cindent.c, src/structs.h,
src/testdir/test_cindent.vim

Patch 8.2.0865

Problem: Syntax foldlevel is taken from the start of the line.
Solution: Add ":syn foldlevel" to be able to use the minimal foldlevel in the line. (Brad King, closes #6087)
Files: runtime/doc/syntax.txt, src/structs.h, src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.0866

Problem: Not enough tests for buffer writing.
Solution: Add more tests. Use CheckRunVimInTerminal in more places. (Yegappan Lakshmanan, closes #6167)
Files: src/testdir/test_arglist.vim, src/testdir/test_match.vim, src/testdir/test_messages.vim, src/testdir/test_netbeans.py, src/testdir/test_netbeans.vim, src/testdir/test_search.vim, src/testdir/test_signals.vim, src/testdir/test_signs.vim, src/testdir/test_startup.vim, src/testdir/test_startup_utf8.vim, src/testdir/test_syntax.vim, src/testdir/test_tabpage.vim, src/testdir/test_timers.vim, src/testdir/test_vimscript.vim, src/testdir/test_writefile.vim

Patch 8.2.0867

Problem: Using {\xxx} for encoding a modifier is not nice.
Solution: Use \<xxx> instead, since it's the same as \<xxx> but producing a different code.
Files: runtime/doc/eval.txt, src/typval.c, src/misc2.c, src/vim.h, src/testdir/test_backspace_opt.vim, src/testdir/test_mapping.vim, src/testdir/test_messages.vim

Patch 8.2.0868

Problem: trim() always trims both ends.
Solution: Add an argument to only trim the beginning or end. (Yegappan Lakshmanan, closes #6126)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.2.0869

Problem: It is not possible to customize the quickfix window contents.
Solution: Add '**quickfixtextfunc**'. (Yegappan Lakshmanan, closes #5465)
Files: runtime/doc/eval.txt, runtime/doc/options.txt, runtime/doc/quickfix.txt, src/option.h, src/optiondefs.h, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.0870

Problem: MS-Windows: Control keys don't work in the GUI.
Solution: Don't set seenModifyOtherKeys for now. (Yasuhiro Matsumoto, closes #6175)
Files: src/gui.c

Patch 8.2.0871

Problem: Cannot use getmarklist() as a method.
Solution: Make getmarklist() work as a method. Add one to the column number to match getpos(). (Yegappan Lakshmanan, closes #6176)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/mark.c, src/testdir/test_marks.vim

Patch 8.2.0872

Problem: XIM code is mixed with multibyte code.

Solution: Move the XIM code to a separate file. (Yegappan Lakshmanan, closes #6177)

Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/gui_xim.c, src/mbyte.c, src/proto.h, src/proto/gui_xim.pro, src/proto/mbyte.pro

Patch 8.2.0873

Problem: A .jl file can be sawfish (lisp) or Julia.

Solution: Do not recognize *.jl as lisp, since it might be Julia. (closes #6178)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0874

Problem: Signals test is a bit flaky.

Solution: Flush the XautoOut file. Delete files that may be left behind from a failure. (Dominique Pellé, closes #6179)

Files: src/testdir/test_signals.vim

Patch 8.2.0875

Problem: Getting attributes for directory entries is slow.

Solution: Add readdirex(). (Ken Takata, closes #5619)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/fileio.c, src/filepath.c, src/proto/fileio.pro, src/proto/filepath.pro, src/testdir/test_functions.vim

Patch 8.2.0876

Problem: :pwd does not give a hint about the scope of the directory

Solution: Make ":verbose pwd" show the scope. (Takuya Fujiwara, closes #5469)

Files: runtime/doc/editing.txt, src/ex_docmd.c, src/testdir/test_cd.vim

Patch 8.2.0877

Problem: Cannot get the search statistics.

Solution: Add the searchcount() function. (Fujiwara Takuya, closes #4446)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/macros.h, src/proto/search.pro, src/search.c, src/testdir/test_search_stat.vim

Patch 8.2.0878

Problem: No reduce() function.

Solution: Add a reduce() function. (closes #5481)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/globals.h, src/list.c, src/proto/list.pro, src/testdir/test_listdict.vim

Patch 8.2.0879

Problem: Compiler warning for unused function argument.

Solution: Add UNUSED.

Files: src/search.c

Patch 8.2.0880 (after 8.2.0877)

Problem: Leaking memory when using searchcount().

Solution: Free the last used search pattern.
Files: src/search.c

Patch 8.2.0881

Problem: Compiler warning for argument type.
Solution: Add type cast. (Mike Williams)
Files: src/ops.c

Patch 8.2.0882

Problem: Leaking memory when using reduce().
Solution: Free the intermediate value.
Files: src/list.c

Patch 8.2.0883

Problem: Memory leak in test 49.
Solution: Free "sfile" from the exception.
Files: src/ex_docmd.c

Patch 8.2.0884

Problem: Searchcount() test fails on slower systems.
Solution: Set a longer timeout.
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.2.0885

Problem: "make shadow" does not link new lua test dir.
Solution: Also link testdir/testluaplugin. (Elimar Riesebieter)
Files: src/Makefile

Patch 8.2.0886

Problem: Cannot use octal numbers in scriptversion 4.
Solution: Add the "0o" notation. (Ken Takata, closes #5304)
Files: runtime/doc/eval.txt, src/charset.c, src/evalfunc.c,
src/testdir/test_eval_stuff.vim, src/testdir/test_functions.vim,
src/vim.h

Patch 8.2.0887

Problem: Searchcount().exact_match is 1 right after a match.
Solution: Use LT_POS() instead of LTREQ_POS(). (closes #6189)
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.2.0888

Problem: Readdirx() returns size -2 for a directory.
Solution: Add missing "else". (Ken Takata, closes #6185)
Files: src/fileio.c, src/testdir/test_functions.vim

Patch 8.2.0889

Problem: Using old style comments.
Solution: Use // comments. (Yegappan Lakshmanan, closes #6190)
Files: src/gui_xim.c

Patch 8.2.0890

Problem: No color in terminal window when 'termguicolors' is set.
Solution: Clear the underline color. (closes #6186)
Files: src/highlight.c

Patch 8.2.0891

Problem: Clang warns for invalid conversion.
Solution: Use zero instead of INVALIDCOLOR.
Files: src/highlight.c

Patch 8.2.0892

Problem: Ubsan warns for undefined behavior.
Solution: Use unsigned instead of signed variable. (Dominique Pellé,
closes #6193)
Files: src/regexp_nfa.c

Patch 8.2.0893

Problem: Assert_equalfile() does not take a third argument.
Solution: Implement the third argument. (Gary Johnson)
Files: runtime/doc/eval.txt, runtime/doc/testing.txt, src/evalfunc.c,
src/testdir/test_assert.vim, src/testing.c

Patch 8.2.0894

Problem: :mkspell can take very long if the word count is high.
Solution: Use long to avoid negative numbers. Increase the limits by 20% if
the compression did not have effect.
Files: src/spellfile.c

Patch 8.2.0895

Problem: :mkspell output does not mention the tree type.
Solution: Back out increasing the limits, it has no effect. Mention the
tree being compressed. Only give a message once per second.
Files: src/spellfile.c

Patch 8.2.0896

Problem: Crash when calling searchcount() with a string.
Solution: Check the argument is a dict. (closes #6192)
Files: src/search.c, src/testdir/test_search_stat.vim

Patch 8.2.0897

Problem: List of functions in patched version is outdated.
Solution: Update the function lists only.
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt

Patch 8.2.0898

Problem: Missing help for a function goes unnoticed.
Solution: Add a test. (Gary Johnson)
Files: src/testdir/test_function_lists.vim, src/testdir/Make_all.mak

Patch 8.2.0899

Problem: Assert_equalfile() does not give a hint about the difference.
Solution: Display the last seen text.
Files: src/testing.c, src/testdir/test_assert.vim

Patch 8.2.0900

Problem: Function list test fails on MS-Windows.
Solution: Make sure the fileformat is "unix"
Files: src/testdir/test_function_lists.vim

Patch 8.2.0901

Problem: Formatting CJK text isn't optimal.
Solution: Properly break CJK lines. (closes #3875)
Files: runtime/doc/change.txt, src/mbyte.c, src/ops.c, src/option.h,
src/proto/mbyte.pro, src/testdir/Make_all.mak, src/textformat.c,
src/testdir/test_cjk_linebreak.vim

Patch 8.2.0902

Problem: Using searchcount() in '**statusline**' causes an error.
Solution: Avoid saving/restoring the search pattern recursively.
(closes #6194)
Files: src/search.c, src/testdir/test_search_stat.vim,
src/testdir/dumps/Test_searchstat_4.dump

Patch 8.2.0903

Problem: comparing WINVER does not work correctly.
Solution: Use arithmetic expansion. (Ozaki Kiichi, closes #6197)
Files: src/Make_cyg_ming.mak

Patch 8.2.0904

Problem: Assuming modifyOtherKeys for rhs of mapping.
Solution: Ignore seenModifyOtherKeys for mapped characters. (closes #6200)
Files: src/getchar.c, src/testdir/test_gui.vim

Patch 8.2.0905

Problem: Test coverage could be better.
Solution: Add a couple of tests. (Dominique Pellé, closes #6202)
Files: src/testdir/test_cmdline.vim, src/testdir/test_ga.vim

Patch 8.2.0906

Problem: When setting '**termguicolors**' SpellBad is no longer red.
Solution: Only use the RGB guisp color for cterm when using the "underline"
or "undercurl" attributes to avoid the background color to be
cleared. Also make t_8u empty when the termresponse indicates a
real xterm. (closes #6207)
Files: src/highlight.c, src/term.c

Patch 8.2.0907

Problem: When using :global clipboard isn't set correctly.
Solution: Set "clip_unnamed_saved" instead of "clip_unnamed". (Christian
Brabandt, closes #6203, closes #6198)
Files: src/clipboard.c, src/testdir/test_global.vim

Patch 8.2.0908

Problem: Crash when changing the function table while listing it.
Solution: Bail out when the function table changes. (closes #6209)
Files: src/userfunc.c, src/testdir/test_timers.vim

Patch 8.2.0909

Problem: Cannot go back to the previous local directory.
Solution: Add "tcd -" and "lcd -". (Yegappan Lakshmanan, closes #4362)
Files: runtime/doc/editing.txt, src/filepath.c, src/ex_docmd.c,
src/structs.h, src/testdir/test_cd.vim, src/window.c

Patch 8.2.0910

Problem: Vim is not reproducibly buildable.
Solution: Use the \$SOURCE_DATE_EPOCH environment variable in configure.
(James McCoy, closes #513) Give a warning about using it.
Files: src/config.h.in, src/config.mk.in, src/configure.ac,
src/auto/configure, src/version.c, src/Makefile

Patch 8.2.0911

Problem: Crash when opening a buffer for the cmdline window fails. (Chris Barber)
Solution: Check do_ecmd() succeeds. Reset got_int if "q" was used at the more prompt. (closes #6211)
Files: src/ex_getln.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_cmdwin_interrupted.dump

Patch 8.2.0912

Problem: A few test cases for CJK formatting are disabled.
Solution: Fix the tests and enable them. (closes #6212)
Files: src/testdir/test_cjk_linebreak.vim

Patch 8.2.0913

Problem: Code for resetting v:register is duplicated.
Solution: Add reset_reg_var().
Files: src/evalvars.c, src/proto/evalvars.pro, src/main.c, src/normal.c

Patch 8.2.0914

Problem: MS-Windows: cannot specify a "modified by" text.
Solution: Add MODIFIED_BY in the MSVC build file. (Chen Lei, closes #1275)
Files: src/Make_mvc.mak

Patch 8.2.0915

Problem: Search() cannot skip over matches like searchpair() can.
Solution: Add an optional "skip" argument. (Christian Brabandt, closes #861)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/testdir/test_syntax.vim,
src/structs.h, src/evalvars.c, src/proto/evalvars.pro

Patch 8.2.0916

Problem: Mapping with partly modifyOtherKeys code does not work.
Solution: If there is no mapping with a separate modifier include the modifier in the key and then try mapping again. (closes #6200)
Files: src/getchar.c, src/proto/getchar.pro, src/edit.c, src/term.c,
src/proto/term.pro, src/testdir/test_termcodes.vim

Patch 8.2.0917

Problem: Quickfix entries do not support a "note" type.
Solution: Add support for "note". (partly by Yegappan Lakshmanan, closes #5527, closes #6216)
Files: runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.2.0918

Problem: Duplicate code for evaluating expression argument.
Solution: Merge the code and make the use more flexible.

Files: src/evalfunc.c, src/eval.c, src/proto/eval.pro, src/evalvars.c,
src/proto/evalvars.pro, src/structs.h

Patch 8.2.0919

Problem: Merging modifier for modifyOtherKeys is done twice.
Solution: Remove the merging done in vgetc().
Files: src/getchar.c, src/ex_getln.c

Patch 8.2.0920

Problem: Writing viminfo fails with a circular reference.
Solution: Use copyID to detect the cycle. (closes #6217)
Files: src/testdir/test_viminfo.vim, src/viminfo.c

Patch 8.2.0921

Problem: **CTRL-W** T in cmdline window causes trouble.
Solution: Disallow **CTRL-W** T in the cmdline window. Add more tests.
(Naruhiko Nishino, closes #6219)
Files: src/testdir/test_cmdline.vim, src/window.c

Patch 8.2.0922

Problem: Search test fails.
Solution: Remove failure tests for calls that no longer fail.
Files: src/testdir/test_search.vim

Patch 8.2.0923

Problem: Cmdline test is slow.
Solution: Use WaitForAssert().
Files: src/testdir/test_cmdline.vim

Patch 8.2.0924

Problem: Cannot save and restore a register properly.
Solution: Add getreginfo() and make setreg() accept a dictionary. (Andy
Massimino, closes #3370)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/proto/register.pro, src/register.c,
src/testdir/test_eval_stuff.vim, src/testdir/test_registers.vim

Patch 8.2.0925

Problem: Getcompletion() does not return command line arguments.
Solution: Add the "cmdline" option. (Shougo, closes #1140)
Files: runtime/doc/eval.txt, src/cmdexpand.c,
src/testdir/test_cmdline.vim

Patch 8.2.0926

Problem: Cmdline test fails on Appveyor.
Solution: Add CR to the commands. (Naruhiko Nishino, closes #6220)
Files: src/testdir/test_cmdline.vim

Patch 8.2.0927

Problem: Some sshconfig and ssdconfig files are not recognized.
Solution: Add filetype patterns.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0928

Problem: Many type casts are used for vim_strnsave().
Solution: Make the length argument size_t instead of int. (Ken Takata, closes #5633) Remove some type casts.
Files: src/misc2.c, src/proto/misc2.pro, src/autocmd.c, src/channel.c, src/cmdexpand.c, src/dict.c, src/diff.c, src/digraph.c, src/eval.c, src/evalfunc.c, src/highlight.c, src/syntax.c

Patch 8.2.0929

Problem: v:register is not cleared after an operator was executed.
Solution: Clear v:register after finishing an operator (Andy Massimino, closes #5305)
Files: src/normal.c, src/testdir/test_registers.vim

Patch 8.2.0930

Problem: Script filetype detection trips over env -S argument.
Solution: Remove "-S" and "--ignore-environment". (closes #5013)
Add tests.
Files: runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.0931

Problem: Some remarks about BeOS remain.
Solution: Remove BeOS remarks from the help and other files. (Emir Sarı, closes #6221)
Files: READMEdir/README_extra.txt, runtime/doc/options.txt, runtime/doc/os_beos.txt, runtime/doc/os_vms.txt, runtime/doc/vi_diff.txt, src/INSTALL

Patch 8.2.0932

Problem: Misspelling spelllang.
Solution: Add an "l". (Dominique Pellé)
Files: src/optionstr.c, src/proto/spell.pro, src/spell.c

Patch 8.2.0933

Problem: **'quickfixtextfunc'** does not get window ID of location list.
Solution: Add "winid" to the dict argument. (Yegappan Lakshmanan, closes #6222)
Files: runtime/doc/quickfix.txt, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.0934

Problem: Running lhelgrep twice in a help window doesn't jump to the help topic.
Solution: Check whether any window with the location list is present. (Yegappan Lakshmanan, closes #6215)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.0935

Problem: Flattening a list with existing code is slow.
Solution: Add flatten(). (Mopp, closes #3676)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/list.c, src/proto/list.pro, src/testdir/Make_all.mak, src/testdir/test_flatten.vim

Patch 8.2.0936

Problem: Some terminals misinterpret the code for getting cursor style.
Solution: Send a sequence to the terminal and check the result. (IWAMOTO Kouichi, closes #2126) Merged with current code.
Files: src/main.c, src/term.c, src/proto/term.pro,
src/testdir/term_util.vim, src/testdir/test_quickfix.vim,
src/testdir/test_terminal.vim, src/testdir/test_startup_utf8.vim,
src/testdir/dumps/Test_balloon_eval_term_01.dump,
src/testdir/dumps/Test_balloon_eval_term_01a.dump,
src/testdir/dumps/Test_balloon_eval_term_02.dump,
src/testdir/dumps/Test_terminal_all_ansi_colors.dump

Patch 8.2.0937

Problem: Asan failure in the flatten() test.
Solution: Free the flattened list.
Files: src/list.c

Patch 8.2.0938

Problem: NFA regexp uses tolower() to compare ignore-case. (Thayne McCombs)
Solution: Use utf_fold() when possible. (ref. neovim #12456)
Files: src/macros.h, src/diff.c, src/regexp_nfa.c,
src/testdir/test_regexp_utf8.vim

Patch 8.2.0939

Problem: checking for term escape sequences is long and confusing
Solution: Refactor code into separate functions.
Files: src/term.c

Patch 8.2.0940 (after 8.2.0939)

Problem: Build failure with tiny features.
Solution: Add #ifdef. Add UNUSED. A bit more cleaning up.
Files: src/term.c

Patch 8.2.0941

Problem: Detecting terminal properties is unstructured.
Solution: Add a table with terminal properties. Set properties when a terminal is detected.
Files: src/term.c

Patch 8.2.0942

Problem: Expanding to local dir after homedir keeps "~/".
Solution: Adjust modify_fname(). (Christian Brabandt, closes #6205, closes #5979)
Files: src/filepath.c, src/testdir/test_fnamemodify.vim

Patch 8.2.0943

Problem: Displaying ^M or ^J depends on current buffer.
Solution: Pass the displayed buffer to transchar(). (closes #6225)
Files: src/drawline.c, src/charset.c, src/proto/charset.pro,
src/ex_cmds.c, src/gui_beval.c, src/message.c,
src/testdir/test_display.vim,
src/testdir/dumps/Test_display_unprintable_01.dump,
src/testdir/dumps/Test_display_unprintable_02.dump

Patch 8.2.0944

Problem: Xxd test leaves file behind.
Solution: Delete the file "XXDfile". (Christian Brabandt, closes #6228)
Files: src/testdir/test_xxd.vim

Patch 8.2.0945

Problem: Cannot use "z=" when 'spell' is off.
Solution: Make "z=" work even when 'spell' is off. (Christian Brabandt, Gary Johnson, closes #6227)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/spell.c, src/spellsuggest.c, src/testdir/test_spell.vim, src/globals.h

Patch 8.2.0946

Problem: Cannot use "q" to cancel a number prompt.
Solution: Recognize "q" instead of ignoring it.
Files: src/misc1.c, src/testdir/test_functions.vim

Patch 8.2.0947

Problem: Readdirx() doesn't handle broken link properly.
Solution: Small fixes to readdirx(). (Christian Brabandt, closes #6226, closes #6213)
Files: src/fileio.c, src/testdir/test_functions.vim

Patch 8.2.0948

Problem: Spell test fails.
Solution: Adjust expected text of the prompt.
Files: src/testdir/test_spell.vim

Patch 8.2.0949

Problem: Strptime() does not use DST.
Solution: Set the tm_isdst field to -1. (Tomáš Janoušek, closes #6230)
Files: src/time.c, src/testdir/test_functions.vim

Patch 8.2.0950

Problem: Tagjump test fails.
Solution: Adjust expected text of the prompt.
Files: src/testdir/test_tagjump.vim

Patch 8.2.0951

Problem: Search stat test has leftover from debugging.
Solution: Remove line that writes a file. (Christian Brabandt, closes #6224)
Files: src/testdir/test_search_stat.vim

Patch 8.2.0952

Problem: No simple way to interrupt Vim.
Solution: Add the SigUSR1 autocommand, triggered by SIGUSR1. (Jacob Hayes, closes #1718)
Files: runtime/doc/autocmd.txt, src/vim.h, src/autocmd.c, src/getchar.c, src/globals.h, src/os_unix.c, src/testdir/test_autocmd.vim

Patch 8.2.0953

Problem: Spell checking doesn't work for CamelCased words.
Solution: Add the "camel" value in the new option 'spelloptions'. (closes #1235)
Files: runtime/doc/options.txt, runtime/doc/spell.txt, src/optiondefs.h,

src/option.h, src/option.c, src/buffer.c, src/optionstr.c,
src/testdir/gen_opt_test.vim, src/testdir/test_spell.vim

Patch 8.2.0954

Problem: Not all desktop files are recognized.
Solution: Add the *.directory pattern. (Eisuke Kawashima, closes #3317)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0955 (after 8.2.0953)

Problem: Build fails.
Solution: Add missing struct change.
Files: src/structs.h

Patch 8.2.0956 (after 8.2.0953)

Problem: Spell test fails.
Solution: Add missing change the spell checking.
Files: src/spell.c

Patch 8.2.0957

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize one variable.
Files: src/spell.c

Patch 8.2.0958

Problem: Not sufficient testing for buffer writing.
Solution: Add a few tests. (Yegappan Lakshmanan, closes #6238)
Files: src/testdir/test_backup.vim, src/testdir/test_writefile.vim

Patch 8.2.0959

Problem: Using **'quickfixtextfunc'** is a bit slow.
Solution: Process a list of entries. (Yegappan Lakshmanan, closes #6234)
Files: runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.2.0960

Problem: Cannot use :import in legacy Vim script.
Solution: Support :import in any Vim script.
Files: src/vim9script.c, src/evalvars.c, src/userfunc.c,
src/testdir/test_vim9_script.vim

Patch 8.2.0961

Problem: MS-Windows: no completion for locales.
Solution: Use the directories in \$VIMRUNTIME/lang to complete locales.
(Christian Brabandt, closes 36248)
Files: src/cmdexpand.c, src/ex_cmds2.c, src/testdir/test_cmdline.vim

Patch 8.2.0962

Problem: Terminal test sometimes hangs on Travis.
Solution: Do show output for this test temporarily.
Files: src/testdir/Makefile

Patch 8.2.0963

Problem: Number increment/decrement does not work with **'virtualedit'**.
Solution: Handle coladd changing. (Christian Brabandt, closes #6240,

closes #923)
Files: runtime/doc/options.txt, runtime/doc/various.txt, src/ops.c,
src/testdir/test_increment.vim

Patch 8.2.0964

Problem: TextYankPost does not provide info about Visual selection.
Solution: Add the **'visual'** key in v:event. (closes #6249)
Files: runtime/doc/autocmd.txt, src/register.c,
src/testdir/test_autocmd.vim

Patch 8.2.0965

Problem: Has_funcundefined() is not used.
Solution: Delete the function. (Dominique Pellé, closes #6242)
Files: src/autocmd.c, src/proto/autocmd.pro

Patch 8.2.0966

Problem: **'shortmess'** flag "n" not used in two places.
Solution: Make use of the "n" flag consistent. (Nick Jensen, closes #6245,
closes #6244)
Files: src/bufwrite.c, src/proto/bufwrite.pro, src/buffer.c,
src/fileio.c, src/testdir/dumps/Test_popup_textprop_corn_5.dump,
src/testdir/dumps/Test_start_with_tabs.dump

Patch 8.2.0967

Problem: Unnecessary type casts for vim_strnsave().
Solution: Remove the type casts.
Files: src/evalvars.c, src/ex_cmds.c, src/ex_eval.c, src/fileio.c,
src/filepath.c, src/findfile.c, src/highlight.c, src/if_ruby.c,
src/insexpand.c, src/json.c, src/mark.c, src/memline.c,
src/menu.c, src/misc1.c, src/ops.c, src/os_win32.c, src/regexp.c,
src/regexp_bt.c, src/regexp_nfa.c, src/register.c, src/search.c,
src/sign.c, src/syntax.c, src/term.c, src/terminal.c, src/undo.c,
src/usercmd.c, src/userfunc.c, src/vim9compile.c, src/if_perl.xs

Patch 8.2.0968

Problem: No proper testing of the **'coptions'** flags.
Solution: Add tests. (Yegappan Lakshmanan, closes #6251)
Files: src/testdir/Make_all.mak, src/testdir/test_coptions.vim,
src/testdir/test_edit.vim, src/testdir/test_normal.vim

Patch 8.2.0969

Problem: Assert_equal() output for dicts is hard to figure out.
Solution: Only show the different items.
Files: src/testing.c, src/testdir/test_assert.vim

Patch 8.2.0970

Problem: Terminal properties are not available in Vim script.
Solution: Add the terminalprops() function.
Files: src/term.c, src/proto/term.pro, src/evalfunc.c, src/main.c,
src/testing.c, src/globals.h, src/testdir/test_termcodes.vim,
runtime/doc/usr_41.txt, runtime/doc/eval.txt,
runtime/doc/testing.txt

Patch 8.2.0971

Problem: Build with tiny features fails.
Solution: Add #ifdef.
Files: src/term.c

Patch 8.2.0972

Problem: Vim9 script variable declarations need a type.
Solution: Make "let var: type" declare a script-local variable.
Files: src/evalvars.c, src/vim9script.c, src/proto/vim9script.pro,
src/globals.h, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.0973

Problem: Vim9: type is not checked when assigning to a script variable.
Solution: Check the type.
Files: src/evalvars.c, src/vim9script.c, src/proto/vim9script.pro,
src/vim9compile.c, src/proto/vim9compile.pro,
src/testdir/test_vim9_script.vim

Patch 8.2.0974

Problem: Vim9: memory leak when script var has wrong type.
Solution: Free the variable name.
Files: src/vim9script.vim

Patch 8.2.0975

Problem: Vim9: script variable does not accept optional s: prefix.
Solution: Adjust the accepted syntax.
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.0976

Problem: Some '**coptions**' not tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6253)
Files: src/testdir/test_cd.vim, src/testdir/test_charsearch.vim,
src/testdir/test_coptions.vim, src/testdir/test_normal.vim

Patch 8.2.0977

Problem: t_8u is made empty for the wrong terminals. (Dominique Pelle)
Solution: Invert the check for TPR_YES. (closes #6254)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.0978

Problem: Leaking memory in termcodes test.
Solution: Set t_8u with set_option_value().
Files: src/term.c

Patch 8.2.0979

Problem: A couple of screendump tests fail.
Solution: Do not redraw when clearing t_8u.
Files: src/term.c

Patch 8.2.0980

Problem: Raku file extension not recognized. (Steven Penny)
Solution: Recognize .raku and .rakumod. (closes #6255)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0981

Problem: Vim9: cannot compile "[var, var] = list".
Solution: Implement list assignment.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/evalvars.c,
src/proto/evalvars.pro, src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.0982

Problem: Insufficient testing for reading/writing files.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6257)
Add "ui_delay" to test_override() and use it for the **CTRL-O** test.
Files: src/testing.c, src/globals.h, src/ui.c, runtime/doc/testing.txt,
src/testdir/test_autocmd.vim, src/testdir/test_edit.vim,
src/testdir/test_filechanged.vim, src/testdir/test_writefile.vim

Patch 8.2.0983

Problem: SConstruct file type not recognized.
Solution: Use python for SConstruct files. (Roland Hieber)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.0984

Problem: Not using previous window when closing a shell popup window.
Solution: Use "prevwin" if it was set. (closes #6267)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.0985

Problem: Simplify() does not remove slashes from "///path".
Solution: Reduce > 2 slashes to one. (closes #6263)
Files: src/findfile.c, src/testdir/test_functions.vim

Patch 8.2.0986 (after 8.2.0985)

Problem: MS-Windows: functions test fails.
Solution: Only simplify ///path on Unix.
Files: src/testdir/test_functions.vim

Patch 8.2.0987

Problem: Vim9: cannot assign to [var; var].
Solution: Assign rest of items to a list.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/list.c,
src/proto/list.pro, src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.0988

Problem: Getting directory contents is always case sorted.
Solution: Add sort options and v:collate. (Christian Brabandt, closes #6229)
Files: runtime/doc/eval.txt, runtime/doc/mlang.txt, src/auto/configure,
src/cmdexpand.c, src/config.h.in, src/configure.ac,
src/evalfunc.c, src/evalvars.c, src/ex_cmds2.c, src/fileio.c,
src/filepath.c, src/globals.h, src/proto/fileio.pro,
src/testdir/test_cmdline.vim, src/testdir/test_functions.vim,
src/vim.h

Patch 8.2.0989

Problem: Crash after resizing a terminal window. (August Masquelier)
Solution: Add check for valid row in libvterm. (closes #6273)
Files: src/libvterm/src/state.c, src/libvterm/src/screen.c

Patch 8.2.0990 (after 8.2.0988)

Problem: Using duplicate error number.
Solution: Use an unused error number. Add a test for it.
Files: src/globals.h, src/testdir/test_functions.vim

Patch 8.2.0991

Problem: Cannot get window type for autocmd and preview window.
Solution: Add types to win_gettype(). (Yegappan Lakshmanan, closes #6277)
Files: runtime/doc/eval.txt, src/evalwindow.c,
src/testdir/test_autocmd.vim, src/testdir/test_preview.vim

Patch 8.2.0992

Problem: Vim9: crash when using :import in the Vim command.
Solution: Give an error when using :import outside of a script.
(closes #6271)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim,
src/testdir/term_util.vim

Patch 8.2.0993

Problem: Vim9 script test fails with normal features.
Solution: Use :func instead of :def for now.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0994

Problem: Vim9: missing function causes compilation error.
Solution: Call test function indirectly.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.0995

Problem: Insufficient testing for the readdir() sort option.
Solution: Add a few more tests. (Christian Brabandt, closes #6278)
Files: src/testdir/test_functions.vim

Patch 8.2.0996

Problem: Using "aucmdwin" in win_gettype() is not ideal.
Solution: Rename to "autocmd".
Files: runtime/doc/eval.txt, src/evalwindow.c,
src/testdir/test_autocmd.vim

Patch 8.2.0997

Problem: Cannot execute a register containing line continuation.
Solution: Concatenate lines where needed. (Yegappan Lakshmanan,
closes #6272)
Files: runtime/doc/repeat.txt, src/register.c,
src/testdir/test_registers.vim

Patch 8.2.0998

Problem: Not all tag code is tested.
Solution: Add a few more test cases. (Yegappan Lakshmanan, closes #6284)
Files: src/testdir/test_tagjump.vim

Patch 8.2.0999

Problem: Moving to next sentence gets stuck on quote.
Solution: When moving to the next sentence doesn't result in moving, advance

a character and try again. (closes #6291)
Files: src/textobject.c, src/testdir/test_textobjects.vim

Patch 8.2.1000
Problem: Get error when leaving Ex mode with :visual and a CmdLineEnter autocommand was used.
Solution: Reset ex_pressedreturn. (closes #6293)
Files: src/ex_docmd.c, src/testdir/test_ex_mode.vim

Patch 8.2.1001
Problem: Vim9: crash with nested "if" and assignment.
Solution: Skip more of the assignment. Do not set ctx_skip when code is reachable.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1002
Problem: Test may fail when run directly.
Solution: Check if g:run_nr exists. (Christian Brabandt, closes #6285)
Files: src/testdir/term_util.vim

Patch 8.2.1003
Problem: Vim9: return type of sort() is too generic.
Solution: Get type from the first argument. (closes #6292)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1004
Problem: Line numbers below filler lines not always updated.
Solution: Don't break out of the win_line() loop too early. (Christian Brabandt, closes #6294, closes #6138)
Files: src/drawline.c, src/testdir/dumps/Test_diff_rnu_01.dump, src/testdir/dumps/Test_diff_rnu_02.dump, src/testdir/dumps/Test_diff_rnu_03.dump, src/testdir/test_diffmode.vim

Patch 8.2.1005
Problem: Vim9: using TRUE/FALSE/MAYBE for ctx_skip is confusing.
Solution: Use an enum value.
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1006
Problem: Vim9: require unnecessary return statement.
Solution: Improve the use of the had_return flag. (closes #6270)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_func.vim

Patch 8.2.1007
Problem: Completion doesn't work after ":r ++arg !".
Solution: Skip over "++arg". (Christian Brabandt, closes #6275, closes #6258)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.1008
Problem: Vim9: no test for disassembling newly added instructions.
Solution: Add a function and check disassembly.

Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.1009

Problem: Vim9: some failures not checked for.

Solution: Add test cases. Remove unused code.

Files: src/testdir/test_vim9_script.vim, src/vim9execute.c

Patch 8.2.1010

Problem: Build failure in libvterm with debug enabled. (John Little)

Solution: Use "->" instead of ".".

Files: src/libvterm/src/state.c

Patch 8.2.1011

Problem: Vim9: some code not tested.

Solution: Add a few more test cases. Reorder checks for clearer error.
Remove unreachable code.

Files: src/evalvars.c, src/vim9script.c, src/vim9execute.c,
src/proto/vim9script.pro, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1012

Problem: Vim9: cannot declare single character script variables.

Solution: Don't see "b:", "s:", etc. as namespace. Fix item size of
sn_var_vals.

Files: src/vim9script.c, src/scriptfile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1013

Problem: Channel tests can be a bit flaky.

Solution: Set the g:test_is_flaky flag in SetUp().

Files: src/testdir/test_channel.vim

Patch 8.2.1014

Problem: Using "name" for a string result is confusing.

Solution: Rename to "end".

Files: src/typval.c

Patch 8.2.1015

Problem: Popup filter gets key with modifier prepended when using
modifyOtherKeys.

Solution: Remove the shift modifier when it is included in the key, also
when the Alt or Meta modifier is used.

Files: src/term.c, src/misc2.c, src/testdir/test_popupwin.vim

Patch 8.2.1016

Problem: Vim9: test fails when channel feature is missing.

Solution: Process an :if command when skipping

Files: src/vim9compile.c

Patch 8.2.1017

Problem: Appveyor output doesn't show MinGW console features.

Solution: List the features of the console build.

Files: ci/appveyor.bat

Patch 8.2.1018

Problem: Typo in enum value. (James McCoy)
Solution: Fix the typo.
Files: src/vim9compile.c

Patch 8.2.1019

Problem: Mapping <M-S-a> does not work in the GUI.
Solution: Move the logic to remove the shift modifier to may_remove_shift_modifier() and also use it in the GUI.
Files: src/gui_gtk_x11.c, src/misc2.c, src/proto/misc2.pro, src/term.c

Patch 8.2.1020

Problem: Popupwin test fails in the GUI.
Solution: Send GUI byte sequence for <C-S-a>.
Files: src/testdir/test_popupwin.vim

Patch 8.2.1021

Problem: Ruby interface not tested enough.
Solution: Add a couple more tests. (Dominique Pellé, closes #6301)
Files: src/testdir/test_ruby.vim

Patch 8.2.1022

Problem: Various parts of code not covered by tests.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6300)
Files: src/testdir/test_blob.vim, src/testdir/test_coptions.vim, src/testdir/test_digraph.vim, src/testdir/test_edit.vim, src/testdir/test_iminsert.vim, src/testdir/test_paste.vim, src/testdir/test_prompt_buffer.vim, src/testdir/test_selectmode.vim, src/testdir/test_tabpage.vim, src/testdir/test_tagjump.vim, src/testdir/test_textformat.vim, src/testdir/test_viminfo.vim, src/testdir/test_virtualedit.vim, src/testdir/test_visual.vim

Patch 8.2.1023

Problem: Vim9: redefining a function uses a new index every time.
Solution: When redefining a function clear the contents and re-use the index.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/structs.h, src/eval.c, src/evalvars.c, src/vim9execute.c

Patch 8.2.1024

Problem: Vim9: no error for using "let g:var = val".
Solution: Add an error.
Files: src/evalvars.c, src/globals.h, src/structs.h, src/vim9compile.c, src/scriptfile.c, src/userfunc.c, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_func.vim

Patch 8.2.1025

Problem: Tabpage menu and tabline not sufficiently tested.
Solution: Add tests. (Yegappan Lakshmanan, closes #6307)
Files: src/testdir/test_digraph.vim, src/testdir/test_tabpage.vim

Patch 8.2.1026

Problem: Vim9: cannot break the line after "->".
Solution: Check for a continuation line after "->", "[" and ".". Ignore trailing white space.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1027

Problem: GUI: multibyte characters do not work in a terminal.
Solution: Do not assume a key is one byte. (closes #6304)
Files: src/gui_gtk_x11.c, src/gui_x11.c

Patch 8.2.1028

Problem: Vim9: no error for declaring buffer, window, etc. variable.
Solution: Give an error. Unify the error messages.
Files: src/evalvars.c, src/globals.h, src/vim9compile.c,
src/proto/vim9compile.pro, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1029

Problem: Vim9: cannot chain function calls with -> at line start.
Solution: Peek ahead for a following line starting with "->". (closes #6306)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1030

Problem: Reducing size of a terminal window may cause a crash.
Solution: Make sure the row and column don't become negative. (closes #6273)
Files: src/libvterm/src/state.c, src/libvterm/src/screen.c

Patch 8.2.1031

Problem: Build failure with Perl5.32.
Solution: Define a few more functions. (Felix Yan, closes #6310)
Files: src/if_perl.xs

Patch 8.2.1032

Problem: Error message for declaring a variable cannot be translated.
Solution: Enclose in _(). Make environment variable a separate message.
Files: src/globals.h, src/vim9compile.c

Patch 8.2.1033

Problem: Not easy to read the test time in the test output.
Solution: Align the times. Make slow tests bold.
Files: src/testdir/runtest.vim

Patch 8.2.1034

Problem: Compiler warning for uninitialized variables.
Solution: Add initializations. (John Marriott)
Files: src/vim9compile.c

Patch 8.2.1035

Problem: setreg() does not always clear the register.
Solution: Clear the register if the dict argument is empty. (Andy Massimino, closes #3370)
Files: src/evalfunc.c, src/testdir/test_registers.vim

Patch 8.2.1036

Problem: Popupwin test fails sometimes.
Solution: Use WaitForAssert() instead of a sleep.
Files: src/testdir/test_popupwin.vim

Patch 8.2.1037

Problem: Vim9: crash when using line continuation inside :def.
Solution: Check for no more lines available.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1038

Problem: Popupwin test fails.
Solution: Fix WaitForAssert() argument.
Files: src/testdir/test_popupwin.vim

Patch 8.2.1039

Problem: Cannot put NUL byte on clipboard.
Solution: Use the text length. (Christian Brabandt, closes #6312, closes #6149)
Files: src/winclip.c, src/testdir/test_registers.vim

Patch 8.2.1040

Problem: Not enough testing for movement commands.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6313)
Files: src/testdir/test_cursor_func.vim, src/testdir/test_functions.vim, src/testdir/test_gf.vim, src/testdir/test_normal.vim, src/testdir/test_options.vim, src/testdir/test_quickfix.vim

Patch 8.2.1041

Problem: Test summary is missing executed count.
Solution: Adjust pattern used for counting.
Files: src/testdir/summarize.vim

Patch 8.2.1042

Problem: Vim9: cannot put an operator on the next line.
Solution: Require a colon before a range to see if that causes problems.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/ex_docmd.c, src/globals.h, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1043

Problem: %a item in 'statusline' not tested.
Solution: Add a test. (Dominique Pellé, closes #6318)
Files: src/testdir/test_statusline.vim

Patch 8.2.1044

Problem: Not all systemd file types are recognized.
Solution: Match several more files. (Guido Cella, closes #6319)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1045

Problem: Vim9: line break before operator does not work.
Solution: Peek the next line for an operator.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1046

Problem: Insufficient tests for src/buffer.c.
Solution: Add more tests. Move comments related tests to a separate file.
(Yegappan Lakshmanan, closes #6325)
Files: src/testdir/Make_all.mak, src/testdir/test_buffer.vim,
src/testdir/test_cmdline.vim, src/testdir/test_comments.vim,
src/testdir/test_normal.vim, src/testdir/test_textformat.vim

Patch 8.2.1047

Problem: Vim9: script cannot use line continuation like in a :def function.
Solution: Pass the getline function pointer to the eval() functions. Use it
for addition and multiplication operators.
Files: src/vim.h, src/structs.h, src/globals.h, src/ex_eval.c,
src/eval.c, src/proto/eval.pro, src/dict.c, src/evalfunc.c,
src/evalvars.c, src/list.c, src/userfunc.c, src/scriptfile.c,
src/proto/scriptfile.pro, src/testdir/test_vim9_expr.vim

Patch 8.2.1048 (after 8.2.1047)

Problem: Build failure without the eval feature.
Solution: Add dummy typedef.
Files: src/structs.h

Patch 8.2.1049 (after 8.2.1047)

Problem: Vim9: leaking memory when using continuation line.
Solution: Keep a pointer to the continuation line in evalarg_T. Centralize
checking for a next command.
Files: src/structs.h, src/eval.c, src/proto/eval.pro, src/beval.c,
src/buffer.c, src/clientserver.c, src/evalvars.c, src/ex_docmd.c,
src/ex_eval.c, src/filepath.c, src/findfile.c, src/fold.c,
src/globals.h, src/if_ole.cpp, src/if_perl.xs, src/if_tcl.c,
src/map.c, src/quickfix.c, src/regexp.c, src/register.c,
src/screen.c, src/userfunc.c

Patch 8.2.1050 (after 8.2.1049)

Problem: Missing change in struct.
Solution: Add missing change.
Files: src/ex_cmds.h

Patch 8.2.1051

Problem: Crash when changing a list while using reduce() on it.
Solution: Lock the list. (closes #6330)
Files: src/list.c, src/testdir/test_listdict.vim

Patch 8.2.1052

Problem: Build failure with older compilers.
Solution: Move declaration to start of block.
Files: src/eval.c

Patch 8.2.1053

Problem: Insufficient testing for **'statusline'** and **'tabline'**.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6333)
Files: src/testdir/test_autocmd.vim, src/testdir/test_statusline.vim,
src/testdir/test_tabline.vim

Patch 8.2.1054

Problem: Not so easy to pass a lua function to Vim.
Solution: Convert a Lua function and closure to a Vim funcref. (Prabir Shrestha, closes #6246)
Files: runtime/doc/if_lua.txt, src/if_lua.c, src/proto/userfunc.pro, src/structs.h, src/testdir/test_lua.vim, src/userfunc.c

Patch 8.2.1055

Problem: No filetype set for pacman config files.
Solution: Recognize pacman.conf and *.hook. (Guido Cella, closes #6335)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1056

Problem: Wrong display when mixing match conceal and syntax conceal.
Solution: Adjust how conceal flags are used. (closes #6327, closes #6303)
Files: src/drawline.c, src/highlight.c, src/testdir/test_matchadd_conceal.vim

Patch 8.2.1057 (after 8.2.1054)

Problem: Cannot build with dynamic Lua.
Solution: Add dll variables.
Files: src/if_lua.c

Patch 8.2.1058

Problem: Multiline conceal causes display errors.
Solution: Do not allow conceal cross over EOL. (closes #6326, closes #4854, closes #6302)
Files: src/drawline.c, src/testdir/test_conceal.vim, src/testdir/test_diffmode.vim

Patch 8.2.1059

Problem: Crash when using :tabonly in an autocommand. (Yegappan Lakshmanan)
Solution: Do not allow the autocommand window to be closed.
Files: src/ex_docmd.c, src/window.c, src/globals.h, src/testdir/test_autocmd.vim

Patch 8.2.1060

Problem: Not all elinks files are recognized.
Solution: Just check for "elinks.conf". (Guido Cella, closes #6337)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1061

Problem: Insufficient testing for src/window.c.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6345)
Files: src/testdir/test_excmd.vim, src/testdir/test_gf.vim, src/testdir/test_options.vim, src/testdir/test_popupwin.vim, src/testdir/test_quickfix.vim, src/testdir/test_tabpage.vim, src/testdir/test_tagjump.vim, src/testdir/test_window_cmd.vim, src/window.c

Patch 8.2.1062

Problem: Vim9: no line break allowed inside "cond ? val1 : val2".
Solution: Check for operator after line break.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1063

Problem: Vim9: no line break allowed before || or &&
Solution: Check for operator after line break.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1064

Problem: Vim9: no line break allowed before comparators.
Solution: Check for comparator after line break.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1065

Problem: Vim9: no line break allowed inside a list.
Solution: Handle line break inside a list in Vim9 script.
Files: src/eval.c, src/proto/eval.pro, src/list.c, src/proto/list.pro,
src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_arglist.vim

Patch 8.2.1066

Problem: Lua arrays are zero based.
Solution: Make Lua arrays one based. (Prabir Shrestha, closes #6347)
Note: this is not backwards compatible.
Files: runtime/doc/if_lua.txt, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.1067

Problem: Expression "!expr->func()" does not work.
Solution: Apply plus and minus earlier. (closes #6348)
Files: src/eval.c, src/proto/eval.pro, src/evalvars.c, src/userfunc.c,
src/testdir/test_expr.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1068

Problem: Vim9: no line break allowed inside a dict.
Solution: Handle line break inside a dict in Vim9 script.
Files: src/eval.c, src/dict.c, src/proto/dict.pro,
src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1069

Problem: Vim9: fail to check for white space in list.
Solution: Add check for white space.
Files: src/list.c

Patch 8.2.1070

Problem: Vim9: leaking memory when lacking white space in dict.
Solution: Clear the typval.
Files: src/dict.c

Patch 8.2.1071

Problem: Vim9: no line break allowed inside a lambda.
Solution: Handle line break inside a lambda in Vim9 script.
Files: src/eval.c, src/proto/eval.pro, src/evalvars.c, src/userfunc.c,
src/proto/userfunc.pro, src/popupwin.c, src/vim9compile.c,
src/ex_eval.c, src/globals.h, src/structs.h,
src/testdir/test_vim9_expr.vim

Patch 8.2.1072

Problem: Missing libvterm test.
Solution: Sync with libvterm revision 768.
Files: src/libvterm/src/state.c, src/libvterm/t/63screen_resize.test

Patch 8.2.1073

Problem: Vim9: no line break allowed in () expression.
Solution: Skip a line break.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1074

Problem: Vim9: no line break allowed after some operators.
Solution: Skip a line break after the operator. Add eval_may_get_next_line() to simplify checking for a line break.
Files: src/eval.c, src/proto/eval.pro, src/dict.c, src/list.c, src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1075

Problem: Vim9: no line break allowed in :echo expression.
Solution: Skip linebreak.
Files: src/eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1076

Problem: Vim9: no line break allowed in :if expression.
Solution: Skip linebreak.
Files: src/eval.c, src/proto/eval.pro, src/evalvars.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1077

Problem: No enough test coverage for highlighting.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6351)
Files: runtime/doc/syntax.txt, src/testdir/test_cmdline.vim, src/testdir/test_highlight.vim, src/testdir/test_match.vim

Patch 8.2.1078

Problem: Highlight and match functionality together in one file.
Solution: Move match functionality to a separate file. (Yegappan Lakshmanan, closes #6352)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/highlight.c, src/match.c, src/proto.h, src/proto/highlight.pro, src/proto/match.pro

Patch 8.2.1079

Problem: Vim9: no line break allowed in a while loop.
Solution: Update stored loop lines when finding line breaks.
Files: src/structs.h, src/globals.h, src/eval.c, src/evalvars.c, src/ex_docmd.c, src/proto/ex_docmd.pro, src/testdir/test_vim9_cmd.vim

Patch 8.2.1080

Problem: Vim9: no line break allowed in a for loop.
Solution: Skip line breaks in for command.
Files: src/eval.c, src/ex_eval.c, src/proto/eval.pro, src/userfunc.c,

src/structs.h, src/globals.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.1081

Problem: Lua: cannot use table.insert() and table.remove().
Solution: Add the list functions. (Prabir Shrestha, closes #6353)
Files: runtime/doc/if_lua.txt, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.1082

Problem: Coverity complains about ignoring dict_add() return value.
Solution: Add (void).
Files: src/evalfunc.c

Patch 8.2.1083

Problem: Crash when using reduce() on a NULL list.
Solution: Only access the list when not NULL.
Files: src/list.c, src/testdir/test_listdict.vim

Patch 8.2.1084

Problem: Lua: registering function has useless code.
Solution: Remove clearing grow arrays.
Files: src/userfunc.c

Patch 8.2.1085

Problem: Coverity complains about ignoring dict_add() return value.
Solution: Add (void).
Files: src/register.c

Patch 8.2.1086

Problem: Possibly using freed memory when text properties used when changing indent of a line.
Solution: Compute the offset before calling ml_replace().
Files: src/indent.c

Patch 8.2.1087

Problem: Possible memory leak when file expansion fails.
Solution: Clear the grow array when returning FAIL. Use an error message instead of an empty string.
Files: src/filepath.c

Patch 8.2.1088

Problem: A very long translation might cause a buffer overflow.
Solution: Truncate the message if needed.
Files: src/fileio.c

Patch 8.2.1089

Problem: Coverity warns for pointer computation.
Solution: Avoid computing a pointer to invalid memory.
Files: src/spellfile.c

Patch 8.2.1090

Problem: May use NULL pointer when skipping over name.
Solution: Always set ll_name_end.
Files: src/eval.c

Patch 8.2.1091

Problem: No check if opening a pty works.
Solution: Check for invalid file descriptor.
Files: src/os_unix.c

Patch 8.2.1092

Problem: Not checking if saving for undo succeeds.
Solution: Bail out if u_savesub() returns FAIL.
Files: src/textprop.c

Patch 8.2.1093

Problem: Python: double free when adding item to dict fails.
Solution: Remove vim_free() call.
Files: src/if_py_both.h

Patch 8.2.1094

Problem: Dead code in libvterm.
Solution: Remove condition that is always true.
Files: src/libvterm/src/pen.c

Patch 8.2.1095

Problem: May use pointer after freeing it when text properties are used.
Solution: Update redo buffer before calling ml_replace().
Files: src/spellsuggest.c

Patch 8.2.1096

Problem: Vim9: return type of getqflist() is wrong.
Solution: Let the return type depend on the arguments. Also for getloclist(). (closes #6357)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1097

Problem: Highlight code not sufficiently tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #6359)
Files: src/testdir/test_filter_cmd.vim, src/testdir/test_highlight.vim

Patch 8.2.1098

Problem: Vim9: cannot use line break in :throw argument.
Solution: Check for line break.
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.1099

Problem: Vim9: cannot use line break in :cexpr argument.
Solution: Check for line break.
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.1100

Problem: Vim9: cannot use line break in :execute, :echomsg and :echoerr argument.
Solution: Check for line break.
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.1101

Problem: No error when using wrong arguments for setqflist() or

setloclist().
Solution: Check for the error.
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1102

Problem: Coverity gets confused by an unnecessary NULL check.
Solution: Remove the check for NULL.
Files: src/quickfix.c

Patch 8.2.1103

Problem: Coverity reports an unnecessary NULL check.
Solution: Remove the check for NULL.
Files: src/eval.c

Patch 8.2.1104

Problem: Coverity warns for possible NULL pointer use.
Solution: Check "pbyts" is not NULL.
Files: src/spellsuggest.c

Patch 8.2.1105

Problem: Insufficient test coverage for Lua.
Solution: Add tests. (Yegappan Lakshmanan, closes #6368) Fix uncovered memory leak. Avoid unnecessary copy/free.
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.1106

Problem: Crash when trying to use s: variable in typed command.
Solution: Don't use the script index when not set. (Ken Takata, closes #6366)
Files: src/vim9compile.c, src/testdir/test_vimscript.vim

Patch 8.2.1107

Problem: **'imactivatefunc'** and **'imstatusfunc'** are not used in the GUI.
Solution: Adjust the #ifdefs. (closes #6367)
Files: runtime/doc/options.txt, src/gui_xim.c, src/testdir/test_iminsert.vim

Patch 8.2.1108

Problem: Mouse left-right scroll is not supported in terminal window.
Solution: Implement mouse codes 6 and 7. (Trygve Aaberge, closes #6363)
Files: src/libvterm/src/mouse.c, src/mouse.c, src/terminal.c, src/testdir/mouse.vim, src/testdir/test_termcodes.vim

Patch 8.2.1109 (after 8.2.1106)

Problem: Still crashing when using s:variable.
Solution: Remove assignment. (Ken Takata)
Files: src/vim9compile.c

Patch 8.2.1110

Problem: Vim9: line continuation does not work in function arguments.
Solution: Pass "evalarg" to get_func_tv(). Fix seeing double quoted string as comment.
Files: src/userfunc.c, src/proto/userfunc.pro, src/eval.c, src/ex_eval.c, src/list.c, src/dict.c, src/proto/eval.pro,

src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.1111

Problem: Inconsistent naming of get_list_tv() and eval_dict().
Solution: Rename get_list_tv() to eval_list(). Similarly for eval_number(), eval_string(), eval_lit_string() and a few others.
Files: src/eval.c, src/list.c, src/proto/list.pro, src/vim9compile.c, src/typval.c, src/proto/typval.pro, src/vim9script.c, src/evalfunc.c, src/evalvars.c, src/proto/evalvars.pro, src/vim9execute.c

Patch 8.2.1112

Problem: Vim9: no line continuation allowed in method call.
Solution: Handle line continuation in expression before method call.
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1113

Problem: No test for verbose output of :call.
Solution: Add a test.
Files: src/testdir/test_user_func.vim

Patch 8.2.1114

Problem: Terminal test sometimes times out.
Solution: Split the test in two parts.
Files: src/testdir/Makefile, src/testdir/Make_all.mak, src/testdir/term_util.vim, src/testdir/test_terminal.vim, src/testdir/test_terminal2.vim

Patch 8.2.1115

Problem: Iminsert test fails when compiled with VIMDLL.
Solution: Change condition. (Ken Takata, closes #6376)
Files: src/testdir/test_iminsert.vim

Patch 8.2.1116

Problem: Vim9: parsing command checks for list twice.
Solution: Adjust how a command is parsed.
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1117

Problem: Coverity warns for using uninitialized field.
Solution: Initialize v_lock.
Files: src/if_lua.c

Patch 8.2.1118

Problem: Condition can never be true, dead code.
Solution: Remove the dead code.
Files: src/move.c

Patch 8.2.1119

Problem: Configure fails with Xcode 12 beta.
Solution: use "return" instead of "exit()". (Nico Weber, closes #6381)
Files: src/configure.ac, src/auto/configure

Patch 8.2.1120

Problem: Python code not tested properly.
Solution: Add more tests and convert old-style test into new-style test.
(Yegappan Lakshmanan, closes #6370)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test86.in, src/testdir/test86.ok,
src/testdir/test_python2.vim

Patch 8.2.1121

Problem: Command completion not working after ++arg.
Solution: Move skipping up. (Christian Brabandt, closes #6382)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.1122

Problem: Vim9: line continuation in dict member not recognized.
Solution: Check for line continuation.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1123

Problem: Python 3 test is old style.
Solution: Turn into new style test. (Yegappan Lakshmanan, closes #6385)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test87.in, src/testdir/test87.ok,
src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1124

Problem: Vim9: no line break allowed in :import command.
Solution: Skip over line breaks.
Files: src/vim9script.c, src/proto/vim9script.pro, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1125

Problem: Vim9: double quote can be a string or a comment.
Solution: Only support comments starting with # to avoid confusion.
Files: src/eval.c, src/proto/eval.pro, src/dict.c, src/list.c,
src/vim9script.c

Patch 8.2.1126

Problem: Vim9: using :copen causes an error.
Solution: Add flag LET_NO_COMMAND in set_var().
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1127

Problem: Vim9: getting a dict member may not work.
Solution: Clear the dict only after copying the item. (closes #6390)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1128

Problem: The write message mentions characters, but it's actually bytes.
Solution: Change "C" to "B" and "characters" to "bytes".
Files: runtime/doc/options.txt, src/fileio.c,
src/testdir/test_cscope.vim, src/testdir/test_netbeans.vim,
src/testdir/dumps/Test_diff_syntax_1.dump,

src/testdir/dumps/Test_long_file_name_1.dump,
src/testdir/dumps/Test_display_unprintable_01.dump,
src/testdir/dumps/Test_tselect_1.dump

Patch 8.2.1129

Problem: Vim9: bar not recognized after not compiled command.
Solution: Check for bar for commands where this is possible. (closes #6391)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1130

Problem: Vim9: bar not recognized after function call
Solution: Skip whitespace. (closes #6391)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1131

Problem: Vim9: error message for returning a value in a function that does not return anything is not clear.
Solution: Add a specific message.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1132

Problem: Vim9: return type of repeat() is not specific enough.
Solution: Return the type of the first argument. (closes #6395)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1133

Problem: Vim9: return type of add() is not specific enough.
Solution: Return the type of the first argument. (closes #6395)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1134

Problem: Vim9: getting a list member may not work.
Solution: Clear the list only after copying the item. (closes #6393)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1135

Problem: Vim9: getting a dict member may not work.
Solution: Clear the dict only after copying the item.
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1136

Problem: Vim9: return type of argv() is always any.
Solution: Use list<string> if there is no argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1137

Problem: Vim9: modifiers not cleared after compiling function.
Solution: Clear command modifiers. (closes #6396)
Files: src/vim9compile.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/testdir/test_vim9_func.vim,
src/testdir/dumps/Test_vim9_silent_echo.dump

Patch 8.2.1138

Problem: Vim9: return type of copy() and deepcopy() is any.

Solution: Use type of the argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1139 (after 8.2.1137)
Problem: Vim9: test for silent echo fails in some environments.
Solution: Use :function instead of :def.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1140
Problem: Vim9: return type of extend() is any.
Solution: Use type of the argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1141
Problem: Vim9: return type of filter() is any.
Solution: Use type of the argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1142
Problem: Vim9: return type of insert() is any.
Solution: Use type of the first argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1143
Problem: Vim9: return type of remove() is any.
Solution: Use the member type of the first argument, if known.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1144
Problem: Vim9: return type of reverse() is any.
Solution: Use the type of the first argument.
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1145
Problem: Vim9: "for" only accepts a list at compile time.
Solution: Also accept a list at runtime.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1146
Problem: Not enough testing for Python.
Solution: Add more tests. Fix uncovered problems. (Yegappan Lakshmanan,
closes #6392)
Files: src/if_py_both.h, src/if_python3.c, src/testdir/shared.vim,
src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1147
Problem: :confirm may happen in cooked mode. (Jason Franklin)
Solution: Switch to raw mode before prompting. (Brandon Pfeifer)
Files: src/message.c, src/testdir/test_excmod.vim

Patch 8.2.1148
Problem: Warning for using int instead of size_t.
Solution: Change "len" argument to size_t. (Mike Williams)

Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9script.c

Patch 8.2.1149

Problem: Vim9: :eval command not handled properly.

Solution: Compile the :eval command. (closes #6408)

Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1150

Problem: ml_get error when using Python. (Yegappan Lakshmanan)

Solution: Check the line number is not out of range. Call "Check" with "fromObj" instead of "from".

Files: src/if_py_both.h, src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1151

Problem: Insufficient test coverage for Python.

Solution: Add more test cases. (Yegappan Lakshmanan, closes #6415)

Files: src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1152

Problem: Vim9: function reference is missing script prefix.

Solution: Use the actual function name instead of the name searched for in the script context. (closes #6412)

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1153

Problem: Vim9: script test fails on some systems.

Solution: Return proper value from Compare().

Files: src/testdir/test_vim9_script.vim

Patch 8.2.1154

Problem: Vim9: crash when using imported function.

Solution: Check for a function type. Set the script context when calling a function. (closes #6412)

Files: src/evalvars.c, src/scriptfile.c, src/proto/scriptfile.pro, src/vim9execute.c, src/structs.h, src/testdir/test_vim9_script.vim

Patch 8.2.1155

Problem: Vim9: cannot handle line break inside lambda.

Solution: Pass the compilation context through. (closes #6407, closes #6409)

Files: src/structs.h, src/vim9compile.c, src/proto/vim9compile.pro, src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.1156

Problem: Vim9: No error for invalid command in compiled function.

Solution: Handle CMD_SIZE.

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1157

Problem: Vim9: dict.name is not recognized as an expression.

Solution: Recognize ".name". (closes #6418)

Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1158 (after 8.2.1155)

Problem: Build error.
Solution: Add missing change to globals.
Files: src/globals.h

Patch 8.2.1159

Problem: Vim9: no error for missing space after a comma.
Solution: Check for white space.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1160

Problem: Vim9: memory leak in allocated types.
Solution: Free the type pointers.
Files: src/vim9script.c, src/userfunc.c, src/vim9compile.c,
src/proto/vim9compile.pro

Patch 8.2.1161

Problem: Vim9: using freed memory.
Solution: Put pointer back in evalarg instead of freeing it.
Files: src/userfunc.c, src/vim9compile.c, src/eval.c, src/proto/eval.pro,
src/structs.h

Patch 8.2.1162

Problem: Crash when using a lambda.
Solution: Check for evalarg to be NULL.
Files: src/userfunc.c

Patch 8.2.1163 (after 8.2.1161)

Problem: Build error.
Solution: Add missing change to globals.
Files: src/globals.h

Patch 8.2.1164

Problem: Text cleared by checking terminal properties not redrawn. (Alexey Radkov)
Solution: Mark the screen characters as invalid. (closes #6422)
Files: src/screen.c, src/proto/screen.pro, src/term.c

Patch 8.2.1165

Problem: Insufficient testing for the Tcl interface.
Solution: Add more tests. (Yegappan Lakshmanan, closes #6423)
Files: src/testdir/test_tcl.vim

Patch 8.2.1166

Problem: Once mouse move events are enabled getchar() returns them.
Solution: Ignore K_MOUSEMOVE in getchar(). (closes #6424)
Files: runtime/doc/eval.txt, src/getchar.c

Patch 8.2.1167

Problem: Vim9: builtin function method call only supports first argument.
Solution: Shift arguments when needed. (closes #6305, closes #6419)
Files: src/evalfunc.c, src/vim9compile.c, src/vim9execute.c,
src/vim9.h, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1168

Problem: Wrong method argument for appendbuflineline().
Solution: Use FEARG_3.
Files: src/evalfunc.c

Patch 8.2.1169

Problem: Write NUL past allocated space using corrupted spell file.
(Markus Vervier)
Solution: Init "c" every time.
Files: src/spellfile.c

Patch 8.2.1170

Problem: Cursor off by one with block paste while '**virtualedit**' is "all".
Solution: Adjust condition. (Hugo Gualandi, closes #6430)
Files: src/register.c, src/testdir/test_registers.vim

Patch 8.2.1171

Problem: Possible crash when out of memory.
Solution: Check for NULL pointer. (Dominique Pellé, closes #6432)
Files: src/syntax.c

Patch 8.2.1172

Problem: Error messages when doing "make clean" in the runtime/doc or
src/tee directories.
Solution: Use "rm -f".
Files: runtime/doc/Makefile, src/tee/Makefile

Patch 8.2.1173

Problem: Tee doesn't build on some systems.
Solution: Include header files. (Dominique Pelle, closes #6431)
Files: src/tee/tee.c

Patch 8.2.1174

Problem: No test for the "recording @x" message.
Solution: Add a test. (Dominique Pellé, closes #6427)
Files: src/testdir/test_registers.vim

Patch 8.2.1175

Problem: Vim9: Cannot split a line before ".member".
Solution: Check for ".member" after line break.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1176

Problem: Vim9: not enough type checking in Vim9 script.
Solution: Use same type checking as in a :def function.
Files: src/vim9compile.c, src/proto/vim9compile.pro,
src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1177

Problem: Terminal2 test sometimes hangs in the GUI.
Solution: Move some tests to other files to further locate the problem.
Set the GUI to a fixed screen size.
Files: src/testdir/test_terminal.vim, src/testdir/test_terminal2.vim,

src/testdir/test_terminal3.vim, src/testdir/Make_all.mak,
src/testdir/runtest.vim

Patch 8.2.1178

Problem: Vim9: filter function recognized as command modifier, leading to a crash.

Solution: Clear cmdmod after freeing items. Do not recognize a command modifier followed by non-white space. (closes #6434)

Files: src/ex_docmd.c, src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1179

Problem: Test_termwinscroll() sometimes hangs in the GUI.

Solution: Skip the test in the GUI.

Files: src/testdir/test_terminal2.vim

Patch 8.2.1180

Problem: Build failure in small version.

Solution: Add #ifdef.

Files: src/ex_docmd.c

Patch 8.2.1181

Problem: Json code not fully tested.

Solution: Add more test coverage. (Dominique Pellé, closes #6433)

Files: src/testdir/test_json.vim

Patch 8.2.1182

Problem: Vim9: no check for whitespace after comma in lambda.

Solution: Give error if white space is missing.

Files: src/userfunc.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.1183

Problem: assert_fails() checks the last error message.

Solution: Check the first error, it is more relevant. Fix all the tests that rely on the old behavior.

Files: runtime/doc/testing.txt, src/message.c, src/globals.h,
src/testing.c, src/testdir/test_autocmd.vim,
src/testdir/test_buffer.vim, src/testdir/test_cd.vim,
src/testdir/test_channel.vim, src/testdir/test_clientserver.vim,
src/testdir/test_cmdline.vim, src/testdir/test_coptions.vim,
src/testdir/test_cscope.vim, src/if_cscope.c,
src/testdir/test_excmod.vim, src/evalvars.c,
src/testdir/test_expr.vim, src/testdir/test_functions.vim,
src/testdir/test_json.vim, src/testdir/test_let.vim,
src/testdir/test_listdict.vim, src/testdir/test_listener.vim,
src/testdir/test_match.vim, src/testdir/test_menu.vim,
src/testdir/test_method.vim, src/testdir/test_normal.vim,
src/testdir/test_popup.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_quickfix.vim,
src/testdir/test_random.vim, src/testdir/test_search.vim,
src/testdir/test_signs.vim, src/testdir/test_spell.vim,
src/testdir/test_substitute.vim, src/testdir/test_syntax.vim,
src/testdir/test_tagjump.vim, src/testdir/test_taglist.vim,
src/testdir/test_terminal.vim, src/testdir/test_textprop.vim,

src/testdir/test_trycatch.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim, src/vim9compile.c,
src/testdir/test_vim9_script.vim, src/testdir/test_viminfo.vim,
src/testdir/test_winbuf_close.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_writefile.vim,
src/testdir/test_regexp_latin.vim, src/testdir/test_utf8.vim,
src/testdir/test_global.vim, src/testdir/test_tagfunc.vim

Patch 8.2.1184 (after 8.2.1183)

Problem: Some tests fail.

Solution: Adjust tests for different assert_fails() behavior. Remove unused variable.

Files: src/testdir/test_assert.vim, src/testdir/test_eval_stuff.vim,
src/evalvars.c

Patch 8.2.1185 (after 8.2.1183)

Problem: Some other tests fail.

Solution: Adjust tests for different assert_fails() behavior.

Files: src/testdir/test_lua.vim, src/testdir/test_tcl.vim

Patch 8.2.1186

Problem: With SGR mouse codes balloon doesn't show up after click.

Solution: Add the MOUSE_RELEASE bits to mouse_code.

Files: src/mouse.c

Patch 8.2.1187

Problem: Terminal2 test sometimes hangs in the GUI on Travis.

Solution: Disable Test_zz2_terminal_guioptions_bang() for now.

Files: src/testdir/test_terminal2.vim

Patch 8.2.1188

Problem: Memory leak with invalid json input.

Solution: Free all keys at the end. (Dominique Pellé, closes #6443,
closes #6442)

Files: src/json.c, src/testdir/test_json.vim

Patch 8.2.1189

Problem: Vim9: line continuation in lambda doesn't always work.

Solution: Do not use a local evalarg unless there isn't one. (closes #6439)

Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1190

Problem: Vim9: checking for Vim9 syntax is spread out.

Solution: Use in_vim9script().

Files: src/vim9script.c, src/dict.c, src/eval.c, src/evalvars.c,
src/ex_docmd.c, src/list.c, src/scriptfile.c, src/userfunc.c

Patch 8.2.1191

Problem: Vim9: crash when function calls itself.

Solution: Add status UF_COMPILING. (closes #6441)

Files: src/structs.h, src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1192

Problem: Lua test fails with older Lua version.
Solution: Adjust expected error messages. (closes #6444)
Files: src/testdir/test_lua.vim

Patch 8.2.1193

Problem: Terminal window not redrawn when dragging a popup window over it.
Solution: Redraw terminal window. (fixes #6438)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_term_01.dump,
src/testdir/dumps/Test_popupwin_term_02.dump

Patch 8.2.1194

Problem: Test failure because shell prompt differs.
Solution: Set the shell prompt.
Files: src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_term_01.dump,
src/testdir/dumps/Test_popupwin_term_02.dump

Patch 8.2.1195

Problem: Clientserver test fails on MS-Windows.
Solution: Expect a different error message.
Files: src/testdir/test_clientserver.vim

Patch 8.2.1196

Problem: Build failure with normal features.
Solution: Add #ifdef.
Files: src/popupwin.c

Patch 8.2.1197

Problem: Clientserver test still fails on MS-Windows.
Solution: Expect a different error message.
Files: src/testdir/test_clientserver.vim

Patch 8.2.1198

Problem: Terminal2 test sometimes hangs in the GUI on Travis.
Solution: Move test function to terminal3 to see if the problem moves too.
Files: src/testdir/test_terminal2.vim, src/testdir/test_terminal3.vim

Patch 8.2.1199

Problem: Not all assert functions are fully tested.
Solution: Test more assert functions.
Files: src/testing.c, src/testdir/test_assert.vim

Patch 8.2.1200

Problem: Vim9: cannot disassemble a lambda function.
Solution: Recognize "<lambda>123" as a function name.
Files: src/vim9execute.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1201

Problem: Vim9: crash when passing number as dict key.
Solution: Check key type to be string. (closes #6449)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1202

Problem: Vim9: crash when calling a closure from a builtin function.
Solution: Use the current execution context. (closes #6441)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1203

Problem: Unused assignments in expression evaluation.
Solution: Move declarations and assignments to inner blocks where possible.
Files: src/eval.c

Patch 8.2.1204

Problem: Vim9: true and false not recognized in Vim9 script.
Solution: Recognize true and false.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1205

Problem: Vim9: && and || work differently when not compiled.
Solution: Keep the value.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1206

Problem: Vim9: crash in expr test when run in the GUI.
Solution: Temporarily comment out two test lines.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1207

Problem: Vim9: crash in expr test when run in the GUI.
Solution: Break out of loop over hashtable also when function got removed and added.
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1208

Problem: Build failure.
Solution: Add missing change.
Files: src/structs.h

Patch 8.2.1209

Problem: Vim9: test failure.
Solution: Add missing changes to hashtable.
Files: src/hashtab.c

Patch 8.2.1210

Problem: Using ht_used when looping through a hashtable is less reliable.
Solution: Use ht_changed in a few more places.
Files: src/userfunc.c, src/if_py_both.h

Patch 8.2.1211 (after 8.2.1118)

Problem: Removed more than dead code.
Solution: Put back the decrement.
Files: src/move.c, src/testdir/test_diffmode.vim

Patch 8.2.1212

Problem: Cannot build with Lua 5.4.
Solution: Use luaL_typeerror instead defining it. (closes #6454)
Files: src/if_lua.c

Patch 8.2.1213

Problem: Mouse codes not tested sufficiently.
Solution: Add more tests for mouse codes. (closes #6436)
Files: src/testdir/test_termcodes.vim

Patch 8.2.1214

Problem: MS-Windows: default _vimrc not correct in silent install mode.
Solution: Add the LoadDefaultVimrc macro. (Ken Takata, closes #6451)
Files: nsis/gvim.nsi

Patch 8.2.1215

Problem: Atari MiNT support is outdated.
Solution: Nobody responded this code is still useful, so let's delete it.
Files: Filelist, src/os_mint.h, src/vim.h, src/Make_mint.mak,
src/digraph.c, src/fileio.c, src/memfile.c, src/os_unix.c,
src/term.c, READMEdir/README_extra.txt, runtime/doc/os_mint.txt,
src/INSTALL

Patch 8.2.1216

Problem: Startup test fails.
Solution: Adjust expected values for deleted lines.
Files: src/testdir/test_startup.vim

Patch 8.2.1217

Problem: Startup test depends on random source file.
Solution: Write a test file to find quickfix errors in.
Files: src/testdir/test_startup.vim

Patch 8.2.1218

Problem: Vim9: cannot use **'text'**->func().
Solution: Recognize string at start of command.
Files: src/vim9compile.c, src/ex_docmd.c, src/testdir/test_vim9_func.vim

Patch 8.2.1219

Problem: Symlink not followed if dirname ends in //.
Solution: Resolve symlink earlier. (Tomáš Janoušek, closes #6454)
Files: src/memline.c, src/testdir/test_swap.vim

Patch 8.2.1220

Problem: memory access error when dragging a popup window over a buffer with folding.
Solution: Avoid going over the end of the cache. (closes #6438)
Files: src/mouse.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_term_01.dump,
src/testdir/dumps/Test_popupwin_term_02.dump,
src/testdir/dumps/Test_popupwin_term_03.dump,
src/testdir/dumps/Test_popupwin_term_04.dump

Patch 8.2.1221

Problem: Memory leak when updating popup window.
Solution: Clear search highlighting.
Files: src/popupwin.c

Patch 8.2.1222

Problem: When using valgrind a Vim command started by a test uses the same log file name which gets overwritten.
Solution: Fix regexp to rename the log file.
Files: src/testdir/shared.vim

Patch 8.2.1223

Problem: Vim9: invalid type error for function default value.
Solution: Use right argument index. (closes #6458)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1224

Problem: Vim9: arguments from partial are not used.
Solution: Put the partial arguments on the stack. (closes #6460)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1225

Problem: Linker errors when building with dynamic Python 3.9.
Solution: Add #defined items. (closes #6461)
Files: src/if_python3.c

Patch 8.2.1226

Problem: MS-Windows: windows positioning wrong when the taskbar is placed at the top or left of the screen.
Solution: Use GetWindowRect and MoveWindow APIs. (Yukihiro Nakadaira, Ken Takata, closes #6455)
Files: src/gui_w32.c

Patch 8.2.1227

Problem: Vim9: allowing both quoted and # comments is confusing.
Solution: Only support # comments in Vim9 script.
Files: runtime/doc/vim9.txt, src/ex_docmd.c, src/proto/ex_docmd.pro, src/vim9compile.c, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1228

Problem: Scrollbars not flush against the window edges when maximised.
Solution: Add padding. (Ken Takata, closes #5602, closes #6466)
Files: src/gui.c, src/gui_athena.c, src/gui_gtk.c, src/gui_haiku.cc, src/gui_mac.c, src/gui_motif.c, src/gui_photon.c, src/gui_w32.c, src/proto/gui_athena.pro, src/proto/gui_gtk.pro, src/proto/gui_haiku.pro, src/proto/gui_mac.pro, src/proto/gui_motif.pro, src/proto/gui_photon.pro, src/proto/gui_w32.pro

Patch 8.2.1229

Problem: Build error without the eval feature.
Solution: Declare starts_with_colon. Make function local.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 8.2.1230

Problem: Vim9: list index error not caught by try/catch.
Solution: Do not bail out if an error is inside try/catch. (closes #6462)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1231

Problem: MS-Windows: GUI code can be cleaned up.
Solution: Do a bit of cleaning up. (Ken Takata, closes #6465)
Files: src/gui_w32.c, src/proto/gui_w32.pro

Patch 8.2.1232

Problem: MS-Windows GUI: Snap cancelled by split command.
Solution: Do not cancel Snap when splitting a window. (Ken Takata, closes #6467)
Files: src/gui_w32.c

Patch 8.2.1233

Problem: Vim9: various errors not caught by try/catch.
Solution: Do not bail out if an error is inside try/catch.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1234

Problem: Lua build problem with old compiler.
Solution: Move declarations to start of the block. (Taro Muraoka, closes #6477)
Files: src/if_lua.c

Patch 8.2.1235

Problem: Not all mouse codes covered by tests.
Solution: Add more tests for the mouse. (Yegappan Lakshmanan, closes #6472)
Files: src/testdir/mouse.vim, src/testdir/test_termcodes.vim

Patch 8.2.1236

Problem: Vim9: a few errors not caught by try/catch.
Solution: Do not bail out if an error is inside try/catch. Fix that a not matching catch doesn't jump to :endtry.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1237

Problem: Changing **'completepopup'** after opening a popup has no effect. (Jay Sitter)
Solution: Close the popup when the options are changed. (closes #6471)
Files: runtime/doc/options.txt, src/popupwin.c, src/proto/popupwin.pro, src/optionstr.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_infopopup_8.dump

Patch 8.2.1238

Problem: Vim9: a few remaining errors not caught by try/catch.
Solution: Do not bail out if an error is inside try/catch.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1239

Problem: "maxwidth" in **'completepopup'** not obeyed. (Jay Sitter)
Solution: Add separate field for value from option. (closes #6470)
Files: src/structs.h, src/popupwin.c, src/popupmenu.c, src/testdir/dumps/Test_popupwin_infopopup_9.dump

Patch 8.2.1240

Problem: GUI tests sometimes fail because of translations.
Solution: Reload the menus without translation. (Taro Muraoka, closes #6486)
Files: src/testdir/runtest.vim

Patch 8.2.1241

Problem: Cannot use getbufinfo() as a method.
Solution: Support using getbufinfo() as a method. (closes #6458)
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_bufwintabinfo.vim

Patch 8.2.1242

Problem: Vim9: no error if calling a function with wrong argument type.
Solution: Check types of arguments. (closes #6469)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.1243

Problem: Vim9: cannot have a comment or empty line halfway a list at script level.
Solution: Skip more than one line if needed.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/eval.c,
src/scriptfile.c

Patch 8.2.1244

Problem: Vim9: in lambda index assumes a list.
Solution: Use the value type to decide about list or dict. (closes #6479)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1245

Problem: Build failure in tiny version.
Solution: Add #ifdef.
Files: src/scriptfile.c

Patch 8.2.1246

Problem: Vim9: comment after assignment doesn't work.
Solution: Skip over white space. (closes #6481)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1247

Problem: Vim9: cannot index a character in a string.
Solution: Add ISN_STRINDEX instruction. (closes #6478)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.1248

Problem: Netbeans test is flaky in the GUI.
Solution: Filter out geometry messages. (Taro Muraoka, closes #6487)
Files: src/testdir/test_netbeans.vim

Patch 8.2.1249

Problem: Vim9: disassemble test fails.
Solution: Change INDEX to LISTINDEX. Add test for STRINDEX.

Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.1250
Problem: Vim9: cannot use the g:, b:, t: and w: namespaces.
Solution: Add instructions to push a dict for the namespaces. (closes #6480)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1251
Problem: Vim9: warning for pointer usage, test failure undetected.
Solution: Fix pointer indirection. Give error when executing function failed for any reason. Fix instruction names.
Files: src/vim9execute.c, src/userfunc.c, src/proto/userfunc.pro

Patch 8.2.1252
Problem: ":marks" may show '<' and '>' mixed up.
Solution: Show the mark position as where '<' and '>' would jump.
Files: src/mark.c, src/testdir/test_marks.vim

Patch 8.2.1253
Problem: CTRL-K in Insert mode gets <CursorHold> inserted. (Roland Puntaier)
Solution: Do not reset did_cursorhold, restore it. (closes #6447)
Files: src/normal.c

Patch 8.2.1254
Problem: MS-Windows: regexp test may fail if 'iskeyword' set wrongly.
Solution: Override the 'iskeyword' value. (Taro Muraoka, closes #6502)
Files: src/testdir/test_regexp_utf8.vim

Patch 8.2.1255
Problem: Cannot use a lambda with quickfix functions.
Solution: Add support for lambda. (Yegappan Lakshmanan, closes #6499)
Files: runtime/doc/eval.txt, runtime/doc/options.txt,
runtime/doc/quickfix.txt, src/channel.c, src/evalvars.c,
src/optionstr.c, src/proto/evalvars.pro, src/proto/quickfix.pro,
src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1256
Problem: Vim9: type wrong after getting dict item in lambda.
Solution: Set the type to "any" after enforcing dict type. (closes #6491)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1257
Problem: Vim9: list unpack doesn't work at the script level.
Solution: Detect unpack assignment better. (closes #6494)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1258 (after 8.2.1253)
Problem: CursorHold does not work well.a (Shane-XB-Qian)
Solution: Only restore did_cursorhold when using :normal.
Files: src/normal.c

Patch 8.2.1259

Problem: Empty group in **'tabline'** may cause using an invalid pointer.
Solution: Set the group start position. (closes #6505)
Files: src/buffer.c, src/testdir/test_tabline.vim

Patch 8.2.1260

Problem: There is no good test for CursorHold.
Solution: Add a test. Remove duplicated test. (Yegappan Lakshmanan, closes #6503)
Files: src/testdir/test_autocmd.vim, src/testdir/test_buffer.vim, src/testdir/test_normal.vim

Patch 8.2.1261

Problem: Vim9: common type of function not tested.
Solution: Add a test. Fix uncovered problems.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1262

Problem: src/ex_cmds.c file is too big.
Solution: Move help related code to src/help.c. (Yegappan Lakshmanan, closes #6506)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/cmdexpand.c, src/ex_cmds.c, src/help.c, src/proto.h, src/proto/ex_cmds.pro, src/proto/help.pro

Patch 8.2.1263

Problem: Vim9: comparators use **'ignorecase'** in Vim9 script.
Solution: Ignore **'ignorecase'**. Use true and false instead of 1 and 0. (closes #6497)
Files: src/eval.c, src/typval.c, src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1264

Problem: Terminal getwinpos() test is a bit flaky.
Solution: Call getwinpos() a bit later.
Files: src/testdir/test_terminal3.vim

Patch 8.2.1265

Problem: Crash with EXITFREE when split() fails.
Solution: Restore **'coptions'**.
Files: src/evalfunc.c

Patch 8.2.1266 (after 8.2.1262)

Problem: Makefile preference were accidentally included.
Solution: Revert the Makefile changes.
Files: src/Makefile

Patch 8.2.1267

Problem: MS-Windows: tests may fail due to \$PROMPT value.
Solution: Set \$PROMPT for testing. (Taro Muraoka, closes #6510)
Files: src/testdir/runtest/vim

Patch 8.2.1268

Problem: Vim9: no error for using double quote comment after :func or :def.
Solution: Only accept double quote when not in Vim9 script and not after :def. (closes #6483)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1269

Problem: Language and locale code spread out.
Solution: Move relevant code to src/locale.c. (Yegappan Lakshmanan, closes #6509)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_morph.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/ex_cmds2.c, src/locale.c, src/main.c, src/proto.h, src/proto/ex_cmds2.pro, src/proto/locale.pro

Patch 8.2.1270

Problem: Vim9: not skipping over function type declaration with only a return type.
Solution: Skip over the return type. (issue #6507)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1271

Problem: Vim9: Error for Funcref function argument type.
Solution: Find the actual function type if possible. (issue #6507)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1272

Problem: Vim9: type not checked if declaration also assigns value.
Solution: Check the type. (issue #6507)
Files: src/eval.c, src/vim9compile.c, src/proto/vim9compile.pro, src/vim9script.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1273

Problem: MS-Windows: terminal test may leave file behind.
Solution: Wait a moment for process to end before deleting the file. (Taro Muraoka, closes #6513)
Files: src/testdir/test_terminal.vim

Patch 8.2.1274

Problem: Vim9: no error for missing white space in assignment at script level.
Solution: Check for white space. (closes #6495)
Files: src/eval.c, src/evalvars.c, src/testdir/test_vim9_script.vim, src/testdir/test_let.vim

Patch 8.2.1275

Problem: Vim9: compiler warning for buffer size.
Solution: Change the offset from 10 to 15. (Dominique Pellé, closes #6518)
Files: src/vim9script.c

Patch 8.2.1276

Problem: MS-Windows: system test may fail if more.exe is installed.
Solution: Explicitly use more.com. (Taro Muraoka, Ken Takata, closes #6517)

Files: src/testdir/test_system.vim

Patch 8.2.1277

Problem: Tests on Travis do not run with EXITFREE.

Solution: Add EXITFREE to all builds to uncover any mistakes.

Files: .travis.yml

Patch 8.2.1278

Problem: Vim9: line break after "->" only allowed in :def function.

Solution: Only allow line break after "->". (closes #6492)

Files: src/vim9compile.c, src/globals.h, src/testdir/test_vim9_expr.vim

Patch 8.2.1279

Problem: Some tests on Travis have EXITFREE duplicated.

Solution: Remove EXITFREE from shadowopt. Add "shadow" to job name.

Files: .travis.yml

Patch 8.2.1280

Problem: Ex command error cannot contain an argument.

Solution: Add ex_errmsg() and translate earlier. Use e_trailing_arg where possible.

Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/buffer.c, src/ex_eval.c, src/match.c, src/testdir/test_tabpage.vim

Patch 8.2.1281

Problem: The "trailing characters" error can be hard to understand.

Solution: Add the trailing characters to the message.

Files: src/cmdhist.c, src/eval.c, src/evalfunc.c, src/evalvars.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_eval.c, src/json.c, src/menu.c, src/quickfix.c, src/sign.c, src/userfunc.c

Patch 8.2.1282

Problem: Vim9: crash when using CheckScriptFailure() in Test_vim9script_call_fail_decl().

Solution: Do not decrement the def_functions len unless the function was newly added.

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1283

Problem: Vim9: error for misplaced -> lacks argument.

Solution: Use the pointer before it was advanced.

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1284

Problem: Vim9: skipping over type includes following white space, leading to an error for missing white space.

Solution: Do not skip over white space after the type.

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1285

Problem: Vim9: argument types are not checked on assignment.

Solution: Check function argument types. (issue #6507)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1286

Problem: Vim9: No error when using a type on a window variable
Solution: Recognize the syntax and give an error. (closes #6521)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1287

Problem: Vim9: crash when using an imported function.
Solution: Add the function type to the imported entry. (closes #6522)
Files: src/vim9script.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1288

Problem: Vim9: cannot use mark in range.
Solution: Use the flag that a colon was seen. (closes #6528)
Files: src/ex_docmd.c, src/testdir/test_vim9_func.vim

Patch 8.2.1289

Problem: Crash when using a custom completion function.
Solution: Initialize all of the expand_T. (closes #6532)
Files: src/cmdexpand.c

Patch 8.2.1290

Problem: Vim9: cannot replace a global function.
Solution: Allow for "!" on a global function. (closes #6524) Also fix that
:delfunc on a :def function only made it empty.
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1291

Problem: Vim9: type of varargs items is not checked.
Solution: Check the list item types. (closes #6523)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1292

Problem: AIDL filetype not recognized.
Solution: Add filetype detection. (Dominique Pellé, closes #6533)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1293

Problem: Vim9: :execute mixes up () expression and function call.
Solution: Do not skip white space when looking for the "(" . (closes #6531)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1294

Problem: Vim9: error when using vim9script in TextYankPost.
Solution: Use EX_LOCKOK instead of the EX_CMDWIN flag for command that can
be used when text is locked. (closes #6529)
Files: src/ex_cmds.h, src/ex_docmd.c

Patch 8.2.1295

Problem: Tests 44 and 99 are old style.
Solution: Convert to new style tests. (Yegappan Lakshmanan, closes #6536)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_vms.mms,
src/testdir/test44.in, src/testdir/test44.ok,
src/testdir/test99.in, src/testdir/test99.ok,

src/testdir/test_regexp_utf8.vim

Patch 8.2.1296

Problem: Some part of using **'smartcase'** was not tested.
Solution: Add more tests. (Dominique Pellé, closes #6538)
Files: src/testdir/test_search.vim

Patch 8.2.1297

Problem: When a test fails it's often not easy to see what the call stack is.
Solution: Add more entries from the call stack in the exception message.
Files: runtime/doc/cmdline.txt, src/scriptfile.c,
src/proto/scriptfile.pro, src/debugger.c, src/ex_docmd.c,
src/ex_eval.c, src/message.c, src/testing.c,
src/testdir/test_expand_func.vim

Patch 8.2.1298

Problem: Compiler warning for unused argument in small version.
Solution: Add UNUSED.
Files: src/scriptfile.c

Patch 8.2.1299

Problem: Compiler warning for using size_t for int and void pointer.
Solution: Add type casts.
Files: src/scriptfile.c

Patch 8.2.1300

Problem: Vim9: optional argument type not parsed properly.
Solution: Skip over the "?". (issue #6507)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/evalvars.c,
src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1301

Problem: Vim9: varargs argument type not parsed properly.
Solution: Skip over the "...". (issue #6507)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1302

Problem: Vim9: varargs arg after optional arg does not work
Solution: Check for the "...". (issue #6507)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1303

Problem: Calling popup_setoptions() resets **'signcolumn'**.
Solution: Only set **'signcolumn'** when creating the popup. (closes #6542)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.1304

Problem: Debug backtrace isn't tested much.
Solution: Add more specific tests. (Ben Jackson, closes #6540)
Files: src/testdir/runtest.vim, src/testdir/test_debugger.vim

Patch 8.2.1305

Problem: Some tests are still old style.

Solution: Convert tests 52 and 70 to new style. (Yegappan Lakshmanan, closes #6544) Fix error in FinishTesting().
Files: src/testdir/runtest.vim, src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_amiga.mak, src/testdir/Make_vms.mms, src/testdir/test52.in, src/testdir/test52.ok, src/testdir/test70.in, src/testdir/test70.ok, src/testdir/test_mzscheme.vim, src/testdir/test_writefile.vim

Patch 8.2.1306

Problem: Checking for first character of dict key is inconsistent.
Solution: Add eval_isdictc(). (closes #6546)
Files: src/eval.c, src/proto/eval.pro, src/vim9compile.c, src/testdir/test_listdict.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_let.vim

Patch 8.2.1307

Problem: popup window width does not include number, fold of sign column width.
Solution: Take number, fold and sign column with into account.
Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_sign_2.dump

Patch 8.2.1308

Problem: Vim9: accidentally using "x" causes Vim to exit.
Solution: Disallow using ":x" or "xit" in Vim9 script. (closes #6399)
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/vim9script.c, src/proto/vim9script.pro, src/ex_docmd.c, src/ex_cmds.c, src/testdir/test_vim9_script.vim

Patch 8.2.1309

Problem: Build failure with tiny version.
Solution: Add #ifdef.
Files: src/ex_cmds.c, src/ex_docmd.c

Patch 8.2.1310

Problem: Configure with Xcode 12 fails to check for tgetent.
Solution: Declare tgetent(). (Ozaki Kiichi, closes #6558)
Files: src/configure.ac, src/auto/configure

Patch 8.2.1311

Problem: Test failures with legacy Vim script.
Solution: Actually check for Vim9 script.
Files: src/vim9script.c

Patch 8.2.1312

Problem: MS-Windows: terminal test may fail if dir.exe exists.
Solution: Use dir.com. (Ken Takata, closes #6557)
Files: src/testdir/test_terminal3.vim

Patch 8.2.1313

Problem: Vim9 script: cannot assign to environment variable.
Solution: Recognize environment variable assignment. (closes #6548) Also options and registers.
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1314

Problem: Vim9: rule for comment after :function is confusing.
Solution: Allow double quoted comment after :function in vim9script.
(closes #6556)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1315

Problem: MS-Windows: test log contains escape sequences.
Solution: Do not use t_md and t_me but ANSI escape sequences. (Ken Takata,
closes #6559)
Files: src/testdir/runtest.vim

Patch 8.2.1316

Problem: Test 42 is still old style.
Solution: Turn it into a new style test. (Yegappan Lakshmanan, closes #6561)
Files: src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_vms.mms,
src/testdir/test42.in, src/testdir/test42.ok,
src/testdir/test_writefile.vim

Patch 8.2.1317

Problem: MS-Windows tests on AppVeyor are slow.
Solution: Use GitHub Actions. (Ken Takata, closes #6569)
Files: Filelist, .github/workflows/ci-windows.yaml, appveyor.yml,
ci/appveyor.bat

Patch 8.2.1318

Problem: No status badge for Github CI.
Solution: Add a badge.
Files: README.md

Patch 8.2.1319

Problem: Status badge for Github CI has wrong link.
Solution: Rename and use the right link
Files: README.md, .github/workflows/ci-windows.yaml

Patch 8.2.1320

Problem: Vim9: cannot declare some single letter variables.
Solution: Do not recognize a colon for a namespace for single letter
variables. (closes #6547)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1321

Problem: GitHub CI also runs on tag push.
Solution: Skip CI on push. (Ken Takata, closes #6571)
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1322

Problem: Vim9: method on double quoted string doesn't work.
Solution: Recognize double quoted string. (closes #6562)
Files: src/ex_docmd.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1323

Problem: Vim9: invalid operators only rejected in :def function.
Solution: Also reject them at script level. (closes #6564)
Files: src/eval.c, src/vim9compile.c, src/proto/vim9compile.pro,
src/testdir/test_vim9_expr.vim

Patch 8.2.1324

Problem: Vim9: line break after "=" does not work.
Solution: Also allow for NUL after "=". (closes #6549)
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1325

Problem: Vim9: using Vim9 script for autoload not tested.
Solution: Add a test. Update help.
Files: runtime/doc/vim9.txt, src/testdir/test_autoload.vim,
src/testdir/sautest/autoload/auto9.vim

Patch 8.2.1326

Problem: Vim9: skipping over white space after list.
Solution: Do not skip white space, a following [] would be misinterpreted.
(closes #6552) Fix a few side effects.
Files: src/list.c, src/dict.c, src/eval.c, src/userfunc.c,
src/testdir/test_functions.vim, src/testdir/test_gn.vim,
src/testdir/test_popupwin.vim, src/testdir/test_tabpage.vim,
src/testdir/test_textprop.vim, src/testdir/test_textobjects.vim

Patch 8.2.1327

Problem: Mac: configure can't find Tcl libraries.
Solution: Adjust configure check. (closes #6575)
Files: src/configure.ac, src/auto/configure

Patch 8.2.1328

Problem: No space allowed before comma in list.
Solution: Legacy Vim script allows it. (closes #6577)
Files: src/dict.c, src/list.c, src/testdir/test_listdict.vim

Patch 8.2.1329

Problem: Vim9: cannot define global function inside :def function.
Solution: Assign to global variable instead of local. (closes #6584)
Files: src/vim9compile.c, src/userfunc.c, src/proto/userfunc.pro,
src/vim9.h, src/vim9execute.c, src/structs.h,
src/misc2.c, src/proto/misc2.pro, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1330

Problem: Github workflow takes longer than needed.
Solution: Do two test runs in parallel instead of sequentially. (Ken Takata,
closes #6579)
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1331

Problem: Vim9: :echo with two lists doesn't work.
Solution: Do not skip white space before []. (closes #6552)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1332

Problem: Vim9: memory leak when using nested global function.
Solution: Delete the function when deleting the instruction. Disable test that still causes a leak.
Files: src/vim9compile.c, src/userfunc.c, src/proto/userfunc.pro, src/testdir/test_vim9_func.vim

Patch 8.2.1333

Problem: Vim9: memory leak when using nested global function.
Solution: Swap from and to when copying the lines.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1334

Problem: Github workflow timeout needs tuning
Solution: Use a 10 minute timeout. Fail when timing out. (Ken Takata, closes #6590)
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1335

Problem: **CTRL-C** in the GUI doesn't interrupt. (Sergey Vlasov)
Solution: Recognize "C" with CTRL modifier as **CTRL-C**. (issue #6565)
Files: src/gui.c, src/proto/gui.pro, src/gui_gtk_x11.c, src/gui_x11.c, src/gui_photon.c

Patch 8.2.1336 (after 8.2.1335)

Problem: Build failure on non-Unix systems.
Solution: Add #ifdef.
Files: src/gui.c

Patch 8.2.1337

Problem: Vim9: cannot use empty key in dict assignment.
Solution: Allow empty key. (closes #6591)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1338

Problem: Vim9: assigning to script-local variable doesn't check type.
Solution: Use the type. (issue #6591)
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1339

Problem: Vim9: assigning to global dict variable doesn't work.
Solution: Guess variable type based in index type. (issue #6591)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1340

Problem: Some tests fail on Cirrus CI and/or with FreeBSD.
Solution: Make **'backupskip'** empty. Do not run tests as root. Check for directory when using viminfo. (Ozaki Kiichi, closes #6596)
Files: .cirrus.yml, src/testdir/test_backup.vim, src/testdir/test_edit.vim, src/testdir/test_viminfo.vim, src/testdir/test_writefile.vim, src/viminfo.c

Patch 8.2.1341

Problem: Build failures.
Solution: Add missing error message.
Files: src/globals.h

Patch 8.2.1342

Problem: Vim9: accidentally using "x" gives a confusing error.
Solution: Disallow using ":t" in Vim9 script. (issue #6399)
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/vim9script.c,
src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1343

Problem: Vim9: cannot find global function when using g: when local
function with the same name exists.
Solution: Find global function when using g:.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1344

Problem: Vim9: No test for trying to redefine global function.
Solution: Add a test.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1345

Problem: Redraw error when using visual block and scroll.
Solution: Add check for w_topline. (closes #6597)
Files: src/drawscreen.c, src/testdir/test_display.vim,
src/testdir/dumps/Test_display_visual_block_scroll.dump

Patch 8.2.1346

Problem: Small build fails.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.1347

Problem: Cannot easily get the script ID.
Solution: Support expand('<SID>').
Files: runtime/doc/map.txt, src/ex_docmd.c,
src/testdir/test_expand_func.vim

Patch 8.2.1348

Problem: Build failure without the eval feature.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.1349

Problem: Vim9: can define a function with the name of an import.
Solution: Disallow using an existing name. (closes #6585)
Files: src/userfunc.c, src/vim9compile.c, src/globals.h,
src/testdir/test_vim9_script.vim

Patch 8.2.1350

Problem: Vim9: no test for error message when redefining function.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1351

Problem: Vim9: no proper error if using namespace for nested function.
Solution: Specifically check for a namespace. (closes #6582)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1352

Problem: Vim9: no error for shadowing a script-local function by a nested function.
Solution: Check for script-local function. (closes #6586)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1353

Problem: Crash when drawing double-wide character in terminal window.
(Masato Nishihata)
Solution: Check getcell() returning NULL. (issue #6141)
Files: src/libvterm/src/screen.c, src/testdir/test_terminal.vim

Patch 8.2.1354

Problem: Test 59 is old style.
Solution: Convert into a new style test. (Yegappan Lakshmanan, closes #6604)
Files: runtime/doc/eval.txt, src/Makefile, src/testdir/Make_all.mak,
src/testdir/Make_vms.mms, src/testdir/test59.in,
src/testdir/test59.ok, src/testdir/test_spell_utf8.vim

Patch 8.2.1355

Problem: Vim9: no error using :let for options and registers.
Solution: Give an error. (closes #6568)
Files: src/evalvars.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1356

Problem: Vim9: cannot get the percent register.
Solution: Check for readable registers instead of writable. (closes #6566)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1357

Problem: Vim9: cannot assign to / register.
Solution: Adjust check for assignment. (issue #6566)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1358

Problem: Vim9: test fails with +dnd is not available.
Solution: Add condition.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1359

Problem: Vim9: cannot assign to / register in Vim9 script.
Solution: Adjust check for assignment in Vim9 script. (closes #6567)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1360

Problem: Stray error for white space after expression.

Solution: Ignore trailing white space. (closes #6608)
Files: src/eval.c, src/testdir/test_filter_map.vim

Patch 8.2.1361

Problem: Error for white space after expression in assignment.
Solution: Skip over white space. (closes #6617)
Files: src/eval.c, src/testdir/test_expr.vim

Patch 8.2.1362

Problem: Last entry of ":set term=xxx" overwritten by error message when '**cmdheight**' is two or more. (Tony Mechelynck)
Solution: Output extra line breaks.
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.1363

Problem: Test trying to run terminal when it is not supported.
Solution: Check if Vim can be run in a terminal.
Files: src/testdir/test_termcodes.vim

Patch 8.2.1364

Problem: Invalid memory access when searching for raw string.
Solution: Check for delimiter match before following quote. (closes #6578)
Files: src/search.c

Patch 8.2.1365

Problem: Vim9: no error for missing white space around operator.
Solution: Check for white space. (closes #6618)
Files: src/eval.c, src/vim9compile.c, src/proto/vim9compile.pro,
src/evalvars.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.1366

Problem: Test 49 is old style.
Solution: Convert several tests to new style. (Yegappan Lakshmanan, closes #6629)
Files: src/testdir/script_util.vim, src/testdir/test49.ok,
src/testdir/test49.vim, src/testdir/test_vimscript.vim

Patch 8.2.1367

Problem: Vim9: no error for missing white space around operator.
Solution: Check for white space around *, / and %.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1368

Problem: Vim9: no error for missing white space around operator.
Solution: Check for white space around <, !=, etc.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1369

Problem: MS-Windows: autocommand test sometimes fails.
Solution: Do not rely on the cat command.
Files: src/testdir/test_autocmd.vim

Patch 8.2.1370

Problem: MS-Windows: warning for using fstat() with stat_T.
Solution: use _fstat64() if available. (Naruhiko Nishino, closes #6625)
Files: src/macros.h

Patch 8.2.1371

Problem: Vim9: no error for missing white space around operator.
Solution: Check for white space around && and ||.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1372

Problem: Vim9: no error for missing white space around operator.
Solution: Check for white space around ? and :.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1373

Problem: Vim9: no error for assigning to non-existing script var.
Solution: Check that in Vim9 script the variable was defined. (closes #6630)
Files: src/vim9compile.c, src/userfunc.c, src/structs.h,
src/testdir/test_vim9_script.vim

Patch 8.2.1374

Problem: Vim9: error for assigning empty list to script variable.
Solution: Use t_unknown for empty list member. (closes #6595)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1375

Problem: Vim9: method name with digit not accepted.
Solution: Use eval_isnamec() instead of eval_isnamec1(). (closes #6613)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1376

Problem: Vim9: expression mapping causes error for using :import.
Solution: Add EX_LOCK_OK to :import and :export. (closes #6606)
Files: src/ex_cmds.h, src/testdir/test_vim9_script.vim

Patch 8.2.1377

Problem: Triggering the ATTENTION prompt causes typeahead to be messed up.
Solution: Increment tb_change_cnt. (closes #6541)
Files: src/getchar.c

Patch 8.2.1378

Problem: Cannot put space between function name and paren.
Solution: Allow this for backwards compatibility.
Files: src/eval.c, src/testdir/test_expr.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1379

Problem: Curly braces expression ending in " }" does not work.
Solution: Skip over white space when checking for "}". (closes #6634)
Files: src/dict.c, src/testdir/test_eval_stuff.vim

Patch 8.2.1380

Problem: Vim9: return type of getreg() is always a string.
Solution: Use list of strings when there are three arguments. (closes #6633)

Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1381

Problem: MS-Windows: crash with Python 3.5 when stdin is redirected.
Solution: Reconnect stdin. (Yasuhiro Matsumoto, Ken Takata, closes #6641)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/if_python3.c

Patch 8.2.1382

Problem: Vim9: using :import in filetype plugin gives an error.
Solution: Allow commands with the EX_LOCK_OK flag. (closes #6636)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1383

Problem: Test 49 is old style.
Solution: Convert test cases to new style. (Yegappan Lakshmanan, closes #6638)
Files: src/testdir/test49.ok, src/testdir/test49.vim, src/testdir/test_vimscript.vim

Patch 8.2.1384

Problem: No ATTENTION prompt for :vimgrep first match file.
Solution: When there is an existing swap file do not keep the dummy buffer. (closes #6649)
Files: src/quickfix.c, src/testdir/runtest.vim, src/testdir/test_quickfix.vim

Patch 8.2.1385

Problem: No testing on ARM.
Solution: Add a test on Travis for ARM. (Ozaki Kiichi, closes #6615)
Files: .travis.yml

Patch 8.2.1386

Problem: Backslash not removed after space in option with space in **'isfname'**.
Solution: Do remove backslash before space, also when it is in **'isfname'**. (Yasuhiro Matsumoto, closes #6651)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.2.1387

Problem: Vim9: cannot assign to single letter variable with type.
Solution: Exclude the colon from the variable name. (closes #6647)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.1388

Problem: Vim9: += only works for numbers.
Solution: Use += as concatenate for a list. (closes #6646)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1389

Problem: File missing from the distribution.
Solution: Add script_util.vim to the list of distributes files.
Files: Filelist

Patch 8.2.1390

Problem: Vim9: type error after storing an option value.
Solution: Drop the type after a STOREOPT instruction. (closes #6632)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1391

Problem: Vim9: no error for shadowing a script function.
Solution: Check for already defined items. (closes #6652)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1392

Problem: Vim9: error line number incorrect after skipping over comment lines.
Solution: Insert empty lines for skipped lines.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1393

Problem: Insufficient testing for script debugging.
Solution: Add more tests. (Ben Jackson)
Files: src/testdir/test_debugger.vim

Patch 8.2.1394

Problem: Vim9: compiling a function interferes with command modifiers.
Solution: Save and restore command modifiers. (closes #6658)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1395

Problem: Vim9: no error if declaring a funcref with a lower case letter.
Solution: Check the name after the type is inferred. Fix confusing name.
Files: src/vim9compile.c, src/dict.c, src/eval.c, src/evalvars.c,
src/proto/evalvars.pro, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1396

Problem: Vim9: no error for unexpectedly returning a value.
Solution: Only set the return type for lambda's. Make using function type in a function reference work.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1397

Problem: Vim9: return type of maparg() not adjusted for fourth argument.
Solution: Check if fourth argument is present. (closes #6645)
Files: src/evalfunc.c, src/testdir/test_maparg.vim

Patch 8.2.1398

Problem: Autoload script sourced twice if sourced directly.
Solution: Do not source an autoload script again. (issue #6644)
Files: src/scriptfile.c, src/testdir/sautest/autoload/sourced.vim

Patch 8.2.1399

Problem: Vim9: may find imported item in wrong script.
Solution: When looking up script-local function use the embedded script ID. (issue #6644)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c,

src/testdir/test_vim9_script.vim

Patch 8.2.1400

Problem: Vim9: test does not delete written files.
Solution: Correct file names.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1401

Problem: Cannot jump to the last used tabpage.
Solution: Add g<Tab> and tabpagr('#'). (Yegappan Lakshmanan, closes #6661, neovim #11626)
Files: runtime/doc/eval.txt, runtime/doc/index.txt,
runtime/doc/tabpage.txt, src/evalwindow.c, src/globals.h,
src/normal.c, src/proto/window.pro, src/testdir/test_tabpage.vim,
src/window.c

Patch 8.2.1402

Problem: s390x tests always fail.
Solution: Temporarily disable s390x tests.
Files: .travis.yml

Patch 8.2.1403

Problem: Vim9: Vim highlighting fails in cmdline window if it uses Vim9 commands.
Solution: Allow using :vim9script, :import and :export while in the cmdline window. (closes #6656)
Files: src/ex_cmds.h, src/testdir/test_vim9_script.vim

Patch 8.2.1404

Problem: Vim9: script test fails in the GUI.
Solution: Use another key to map. Improve cleanup.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1405

Problem: Vim9: vim9compile.c is getting too big.
Solution: Split off type code to vim9type.c.
Files: Filelist, src/vim9compile.c, src/proto/vim9compile.pro,
src/vim9type.c, src/proto/vim9type.pro, src/proto.h,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Makefile

Patch 8.2.1406

Problem: Popupwindow lacks scrollbar if no "maxheight" is used.
Solution: Compute the max height depending on the position. (closes #6664)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_toohigh_1.dump,
src/testdir/dumps/Test_popupwin_toohigh_2.dump

Patch 8.2.1407

Problem: Vim9: type of list and dict only depends on first item.
Solution: Use all items to decide about the type.
Files: src/vim9compile.c, src/vim9type.c, src/proto/vim9type.pro,
src/testdir/test_vim9_expr.vim, runtime/doc/vim9.txt

Patch 8.2.1408

Problem: Vim9: type casting not supported.
Solution: Introduce type casting.
Files: runtime/doc/vim9.txt, src/vim9compile.c,
src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1409

Problem: Npmrc and php.ini filetypes not recognized.
Solution: Add filetype detection. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1410

Problem: Adding compiler plugin requires test change.
Solution: Include compiler plugin and adjust test.
Files: src/testdir/test_compiler.vim, runtime/compiler/xo.vim

Patch 8.2.1411

Problem: when splitting a window localdir is copied but prevdir is not.
Solution: Also copy prevdir. (closes #6667)
Files: src/window.c, src/testdir/test_cd.vim

Patch 8.2.1412

Problem: Vim: not operator does not result in boolean.
Solution: Make type depend on operator. (issue 6678) Fix using "false" and "true" in Vim9 script.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1413 (after 8.2.1401)

Problem: Previous tab page not usable from an Ex command.
Solution: Add the "#" argument for :tabnext et al. (Yegappan Lakshmanan, closes #6677)
Files: runtime/doc/tabpage.txt, src/ex_docmd.c, src/window.c,
src/testdir/test_tabpage.vim

Patch 8.2.1414

Problem: Popupwindow missing last couple of lines when cursor is in the first line.
Solution: Compute the max height also when top aligned. (closes #6664)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_toohigh_3.dump,
src/testdir/dumps/Test_popupwin_nospace.dump

Patch 8.2.1415

Problem: Closing a popup window with **CTRL-C** interrupts 'statusline' if it calls a function.
Solution: Reset got_int while redrawing. (closes #6675)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_ctrl_c.dump

Patch 8.2.1416

Problem: Vim9: boolean evaluation does not work as intended.
Solution: Use tv2bool() in Vim9 script. (closes #6681)
Files: src/eval.c, src/testdir/test_vim9_expr.vim, src/testdir/vim9.vim

Patch 8.2.1417

Problem: Test 49 is old style.
Solution: Convert more parts to new style test. (Yegappan Lakshmanan, closes #6682)
Files: src/testdir/test49.ok, src/testdir/test49.vim, src/testdir/test_vimscript.vim

Patch 8.2.1418

Problem: Vim9: invalid error for missing white space after function.
Solution: Do not skip over white space. (closes #6679)
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1419

Problem: Vim9: not operator applied too early.
Solution: Implement the "numeric_only" argument. (closes #6680)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1420

Problem: Test 49 is old style.
Solution: Convert remaining parts to new style. Remove obsolete items. (Yegappan Lakshmanan, closes #6683)
Files: Filelist, runtime/doc/testing.txt, src/Make_mvc.mak, src/Makefile, src/testdir/Make_all.mak, src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_vms.mms, src/testdir/Makefile, src/testdir/README.txt, src/testdir/test49.in, src/testdir/test49.ok, src/testdir/test49.vim, src/testdir/test_quickfix.vim, src/testdir/test_vimscript.vim

Patch 8.2.1421

Problem: Vim9: handling "+" and "-" before number differs from Vim script.
Solution: Use the same sequence of commands.
Files: src/vim9compile.c

Patch 8.2.1422

Problem: The Mac GUI implementation is outdated and probably doesn't even work.
Solution: Remove the Mac GUI code. The MacVim project provides the supported Vim GUI version.
Files: Filelist, src/gui_mac.c, src/proto/gui_mac.pro, src/proto.h, src/Makefile, src/configure.ac, src/auto/configure, src/evalfunc.c, src/fileio.c, src/gui.c, src/if_mzsch.c, src/main.c, src/misc2.c, src/mouse.c, src/os_mac_conv.c, src/os_unix.c, src/feature.h, src/globals.h, src/gui.h, src/option.h, src/optiondefs.h, src/os_mac.h, src/structs.h, src/vim.h, src/INSTALLmac.txt

Patch 8.2.1423

Problem: Vim9: find global function when looking for script-local.
Solution: Don't strip prefix if name starts with "s:". (closes #6688)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1424 (after 8.2.1422)

Problem: Mac build fails.

Solution: Adjust configure to not fall back to Athena. Adjust some other files.
Files: src/configure.ac, src/auto/configure, src/os_macosx.m, src/version.c

Patch 8.2.1425

Problem: Vim9: cannot use call() without :call.
Solution: Do not skip over "call(". (closes #6689)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1426

Problem: Vim9: cannot call autoload function in :def function.
Solution: Load the autoload script. (closes #6690)
Files: src/vim9execute.c, src/vim9compile.c, src/scriptfile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1427

Problem: Vim9: cannot use a range with marks in :def function.
Solution: Parse range after colon. (closes #6686)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1428

Problem: Vim9: :def function does not abort on nested function error.
Solution: Check whether an error message was given. (closes #6691)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1429

Problem: Vim9: no error for missing white after : in dict.
Solution: Check for white space. (closes #6671) Also check that there is no white before the :.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.1430

Problem: Vim9: error for missing comma instead of extra white space.
Solution: Check if comma can be found after white space. (closes #6668)
Also check for extra white space in literal dict. (closes #6670)
Files: src/list.c, src/dict.c, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1431

Problem: Vim9: no error for white space before comma in dict.
Solution: Check for extra white space. (closes #6674)
Files: src/vim9compile.c, src/dict.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1432

Problem: Various inconsistencies in test files.
Solution: Add modelines where they were missing. Use Check commands instead of silently skipping over tests. Adjust indents and comments. (Ken Takata, closes #6695)
Files: src/testdir/test_arglist.vim, src/testdir/test_assert.vim, src/testdir/test_autochdir.vim, src/testdir/test_autocmd.vim, src/testdir/test_autoload.vim, src/testdir/test_balloon.vim, src/testdir/test_balloon_gui.vim, src/testdir/test_behave.vim,

src/testdir/test_blockedit.vim, src/testdir/test_breakindent.vim,
src/testdir/test_buflines.vim, src/testdir/test_bufwintabinfo.vim,
src/testdir/test_cd.vim, src/testdir/test_changedtick.vim,
src/testdir/test_changelist.vim, src/testdir/test_channel.vim,
src/testdir/test_checkpath.vim, src/testdir/test_cindent.vim,
src/testdir/test_cjk_linebreak.vim,
src/testdir/test_clientserver.vim,
src/testdir/test_close_count.vim, src/testdir/test_cmdline.vim,
src/testdir/test_command_count.vim,
src/testdir/test_comparators.vim, src/testdir/test_compiler.vim,
src/testdir/test_crypt.vim, src/testdir/test_cursorline.vim,
src/testdir/test_curswant.vim, src/testdir/test_debugger.vim,
src/testdir/test_delete.vim, src/testdir/test_diffmode.vim,
src/testdir/test_digraph.vim, src/testdir/test_display.vim,
src/testdir/test_edit.vim, src/testdir/test_envIRON.vim,
src/testdir/test_erasebackward.vim,
src/testdir/test_escaped_glob.vim, src/testdir/test_ex_equal.vim,
src/testdir/test_ex_undo.vim, src/testdir/test_ex_z.vim,
src/testdir/test_exec_while_if.vim, src/testdir/test_exists.vim,
src/testdir/test_exists_autocmd.vim, src/testdir/test_exit.vim,
src/testdir/test_expand_dllpath.vim,
src/testdir/test_expr_utf8.vim, src/testdir/test_feedkeys.vim,
src/testdir/test_file_size.vim, src/testdir/test_fileformat.vim,
src/testdir/test_filter_cmd.vim,
src/testdir/test_find_complete.vim, src/testdir/test_findfile.vim,
src/testdir/test_fixeol.vim, src/testdir/test_flatten.vim,
src/testdir/test_fnameescape.vim, src/testdir/test_fold.vim,
src/testdir/test_functions.vim, src/testdir/test_ga.vim,
src/testdir/test_getcwd.vim, src/testdir/test_getvar.vim,
src/testdir/test_glob2regpat.vim, src/testdir/test_global.vim,
src/testdir/test_gui.vim, src/testdir/test_gui_init.vim,
src/testdir/test_highlight.vim, src/testdir/test_hlsearch.vim,
src/testdir/test_iminsert.vim,
src/testdir/test_increment_dbcs.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_interrupt.vim,
src/testdir/test_job_fails.vim, src/testdir/test_join.vim,
src/testdir/test_json.vim, src/testdir/test_jumplist.vim,
src/testdir/test_jumps.vim, src/testdir/test_lambda.vim,
src/testdir/test_langmap.vim, src/testdir/test_largefile.vim,
src/testdir/test_lineending.vim, src/testdir/test_listchars.vim,
src/testdir/test_listener.vim, src/testdir/test_listlbr.vim,
src/testdir/test_listlbr_utf8.vim,
src/testdir/test_makeencoding.vim, src/testdir/test_man.vim,
src/testdir/test_mapping.vim, src/testdir/test_marks.vim,
src/testdir/test_matchadd_conceal.vim,
src/testdir/test_matchadd_conceal_utf8.vim,
src/testdir/test_memory_usage.vim, src/testdir/test_menu.vim,
src/testdir/test_messages.vim, src/testdir/test_mksession.vim,
src/testdir/test_modeline.vim,
src/testdir/test_nested_function.vim, src/testdir/test_number.vim,
src/testdir/test_options.vim, src/testdir/test_packadd.vim,
src/testdir/test_partial.vim, src/testdir/test_paste.vim,
src/testdir/test_plus_arg_edit.vim, src/testdir/test_preview.vim,
src/testdir/test_profile.vim, src/testdir/test_prompt_buffer.vim,

src/testdir/test_quickfix.vim, src/testdir/test_quotestar.vim,
src/testdir/test_random.vim, src/testdir/test_recover.vim,
src/testdir/test_regex_char_classes.vim,
src/testdir/test_regexp_latin.vim, src/testdir/test_registers.vim,
src/testdir/test_rename.vim, src/testdir/test_retab.vim,
src/testdir/test_scriptnames.vim, src/testdir/test_scroll_opt.vim,
src/testdir/test_scrollbind.vim, src/testdir/test_search_stat.vim,
src/testdir/test_searchpos.vim, src/testdir/test_set.vim,
src/testdir/test_sha256.vim, src/testdir/test_shift.vim,
src/testdir/test_shortpathname.vim, src/testdir/test_signs.vim,
src/testdir/test_sort.vim, src/testdir/test_sound.vim,
src/testdir/test_source_utf8.vim, src/testdir/test_spellfile.vim,
src/testdir/test_startup.vim, src/testdir/test_startup_utf8.vim,
src/testdir/test_stat.vim, src/testdir/test_suspend.vim,
src/testdir/test_swap.vim, src/testdir/test_syntax.vim,
src/testdir/test_tab.vim, src/testdir/test_tabline.vim,
src/testdir/test_tagcase.vim, src/testdir/test_tagjump.vim,
src/testdir/test_taglist.vim, src/testdir/test_termcodes.vim,
src/testdir/test_termencoding.vim, src/testdir/test_terminal.vim,
src/testdir/test_terminal2.vim, src/testdir/test_terminal3.vim,
src/testdir/test_terminal_fail.vim,
src/testdir/test_true_false.vim,
src/testdir/test_utf8_comparisons.vim,
src/testdir/test_vartabs.vim, src/testdir/test_version.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_winbar.vim,
src/testdir/test_winbuf_close.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_window_id.vim,
src/testdir/test_windows_home.vim, src/testdir/test_wnext.vim,
src/testdir/test_wordcount.vim, src/testdir/test_writefile.vim,
src/testdir/test_xxd.vim

Patch 8.2.1433

Problem: Vim9: cannot mingle comments in multi-line lambda.
Solution: Skip over NULL lines. (closes #6694)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1434

Problem: Vim9: crash when lambda uses outer function argument.
Solution: Set the flag that the outer context is used.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1435

Problem: Vim9: always converting to string for "." leads to mistakes.
Solution: Only automatically convert simple types.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/vim9.h,
src/vim9execute.c, src/proto/vim9execute.pro, src/eval.c,
src/evalfunc.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1436

Problem: Function implementing :substitute has unexpected name.
Solution: Rename from do_sub() to ex_substitute().
Files: src/ex_cmds.c, src/proto/ex_cmds.pro, src/ex_docmd.c,
src/ex_cmds.h

Patch 8.2.1437

Problem: Vim9: **'statusline'** is evaluated using Vim9 script syntax.
Solution: Always use legacy script syntax.
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.1438

Problem: Missing tests for interrupting script execution from debugger.
Solution: Add tests. (Yegappan Lakshmanan, closes #6697)
Files: src/testdir/test_debugger.vim

Patch 8.2.1439

Problem: Tiny and small builds have no test coverage.
Solution: Restore tests that do not depend on the +eval feature.
(Ken Takata, closes #6696)
Files: .travis.yml, Filelist, Makefile, runtime/doc/testing.txt,
src/Make_mvc.mak, src/Makefile, src/testdir/Make_all.mak,
src/testdir/Make_amiga.mak, src/testdir/Make_dos.mak,
src/testdir/Make_ming.mak, src/testdir/Make_vms.mms,
src/testdir/Makefile, src/testdir/runtest.vim,
src/testdir/test1.in, src/testdir/test1.ok, src/testdir/test20.in,
src/testdir/test20.ok, src/testdir/test21.in,
src/testdir/test21.ok, src/testdir/test22.in,
src/testdir/test22.ok, src/testdir/test23.in,
src/testdir/test23.ok, src/testdir/test24.in,
src/testdir/test24.ok, src/testdir/test25.in,
src/testdir/test25.ok, src/testdir/test26.in,
src/testdir/test26.ok, src/testdir/test27.in,
src/testdir/test27.ok, src/testdir/test_options.vim

Patch 8.2.1440

Problem: Debugger code insufficiently tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #6700)
Files: src/testdir/test_debugger.vim, src/testdir/test_vimscript.vim

Patch 8.2.1441

Problem: Running tests in tiny version gives error for summarize.vim.
Solution: Set **'coptions'** to allow for line continuation. Restore
redirecting test output to /dev/null.
Files: src/testdir/summarize.vim, src/testdir/Makefile

Patch 8.2.1442

Problem: Outdated references to the Mac Carbon GUI.
Solution: Remove or update references. (Yee Cheng Chin, closes #6703)
Files: READMEdir/README_extra.txt, src/Makefile, src/configure.ac,
src/auto/configure, src/gui_haiku.cc, src/os_macosx.m,
src/testdir/test_iminsert.vim, src/vim.h

Patch 8.2.1443

Problem: Vim9: crash when interrupting a nested :def function.
Solution: Push a dummy return value onto the stack. (closes #6701)
Files: src/vim9execute.c

Patch 8.2.1444

Problem: Error messages are spread out and names can be confusing.
Solution: Start moving error messages to a separate file and use clear names.
Files: Filelist, src/vim.h, src/globals.h, src/errors.h, src/Makefile, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/dict.c, src/evalvars.c, src/ex_docmd.c, src/list.c, src/userfunc.c, src/vim9compile.c, src/vim9execute.c, src/vim9script.c, src/vim9type.c

Patch 8.2.1445

Problem: Vim9: function expanded name is cleared when sourcing a script again.
Solution: Only clear the expanded name when deleting the function. (closes #6707)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1446

Problem: Vim9: line number in error message is not correct.
Solution: Set SOURCING_LNUM before calling emsg(). (closes #6708)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1447

Problem: Vim9: return type of keys() is list<any>.
Solution: Should be list<string>. (closes #6711)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1448

Problem: Test 77a for VMS depends on small.vim which does not exist.
Solution: Use the 'silent while 0' trick. (issue #6696)
Files: src/testdir/test77a.in

Patch 8.2.1449

Problem: Some test makefiles delete files that are not generated.
Solution: Remove the deletion commands.
Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Make_amiga.mak, src/testdir/Make_vms.mms

Patch 8.2.1450

Problem: Vim9: no check that script-local items don't become global.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1451

Problem: Vim9: list type at script level only uses first item.
Solution: Use all members, like in a compiled function. (closes #6712)
Also for dictionary.
Files: src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1452

Problem: Vim9: dead code in to_name_end().
Solution: Remove check for lambda and dict, it won't be used.
Files: src/vim9compile.c

Patch 8.2.1453

Problem: Vim9: failure to compile lambda not tested.
Solution: Add a test case.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1454

Problem: Vim9: failure invoking lambda with wrong arguments.
Solution: Handle invalid arguments. Add a test.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.1455

Problem: Vim9: crash when using typecast before constant.
Solution: Generate constant before checking type. Add tests.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1456

Problem: MS-Windows: test files are not deleted.
Solution: use "del" instead of \$(DEL).
Files: src/testdir/Make_dos.mak

Patch 8.2.1457

Problem: Vim9: the output of :disassemble cannot be interrupted.
Solution: Check got_int. (closes #6715)
Files: src/vim9execute.c

Patch 8.2.1458

Problem: .gawk files not recognized.
Solution: Recognize .gawk files. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1459

Problem: Vim9: declaring a script variable at the script level does not infer the type.
Solution: Get the type from the value. (closes #6716)
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1460

Problem: Error messages are spread out.
Solution: Move more messages into errors.h.
Files: src/errors.h, src/globals.h, src/vim9compile.c, src/vim9execute.c,
src/vim9script.c, src/vim9type.c, src/scriptfile.c, src/ex_cmds.c,
src/ex_docmd.c, src/match.c, src/eval.c, src/evalvars.c,
src/userfunc.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1461

Problem: Vim9: string indexes are counted in bytes.
Solution: Use character indexes. (closes #6574)
Files: runtime/doc/eval.txt, src/eval.c, src/proto/eval.pro,
src/vim9execute.c, src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1462

Problem: Vim9: string slice not supported yet.

Solution: Add support for string slicing.
Files: src/errors.h, src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/eval.c, src/proto/eval.pro, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1463

Problem: Vim9: list slice not supported yet.
Solution: Add support for list slicing.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/eval.c, src/list.c, src/proto/list.pro, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1464

Problem: Vim9: build warning for unused variable.
Solution: Delete the variable declaration.
Files: src/vim9execute.c

Patch 8.2.1465

Problem: Vim9: subscript not handled properly.
Solution: Adjust error message. Remove dead code. Disallow string to number conversion in scripts.
Files: src/errors.h, src/vim9compile.c, src/vim9execute.c, src/eval.c, src/typval.c, src/list.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1466

Problem: Vim9: cannot index or slice a variable with type "any".
Solution: Add runtime index and slice.
Files: src/eval.c, src/proto/eval.pro, src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/errors.h, src/list.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1467

Problem: Vim9: :echomsg doesn't like a dict argument.
Solution: Convert arguments like in legacy script. (closes #6717)
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1468

Problem: Vim9: invalid error for missing white space.
Solution: Don't skip over white space after index. (closes #6718)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1469

Problem: Vim9: cannot assign string to string option.
Solution: Change checks for option value. (closes #6720)
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1470

Problem: Errors in spell file not tested.
Solution: Add test for spell file errors. (Yegappan Lakshmanan, closes #6721)

Files: src/testdir/test_spellfile.vim

Patch 8.2.1471

Problem: :const only locks the variable, not the value.

Solution: Lock the value as ":lockvar 1 var" would do. (closes #6719)

Files: src/evalvars.c, src/testdir/test_const.vim

Patch 8.2.1472

Problem: ":argdel" does not work like "::.argdel" as documented. (Alexey Demin)

Solution: Make ":argdel" work like "::.argdel". (closes #6727)

Also fix giving the error "% more files to edit".

Files: src/arglist.c, src/ex_docmd.c, src/testdir/test_arglist.vim

Patch 8.2.1473

Problem: Items in a list given to :const can still be modified.

Solution: Work like ":lockvar! name" but don't lock referenced items.

Make locking a blob work.

Files: runtime/doc/eval.txt, src/evalvars.c, src/eval.c,
src/testdir/test_const.vim

Patch 8.2.1474

Problem: /usr/lib/udev/rules.d not recognized as udevrules.

Solution: Adjust match pattern. (Haochen Tong, closes 36722)

Files: runtime/autoload/dist/ft.vim, src/testdir/test_filetype.vim

Patch 8.2.1475

Problem: Vim9: can't use v:true for option flags.

Solution: Add tv_get_bool_chk(). (closes #6725)

Files: src/typval.c, src/proto/typval.pro, src/channel.c

Patch 8.2.1476 (after 8.2.1474)

Problem: Filetype test fails on MS-Windows.

Solution: Remove "^" from pattern.

Files: runtime/autoload/dist/ft.vim

Patch 8.2.1477

Problem: Vim9: error when using bufnr('%').

Solution: Don't give an error for using a string argument. (closes #6723)

Files: src/evalbuffer.c, src/testdir/test_vim9_func.vim

Patch 8.2.1478

Problem: Vim9: cannot use "true" for some popup options.

Solution: Add dict_get_bool(). (closes #6725)

Files: src/dict.c, src/proto/dict.pro, src/popupwin.c

Patch 8.2.1479

Problem: Vim9: error for list index uses wrong line number.

Solution: Set source line number. (closes #6724) Add a way to assert the
line number of the error with assert_fails().

Files: runtime/doc/testing.txt, src/vim9execute.c, src/testing.c,
src/evalfunc.c, src/message.c, src/globals.h, src/testdir/vim9.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1480

Problem: Vim9: skip expression in search() gives error.
Solution: use tv_get_bool() eval_expr_to_bool(). (closes #6729)
Files: src/eval.c, src/typval.c, src/proto/typval.pro,
src/testdir/test_vim9_func.vim

Patch 8.2.1481

Problem: Vim9: line number reported with error may be wrong.
Solution: Check line number in tests.
Files: src/testdir/test_vim9_expr.vim, src/testdir/vim9.vim,
src/vim9execute.c

Patch 8.2.1482

Problem: Vim9: crash when using a nested lambda.
Solution: Do not clear the growarray when not evaluating. Correct pointer
when getting the next line. (closes #6731)
Files: src/eval.c, src/scriptfile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1483

Problem: Vim9: error for using special as number when returning "false"
from a popup filter.
Solution: Use tv_get_bool(). (closes #6733)
Files: src/popupwin.c

Patch 8.2.1484

Problem: Flaky failure in assert_fails().
Solution: Only used fourth argument if there is a third argument.
Files: src/testing.c

Patch 8.2.1485

Problem: Vim9: readdirex() expression doesn't accept bool.
Solution: Accept both -1 and bool. (closes #6737)
Files: src/filepath.c, src/testdir/test_vim9_func.vim

Patch 8.2.1486

Problem: Vim9: readdir() expression doesn't accept bool.
Solution: Merge with code for readdirex(). (closes #6737)
Files: src/filepath.c, src/testdir/test_vim9_func.vim

Patch 8.2.1487

Problem: Travis: installing snd-dummy is not always useful.
Solution: Only install snd-dummy on amd64. (Ozaki Kiichi, closes #6738)
Files: .travis.yml, ci/load-snd-dummy.sh

Patch 8.2.1488

Problem: Text does not scroll when inserting above first line.
Solution: Adjust off-by-one error. (Ken Takata, closes #6739)
Files: src/drawscreen.c, src/testdir/test_display.vim,
src/testdir/dumps/Test_display_scroll_at_topline.dump

Patch 8.2.1489

Problem: Vim9: error when setting an option with setbufvar().
Solution: Do not get a number from a string value. (closes #6740)
Files: src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.1490

Problem: Vim9: using /= with float and number doesn't work.
Solution: Better support assignment with operator. (closes #6742)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1491

Problem: Vim9: crash when compiling heredoc lines start with comment.
Solution: Skip over NULL pointers. Do not remove comment and empty lines when fetching function lines. (closes #6743)
Files: src/vim9compile.c, src/scriptfile.c, src/proto/scriptfile.pro, src/structs.h, src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_cmds.h, src/autocmd.c, src/proto/autocmd.pro, src/ex_getln.c, src/proto/ex_getln.pro, src/userfunc.c, src/proto/userfunc.pro, src/evalfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1492

Problem: Build failures.
Solution: Move typedef out of #ifdef. Adjust argument types. Discover America.
Files: src/structs.h, src/ex_docmd.c

Patch 8.2.1493

Problem: Not enough test coverage for the spell file handling.
Solution: Add spell file tests. (Yegappan Lakshmanan, closes #6728)
Files: src/spellfile.c, src/testdir/test_spellfile.vim

Patch 8.2.1494

Problem: Missing change to calling eval_getline().
Solution: Change last argument.
Files: src/eval.c

Patch 8.2.1495

Problem: "make clean" may delete too many files.
Solution: Do not delete \$APPDIR. (closes #6751)
Files: src/Makefile

Patch 8.2.1496

Problem: Vim9: cannot use " #" in a mapping.
Solution: Do not remove a comment with the EX_NOTRLCOM flag. (closes #6746)
Files: src/ex_docmd.c, src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1497

Problem: CursorHold test is flaky. (Jakub Kądziołka)
Solution: Use WaitForAssert() (closes #6754)
Files: src/testdir/test_autocmd.vim

Patch 8.2.1498

Problem: On slow systems tests can be flaky.
Solution: Use TermWait() instead of term-wait(). (Yegappan Lakshmanan, closes #6756)
Files: src/testdir/test_digraph.vim, src/testdir/test_display.vim, src/testdir/test_popupwin.vim, src/testdir/test_termcodes.vim,

src/testdir/test_terminal.vim, src/testdir/test_terminal3.vim,
src/testdir/test_writefile.vim

Patch 8.2.1499

Problem: Vim9: error when using "\$" with col().
Solution: Reorder getting the column value. (closes #6744)
Files: src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.1500

Problem: Vim9: error when using address without a command.
Solution: Execute the range itself. (closes #6747)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1501

Problem: Vim9: concatenating to constant reverses order.
Solution: Generate constant before option, register and environment
variable. (closes #6757)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1502

Problem: Vim9: can use += with a :let command at script level.
Solution: Give an error.
Files: src/evalvars.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.1503

Problem: Vim9: error for an autocmd defined in a :def function in legacy
Vim script.
Solution: Don't check the variable type. (closes #6758)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.1504

Problem: Vim9: white space checks are only done for a :def function.
Solution: Also do checks at the script level. Adjust the name of a few
error messages.
Files: src/userfunc.c, src/errors.h, src/dict.c, src/list.c,
src/vim9compile.c, src/vim9script.c, src/vim9type.c,
src/evalvars.c, src/testdir/test_vim9_expr.vim,
src/testdir/vim9.vim

Patch 8.2.1505

Problem: Not all file read and writecode is tested.
Solution: Add a few tests. (Dominique Pellé, closes #6764)
Files: src/testdir/test_eval_stuff.vim, src/testdir/test_fnamemodify.vim,
src/testdir/test_functions.vim

Patch 8.2.1506

Problem: Vim9: no error when using a number other than 0 or 1 as bool.
Solution: Check the number is 0 or 1.
Files: src/errors.h, src/typval.c, src/testdir/test_vim9_func.vim

Patch 8.2.1507

Problem: Using malloc() directly.
Solution: Use ALLOC_ONE(). Remove superfluous typecast. (Hussam al-Homsi,

closes #6768)
Files: src/eval.c, src/memline.c, src/vimrun.c

Patch 8.2.1508

Problem: Not all debugger commands covered by tests.
Solution: Add tests for going up/down in the stack. (Ben Jackson,
 closes #6765)
Files: src/testdir/test_debugger.vim

Patch 8.2.1509

Problem: Vertical separator is cleared when dragging a popup window using a
 multi-byte character for the border.
Solution: Only clear the character before the window if it is using a
 multi-byte character. (closes #6766)
Files: src/screen.c

Patch 8.2.1510

Problem: Using "var" in a :def function may refer to a legacy Vim script
 variable.
Solution: Require using "s:" to refer to a legacy Vim script variable.
 (closes #6771)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1511

Problem: Putting a string in Visual block mode ignores multi-byte
 characters.
Solution: Adjust the column for Visual block mode. (closes #6767)
Files: src/register.c, src/testdir/test_visual.vim

Patch 8.2.1512

Problem: Failure after ternary expression fails.
Solution: Restore eval_flags. (Yasuhiro Matsumoto, closes #6776)
Files: src/eval.c, src/testdir/test_vimscript.vim,
 src/testdir/test_vim9_expr.vim

Patch 8.2.1513

Problem: Cannot interrupt shell used for filename expansion. (Dominique
 Pellé)
Solution: Do set tmode in mch_delay(). (closes #6770)
Files: src/vim.h, src/os_unix.c, src/proto/os_unix.pro, src/term.c,
 src/channel.c, src/if_cscope.c, src/os_amiga.c, src/ui.c,
 src/proto/os_amiga.pro, src/os_win32.c, src/proto/os_win32.pro

Patch 8.2.1514

Problem: Multibyte vertical separator is cleared when dragging a popup
 window using a multi-byte character for the border.
Solution: Only clear the character before the window if it is double width.
 (closes #6766)
Files: src/screen.c

Patch 8.2.1515

Problem: Vim9: can create s:var in legacy script but cannot unlet.
Solution: Allow :unlet for legacy script var.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1516

Problem: Vim9: error for :exe has wrong line number.
Solution: Set line number before calling do_cmdline_cmd(). (closes #6774)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1517

Problem: Cannot easily get the character under the cursor.
Solution: Add the {chars} argument to strpart().
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_functions.vim

Patch 8.2.1518

Problem: Vim9: cannot assign to local option.
Solution: Skip over "&l:" and "&g:". (closes #6749)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/testdir/vim9.vim,
src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1519

Problem: Vim9: Ex command default range is not set.
Solution: When range is not given use default. (closes #6779)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.1520

Problem: Vim9: **CTRL-]** used in :def function does not work.
Solution: Omit count or prepend colon. (closes #6769)
Files: src/normal.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1521

Problem: Reading past end of buffer when reading spellfile. (Yegappan Lakshmanan)
Solution: Store the byte length and check for it.
Files: src/spellfile.c, src/spell.h

Patch 8.2.1522

Problem: Not enough test coverage for the spell file handling.
Solution: Add spell file tests. (Yegappan Lakshmanan, closes #6763)
Files: src/testdir/test_spellfile.vim

Patch 8.2.1523

Problem: Still not enough test coverage for the spell file handling.
Solution: Add spell file tests. (Yegappan Lakshmanan, closes #6790)
Files: src/testdir/test_spellfile.vim

Patch 8.2.1524

Problem: No longer get an error for string concatenation with float. (Tsuyoshi Cho)
Solution: Only convert float for Vim9 script. (closes #6787)
Files: src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.2.1525

Problem: Messages from tests were not always displayed.
Solution: Always show messages, the timing is always useful. (Ken Takata, closes #6792)

Files: src/testdir/Make_dos.mak, src/testdir/Make_ming.mak,
src/testdir/Makefile

Patch 8.2.1526

Problem: Line in testdir Makefile got commented out. (Christian Brabandt)
Solution: Revert.
Files: src/testdir/Makefile

Patch 8.2.1527

Problem: Vim9: cannot use a function name as a function reference at script
level.
Solution: Check if a name is a function name. (closes #6789)
Files: src/evalvars.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1528

Problem: Vim9: :endif not found after "if false".
Solution: When skipping still check for a following command. (closes #6797)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1529

Problem: Vim9: :elseif may be compiled when not needed.
Solution: Do evaluate the :elseif expression.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1530

Problem: Vim9: test fails on MS-Windows.
Solution: Skip Ex command inside "if false".
Files: src/vim9compile.c

Patch 8.2.1531

Problem: Vim9: test still fails on MS-Windows.
Solution: When skipping expect function to be NULL.
Files: src/vim9compile.c

Patch 8.2.1532

Problem: Compiler warning for conversion of size_t to long.
Solution: Add type cast.
Files: src/eval.c

Patch 8.2.1533

Problem: Vim9: error when passing getreginfo() result to setreg().
Solution: Use dict_get_bool() for "isunnamed". (closes #6784)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1534

Problem: Vim9: type error for argument type is not at call position.
Solution: Set the context and stack after checking the arguments.
(issue #6785)
Files: src/userfunc.c, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1535

Problem: It is not possible to specify cell widths of characters.
Solution: Add setcellwidths().

Files: runtime/doc/eval.txt, runtime/doc/options.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/mbyte.c,
src/proto/mbyte.pro, src/errors.h, src/testdir/test_utf8.vim

Patch 8.2.1536

Problem: Cannot get the class of a character; emoji widths are wrong in some environments.

Solution: Add charclass(). Update some emoji widths. Add script to check emoji widths.

Files: Filelist, runtime/doc/eval.txt, runtime/doc/usr_41.txt,
src/evalfunc.c, src/mbyte.c, src/proto/mbyte.pro,
src/testdir/emoji_list.vim, src/testdir/test_functions.vim

Patch 8.2.1537

Problem: Memory access error when using setcellwidths().

Solution: Use array and pointers correctly.

Files: src/mbyte.c, src/errors.h, src/testdir/test_utf8.vim

Patch 8.2.1538

Problem: Python: iteration over vim objects fails to keep reference.

Solution: Keep a reference for the object. (Paul Ollis, closes #6803, closes #6806)

Files: src/if_py_both.h, src/testdir/test_python3.vim

Patch 8.2.1539

Problem: Using invalid script ID causes a crash.

Solution: Check the script ID to be valid. (closes #6804)

Files: src/globals.h, src/evalvars.c, src/profiler.c, src/scriptfile.c,
src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1540

Problem: The user cannot try out emoji character widths.

Solution: Move the emoji script to the runtime/tools directory.

Files: Filelist, src/testdir/emoji_list.vim, runtime/tools/emoji_list.vim

Patch 8.2.1541

Problem: Vim9: cannot find function reference for s:Func.

Solution: Recognize <SNR> prefix. (closes #6805)

Files: src/userfunc.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1542

Problem: Vim9: test with invalid SID does not work in the GUI.

Solution: Skip the test in the GUI.

Files: src/testdir/test_vim9_script.vim

Patch 8.2.1543

Problem: Vim9: test with invalid SID is skipped in the GUI.

Solution: Read the **CTRL-C** that feedkeys() put in typeahead.

Files: src/testdir/test_vim9_script.vim

Patch 8.2.1544

Problem: Cannot translate messages in a Vim script.

Solution: Add gettext(). Try it out for a few messages in the options

Files: window.
Filelist, src/po/Makefile, src/po/README.txt, runtime/optwin.vim,
src/evalfunc.c, src/po/tojavascript.vim, src/po/fixfilenames.vim,
runtime/doc/eval.txt, runtime/doc/usr_41.txt

Patch 8.2.1545

Problem: ch_logfile() is unclear about closing when forking.
Solution: Adjust the log messages.
Files: src/channel.c, src/os_unix.c

Patch 8.2.1546

Problem: Build rule for Vim.app is unused.
Solution: Delete the related build rules.
Files: src/Makefile

Patch 8.2.1547

Problem: Various comment problems.
Solution: Update comments.
Files: src/arglist.c, src/map.c, src/mbyte.c, src/tag.c, src/undo.c,
src/testdir/README.txt, src/testdir/test_put.vim,
src/libvterm/README

Patch 8.2.1548

Problem: Cannot move position of "%%" in message translations. (Emir Sarı)
Solution: Improve the check script.
Files: src/po/check.vim

Patch 8.2.1549

Problem: The "r" command fails for keys with modifiers if 'esckeys' is off
and modifyOtherKeys is used. (Lauri Tirkkonen)
Solution: Temporarily disable bracketed paste and modifyOtherKeys if
'esckeys' is off. (closes #6809)
Files: src/normal.c

Patch 8.2.1550

Problem: Vim9: bufname('%') gives an error.
Solution: Only give an error for wrong argument type. (closes #6807)
Files: src/evalbuffer.c, src/testdir/test_vim9_func.vim

Patch 8.2.1551

Problem: Vim9: error for argument type does not mention the number.
Solution: Pass the argument number to where the error is given.
Files: src/vim9type.c, src/proto/vim9type.pro, src/vim9compile.c,
src/vim9execute.c, src/vim9script.c, src/eval.c,
src/testdir/test_vim9_func.vim

Patch 8.2.1552

Problem: Warnings from asan with clang-11. (James McCoy)
Solution: Avoid using a NULL pointer. (issue #6811)
Files: src/fold.c

Patch 8.2.1553 (after 8.2.1552)

Problem: Crash in edit test.
Solution: Avoid using invalid pointer.

Files: src/fold.c

Patch 8.2.1554

Problem: Crash in normal test.

Solution: Skip adjusting marks if there are no folds.

Files: src/fold.c

Patch 8.2.1555

Problem: Not all tests are executed on Github Actions.

Solution: Copy "src" to "src2" earlier. Recognize "src2" in a couple more places. Add two tests to the list of flaky tests. (Ken Takata, closes #6798)

Files: .github/workflows/ci-windows.yaml, src/testdir/runtest.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1556

Problem: Cursorline highlighting always overrules sign highlighting.

Solution: Combine the highlighting, use the priority to decide how. (closes #6812)

Files: runtime/doc/sign.txt, src/structs.h, src/drawline.c, runtime/pack/dist/opt/termdebug/plugin/termdebug.vim, src/testdir/test_signs.vim, src/testdir/dumps/Test_sign_cursor_5.dump, src/testdir/dumps/Test_sign_cursor_6.dump

Patch 8.2.1557

Problem: Crash in :vimgrep when started as "vim -n". (Raul Segura)

Solution: Check mfp pointer. (Yegappan Lakshmanan, closes #6827)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1558

Problem: Signs test fails.

Solution: Add missing change to sign.c.

Files: src/sign.c

Patch 8.2.1559

Problem: s390x tests work again.

Solution: re-enable s390x tests. (James McCoy, closes #6829)

Files: .travis.yml

Patch 8.2.1560

Problem: Using NULL pointers in some code. (James McCoy)

Solution: Avoid adding to a NULL pointer. Use byte as unsigned.

Files: src/fold.c, src/eval.c, src/spellsuggest.c, src/spellfile.c, src/vim9compile.c

Patch 8.2.1561

Problem: Using NULL pointers in fold code.

Solution: Avoid using a NULL pointer. (Dominique Pellé, closes #6831, closes #6831)

Files: src/fold.c

Patch 8.2.1562

Problem: Vim9: error when using "%" where a buffer is expected.

Solution: Add tv_get_buf_from_arg(). (closes #6814)
Files: src/typval.c, src/proto/typval.pro, src/evalbuffer.c,
src/testdir/test_vim9_func.vim

Patch 8.2.1563

Problem: Vim9: error when using '%' with setbufvar() or getbufvar().
Solution: Use tv_get_buf_from_arg(). (closes #6816)
Files: src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.1564

Problem: A few remaining errors from ubsan.
Solution: Avoid the warnings. (Dominique Pellé, closes #6837)
Files: src/spellfile.c, src/spellsuggest.c, src/viminfo.c

Patch 8.2.1565

Problem: Spellfile test sometimes fails.
Solution: Check running into the end of the file.
Files: src/spellfile.c

Patch 8.2.1566

Problem: Not all Bazel files are recognized.
Solution: Add *.bazel and *.BUILD. (closes #6836)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1567

Problem: No example to use ubsan with clang.
Solution: Add example commands. (Dominique Pellé, issue #6811)
Files: src/Makefile

Patch 8.2.1568

Problem: prop_find() skips properties in the same line if "skipstart" is used.
Solution: Use "continue" instead of "break". (closes #6840)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.1569

Problem: Vim9: fixes for functions not tested; failure in getchangelist().
Solution: Add tests. (closes #6813, closes #6815, closes #6817)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1570

Problem: Configure check for dirfd() does not work on HP-UX. (Michael Osipov)
Solution: Use AC_TRY_LINK instead of AC_TRY_COMPILE. (closes #6838)
Files: src/configure.ac, src/auto/configure, src/fileio.c, src/globals.h

Patch 8.2.1571

Problem: Vim9: count() third argument cannot be "true".
Solution: Use tv_get_bool_chk(). (closes #6818)
Files: src/typval.c, src/list.c, src/testdir/test_vim9_func.vim

Patch 8.2.1572

Problem: Vim9: expand() does not take "true" as argument.
Solution: Use tv_get_bool_chk(). (closes #6819)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1573

Problem: Vim9: getreg() does not take "true" as argument.
Solution: Use tv_get_bool_chk(). (closes #6820)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1574

Problem: Vim9: glob() does not take "true" as argument.
Solution: Use tv_get_bool_chk(). (closes #6821)
Files: src/filepath.c, src/testdir/test_vim9_func.vim

Patch 8.2.1575

Problem: Vim9: globpath() does not take "true" as argument.
Solution: Use tv_get_bool_chk(). (closes #6821)
Files: src/filepath.c, src/testdir/test_vim9_func.vim

Patch 8.2.1576

Problem: Vim9: index() does not take "true" as argument.
Solution: Use tv_get_bool_chk(). (closes #6823)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1577

Problem: Vim9: hasmapto(), mapcheck() and maparg() do not take "true" as argument.
Solution: Use tv_get_bool(). (closes #6822, closes #6824)
Files: src/evalfunc.c, src/map.c, src/testdir/test_vim9_func.vim

Patch 8.2.1578

Problem: Vim9: popup_clear() does not take "true" as argument.
Solution: Use tv_get_bool(). (closes #6826)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.1579

Problem: Reports from asan are not optimal.
Solution: Use clang with ubsan. (James McCoy, closes #6811)
Files: .travis.yml

Patch 8.2.1580

Problem: Wildmenu does not work properly.
Solution: Do not call may_do_incsearch_highlighting() if completion is in progress.
Files: src/ex_getln.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_1.dump,
src/testdir/dumps/Test_wildmenu_2.dump,
src/testdir/dumps/Test_wildmenu_3.dump,
src/testdir/dumps/Test_wildmenu_4.dump

Patch 8.2.1581

Problem: Using line() for global popup window doesn't work.
Solution: Set tabpage to "curtab". (closes #6847)
Files: src/evalwindow.c, src/testdir/test_popupwin.vim

Patch 8.2.1582

Problem: The channel log does not show typed text.

Solution: Add raw typed text to the log file.
Files: src/ui.c, src/os_win32.c

Patch 8.2.1583

Problem: MS-Windows: cannot easily measure code coverage.
Solution: Add the COVERAGE option. (Ken Takata, closes #6842)
Files: src/Make_cyg_ming.mak

Patch 8.2.1584

Problem: Vim9: cannot use "true" for "skipstart" in prop_find().
Solution: Use dict_get_bool() instead of tv_get_number(). (closes #6852)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.1585

Problem: Messages in errors.h not translated, xgettext on MS-Windows not fully supported.
Solution: Add errors.h to list of input files. Update MS-Windows makefiles to improve message translations. (Ken Takata, closes #6858)
Files: src/po/Make_cyg.mak, src/po/Make_ming.mak, src/po/Make_mvc.mak, src/po/Makefile, src/po/README.txt, src/po/fixfilenames.vim

Patch 8.2.1586

Problem: :resize command not fully tested.
Solution: Add a couple of tests. (Dominique Pellé, closes #6857)
Files: src/testdir/test_window_cmd.vim

Patch 8.2.1587

Problem: Loop for handling keys for the command line is too long.
Solution: Move wild menu handling to separate functions. (Yegappan Lakshmanan, closes #6856)
Files: src/cmdexpand.c, src/proto/cmdexpand.pro, src/ex_getln.c

Patch 8.2.1588

Problem: Cannot read back the prompt of a prompt buffer.
Solution: Add prompt_getprompt(). (Ben Jackson, closes #6851)
Files: runtime/doc/channel.txt, runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/channel.c, src/edit.c, src/evalfunc.c, src/proto/channel.pro, src/proto/edit.pro, src/testdir/test_prompt_buffer.vim

Patch 8.2.1589

Problem: Term_start() options for size are overruled by **'termwinsize'**. (Sergey Vlasov)
Solution: Set **'termwinsize'** to the specified size.
Files: src/terminal.c, src/testdir/test_terminal2.vim, src/testdir/term_util.vim

Patch 8.2.1590

Problem: Vim9: bufnr() doesn't take "true" argument.
Solution: use tv_get_bool_chk(). (closes #6863)
Files: src/evalbuffer.c, src/testdir/test_vim9_func.vim

Patch 8.2.1591

Problem: Using winheight('.') in tests works but is wrong.

Solution: Use winheight(0). (issue #6863)
Files: src/testdir/test_functions.vim, src/testdir/test_quickfix.vim

Patch 8.2.1592

Problem: Vim9: passing "true" to char2nr() fails.
Solution: Use tv_get_bool_chk(). (closes #6865)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1593

Problem: Tests do not check the error number properly.
Solution: Add a colon after the error number. (closes #6869)
Files: src/testdir/test_assert.vim, src/testdir/test_autocmd.vim,
src/testdir/test_backspace_opt.vim, src/testdir/test_channel.vim,
src/testdir/test_clientserver.vim, src/testdir/test_cmdline.vim,
src/testdir/test_const.vim, src/testdir/test_cscope.vim,
src/testdir/test_eval_stuff.vim, src/testdir/test_functions.vim,
src/testdir/test_global.vim, src/testdir/test_gui.vim,
src/testdir/test_hlsearch.vim, src/testdir/test_lambda.vim,
src/testdir/test_let.vim, src/testdir/test_listdict.vim,
src/testdir/test_move.vim, src/testdir/test_normal.vim,
src/testdir/test_popupwin.vim, src/testdir/test_put.vim,
src/testdir/test_quickfix.vim, src/testdir/test_rename.vim,
src/testdir/test_search.vim, src/testdir/test_signs.vim,
src/testdir/test_substitute.vim, src/testdir/test_syntax.vim,
src/testdir/test_tagfunc.vim, src/testdir/test_tagjump.vim,
src/testdir/test_taglist.vim, src/testdir/test_terminal.vim,
src/testdir/test_terminal2.vim, src/testdir/test_textprop.vim,
src/testdir/test_timers.vim, src/testdir/test_true_false.vim,
src/testdir/test_user_func.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_vimscript.vim,
src/testdir/test_winbar.vim, src/testdir/test_winbuf_close.vim,
src/testdir/test_window_cmd.vim, src/testdir/test_writefile.vim

Patch 8.2.1594

Problem: Pull requests on github do not notify a maintainer.
Solution: Add a CODEOWNERS file with a few initial entries.
Files: Filelist, .github/CODEOWNERS

Patch 8.2.1595

Problem: Cannot easily see what Vim sends to the terminal.
Solution: Write output to the channel log if it contains terminal control sequences. Avoid warnings for tputs() argument.
Files: src/term.c, src/globals.h, src/edit.c, src/normal.c, src/optionstr.c

Patch 8.2.1596

Problem: Using win_screenpos('.') in tests works but is wrong.
Solution: Use win_screenpos(0).
Files: src/testdir/test_terminal3.vim

Patch 8.2.1597

Problem: The channel source file is too big.
Solution: Move job related code to a new source file.
Files: Filelist, src/Makefile, src/Make_mvc.mak, src/Make_cyg_ming.mak,

src/channel.c, src/proto/channel.pro, src/job.c,
src/proto/job.pro, src/proto.h, src/edit.c, src/proto/edit.pro,
src/globals.h, src/configure.ac, src/auto/configure

Patch 8.2.1598

Problem: Starting a hidden terminal resizes the current window.
Solution: Do not resize the current window for a hidden terminal.
(closes #6872)
Files: src/terminal.c, src/testdir/test_terminal2.vim

Patch 8.2.1599

Problem: Missing line end when skipping a long line with :cgetfile.
Solution: Fix off-by-one error. (closes #6870)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1600

Problem: Vim9: cannot use "true" with deepcopy().
Solution: Use tv_get_bool_chk(). (closes #6867)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim,
src/testdir/test_listdict.vim

Patch 8.2.1601

Problem: Vim9: cannot use "true" with garbagecollect().
Solution: Use tv_get_bool(). (closes #6871)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1602

Problem: Vim9: cannot use "true" with getbufinfo().
Solution: Use dict_get_bool(). (closes #6873)
Files: src/evalbuffer.c, src/testdir/test_vim9_func.vim

Patch 8.2.1603

Problem: Vim9: cannot use "true" with getchar().
Solution: use tv_get_bool_chk(). (closes #6874)
Files: src/getchar.c, src/testdir/test_vim9_func.vim

Patch 8.2.1604

Problem: Vim9: cannot use "true" with getcompletion().
Solution: use tv_get_bool_chk(). (closes #6875)
Files: src/cmdexpand.c, src/testdir/test_vim9_func.vim

Patch 8.2.1605

Problem: Default maintainer on github is wrong.
Solution: Use Bram's account.
Files: .github/CODEOWNERS

Patch 8.2.1606

Problem: Vim9: cannot use "true" with has().
Solution: Use tv_get_bool(). (closes #6876)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1607

Problem: Vim9: getchar() test fails on MS-Windows.
Solution: First consume any available input.

Files: src/testdir/test_vim9_func.vim

Patch 8.2.1608

Problem: Vim9: getchar() test fails with GUI.
Solution: Avoid that getchar(0) gets stuck on K_IGNORE.
Files: src/getchar.c

Patch 8.2.1609

Problem: Vim9: test fails when build without +channel.
Solution: Add check for +channel. (closes #6879)
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1610

Problem: Vim9: cannot pass "true" to list2str() and str2list().
Solution: Use tv_get_bool_chk(). (closes #6877)
Files: src/evalfunc.c, src/list.c, src/testdir/test_vim9_func.vim

Patch 8.2.1611

Problem: Vim9: cannot pass "true" to nr2char().
Solution: use tv_get_bool_chk(). (closes #6878)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1612

Problem: Vim9: cannot pass "true" to prop_remove().
Solution: Use dict_get_bool(). (closes #6853)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.1613

Problem: Vim9: cannot pass "true" to prop_type_add().
Solution: Use tv_get_bool(). (closes #6850)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.1614

Problem: Vim9: cannot pass "true" to searchcount().
Solution: Use tv_get_bool_chk(). (closes #6854)
Files: src/search.c, src/testdir/test_vim9_func.vim

Patch 8.2.1615

Problem: Vim9: cannot pass "true" to searchdecl().
Solution: use tv_get_bool_chk(). (closes #6881)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1616

Problem: Vim9: cannot pass "true" to synID().
Solution: Use tv_get_bool_chk(). (closes #6860)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1617

Problem: Vim9: cannot pass "true" to win_splitmove().
Solution: Use dict_get_bool(). (closes #6862) Alphabetize test functions.
Files: src/evalwindow.c, src/testdir/test_vim9_func.vim

Patch 8.2.1618

Problem: Vim9: cannot pass "true" to setloclist().

Solution: Use dict_get_bool(). (closes #6882)
Files: src/quickfix.c, src/testdir/test_vim9_func.vim

Patch 8.2.1619

Problem: Vim9: cannot pass "true" to spellsuggest().
Solution: Use tv_get_bool_chk(). (closes #6883)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1620

Problem: searchcount() test fails.
Solution: Restore default flag value.
Files: src/search.c

Patch 8.2.1621

Problem: Crash when using submatch(0, 1) in substitute().
Solution: Increment reference count. (closes #6887)
Files: src/regexp.c, src/testdir/test_substitute.vim

Patch 8.2.1622

Problem: Loop to handle keys for the command line is too long.
Solution: Move code to functions. (Yegappan Lakshmanan, closes #6880)
Files: src/ex_getln.c

Patch 8.2.1623

Problem: Vim9: using :call where it is not needed.
Solution: Remove :call. (closes #6892)
Files: src/testdir/test_maparg.vim, src/testdir/test_textprop.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1624

Problem: Vim9: cannot pass "true" to split(), str2nr() and strchr().
Solution: Use tv_get_bool_chk(). (closes #6884, closes #6885, closes #6886)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1625

Problem: Compiler warning for use of fptr_T.
Solution: Make the type less strict.
Files: src/regexp.c

Patch 8.2.1626

Problem: Test for strchr() fails with different error number.
Solution: Adjust the error number.
Files: src/testdir/test_utf8.vim

Patch 8.2.1627

Problem: Vim9: cannot pass "true" to submatch(), term_gettty() and term_start()
Solution: Use tv_get_bool_chk(). (closes #6888, closes #6890, closes #6889)
Files: src/evalfunc.c, src/terminal.c, src/job.c,
src/testdir/test_vim9_func.vim

Patch 8.2.1628

Problem: Vim9: cannot pass "true" to timer_paused().
Solution: Use tv_get_bool(). (closes #6891)
Files: src/time.c, src/testdir/test_vim9_func.vim

Patch 8.2.1629

Problem: Test fails without terminal feature.
Solution: Check for terminal feature.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1630

Problem: Terminal test fails.
Solution: Correct argument to term_start(). Correct error number.
Files: src/testdir/test_terminal.vim, src/testdir/test_terminal2.vim

Patch 8.2.1631

Problem: test_fails() does not check the context of the line number.
Solution: Use another argument to specify the context of the line number.
Files: runtime/doc/testing.txt, runtime/doc/eval.txt,
src/testdir/test_vim9_func.vim, src/testing.c, src/globals.h,
src/evalfunc.c, src/message.c

Patch 8.2.1632

Problem: Not checking the context of test_fails().
Solution: Add the line number and context arguments. Give error if
assert_fails() argument types are wrong.
Files: src/testing.c, src/errors.h, src/testdir/test_assert.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1633

Problem: Some error messages are internal but do not use iemsg().
Solution: Use iemsg(). (Dominique Pellé, closes #6894)
Files: src/regexp.c, src/regexp_bt.c, src/regexp_nfa.c

Patch 8.2.1634

Problem: Loop to handle keys for the command line is too long.
Solution: Move a few more parts to separate functions. (Yegappan Lakshmanan,
closes #6895)
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.1635

Problem: No digraph for 0x2022 BULLET.
Solution: Use "oo". (Hans Ginzel, closes #6904)
Files: src/digraph.c, runtime/doc/digraph.txt

Patch 8.2.1636

Problem: Get stuck if a popup filter causes an error.
Solution: Check whether the function can be called and does not cause an
error. (closes #6902)
Files: src/structs.h, src/popupwin.c, src/testdir/test_popupwin.vim
src/testdir/dumps/Test_popupwin_wrong_name.dump,
src/testdir/dumps/Test_popupwin_three_errors_1.dump,
src/testdir/dumps/Test_popupwin_three_errors_2.dump

Patch 8.2.1637

Problem: Vim9: :put ={expr} does not work inside :def function.
Solution: Add ISN_PUT. (closes #6397)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/register.c,
src/proto/register.pro, src/edit.c, src/ex_docmd.c, src/mouse.c,
src/normal.c, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1638

Problem: Leaking memory when popup filter function can't be called.
Solution: Don't return too soon.
Files: src/popupwin.c

Patch 8.2.1639

Problem: Options window cannot be translated.
Solution: Get the translation for "local to" texts once and use them in many
places. Fix that '**whichwrap**' is not a local option. (issue #6800)
Files: runtime/optwin.vim

Patch 8.2.1640

Problem: Amiga: missing header for getgrgid().
Solution: Add the grp.h header. (Ola Söder, closes #6906)
Files: src/os_amiga.h

Patch 8.2.1641

Problem: Vim9: cannot use 0 or 1 where a bool is expected.
Solution: Allow using 0 and 1 for a bool type. (closes #6903)
Files: src/vim9compile.c, src/vim9type.c, src/proto/vim9type.pro,
src/structs.h, src/testdir/test_vim9_script.vim

Patch 8.2.1642

Problem: Options test fails.
Solution: Correct call to OptionG().
Files: runtime/optwin.vim

Patch 8.2.1643

Problem: Vim9: :defcompile compiles dead functions.
Solution: Skip over dead functions.
Files: src/userfunc.c

Patch 8.2.1644

Problem: Vim9: cannot assign 1 and 0 to bool at script level.
Solution: Add the TTFLAG_BOOL_OK flag to the type. Fix name of test
function.
Files: src/vim9type.c, src/testdir/test_vim9_script.vim,
src/testdir/vim9.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1645

Problem: GTK3: icons become broken images when resized.
Solution: Use gtk_image_new_from_icon_name(). (closes #6916)
Fix compiler warnings.
Files: src/gui_gtk_x11.c

Patch 8.2.1646

Problem: Amiga: Unnecessary #include.

Solution: Remove the #include. (Ola Söder, closes #6908)
Files: src/version.c

Patch 8.2.1647

Problem: Vim9: result of expression with && and || cannot be assigned to a bool variable.

Solution: Add the TTFLAG_BOOL_OK flag and convert the value if needed.

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1648

Problem: Amiga: no common build file for Amiga (-like) systems.

Solution: Turn Make_morph.mak into Make_ami.mak. (Ola Söder, closes #6805)

Files: Filelist, src/Make_ami.mak, src/Make_morph.mak, src/INSTALLami.txt

Patch 8.2.1649

Problem: GTK3: using old file chooser.

Solution: Use native file chooser on GTK 3.20 and above. (Yogeshwar Velingker, closes #6909)

Files: src/gui_gtk.c

Patch 8.2.1650

Problem: Vim9: result of && and || expression cannot be assigned to a bool at the script level.

Solution: Add the VAR_BOOL_OK flag. Convert to bool when needed.

Files: src/structs.h, src/vim9type.c, src/proto/vim9type.pro,
src/vim9script.c, src/evalvars.c, src/eval.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1651

Problem: Spellfile code not completely tested.

Solution: Add a few more test cases. (Yegappan Lakshmanan, closes #6918)

Files: src/testdir/test_spellfile.vim

Patch 8.2.1652

Problem: Cannot translate lines in the options window.

Solution: Use the AddOption() function to split descriptions where indicated by a line break. (issue #6800)

Files: runtime/optwin.vim

Patch 8.2.1653

Problem: Expand('<stack>') does not include the final line number.

Solution: Add the line number. (closes #6927)

Files: src/vim.h, src/scriptfile.c, src/proto/scriptfile.pro,
src/debugger.c, src/ex_docmd.c, src/ex_eval.c, src/message.c,
src/testing.c, src/testdir/test_expand_func.vim

Patch 8.2.1654

Problem: When job writes to hidden buffer current window has display errors. (Johnny McArthur)

Solution: Use aucmd_prepbuf() instead of switch_to_win_for_buf().
(closes #6925)

Files: src/channel.c

Patch 8.2.1655

Problem: Cannot build with Strawberry Perl 5.32.0.
Solution: Use Perl_sv_2pvbyte_flags. (closes #6921)
Files: src/if_perl.xs

Patch 8.2.1656

Problem: Vim9: callstack wrong if :def function calls :def function.
Solution: Set the line number before calling. (closes #6914)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1657

Problem: Vim9: no proper error for nested ":def!".
Solution: Check for "!". (closes #6920)
Files: src/errors.h, src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1658

Problem: Expand('<stack>') has trailing "..".
Solution: Remove the "..". (closes #6927)
Files: src/scriptfile.c, src/testdir/test_expand_func.vim

Patch 8.2.1659

Problem: Spellfile code not completely tested.
Solution: Add a few more test cases. (Yegappan Lakshmanan, closes #6929)
Files: src/testdir/test_spell.vim, src/testdir/test_spellfile.vim

Patch 8.2.1660

Problem: Assert functions require passing expected result as the first argument, which isn't obvious.
Solution: Use a method, as in "runtest()->assert_equal(expected)".
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1661

Problem: Cannot connect to 127.0.0.1 for host with only IPv6 addresses.
Solution: pass AI_V4MAPPED flag to getaddrinfo. (Filipe Brandenburger, closes #6931)
Files: src/channel.c

Patch 8.2.1662

Problem: :mksession does not restore shared terminal buffer properly.
Solution: Keep a hashtable with terminal buffers. (Rob Pilling, closes #6930)
Files: src/hashtab.c, src/proto/terminal.pro, src/session.c, src/terminal.c, src/testdir/test_mksession.vim

Patch 8.2.1663

Problem: Options window entries cannot be translated.
Solution: Use AddOption() for all explanations. (closes #6800)
Files: runtime/optwin.vim

Patch 8.2.1664

Problem: Memory leak when using :mkview with a terminal buffer.
Solution: Don't use a hashtable for :mkview. (Rob Pilling, closes #6935)
Files: src/session.c, src/terminal.c, src/testdir/test_mksession.vim

Patch 8.2.1665

Problem: Cannot do fuzzy string matching.
Solution: Add matchfuzzy(). (Yegappan Lakshmanan, closes #6932)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/proto/search.pro, src/search.c, src/testdir/test_functions.vim

Patch 8.2.1666

Problem: The initial value of '**backupskip**' can have duplicate items.
Solution: Remove duplicates, like when it is set later. (Tom Ryder,
closes #6940)
Files: src/option.c, src/testdir/test_options.vim

Patch 8.2.1667

Problem: Local function name cannot shadow a global function name.
Solution: Ignore global functions when checking a script-local or scoped
function name. (closes #6926)
Files: src/vim9compile.c, src/userfunc.c, src/proto/userfunc.pro,
src/testdir/test_vim9_func.vim

Patch 8.2.1668

Problem: Vim9: not accepting 0 or 1 as bool when type is any.
Solution: Convert the type with the CHECKTYPE instruction. (closes #6913)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1669

Problem: Vim9: memory leak when storing a value fails.
Solution: Free the value when not storing it.
Files: src/evalvars.c

Patch 8.2.1670

Problem: A couple of gcc compiler warnings.
Solution: Initialize local variables. (Dominique Pellé, closes #6944)
Files: src/memline.c, src/option.c

Patch 8.2.1671

Problem: Vim9: stray error for missing white space.
Solution: Do not skip over white space after member. (closes #6917)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1672

Problem: v_lock is used when it is not initialized. (Yegappan Lakshmanan)
Solution: Initialize the typval in eval1().
Files: src/eval.c

Patch 8.2.1673

Problem: complete_info() selected index has an invalid value. (Ben Jackson)
Solution: Set the index when there is only one match. (closes #6945)
Add test for complete_info().
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.1674

Problem: Vim9: internal error when using variable that was not set.
Solution: Give a meaningful error. (closes #6937)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.1675

Problem: MinGW: testdir makefile deletes non-existing file.
Solution: Use another way to delete the output file if it already exists.
(Michael Soyka)
Files: src/testdir/Make_ming.mak

Patch 8.2.1676

Problem: Compiler warnings for function typecast.
Solution: Add an intermediate cast to "void *".
Files: src/os_unix.c

Patch 8.2.1677

Problem: Memory access errors when calling setloclist() in an autocommand.
Solution: Give an error if the list was changed unexpectedly. (closes #6946)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1678

Problem: Crash when using ":set" after ":ownsyntax". (Dhiraj Mishra)
Solution: Make sure '**spelloptions**' is not NULL. (closes #6950)
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.1679

Problem: Vim9: ":*" is not recognized as a range.
Solution: Move recognizing "*" into skip_range(). (closes #6938)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/cmdexpand.c,
src/ex_getln.c, src/userfunc.c, src/vim9compile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.1680

Problem: Vim9: line number for compare error is wrong.
Solution: Set SOURCING_LNUM. (closes #6936)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1681

Problem: Vim9: unnecessary :call commands in tests.
Solution: Remove the commands. (issue #6936)
Files: src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1682

Problem: Vim9: const works in an unexpected way.
Solution: ":const" only disallows changing the variable, not the value.
Make "list[0] = 9" work at the script level.
Files: src/vim9compile.c, src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1683

Problem: Vim9: assignment test fails.
Solution: Include changes to find Ex command.
Files: src/ex_docmd.c

Patch 8.2.1684

Problem: "gF" does not use line number after file in Visual mode.
Solution: Look for ":123" after the Visual area. (closes #6952)
Files: src/findfile.c, src/testdir/test_gf.vim

Patch 8.2.1685

Problem: Vim9: cannot declare a constant value.
Solution: Introduce ":const!".
Files: runtime/doc/vim9.txt, src/ex_cmds.h, src/vim9compile.c,
src/vim9.h, src/vim9execute.c, src/evalvars.c,
src/proto/evalvars.pro, src/errors.h, src/vim.h, src/eval.c,
src/testdir/test_vim9_script.vim

Patch 8.2.1686

Problem: Vim9: "const!" not sufficiently tested.
Solution: Add a few more test cases. Fix type checking.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1687

Problem: Vim9: out of bounds error.
Solution: Check that cmdidx is not negative.
Files: src/vim9compile.c

Patch 8.2.1688

Problem: Increment/decrement removes text property.
Solution: Insert the new number before deleting the old one. (closes #6962)
Files: src/ops.c, src/testdir/test_textprop.vim

Patch 8.2.1689

Problem: **'colorcolumn'** doesn't show in indent.
Solution: Also draw the column when draw_state is WL_BRI or WL_SBR.
(Alexey Demin, closes #6948, closes #6619)
Files: src/drawline.c, src/testdir/dumps/Test_colorcolumn_2.dump,
src/testdir/dumps/Test_colorcolumn_3.dump,
src/testdir/test_highlight.vim

Patch 8.2.1690

Problem: Text properties not adjusted for "I" in Visual block mode.
Solution: Call inserted_bytes(). (closes #6961)
Files: src/ops.c, src/change.c, src/proto/change.pro,
src/testdir/test_textprop.vim

Patch 8.2.1691

Problem: Vim9: list<any> is not accepted where list<number> is expected.
Solution: Add functions to allocate and free a type_T, use it in
ISN_CHECKTYPE. (closes #6959)
Files: src/vim9.h, src/globals.h, src/vim9compile.c, src/vim9execute.c,
src/vim9type.c, src/proto/vim9type.pro, src/errors.h,
src/evalfunc.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1692

Problem: Build fails because TTFLAG_STATIC is missing.
Solution: Include missing change.
Files: src/structs.h

Patch 8.2.1693

Problem: "hi def" does not work for cleared highlight.

Solution: Check the "sg_cleared" flag. (Maxim Kim, closes #6956,
closes #4405)
Files: src/highlight.c, src/testdir/test_highlight.vim

Patch 8.2.1694

Problem: Compiler warning for loss if data.
Solution: Add typecast.
Files: src/ops.c

Patch 8.2.1695

Problem: Vim9: crash when using varargs type "any".
Solution: Check if uf_va_type is &t_any. (closes #6957)
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.1696

Problem: Unused (duplicate) macros.
Solution: Remove the macros.
Files: src/spell.c

Patch 8.2.1697

Problem: Inconsistent capitalization of error messages.
Solution: Always start with a capital.
Files: src/errors.h, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim,
src/testdir/test_assert.vim

Patch 8.2.1698

Problem: Cannot lock a variable in legacy Vim script like in Vim9.
Solution: Make ":lockvar 0" work.
Files: runtime/doc/eval.txt, src/evalvars.c, src/proto/evalvars.pro,
src/dict.c, src/eval.c, src/list.c, src/typval.c, src/userfunc.c,
src/testdir/test_const.vim, src/testdir/test_listdict.vim

Patch 8.2.1699

Problem: Build failure due to missing error message.
Solution: Add error message.
Files: src/errors.h

Patch 8.2.1700

Problem: Vim9: try/catch causes wrong value to be returned.
Solution: Reset tcd_return. (closes #6964)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1701

Problem: Vim9: sort("i") does not work.
Solution: Don't try getting a number for a string argument. (closes #6958)
Files: src/list.c, src/testdir/test_vim9_func.vim

Patch 8.2.1702

Problem: Crash when using undo after deleting folded lines.
Solution: Check for NULL pointer. (closes #6968)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.1703

Problem: ":highlight clear" does not restore default link.
Solution: Remember the default link and restore it. (Antony Scriven, closes #6970, closes #4405)
Files: runtime/doc/syntax.txt, src/highlight.c, src/testdir/test_highlight.vim

Patch 8.2.1704

Problem: Vim9: crash in for loop when autoload script has an error.
Solution: Reset suppress_errthrow. Check for NULL list. (closes #6967)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1705

Problem: "verbose hi Name" reports incorrect info after ":hi clear".
Solution: Store the script context. (Antony Scriven, closes #6975)
Files: src/highlight.c, src/testdir/test_highlight.vim

Patch 8.2.1706

Problem: Vim9: crash after running into the "Multiple closures" error.
Solution: When a function fails still update any closures. (closes #6973)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1707

Problem: Small inconsistency in highlight test.
Solution: Use one argument for :execute. (Antony Scriven, #6975)
Files: src/testdir/test_highlight.vim

Patch 8.2.1708

Problem: Vim9: error message for function has unprintable characters.
Solution: use printable_func_name(). (closes #6965)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1709

Problem: Vim9: memory leak when using multiple closures.
Solution: Free the partial.
Files: src/vim9execute.c

Patch 8.2.1710

Problem: Vim9: list of list type can be wrong.
Solution: Use VAR_UNKNOWN for empty list. Recognize VAR_UNKNOWN when looking for a common type. (closes #6979)
Files: src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1711

Problem: Vim9: leaking memory when using partial.
Solution: Do delete the function even when it was compiled.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/vim9execute.c

Patch 8.2.1712

Problem: Vim9: leaking memory when calling a lambda.
Solution: Decrement function reference from ISN_DCALL.
Files: src/vim9compile.c, src/userfunc.c, src/proto/userfunc.pro

Patch 8.2.1713

Problem: Motif GUI: crash when setting menu colors. (Andrzej Bylicki)
Solution: Add {} to make "n" incremented correctly. (closes #6989,
closes #5948)
Files: src/gui_motif.c

Patch 8.2.1714

Problem: Text properties corrupted with substitute command. (Filipe
Brandenburger)
Solution: Get the changed line again after using u_savesub(). (closes #6984)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.1715

Problem: Motif GUI: commented out code missed {}.
Solution: Add {} and reenable the code. (similar to #6989)
Files: src/gui_motif.c

Patch 8.2.1716

Problem: Options window has duplicate translations.
Solution: Make one entry for "global or local to buffer". Fix wrong text.
(closes #6983)
Files: runtime/optwin.vim

Patch 8.2.1717

Problem: MS-Windows installer doesn't have Russian translations.
Solution: Add Russian translations. (closes #6985)
Files: nsis/gvim.nsi, nsis/lang/russian.nsi

Patch 8.2.1718

Problem: Vim9: :def function disallows "firstline" and "lastline" argument
names for no good reason.
Solution: Don't check the arguments for a :def function. (closes #6986)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1719

Problem: Vim9: no error if comma is missing in between arguments.
Solution: Give an error message.
Files: src/errors.h, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1720

Problem: Vim9: memory leak with heredoc that isn't executed. (Dominique
Pellé)
Solution: Don't clear the list items. (closes #6991)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1721

Problem: MS-Windows installer doesn't work.
Solution: Write "Russian" in ASCII. (closes #6995, see #6985).
Files: nsis/lang/russian.nsi

Patch 8.2.1722

Problem: Vim9: cannot assign a lambda to a variable of type function.
Solution: Allow for assigning a partial to a variable of type function.
(Naruhiko Nishino, closes #6996)

Files: src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1723

Problem: Vim9: Variable argument name cannot start with underscore.

Solution: Use eval_isnamec1(). (closes #6988)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1724

Problem: Vim9: assignment tests spread out.

Solution: Create new test file for assignment tests.

Files: src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim,
src/testdir/Make_all.mak

Patch 8.2.1725

Problem: Not all Pascal files are recognized.

Solution: Add filetype patterns. (Doug Kearns)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1726

Problem: Fuzzy matching only works on strings.

Solution: Support passing a dict. Add matchfuzzypos() to also get the match positions. (Yegappan Lakshmanan, closes #6947)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/proto/search.pro, src/search.c, src/testdir/Make_all.mak,
src/testdir/test_functions.vim, src/testdir/test_matchfuzzy.vim

Patch 8.2.1727

Problem: A popup created with "cursorline" will ignore "firstline".

Solution: When both "cursorline" and "firstline" are present put the cursor on "firstline". (closes #7000) Add the "winid" argument to getcurpos().

Files: runtime/doc/eval.txt, src/evalfunc.c, src/popupwin.c,
src/evalwindow.c, src/testdir/test_popupwin.vim,
src/testdir/test_functions.vim

Patch 8.2.1728

Problem: Compiler warning for using uninitialized variable. (John Marriott)

Solution: Initialize "neighbor".

Files: src/search.c

Patch 8.2.1729

Problem: Endless loop when ":normal" feeds popup window filter.

Solution: Add the ex_normal_busy_done flag.

Files: src/globals.h, src/getchar.c, src/evalfunc.c, src/ex_docmd.c,
src/menu.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_normal_cmd.dump

Patch 8.2.1730

Problem: Vim9: cannot use member of unknown type.

Solution: When type is unknown use "any". (closes #6997)

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1731

Problem: Vim9: cannot use += to append to empty NULL list.
Solution: Copy the list instead of extending it. (closes #6998)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1732

Problem: Stuck when win_execute() for a popup causes an error.
Solution: Disable the filter callback on error. (issue #6999)
Files: src/popupwin.c, src/testdir/term_util.vim,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_win_execute.dump

Patch 8.2.1733

Problem: Vim9: memory leaks when using nested function.
Solution: Free function when compilation fails.
Files: src/vim9compile.c

Patch 8.2.1734

Problem: Vim9: cannot use a funcref for a closure twice.
Solution: Instead of putting the funcref on the stack use a growarray on the execution context.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1735

Problem: Github actions appear to timeout too soon.
Solution: use "timeout" instead of "ping".
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1736

Problem: Failure to compile a pattern not tested much.
Solution: Add tests where a pattern fails to compile. (Yegappan Lakshmanan, closes #7004)
Files: src/testdir/gen_opt_test.vim, src/testdir/test_arglist.vim,
src/testdir/test_autocmd.vim, src/testdir/test_buffer.vim,
src/testdir/test_checkpath.vim, src/testdir/test_cmdline.vim,
src/testdir/test_debugger.vim, src/testdir/test_functions.vim,
src/testdir/test_history.vim, src/testdir/test_listdict.vim,
src/testdir/test_options.vim, src/testdir/test_search_stat.vim,
src/testdir/test_sort.vim, src/testdir/test_substitute.vim,
src/testdir/test_syntax.vim, src/testdir/test_tagjump.vim,
src/testdir/test_user_func.vim

Patch 8.2.1737

Problem: Cursor line highlight in popup window is not always updated.
Solution: Check if the cursor has moved. (closes #7010)
Files: src/popupwin.c, src/testdir/test_popupwin.vim
src/testdir/dumps/Test_popupwin_win_execute_cursorline.dump

Patch 8.2.1738

Problem: Mac: str2float() recognizes comma instead of decimal point.
Solution: Set LC_NUMERIC to "C". (closes #7003)
Files: src/os_mac_conv.c

Patch 8.2.1739

Problem: Vim9: crash when compiling a manually defined function. (Antony Scriven)
Solution: Check that the script ID is positive. (closes #7012)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.1740

Problem: Test fails without the terminal feature.
Solution: Skip test if the terminal feature is not available.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1741

Problem: pathshorten() only supports using one character.
Solution: Add an argument to control the length. (closes #7006)
Files: runtime/doc/eval.txt, src/evalfunc.c, src/filepath.c, src/proto/filepath.pro, src/testdir/test_functions.vim

Patch 8.2.1742

Problem: Test still fails without the terminal feature.
Solution: Put check for terminal feature in separate function.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.1743

Problem: Cannot build without the eval feature.
Solution: Move shorten_dir outside of #ifdef.
Files: src/filepath.c

Patch 8.2.1744

Problem: Vim9: using ":const!" is weird.
Solution: Use "var" - "final" - "const" like Dart. "let" still works for now.
Files: runtime/doc/vim9.txt, src/ex_cmds.h, src/errors.h, src/evalvars.c, src/proto/evalvars.pro, src/cmdexpand.c, src/eval.c, src/ex_docmd.c, src/vim9compile.c, src/vim9execute.c, src/vim9script.c, src/vim.h, src/ex_cmdidxs.h, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1745

Problem: Tiny version doesn't build.
Solution: Add dummy ex_var() function.
Files: src/ex_docmd.c

Patch 8.2.1746

Problem: Vim9: Cannot use "fina" for "finally". (Naruhiko Nishino)
Solution: Specifically check for "fina". (closes #7020)
Files: src/ex_docmd.c, src/testdir/test_trycatch.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1747

Problem: Result of expand() unexpectedly depends on '**completeslash**'.
Solution: Temporarily reset '**completeslash**'. (Yasuhiro Matsumoto, closes #7021)
Files: src/evalfunc.c, src/testdir/test_ins_complete.vim

Patch 8.2.1748

Problem: Closing split window in other tab may cause a crash.
Solution: Set tp_curwin properly. (Rob Pilling, closes #7018)
Files: src/window.c, src/testdir/test_winbuf_close.vim

Patch 8.2.1749

Problem: Vim9: crash when closure fails in nested function.
Solution: Handle function returns before dereferencing remaining closures.
(closes #7008)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1750

Problem: Setting firstline with popup_setoptions() fails if cursorline is set.
Solution: Use apply_options(). Update the popup before applying "zz".
(closes #7010)
Files: src/popupwin.c, src/proto/popupwin.pro, src/move.c,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_win_execute_cursorline.dump,
src/testdir/dumps/Test_popupwin_set_firstline_1.dump,
src/testdir/dumps/Test_popupwin_set_firstline_2.dump

Patch 8.2.1751

Problem: Using 2 where bool is expected may throw an error.
Solution: Make this backwards compatible.
Files: src/typval.c, src/evalfunc.c, src/testdir/test_search.vim,
src/testdir/test_terminal2.vim

Patch 8.2.1752

Problem: GTK GUI: cannot map alt-? with <A-?>. (Ingo Karkat)
Solution: Adjust the characters for which the shift modifier is removed.
(closes #7016) Make Motif and Win32 use the same function as GTK.
Files: src/misc2.c, src/gui_x11.c, src/gui_w32.c,
src/testdir/test_termcodes.vim

Patch 8.2.1753

Problem: Vim9: crash when using import at script level.
Solution: Give a "not implemented yet" error. (closes #7026)
Files: src/evalvars.c

Patch 8.2.1754

Problem: Completion with spell checking not tested.
Solution: Add a test case. (Dominique Pellé, closes #7024)
Files: src/testdir/test_spell.vim

Patch 8.2.1755

Problem: Vim9: crash when using invalid heredoc marker. (Dhiraj Mishra)
Solution: Check for NULL list. (closes #7027) Fix comment character.
Files: src/vim9compile.c, src/evalvars.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.1756

Problem: Vim9: :let will soon be disallowed.
Solution: Add v:disallow_let temporarily. Fix tests.

Files: src/vim.h, src/errors.h, src/evalvars.c, src/vim9compile.c,
src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1757

Problem: Mac: default locale is lacking the encoding.
Solution: Add ".UTF-8 to the locale. (Yee Cheng Chin, closes #7022)
Files: src/os_mac_conv.c, src/testdir/test_envIRON.vim

Patch 8.2.1758

Problem: Vim9: type of unmaterialized list is wrong.
Solution: Use list<number>.
Files: src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1759

Problem: Vim9: Some tests are still using :let.
Solution: Change more declarations to use :var.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1760

Problem: Vim9: crash when end marker is missing. (Dhiraj Mishra)
Solution: Check for end of function lines. (closes #7031)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1761

Problem: Vim9: Some tests are still using :let.
Solution: Change more declarations to use :var.
Files: src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1762

Problem: When a timer uses :stopinsert Insert mode completion isn't
stopped. (Stanley Chan)
Solution: Call ins_compl_prep(ESC).
Files: src/edit.c, src/testdir/test_ins_complete.vim,
src/testdir/dumps/Test_pum_stopped_by_timer.dump

Patch 8.2.1763

Problem: Vim9: cannot use "true" for popup window scrollbar option.
Solution: use dict_get_bool(). (closes #7029)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.1764

Problem: Vim9: no error when assigning to script var with wrong type.
Solution: Fix off-by-one error. (closes #7028)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1765

Problem: Vim9: some tests use "var var".
Solution: Use "var name". (closes #7032)
Files: src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1766

Problem: Vim9: Some tests are still using :let.

Solution: Change the last few declarations to use :var.
Files: src/testdir/runtest.vim, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_maparg.vim,
src/testdir/test_popupwin.vim, src/testdir/test_textprop.vim

Patch 8.2.1767

Problem: Vim9: test fails with python support.
Solution: Use "let" in legacy function.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1768

Problem: Cannot use the help menu from a terminal window.
Solution: Add ":tlnoremenu" commands. (Yee Cheng Chin, closes #7023)
Files: runtime/menu.vim, src/testdir/test_gui.vim

Patch 8.2.1769

Problem: A popup filter interferes with using :normal to move the cursor in a popup.
Solution: Do not invoke the filter when ex_normal_busy is set.
Files: runtime/doc/popup.txt, src/getchar.c, src/evalfunc.c,
src/ex_docmd.c, src/menu.c, src/globals.h,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_normal_cmd.dump

Patch 8.2.1770

Problem: Invalid memory use when using SpellFileMissing autocmd.
Solution: Add test case. (Dominique Pellé, closes #7036) Fix using a window that was closed.
Files: src/spell.c, src/testdir/test_spell.vim

Patch 8.2.1771

Problem: synIDattr() cannot get the value of ctermul.
Solution: Add the "ul" value for "what". (closes #7037)
Files: runtime/doc/eval.txt, src/highlight.c, src/evalfunc.c,
src/testdir/test_highlight.vim

Patch 8.2.1772

Problem: Cannot use **CTRL-W** <Down> to move out of a terminal window.
Solution: Use special_to_buf() instead of mb_char2bytes(). (closes #7045)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.1773

Problem: Crash when calling mapset() with a list as first argument.
Solution: Check for NULL. (closes #7040)
Files: src/map.c, src/testdir/test_maparg.vim

Patch 8.2.1774

Problem: GTK: hang when forced to exit.
Solution: Do not clean up "mainwin" when really_exiting is set.
(Zdenek Dohnal, closes #7042)
Files: src/gui_gtk_x11.c

Patch 8.2.1775

Problem: MS-Windows: adding a long quickfix list is slow.
Solution: Shorten the buffer name only for the first entry. (Yegappan Lakshmanan, closes #7039, closes #7033)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1776

Problem: Filetype.vim may be loaded twice.
Solution: Do "syntax on" after "filetype on". (Adam Stankiewicz, closes #7049)
Files: runtime/defaults.vim

Patch 8.2.1777

Problem: Vim9: some assignment tests in the wrong file.
Solution: Move assignment tests to test_vim9_assign.
Files: src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1778

Problem: Vim9: returning from a partial call clears outer context, causing a crash.
Solution: Put the outer context in the stack frame. (closes #7044)
Files: src/vim9execute.c, src/vim9.h, src/testdir/test_vim9_func.vim

Patch 8.2.1779

Problem: Some debian changelog files are not recognized.
Solution: Add */debian/changelog. (Jason Franklin)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1780

Problem: Statusline not updated when splitting windows.
Solution: Call status_redraw_all(). (Jason Franklin, closes #5496)
Files: src/window.c, src/testdir/test_statusline.vim

Patch 8.2.1781

Problem: Writing to prompt buffer interferes with insert mode.
Solution: Use win_enter() instead of just setting "curwin". (Ben Jackson, closes #7035)
Files: src/autocmd.c, src/testdir/test_prompt_buffer.vim

Patch 8.2.1782

Problem: Vim9: cannot pass boolean to mapset().
Solution: Use get_tv_bool(). (closes #7041)
Files: src/map.c, src/testdir/test_vim9_func.vim

Patch 8.2.1783 (after 8.2.1781)

Problem: Try-catch test fails.
Solution: Don't call win_enter(), only call entering_window().
Files: src/autocmd.c, src/window.c, src/proto/window.pro, src/testdir/runtest.vim

Patch 8.2.1784

Problem: commits are not scanned for security problems
Solution: Enable Github code scanning. (Christian Brabandt, closes #7057)

Files: .github/workflows/codeql-analysis.yml

Patch 8.2.1785
Problem: Compiler warning for strcpy() out of bounds. (Christian Brabandt)
Solution: use memmove() instead.
Files: src/dict.c

Patch 8.2.1786
Problem: Various Normal mode commands not fully tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #7059)
Files: src/testdir/test_normal.vim, src/testdir/test_regexp_utf8.vim, src/testdir/test_registers.vim, src/testdir/test_spellfile.vim, src/testdir/test_tagjump.vim, src/testdir/test_visual.vim

Patch 8.2.1787
Problem: Crash with 'incsearch' and very long line.
Solution: Check whether regprog becomes NULL. (closes #7063)
Files: src/search.c, src/testdir/test_search.vim

Patch 8.2.1788
Problem: Vim9: still allows :let for declarations.
Solution: Make the default for v:disallow_let one. It can still be set to zero to allow for using :let.
Files: src/evalvars.c, src/testdir/runtest.vim

Patch 8.2.1789
Problem: Vim9: crash with invalid list constant. (Dhiraj Mishra)
Solution: Return FAIL when compiling the list fails. (closes #7066)
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_expr.vim

Patch 8.2.1790
Problem: MS-Windows with Python: crash when executed from Vifm.
Solution: Use NUL instead of CONIN. (Ken Takata, closes #7061, closes #7053)
Files: src/if_python3.c

Patch 8.2.1791
Problem: Vim9: debugger test fails.
Solution: Use "var" instead of "let".
Files: src/testdir/test_debugger.vim

Patch 8.2.1792
Problem: Configure does not recognize Racket 6.1+.
Solution: Add a check for "rktio". (closes #7062)
Files: src/configure.ac, src/auto/configure

Patch 8.2.1793
Problem: Not consistently giving the "is a directory" warning.
Solution: Adjust check for illegal file name and directory. (Yasuhiro Matsumoto, closes #7067)
Files: src/fileio.c, src/testdir/test_edit.vim

Patch 8.2.1794
Problem: No falsy Coalescing operator.
Solution: Add the "??" operator. Fix mistake with function argument count.

Files: runtime/doc/eval.txt, src/eval.c, src/vim9compile.c,
src/vim9type.c, src/testdir/test_expr.vim,
src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1795

Problem: Vim9: operators && and || have a confusing result.
Solution: Make the result a boolean.
Files: runtime/doc/vim9.txt, src/eval.c, src/vim9compile.c,
src/vim9execute.c, src/vim9type.c, src/structs.h, src/vim9.h,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1796

Problem: Vim9: invalid memory access with weird function name. (Dhiraj Mishra)
Solution: Check the name is valid. Add a test.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1797

Problem: Vim9: some parts of the code not tested.
Solution: Add a few tests.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1798

Problem: Vim9: ternary operator condition is too permissive.
Solution: Use tv_get_bool_chk().
Files: runtime/doc/vim9.txt, src/eval.c, src/vim9compile.c,
src/vim9execute.c, src/testdir/vim9.vim,
src/testdir/test_expr.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_script.vim

Patch 8.2.1799

Problem: Some Normal mode commands not fully tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #7073)
Files: src/testdir/test_gf.vim, src/testdir/test_goto.vim,
src/testdir/test_normal.vim, src/testdir/test_registers.vim,
src/testdir/test_startup.vim, src/testdir/test_tabpage.vim,
src/testdir/test_visual.vim

Patch 8.2.1800

Problem: Vim9: memory leak if "if" condition is invalid.
Solution: Free ppconst earlier.
Files: src/vim9compile.c

Patch 8.2.1801

Problem: Undo file not found when using ":args" or ":next".
Solution: Handle like editing another file. (closes #7072)
Files: src/ex_cmds.c, src/testdir/test_undo.vim

Patch 8.2.1802

Problem: Vim9: crash with unterminated dict. (Dhiraj Mishra)
Solution: Return empty string instead of NULL. (closes #7084)

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1803

Problem: A few failures are not tested.

Solution: Test a few failures. (Dominique Pellé, closes #7075)

Files: src/testdir/test_arglist.vim, src/testdir/test_cmdline.vim,
src/testdir/test_json.vim, src/testdir/test_listdict.vim

Patch 8.2.1804

Problem: resolve('/') returns an empty string.

Solution: Don't remove single slash. (closes #7074)

Files: src/filepath.c, src/testdir/test_functions.vim

Patch 8.2.1805

Problem: Unix: terminal mode changed when using ":shell".

Solution: Avoid calling settmode() when not needed. (issue #7079)

Files: src/os_unix.c

Patch 8.2.1806

Problem: MS-Windows with Python: Vim freezes after import command.

Solution: Use either "NUL" or "CONIN\$" when reopening stdin. (Yasuhiro
Matsumoto, closes #7083)

Files: src/if_python3.c

Patch 8.2.1807

Problem: Can use :help in a terminal popup window.

Solution: Give an error. (closes #7088)

Files: src/help.c, src/testdir/test_popupwin.vim

Patch 8.2.1808

Problem: No test coverage for ":spelldump!".

Solution: Add a test. (Dominique Pellé, closes #7089)

Files: src/testdir/test_spell.vim

Patch 8.2.1809

Problem: Mapping some keys with Ctrl does not work properly.

Solution: For terminal, GTK and Motif handle "@", "^" and "_" codes.

Files: src/misc2.c, src/proto/misc2.pro, src/term.c, src/gui_gtk_x11.c,
src/gui_x11.c, src/testdir/test_termcodes.vim

Patch 8.2.1810

Problem: Some code in normal.c not covered by tests.

Solution: Add normal mode tests. (Yegappan Lakshmanan, closes #7086)

Files: src/testdir/test_charsearch.vim, src/testdir/test_normal.vim

Patch 8.2.1811

Problem: Mapping Ctrl-key does not work for '{', '}' and '|'.

Solution: Remove the shift modifier. (closes #6457)

Files: runtime/doc/map.txt, src/misc2.c, src/testdir/test_termcodes.vim

Patch 8.2.1812

Problem: Vim9: nested closure throws an internal error.

Solution: Do not skip a local variable with a partial. (closes #7065)

Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1813

Problem: Vim9: can assign wrong type to script dict. (Christian J. Robinson)
Solution: Check the type if known.
Files: src/structs.h, src/eval.c, src/vim9script.c,
src/proto/vim9script.pro, src/proto/evalvars.pro,
src/testdir/test_vim9_script.vim

Patch 8.2.1814 (after 8.2.1813)

Problem: Missing change to remove "static".
Solution: Add the change.
Files: src/evalvars.c

Patch 8.2.1815

Problem: Vim9: memory leak when using function reference.
Solution: Temporarily disable the test.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.1816

Problem: Vim9: another memory leak when using function reference.
Solution: Temporarily disable the tests.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1817

Problem: Vim9: wrong instruction when reusing a local variable spot.
Solution: Clear a newly allocated local variable. (closes #7080)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1818

Problem: SE Linux: deprecation warning for security_context_t.
Solution: Use "char *" instead. (James McCoy, closes #7093)
Files: src/os_unix.c

Patch 8.2.1819

Problem: Vim9: Memory leak when using a closure.
Solution: Compute the minimal refcount in the funcstack. Reenable disabled tests.
Files: src/vim9execute.c, src/proto/vim9execute.pro, src/structs.h,
src/eval.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.1820

Problem: Vim9: crash when error happens in timer callback.
Solution: Check that current_exception is not NULL. (closes #7100)
Files: src/ex_docmd.c

Patch 8.2.1821

Problem: Vim9: concatenating to a NULL list doesn't work.
Solution: Handle a NULL list like an empty list. (closes #7064)
Files: src/list.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1822 (after 8.2.1821)

Problem: List test doesn't fail.
Solution: Adjust the test for NULL list handling.

Files: src/testdir/test_listdict.vim

Patch 8.2.1823

Problem: "gN" does not select the matched string.
Solution: Move the cursor to the start of the match.
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.2.1824

Problem: Vim9: variables at the script level escape their scope.
Solution: When leaving a scope remove variables declared in it.
Files: src/structs.h, src/ex_eval.c, src/evalvars.c,
src/proto/evalvars.pro, src/testdir/test_vim9_script.vim

Patch 8.2.1825

Problem: Vim9: accessing freed memory.
Solution: Clear sv_name when the variable is deleted.
Files: src/ex_eval.c

Patch 8.2.1826

Problem: Vim9: cannot use a {} block at script level.
Solution: Recognize a {} block.
Files: src/ex_docmd.c, src/ex_cmds.h, src/ex_cmdidxs.h, src/ex_eval.c,
src/structs.h, src/proto/ex_eval.pro, src/errors.h,
src/testdir/test_vim9_script.vim

Patch 8.2.1827

Problem: Filetype detection does not test enough file names.
Solution: Test more file names. (Adam Stankiewicz, closes #7099)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1828

Problem: Build failure without the +eval feature.
Solution: Add dummies for ex_block and ex_endblock.
Files: src/ex_docmd.c

Patch 8.2.1829

Problem: Warnings when executing Github actions.
Solution: Use another method to set environment variables. (Ken Takata,
closes #7107)
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1830

Problem: MS-Windows: Python3 issue with stdin.
Solution: Check if stdin is readable. (Ken Takata, closes #7106)
Files: src/if_python3.c

Patch 8.2.1831

Problem: File missing from distribution.
Solution: Add the github code analyses file.
Files: Filelist

Patch 8.2.1832

Problem: readdirex() error is displayed as a message. (Yegappan Lakshmanan)
Solution: Use semsg() instead of msg().

Files: src/fileio.c, src/testdir/test_functions.vim

Patch 8.2.1833

Problem: When reading from stdin dup() is called twice.
Solution: Remove the dup() in main.c. (Ken Takata, closes #7110)
Files: src/main.c

Patch 8.2.1834

Problem: PyEval_InitThreads() is deprecated in Python 3.9.
Solution: Do not call PyEval_InitThreads in Python 3.9 and later. (Ken Takata, closes #7113) Avoid warnings for functions.
Files: src/if_python3.c, src/if_py_both.h

Patch 8.2.1835

Problem: ":help ??" finds the "!!" tag.
Solution: Do not translate "?" into ".". (Naruhiko Nishino, closes #7114, closes #7115)
Files: src/help.c, src/testdir/test_help_tagjump.vim

Patch 8.2.1836

Problem: Autocmd test fails on pacifist systems.
Solution: Check that /bin/kill exists. (James McCoy, closes #7117)
Tune the timing, make the autocmd test run faster.
Files: src/testdir/test_autocmd.vim

Patch 8.2.1837

Problem: Using "gn" after "gN" does not work.
Solution: Extend the other end of the Visual area. (closes #7109)
Files: src/search.c, src/testdir/test_gn.vim

Patch 8.2.1838

Problem: Vim9: cannot insert a comment line in an expression.
Solution: Skip comment lines at the script level. (closes #7111)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1839

Problem: Vim9: memory leaks reported in assign test.
Solution: Move the failing job_start() call to separate test files, it causes false leakreports.
Files: src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_fails.vim, src/testdir/Make_all.mak

Patch 8.2.1840

Problem: Vim9: error message is not clear about compilation error.
Solution: Say "compiling" instead of "processing".
Files: src/vim9compile.c, src/message.c, src/globals.h, src/testdir/test_vim9_func.vim

Patch 8.2.1841

Problem: Vim9: test for compilation error fails in normal build.
Solution: Invoke CheckRunVimInTerminal in a separate function.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.1842

Problem: Crash when USE_FNAME_CASE is defined and using :browse.
Solution: Don't use read-only memory for ".". (Yegappan Lakshmanan, closes #7123)
Files: src/ex_cmds.c, src/ex_docmd.c, src/testdir/test_edit.vim

Patch 8.2.1843

Problem: Netbeans: with huge buffer number memory allocation may fail.
Solution: Check for size overflow.
Files: src/netbeans.c

Patch 8.2.1844

Problem: Using "q" at the more prompt doesn't stop a long message.
Solution: Check for "got_int". (closes #7122)
Files: src/message.c, src/testdir/test_messages.vim, src/testdir/dumps/Test_quit_long_message.dump

Patch 8.2.1845

Problem: Vim9: function defined in a block can't use variables defined in that block.
Solution: First step: Make a second hashtable that holds all script variables, also block-local ones, with more information.
Files: src/structs.h, src/evalvars.c, src/ex_eval.c, src/vim9script.c, src/proto/vim9script.pro, src/scriptfile.c

Patch 8.2.1846

Problem: Vim9: variables declared in a local block are not found in when a function is compiled.
Solution: Look for script variables in sn_all_vars.
Files: src/structs.h, src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/proto/userfunc.pro, src/ex_eval.c, src/vim9script.c, src/proto/vim9script.pro, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.1847

Problem: Vim9: using negative value for unsigned type.
Solution: Use zero instead of -1.
Files: src/vim9compile.c

Patch 8.2.1848

Problem: Crash when passing a NULL string or list to popup_settext().
Solution: Check for NULL pointers. (closes #7132)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.1849

Problem: Vim9: garbage collection frees block-local variables.
Solution: Mark all script variables as used.
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.1850

Problem: "vat" does not select tags correctly over line break.
Solution: Adjust the search pattern. (Aufar Gilbran, closes #7136)
Files: src/textobject.c, src/testdir/test_textobjects.vim

Patch 8.2.1851

Problem: Vim9: "!" followed by space incorrectly used.
Solution: Skip over trailing spaces. (closes #7131)
Files: src/eval.c, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1852

Problem: map() returning zero for NULL list is unexpected.
Solution: Return the empty list. (closes #7133)
Files: src/list.c, src/testdir/test_filter_map.vim,
src/testdir/test_blob.vim

Patch 8.2.1853

Problem: "to_f" is recognized at "topleft" modifier.
Solution: Do not recognize modifier when "_" follows. (closes #7019)
Files: src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1854

Problem: Vim9: crash when throwing exception for NULL string. (Dhiraj Mishra)
Solution: Handle NULL string like empty string. (closes #7139)
Files: src/vim9execute.c, src/errors.h, src/testdir/test_vim9_script.vim

Patch 8.2.1855

Problem: Vim9: get error message when nothing is wrong.
Solution: Check called_emsg instead of did_emsg. (closes #7143)
Files: src/vim9compile.c, src/vim9execute.c, src/errors.h

Patch 8.2.1856

Problem: "2resize" uses size of current window. (Daniel Steinberg)
Solution: Use size of resized window. (Yasuhiro Matsumoto, closes #7152)
Files: src/ex_docmd.c, src/testdir/test_window_cmd.vim

Patch 8.2.1857

Problem: Vim9: using job_status() on an unused var gives an error.
Solution: Return "fail". (closes #7158)
Files: src/job.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1858

Problem: Vim9: filter functions return number instead of bool.
Solution: Return v:true instead of one. (closes #7144)
Files: src/popupwin.c, src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.1859

Problem: Vim9: crash in unpack assignment.
Solution: Make sure an error message is turned into an exception.
(closes #7159)
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.1860

Problem: Vim9: memory leak when throwing empty string.
Solution: Free the empty string.
Files: src/vim9execute.c

Patch 8.2.1861

Problem: Vim9: no specific error when parsing lambda fails.
Solution: Also give syntax errors when not evaluating. (closes #7154)
Files: src/dict.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1862

Problem: vim9: memory leak when compiling lambda fails.
Solution: Call clear_evalarg().
Files: src/vim9compile.c

Patch 8.2.1863

Problem: Json code not sufficiently tested.
Solution: Add more test cases. (Dominique Pellé, closes #7166)
Files: src/testdir/test_json.vim

Patch 8.2.1864

Problem: Vim9: no error for wrong list type.
Solution: Add flag to indicate a constant. (closes #7160)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1865

Problem: Vim9: add() does not check type of argument.
Solution: Inline the add() call. (closes #7160)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/errors.h,
src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1866

Problem: Vim9: appending to pushed blob gives wrong result.
Solution: Set ga_maxlen when copying a blob.
Files: src/blob.c, src/testdir/test_vim9_func.vim

Patch 8.2.1867

Problem: Vim9: argument to add() not checked for blob.
Solution: Add the BLOBAPPEND instruction.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/errors.h,
src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1868

Problem: Vim9: no error for missing space after comma in dict.
Solution: Check for white space. (closes #6672)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1869

Problem: Vim9: memory leak when using add().
Solution: Free the added item.
Files: src/vim9execute.c

Patch 8.2.1870

Problem: Vim9: no need to keep all script variables.
Solution: Only keep script variables when a function was defined that could use them. Fix freeing static string on exit.
Files: src/vim9script.c, src/proto/vim9script.pro, src/structs.h,
src/ex_eval.c, src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.1871

Problem: Using %v in **'errorformat'** may fail before %Z.
Solution: Set qf_viscol only when qf_col is set. (closes #7169)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1872

Problem: Matchfuzzy() does not prefer sequential matches.
Solution: Give sequential matches a higher bonus. (Christian Brabandt, closes #7140)
Files: src/search.c, src/testdir/test_matchfuzzy.vim

Patch 8.2.1873

Problem: Vim9: missing white space when using <f-args>.
Solution: Add spaces. (Christian J. Robinson)
Files: src/usercmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1874

Problem: Can't do something just before leaving Insert mode.
Solution: Add the InsertLeavePre autocommand event. (closes #7177)
Files: runtime/doc/autocmd.txt, src/edit.c, src/vim.h, src/autocmd.c, src/testdir/test_edit.vim

Patch 8.2.1875

Problem: Warning when building GTK gui.
Solution: Add missing function parameter.
Files: src/gui_gtk_f.c

Patch 8.2.1876

Problem: Vim9: argument types for builtin functions are not checked at compile time.
Solution: Add an argument type checking mechanism. Implement type checks for one function.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9compile.c, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_builtin.vim, src/testdir/Make_all.mak

Patch 8.2.1877 (after 8.2.1876)

Problem: Test for function list fails.
Solution: Move "obsolete" comments one line up.
Files: src/evalfunc.c

Patch 8.2.1878

Problem: GTK: error for redefining function. (Tony Mechelynck)
Solution: Remove "gtk_" prefix from local functions and prepend "gui_" to global functions.
Files: src/gui_gtk_f.c, src/gui_gtk_f.h, src/gui_gtk.c, src/gui_gtk_x11.c

Patch 8.2.1879

Problem: Vim9: argument types of insert() not checked when compiling.
Solution: Add argument type checks for insert().
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.1880

Problem: Vim9: Asan complains about adding zero to NULL.
Solution: Check for argument count first.
Files: src/vim9compile.c

Patch 8.2.1881

Problem: Cannot build with GTK3.
Solution: Adjust form functions.
Files: src/gui_gtk_f.c

Patch 8.2.1882

Problem: Vim9: v:disallow_let is no longer needed.
Solution: Remove v:disallow_let.
Files: src/evalvars.c, src/vim.h, src/vim9compile.c

Patch 8.2.1883

Problem: Compiler warnings when using Python.
Solution: Adjust PyCFunction to also have the second argument. Use "int" return type for some functions. Insert "(void *)" to get rid of the remaining warnings.
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c

Patch 8.2.1884

Problem: Compiler warning for uninitialized variable. (John Marriott)
Solution: Initialize with NULL.
Files: src/vim9compile.c, src/evalfunc.c

Patch 8.2.1885

Problem: Filetype tests unnecessarily creates swap files.
Solution: Disable 'swapfile'. (Ken Takata, closes #7183)
Files: src/testdir/test_filetype.vim

Patch 8.2.1886

Problem: Using ":silent!" in a popup filter has unexpected effect.
Solution: Use did_emsg instead of called_emsg. (closes #7178)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.1887

Problem: Github actions not optimally configured.
Solution: Run CI on any pushed branches. Set fail-fast. (Ozaki Kiichi, closes #7184)
Files: .github/workflows/ci-windows.yaml

Patch 8.2.1888

Problem: Vim9: Getbuflines(-1, 1, '\$') gives an error.
Solution: Return an empty list. (closes #7180)
Files: src/evalbuffer.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.1889

Problem: Vim9: erroneous error for missing white space after `{}`.
Solution: Don't skip over white space after `{}`. (issue #7167)
Files: src/dict.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1890

Problem: Vim9: strange error for subtracting from a list.
Solution: Check getting a number, not a string. (closes #7167)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1891

Problem: Vim9: skipping over expression doesn't handle line breaks.
Solution: Pass evalarg to skip_expr(). (closes #7157)
Files: src/vim9compile.c, src/eval.c, src/proto/eval.pro, src/ex_docmd.c,
src/misc1.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1892

Problem: Valgrind warns for using uninitialized access in tests.
Solution: Fix condition for breaking out of loop. (Dominique Pellé,
closes #7187)
Files: src/terminal.c

Patch 8.2.1893

Problem: Fuzzy matching does not support multiple words.
Solution: Add support for matching white space separated words. (Yegappan
Lakshmanan, closes #7163)
Files: runtime/doc/eval.txt, src/search.c,
src/testdir/test_matchfuzzy.vim

Patch 8.2.1894

Problem: Vim9: command modifiers are not supported.
Solution: Support "silent" and "silent!".
Files: src/structs.h, src/vim9compile.c, src/vim9.h, src/vim9execute.c,
src/evalvars.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_cmd.vim

Patch 8.2.1895 (after 8.2.1894)

Problem: Vim9: silent command modifier test fails.
Solution: Add missing changes.
Files: src/ex_docmd.c

Patch 8.2.1896

Problem: Valgrind warns for using uninitialized memory.
Solution: NUL terminate the SmcOpenConnection() error message. (Dominique
Pellé, closes #7194)
Files: src/os_unix.c

Patch 8.2.1897

Problem: Command modifiers are saved and set inconsistently.
Solution: Separate parsing and applying command modifiers. Save values in
cmdmod_T.
Files: src/structs.h, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/ex_cmds.h, src/vim9compile.c

Patch 8.2.1898

Problem: Command modifier parsing always uses global cmdmod.
Solution: Pass in cmdmod_T to use. Rename struct fields consistently.
Files: src/structs.h, src/arglist.c, src/buffer.c, src/bufwrite.c,
src/diff.c, src/change.c, src/cmdhist.c, src/edit.c,
src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_getln.c,

src/fileio.c, src/filepath.c, src/gui.c, src/gui_gtk_x11.c,
src/help.c, src/if_cscope.c, src/indent.c, src/mark.c,
src/memline.c, src/message.c, src/option.c, src/ops.c,
src/os_unix.c, src/quickfix.c, src/register.c, src/scriptfile.c,
src/search.c, src/session.c, src/tag.c, src/terminal.c,
src/textformat.c, src/usercmd.c, src/vim9compile.c, src/window.c,
src/proto/ex_docmd.pro

Patch 8.2.1899

Problem: Crash in out-of-memory situation.
Solution: Bail out if shell_name is NULL. (Dominique Pellé, closes #7196)
Files: src/ex_cmds.c

Patch 8.2.1900

Problem: Vim9: command modifiers do not work.
Solution: Make most command modifiers work.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/usercmd.c, src/proto/usercmd.pro, src/scriptfile.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.1901

Problem: Variable completion does not work in command line window.
Solution: Use the "prevwin". (closes #7198)
Files: src/evalvars.c, src/testdir/test_ins_complete.vim

Patch 8.2.1902

Problem: Default option values are changed when using :badd for an existing buffer.
Solution: When calling buflist_new() pass a zero line number. (closes #7195)
Files: src/ex_cmds.c, src/testdir/test_buffer.vim

Patch 8.2.1903 (after 8.2.1902)

Problem: Buffer test fails with normal features.
Solution: Use 'numberwidth' instead of 'conceallevel' in the test.
Files: src/testdir/test_buffer.vim

Patch 8.2.1904

Problem: Still using default option values after using ":badd +1".
Solution: Find a window where options were set. Don't set the window when using ":badd".
Files: src/buffer.c, src/ex_cmds.c, src/vim.h,
src/testdir/test_buffer.vim

Patch 8.2.1905

Problem: The wininfo list may contain stale entries.
Solution: When closing a window remove any other entry where the window pointer is NULL.
Files: src/buffer.c, src/proto/buffer.pro, src/window.c

Patch 8.2.1906

Problem: Warning for signed/unsigned.
Solution: Use size_t instead of int. (Mike Williams)
Files: src/proto/usercmd.pro, src/usercmd.c, src/vim9execute.c

Patch 8.2.1907

Problem: Complete_info().selected may be wrong.
Solution: Update cp_number if it was never set. (issue #6945)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.1908

Problem: Lua is initialized even when not used.
Solution: Put lua_init() after check for "eap->skip". (Christian Brabandt, closes #7191). Avoid compiler warnings.
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.1909

Problem: Number of status line items is limited to 80.
Solution: Dynamically allocate the arrays. (Rom Grk, closes #7181)
Files: runtime/doc/options.txt, src/buffer.c, src/optionstr.c, src/proto/buffer.pro, src/screen.c, src/structs.h, src/testdir/test_options.vim, src/testdir/test_statusline.vim, src/vim.h

Patch 8.2.1910

Problem: Reading past the end of the command line.
Solution: Check for NUL. (closes #7204)
Files: src/ex_docmd.c, src/testdir/test_edit.vim

Patch 8.2.1911

Problem: Tiny build fails.
Solution: Add #ifdef.
Files: src/insexpand.c

Patch 8.2.1912

Problem: With Python 3.9 some tests fail.
Solution: Take into account the different error message. (James McCoy, closes #7210)
Files: src/testdir/test_python3.vim

Patch 8.2.1913

Problem: GTK GUI: rounding for the cell height is too strict.
Solution: Round up above 15/16 of a pixel. (closes #7203)
Files: src/gui_gtk_x11.c

Patch 8.2.1914

Problem: Vim9: cannot put line break in expression for '=' register.
Solution: Pass fgetline to set_expr_line(). (closes #7209)
Files: src/register.c, src/proto/register.pro, src/ex_docmd.c, src/eval.c, src/proto/eval.pro, src/misc2.c, src/testdir/test_vim9_script.vim

Patch 8.2.1915

Problem: Vim9: error for wrong number of arguments is not useful.
Solution: Mention whatever we have for the name. (closes #7208)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.1916

Problem: Vim9: function call is aborted even when "silent!" is used.

Solution: Use did_emsg instead of called_emsg. (closes #7213)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1917

Problem: No test for improved Man command.
Solution: Test that shell arguments are properly escaped.
Files: src/testdir/test_man.vim

Patch 8.2.1918

Problem: Vim9: E1100 mentions :let.
Solution: Mention "var". (closes #7207)
Files: src/vim9script.c, src/errors.h

Patch 8.2.1919

Problem: Assert_fails() setting emsg_silent changes normal execution.
Solution: Use a separate flag in_assert_fails.
Files: src/testing.c, src/globals.h, src/buffer.c, src/change.c, src/fileio.c, src/insexpand.c, src/message.c, src/misc1.c, src/normal.c, src/screen.c, src/term.c, src/vim9execute.c, src/testdir/test_vim9_func.vim, src/testdir/gen_opt_test.vim, src/testdir/test_autocmd.vim, src/testdir/test_mapping.vim, src/testdir/test_popup.vim, src/testdir/test_terminal.vim

Patch 8.2.1920

Problem: Listlbr test fails when run after another test.
Solution: Add test separately to list of test targets.
Files: src/testdir/Make_all.mak, src/testdir/test_alot_utf8.vim

Patch 8.2.1921

Problem: Fuzzy matching does not recognize path separators.
Solution: Add a bonus for slash and backslash. (Yegappan Lakshmanan, closes #7225)
Files: src/search.c, src/testdir/test_matchfuzzy.vim

Patch 8.2.1922

Problem: Win32: scrolling doesn't work properly when part of window is off-screen.
Solution: Fall back to GDI scrolling if part of the window is off-screen. Handle multi-monitor setup better. (Ken Takata, closes #7219)
Files: src/gui_w32.c

Patch 8.2.1923

Problem: Vim9: "filter" command modifier doesn't work.
Solution: Check for space on char before argument. (closes #7216, closes #7222)
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1924

Problem: Vim9: crash when indexing dict with NULL key.
Solution: Use empty string instead of NULL. (closes #7229) Make error message more useful for empty string.
Files: src/vim9execute.c, src/globals.h, src/testdir/test_vim9_expr.vim

Patch 8.2.1925 (after 8.2.1924)

Problem: List/dict test fails.
Solution: Correct expected exception.
File: src/testdir/test_listdict.vim

Patch 8.2.1926

Problem: Cannot use a space in '**spellfile**'. (Filipe Brandenburger)
Solution: Permit using a space. (closes #7230)
Files: src/spell.c, src/testdir/gen_opt_test.vim

Patch 8.2.1927

Problem: Vim9: get unknown error with an error in a timer function.
Solution: Use did_emsg instead of called_emsg. (closes #7231)
Files: src/vim9compile.c, src/vim9execute.c

Patch 8.2.1928

Problem: Vim9: "silent!" not effective when list index is wrong.
Solution: Ignore list index failure when emsg_silent is set. (closes #7232)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.1929

Problem: MS-Windows: problem loading Perl 5.32.
Solution: Define NO_THREAD_SAFE_LOCALE. (Ken Takata, closes #7234)
Files: src/if_perl.xs

Patch 8.2.1930

Problem: Wrong input if removing shift results in special key code.
Solution: Handle special key codes. (closes #7189)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.1931

Problem: Vim9: arguments of extend() not checked at compile time.
Solution: Add argument type checking for extend().
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.1932

Problem: Compiler warnings when building with Athena GUI.
Solution: Fix function signatures.
Files: src/gui_at_fs.c

Patch 8.2.1933

Problem: Cannot sort using locale ordering.
Solution: Add a flag for :sort and sort() to use the locale. (Dominique Pellé, closes #7237)
Files: runtime/doc/change.txt, runtime/doc/eval.txt, src/ex_cmds.c, src/list.c, src/testdir/test_sort.vim

Patch 8.2.1934

Problem: Vim9: command modifiers in :def function not tested.
Solution: Add tests. Fix using modifier before filter command.
Files: src/ex_docmd.c, src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1935 (after 8.2.1933)

Problem: Sort test fails on Mac.
Solution: Disable the sort test with locale on Mac.

Files: src/testdir/test_sort.vim

Patch 8.2.1936

Problem: Session sets the local '**scrolloff**' value to the global value.
Solution: Do not let restoring the global option value change the local value.

Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.1937

Problem: Vim9: test for confirm modifier fails in some situations.
Solution: Add a short wait. Handle failure better.

Files: src/testdir/term_util.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.1938

Problem: Wiping out a terminal buffer makes some tests fail.

Solution: Do not wipe out the terminal buffer unless wanted.

Files: src/testdir/term_util.vim, src/testdir/test_terminal.vim,
src/testdir/test_terminal3.vim

Patch 8.2.1939

Problem: Invalid memory access in Ex mode with global command.

Solution: Make sure the cursor is on a valid line. (closes #7238)

Files: src/move.c, src/testdir/test_ex_mode.vim

Patch 8.2.1940

Problem: Vim9: browse modifier test fails on Mac.

Solution: Only test when the +browse feature is available.

Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.1941

Problem: Ex mode test fails on MS-Windows with GUI.

Solution: Skip the test when using gvim.

Files: src/testdir/test_ex_mode.vim

Patch 8.2.1942

Problem: Insufficient test coverage for the Netbeans interface.

Solution: Add more tests. Fix an uncovered bug. (Yegappan Lakshmanan,
closes #7240)

Files: runtime/doc/netbeans.txt, src/mouse.c,
src/testdir/test_netbeans.py, src/testdir/test_netbeans.vim,
src/testdir/test_quickfix.vim

Patch 8.2.1943

Problem: Vim9: wrong error message when colon is missing.

Solution: Check for a missing colon. (issue #7239)

Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1944

Problem: Netbeans test is flaky.

Solution: Add a short delay. (Yegappan Lakshmanan, closes #7246)

Files: src/testdir/test_netbeans.vim

Patch 8.2.1945

Problem: Crash when passing NULL function to reduce().

Solution: Check for NULL pointer and give an error. (Dominique Pellé, closes #7243)
Files: src/list.c, src/errors.h, src/testdir/test_listdict.vim

Patch 8.2.1946

Problem: sort() with NULL string not tested.
Solution: Add a test. use v:collate. (Dominique Pellé, closes #7247)
Files: src/testdir/test_sort.vim

Patch 8.2.1947

Problem: Crash when using "zj" without folds. (Sean Dewar)
Solution: Check for at least one fold. (closes #7245)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.1948

Problem: GUI: crash when handling message while closing a window. (Srinath Avadhanula)
Solution: Don't handle message while closing a window. (closes #7250)
Files: src/window.c, src/globals.h, src/getchar.c

Patch 8.2.1949

Problem: Vim9: using extend() on null dict is silently ignored.
Solution: Give an error message. Initialize a dict variable with an empty dictionary. (closes #7251)
Files: src/errors.h, src/list.c, src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.1950

Problem: Vim9: crash when compiling function fails when getting type.
Solution: Handle NULL type. (closes #7253)
Files: src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1951 (after 8.2.1949)

Problem: Test for list and dict fails.
Solution: Adjust for using an empty list/dict for a null one.
Files: src/testdir/test_listdict.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.1952

Problem: Vim9: crash when using a NULL dict key.
Solution: Use a NULL dict key like an empty string. (closes #7249)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.1953

Problem: Vim9: extra "unknown" error after other error.
Solution: Restore did_emsg count after EXEC instruction. (closes #7254)
Improve error message from assert_fails()
Files: src/vim9execute.c, src/testing.c, src/testdir/test_vim9_script.vim, src/testdir/test_assert.vim

Patch 8.2.1954

Problem: Vim9: not all command modifiers are tested.
Solution: Add tests for "keep" modifiers. Fix that marks are lost even though ":lockmarks" is used.

Files: src/ex_cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.1955

Problem: Vim9: not all command modifiers are tested.

Solution: Add tests for remaining modifiers.

Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.1956

Problem: Vim9: cannot specify argument types for lambda.

Solution: Allow adding argument types. Check arguments when calling a function reference.

Files: src/userfunc.c, src/proto/userfunc.pro, src/vim9compile.c,
src/eval.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.1957

Problem: Diff and cursorcolumn highlighting don't mix.

Solution: Fix condition for what attribute to use. (Christian Brabandt, closes #7258, closes #7260)

Files: src/drawline.c, src/testdir/dumps/Test_diff_cuc_01.dump,
src/testdir/dumps/Test_diff_cuc_02.dump,
src/testdir/dumps/Test_diff_cuc_03.dump,
src/testdir/dumps/Test_diff_cuc_04.dump,
src/testdir/test_diffmode.vim

Patch 8.2.1958 (after 8.2.1956)

Problem: Build failure with timers.

Solution: Add missing change.

Files: src/popupwin.c

Patch 8.2.1959

Problem: Crash when terminal buffer name is made empty. (Dhiraj Mishra)

Solution: Fall back to "[No Name]". (closes #7262)

Files: src/buffer.c, src/proto/buffer.pro, src/terminal.c,
src/testdir/test_terminal.vim

Patch 8.2.1960

Problem: Warning for uninitialized variable.

Solution: Initialize the variable.

Files: src/evalfunc.c

Patch 8.2.1961

Problem: Various comments can be improved.

Solution: Various comment adjustments.

Files: src/dict.c, src/structs.h, src/time.c, src/testdir/shared.vim,
src/testdir/test_netbeans.vim, src/gui_motif.c

Patch 8.2.1962

Problem: Netbeans may access freed memory.

Solution: Check the buffer pointer is still valid. Add a test. (Yegappan Lakshmanan, closes #7248)

Files: src/netbeans.c, src/testdir/test_netbeans.vim

Patch 8.2.1963

Problem: Crash when using a popup window with "latin1" encoding.
Solution: Don't use ScreenLinesUC when enc_utf8 is false. (closes #7241)
Files: src/screen.c, src/terminal.c, src/testdir/test_popupwin.vim

Patch 8.2.1964

Problem: Not all ConTeXt files are recognized.
Solution: Add two patterns. (closes #7263)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.1965

Problem: Vim9: tests fail without the channel feature.
Solution: Check if the channel feature is present. (Dominique Pellé, closes #7270)
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1966

Problem: Popup becomes current window after closing a terminal window.
Solution: When restoring the window after executing autocommands, check that the window ID is still the same. (Naruhiko Nishino, closes #7272)
Files: src/autocmd.c, src/window.c, src/proto/window.pro, src/structs.h, src/testdir/test_popupwin.vim

Patch 8.2.1967

Problem: The session file does not restore the alternate file.
Solution: Add ":balt". Works like ":badd" and also sets the buffer as the alternate file. Use it in the session file. (closes #7269, closes #6714)
Files: runtime/doc/windows.txt, src/ex_cmds.h, src/ex_cmdidxs.h, src/ex_docmd.c, src/vim.h, src/ex_cmds.c, src/session.c, src/testdir/test_buffer.vim

Patch 8.2.1968

Problem: Vim9: has() assumes a feature does not change dynamically.
Solution: Check whether a feature may change dynamically. (closes #7265)
Files: src/vim9compile.c, src/evalfunc.c, src/proto/evalfunc.pro, src/testdir/test_vim9_disassemble.vim

Patch 8.2.1969

Problem: Vim9: map() may change the list or dict item type.
Solution: Add mapnew().
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/list.c, src/proto/list.pro, src/testdir/test_filter_map.vim

Patch 8.2.1970

Problem: It is easy to make mistakes when cleaning up swap files after the system crashed.
Solution: Warn for the process still running after recovery. Do not automatically delete a swap file created on another system. (David Fries, closes #7273)
Files: src/memline.c, src/testdir/test_swap.vim

Patch 8.2.1971

Problem: Memory leak when map() fails.

Solution: Clear the typval.
Files: src/list.c

Patch 8.2.1972

Problem: Crash when recreating nested fold.
Solution: Check for empty growarray. (closes #7278)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.1973

Problem: Finding a patch number can be a bit slow.
Solution: Use binary search. (closes #7279)
Files: src/version.c

Patch 8.2.1974

Problem: Vim9: test for has('gui_running') fails with VIMDLL.
Solution: Adjust the #ifdef. (Ken Takata, closes #7276)
Files: src/evalfunc.c

Patch 8.2.1975

Problem: Win32: memory leak when encoding conversion fails.
Solution: Free the allocated memory. (Ken Takata, closes #7277)
Files: src/os_win32.c

Patch 8.2.1976

Problem: Cannot backspace in prompt buffer after using cursor-left. (Maxim Kim)
Solution: Ignore "arrow_used" in a prompt buffer. (closes #7281)
Files: src/edit.c, src/testdir/test_prompt_buffer.vim

Patch 8.2.1977

Problem: Vim9: error for using a string in a condition is confusing.
Solution: Give a more specific error. Also adjust the compile time type checking for || and &&.
Files: src/vim9compile.c, src/vim9execute.c, src/proto/vim9execute.pro, src/typval.c, src/errors.h, src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.1978

Problem: Making a mapping work in all modes is complicated.
Solution: Add the <Cmd> special key. (Yegappan Lakshmanan, closes #7282, closes 4784, based on patch by Bjorn Linse)
Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, runtime/doc/map.txt, src/edit.c, src/errors.h, src/ex_docmd.c, src/ex_getln.c, src/getchar.c, src/insexpand.c, src/keymap.h, src/map.c, src/misc2.c, src/normal.c, src/ops.c, src/proto/getchar.pro, src/screen.c, src/terminal.c, src/testdir/test_mapping.vim

Patch 8.2.1979

Problem: "term_opencmd" option of term_start() is truncated. (Sergey Vlasov)
Solution: Allocate the buffer to hold the command. (closes #7284)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.1980

Problem: Vim9: some tests are not done at the script level.
Solution: Use CheckDefAndScriptSuccess() in more places. Fix uncovered problems.
Files: src/eval.c, src/list.c, src/scriptfile.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.1981

Problem: MinGW: parallel compilation might fail.
Solution: Add dependencies on \$(OUTDIR). (Masamichi Abe, closes #7287)
Files: src/Make_cyg_ming.mak

Patch 8.2.1982

Problem: Quickfix window not updated when adding invalid entries.
Solution: Update the quickfix buffer properly. (Yegappan Lakshmanan, closes #7291, closes #7271)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.1983

Problem: ml_get error when using <Cmd> to open a terminal.
Solution: If the window changed reset the incsearch state. (closes #7289)
Files: src/ex_getln.c, src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_from_cmd.dump

Patch 8.2.1984

Problem: Cannot use :vimgrep in omni completion, causing C completion to fail.
Solution: Add the EX_LOCK_OK flag to :vimgrep. (closes #7292)
Files: src/ex_cmds.h, src/testdir/test_quickfix.vim

Patch 8.2.1985

Problem: Crash when closing terminal popup with <Cmd> mapping.
Solution: Check b_term is not NULL. (closes #7294)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.1986

Problem: Expression test is flaky on Appveyor.
Solution: Temporarily disable the test in MS-Windows.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.1987

Problem: MS-Windows: Win32.mak is no longer needed.
Solution: Do not include Win32.mak. (Jason McHugh, closes #7290)
Files: src/Make_mvc.mak, src/INSTALLpc.txt

Patch 8.2.1988

Problem: Still in Insert mode when opening terminal popup with a <Cmd> mapping in Insert mode.
Solution: Exit Insert mode. (closes #7295)
Files: src/edit.c, src/testdir/test_terminal.vim

Patch 8.2.1989

Problem: Info popup triggers WinEnter and WinLeave autocommands.

Solution: Suppress autocommands for the info popup. (closes #7296)
Files: src/popupmenu.c, src/testdir/test_popupwin.vim

Patch 8.2.1990

Problem: Cursor position wrong in terminal popup with finished job.
Solution: Only add the top and left offset when not done already.
(closes #7298)
Files: src/popupwin.c, src/structs.h, src/drawline.c, src/move.c,
src/terminal.c, src/testdir/dumps/Test_terminal_popup_m1.dump

Patch 8.2.1991

Problem: Coverity warns for not using the ga_grow() return value.
Solution: Bail out if ga_grow() fails. (Yegappan Lakshmanan, closes #7303)
Files: src/getchar.c

Patch 8.2.1992

Problem: Build fails with small features.
Solution: Add #ifdef.
Files: src/move.c

Patch 8.2.1993

Problem: Occasional failure of the netbeans test.
Solution: Add "silent!". (Yegappan Lakshmanan, closes #7304)
Files: src/testdir/test_netbeans.vim

Patch 8.2.1994 (after 8.2.1981)

Problem: MS-Windows: MinGW always does a full build.
Solution: Only check if \$OUTDIR exists. (Masamichi Abe, closes #7311)
Files: src/Make_cyg_ming.mak

Patch 8.2.1995

Problem: The popup menu can cause too much redrawing.
Solution: Reduce the length of the displayed text. (Yasuhiro Matsumoto,
closes #7306)
Files: src/popupmenu.c

Patch 8.2.1996

Problem: Vim9: invalid error for argument of extend().
Solution: Check if the type could match. (closes #7299)
Files: src/evalfunc.c, src/vim9compile.c, src/proto/vim9compile.pro,
src/vim9type.c, src/proto/vim9type.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.1997

Problem: Window changes when using bufload() while in a terminal popup.
Solution: When searching for a window by ID also find a popup window.
(closes #7307)
Files: src/window.c, src/testdir/test_terminal.vim

Patch 8.2.1998

Problem: Terminal Cmd test sometimes fails to close popup.
Solution: Add "term_finish" option.
Files: src/testdir/test_terminal.vim

Patch 8.2.1999

Problem: Terminal popup test sometimes fails.
Solution: Wait for the popup to close.
Files: src/testdir/test_terminal.vim

Patch 8.2.2000

Problem: Vim9: dict.key assignment not implemented yet.
Solution: Implement dict.key assignment. (closes #7312)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2001

Problem: Vim9: :def function does not apply **'maxfuncdepth'**.
Solution: Use **'maxfuncdepth'**. (issue #7313)
Files: src/vim9execute.c, src/userfunc.c, src/proto/userfunc.pro,
src/testdir/test_vim9_func.vim

Patch 8.2.2002

Problem: Vim9: lambda argument shadowed by function name.
Solution: Let function name be shadowed by lambda argument. (closes #7313)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2003

Problem: Build error with +conceal but without +popupwin.
Solution: Add #ifdef. (Tom Ryder, closes #7316)
Files: src/drawline.c

Patch 8.2.2004 (after 8.2.2002)

Problem: Compiler warning for uninitialized variable.
Solution: Initialize "ufunc". (John Marriott)
Files: src/vim9compile.c

Patch 8.2.2005

Problem: Redoing a mapping with <Cmd> doesn't work properly.
Solution: Fill the redo buffer. Use "<SNR>" instead of a key code.
(closes #7282)
Files: src/ops.c, src/getchar.c, src/testdir/test_mapping.vim

Patch 8.2.2006

Problem: .pbtxt files are not recognized.
Solution: Recognize .pbtxt as protobuf text buffers. (closes #7326)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2007

Problem: Test for insert mode in popup is not reliable.
Solution: Wait for the popup to disappear. (Ozaki Kiichi, closes #7321)
Files: src/testdir/test_terminal.vim

Patch 8.2.2008

Problem: MS-Windows GUI: handling channel messages lags.
Solution: Reduce the wait time from 100 to 10 msec. (closes #7097)
Files: src/gui_w32.c

Patch 8.2.2009

Problem: MS-Windows: setting \$LANG in gvimext only causes problems.

Solution: Do not set \$LANG. (Ken Takata, closes #7325)
Files: src/GvimExt/gvimext.cpp

Patch 8.2.2010

Problem: Vim9: compiling fails for unreachable return statement.
Solution: Fix it. (closes #7319)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2011

Problem: "syn sync" reports a very large number.
Solution: Use "at the first line".
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.2012

Problem: Vim9: confusing error message when using bool wrongly.
Solution: Mention "Bool" instead of "Special". (closes #7323)
Files: src/typval.c, src/errors.h, src/testdir/test_vim9_expr.vim

Patch 8.2.2013

Problem: Vim9: not skipping white space after unary minus.
Solution: Skip whitespace. (closes #7324)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2014

Problem: Using **CTRL-O** in a prompt buffer moves cursor to start of the line.
Solution: Do not move the cursor when restarting edit. (closes #7330)
Files: src/job.c, src/testdir/test_prompt_buffer.vim

Patch 8.2.2015

Problem: Vim9: literal dict #{} is not like any other language.
Solution: Support the JavaScript syntax.
Files: runtime/doc/vim9.txt, src/vim9compile.c,
src/proto/vim9compile.pro, src/errors.h,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.2016

Problem: Swap file test is a little flaky.
Solution: Don't set a byte to a fixed value, increment it.
Files: src/testdir/test_swap.vim

Patch 8.2.2017 (after 8.2.2015)

Problem: Missing part of the dict change.
Solution: Also change the script level dict.
Files: src/dict.c

Patch 8.2.2018

Problem: Vim9: script variable not found from lambda.
Solution: In a lambda also check the script hashtab for a variable without a scope. (closes #7329)
Files: src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.2019 (after 8.2.2016)

Problem: Swap file test fails on MS-Windows.

Solution: Add four to the process ID. (Ken Takata, closes #7333)
Files: src/testdir/test_swap.vim

Patch 8.2.2020

Problem: Some compilers do not like the "namespace" argument.
Solution: Rename to "use_namespace". (closes #7332)
Files: src/vim9compile.c, src/proto/vim9compile.pro

Patch 8.2.2021

Problem: Vim9: get E1099 when autocommand resets did_emsg.
Solution: Add did_emsg_cumul. (closes #7336)
Files: src/globals.h, src/ex_docmd.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2022

Problem: Vim9: star command recognized erroneously.
Solution: Give an error for missing colon. (issue #7335)
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2023

Problem: Vim: memory leak when :execute fails.
Solution: Clear the growarray.
Files: src/vim9execute.c

Patch 8.2.2024

Problem: Flicker when redrawing a popup with a title and border.
Solution: Do not redraw the border where the title is displayed. (Naruhiko Nishino, closes #7334)
Files: src/popupwin.c

Patch 8.2.2025

Problem: Amiga: Not all colors are used on OS4.
Solution: Adjust the #ifdef to include __amigaos4__. (Ola Söder, closes #7328)
Files: src/term.c

Patch 8.2.2026

Problem: Coverity warns for possibly using not NUL terminated string.
Solution: Put a NUL in b0_hname just in case.
Files: src/memline.c

Patch 8.2.2027

Problem: Coverity warns for uninitialized field.
Solution: Set "v_lock".
Files: src/list.c

Patch 8.2.2028

Problem: Coverity warns for using an uninitialized variable.
Solution: Initialize to NULL.
Files: src/eval.c

Patch 8.2.2029

Problem: Coverity warns for not checking return value.
Solution: Check that u_save_cursor() returns OK.

Files: src/ops.c

Patch 8.2.2030

Problem: Some tests fail on Mac.

Solution: Avoid Mac test failures. Add additional test for wildmenu.
(Yegappan Lakshmanan, closes #7341)

Files: src/testdir/runtest.vim, src/testdir/test_cmdline.vim,
src/testdir/test_options.vim, src/testdir/test_popupwin.vim

Patch 8.2.2031

Problem: Some tests fail when run under valgrind.

Solution: Avoid timing problems.

Files: src/testdir/test_vim9_func.vim, src/testdir/test_channel.vim,
src/testdir/test_clientserver.vim, src/testdir/test_debugger.vim,
src/testdir/test_quotestar.vim

Patch 8.2.2032

Problem: Cabalconfig and cabalproject filetypes not recognized.

Solution: Detect more cabal files. (Marcin Szamotulski, closes #7339)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2033

Problem: Vim9: :def without argument gives compilation error.

Solution: Add the DEF instruction. (closes #7344)

Files: src/ex_docmd.c, src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/userfunc.c, src/proto/userfunc.pro,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.2034

Problem: Vim9: list unpack in for statement not compiled yet.

Solution: Compile list unpack. (closes #7345)

Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/errors.h,
src/eval.c, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2035

Problem: MS-Windows: some tests may fail.

Solution: Avoid test failures. (Yegappan Lakshmanan, closes #7346)

Files: src/testdir/test_channel.vim, src/testdir/test_ex_mode.vim,
src/testdir/test_functions.vim

Patch 8.2.2036

Problem: Current buffer is messed up if creating a new buffer for the
quickfix window fails.

Solution: Check that creating the buffer succeeds. (closes #7352)

Files: src/quickfix.c, src/testdir/test_quickfix.vim,
src/testdir/dumps/Test_quickfix_window_fails.dump

Patch 8.2.2037

Problem: Compiler test depends on list of compiler plugins.

Solution: Compare with the actual list of compiler plugins.

Files: src/testdir/test_compiler.vim

Patch 8.2.2038

Problem: Compiler test fails on MS-Windows.
Solution: Sort the found compiler plugin names.
Files: src/testdir/test_compiler.vim

Patch 8.2.2039

Problem: Viminfo is not written when creating a new file.
Solution: Set "b_marks_read" in the new buffer. (Christian Brabandt, closes #7350)
Files: src/bufwrite.c, src/testdir/test_viminfo.vim

Patch 8.2.2040

Problem: Terminal buffer disappears even when 'bufhidden' is "hide". (Sergey Vlasov)
Solution: Check 'bufhidden' when a terminal buffer becomes hidden. (closes #7358)
Files: src/buffer.c, src/testdir/test_terminal.vim

Patch 8.2.2041

Problem: Haskell filetype not optimally recognized.
Solution: Recognize all *.hsc files as Haskell. (Marcin Szamotulski, closes #7354)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2042

Problem: Build failure with +profile but without +reltime.
Solution: Adjust #ifdef. (Christian Brabandt, closes #7361)
Files: src/syntax.c

Patch 8.2.2043

Problem: GTK3: white border around text stands out.
Solution: Use current theme color. (closes #7357, issue #349)
Files: src/gui_gtk_x11.c

Patch 8.2.2044

Problem: MS-Windows: swap file test sometimes fails.
Solution: Use a more reliable way to change the process ID. When "timeout" fails use "ping" to wait up to ten minutes. (Ken Takata, closes #7365)
Files: .github/workflows/ci-windows.yaml, src/testdir/test_swap.vim

Patch 8.2.2045

Problem: Highlighting a character too much with incsearch.
Solution: Check "search_match_endcol". (Christian Brabandt, closes #7360)
Files: src/drawline.c, src/testdir/test_search.vim, src/testdir/dumps/Test_incsearch_newline1.dump, src/testdir/dumps/Test_incsearch_newline2.dump, src/testdir/dumps/Test_incsearch_newline3.dump, src/testdir/dumps/Test_incsearch_newline4.dump, src/testdir/dumps/Test_incsearch_newline5.dump

Patch 8.2.2046

Problem: Some test failures don't give a clear error.
Solution: Use assert_match() and assert_fails() instead of assert_true().

(Ken Takata, closes #7368)
Files: src/testdir/test_autocmd.vim, src/testdir/test_backspace_opt.vim

Patch 8.2.2047

Problem: Amiga: FEAT_ARP defined when it should not.
Solution: Adjust #ifdef. (Ola Söder, closes #7370)
Files: src/feature.h

Patch 8.2.2048

Problem: Amiga: obsolete code.
Solution: Remove the unused lines. (Ola Söder, closes #7373)
Files: src/gui.c

Patch 8.2.2049

Problem: Amiga: obsolete function.
Solution: Remove the function. (Ola Söder, closes #7374)
Files: src/memfile.c

Patch 8.2.2050

Problem: Search test contains unneeded sleeps.
Solution: Rename the function, remove sleeps. (Christian Brabandt, closes #7369)
Files: src/testdir/test_search.vim

Patch 8.2.2051

Problem: Vim9: crash when aborting a user function call.
Solution: Do not use the return value when aborting. (closes #7372)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2052

Problem: Vim9: "edit +4 fname" gives an error. (Naruhiko Nishino)
Solution: Allow using a range in the +cmd argument. (closes #7364)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/vim.h, src/ex_cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2053

Problem: Vim9: lambda doesn't accept argument types.
Solution: Optionally accept argument types at the script level.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2054

Problem: Amiga: FEAT_ARP defined when it should not.
Solution: Adjust "||" to "&&" in #ifdef. (Ola Söder, closes #7375)
Files: src/feature.h

Patch 8.2.2055

Problem: MS-Windows: two Vim instances may use the same temp file.
Solution: Use the process ID for the temp name. (Ken Takata, closes #7378)
Files: src/fileio.c

Patch 8.2.2056

Problem: Configure fails when building with the "implicit-function-declaration" error enabled, specifically on Mac.
Solution: Declare the functions like in the source code. (suggestion by

Clemens Lang, closes #7380)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2057

Problem: Getting the selection may trigger TextYankPost autocmd.
Solution: Only trigger the autocommand when yanking in Vim, not for getting the selection. (closes #7367)
Files: src/clipboard.c, src/normal.c, src/register.c, src/testdir/test_autocmd.vim

Patch 8.2.2058

Problem: Using mkview/loadview changes the jumplist.
Solution: Use ":keepjumps". Don't let ":badd" or ":balt" change the jumplist. (closes #7371)
Files: src/session.c, src/ex_docmd.c, src/testdir/test_mksession.vim

Patch 8.2.2059

Problem: Amiga: can't find plugins.
Solution: Do not use "*" in the pattern. (Ola Söder, closes #7384)
Files: src/main.c

Patch 8.2.2060

Problem: Check for features implemented with "if".
Solution: Use the Check commands. (Ken Takata, closes #7383)
Files: src/testdir/test_autocmd.vim, src/testdir/test_compiler.vim, src/testdir/test_delete.vim, src/testdir/test_diffmode.vim, src/testdir/test_expr.vim, src/testdir/test_fold.vim

Patch 8.2.2061

Problem: Vim9: E1030 error when using empty string for term_sendkeys().
Solution: Don't check for an invalid type unless the terminal can't be found. (closes #7382)
Files: src/terminal.c, src/testdir/test_termcodes.vim

Patch 8.2.2062

Problem: <Cmd> does not handle **CTRL-V**.
Solution: Call get_literal() after encountering **CTRL-V**. (closes #7387)
Files: src/getchar.c, src/testdir/test_mapping.vim

Patch 8.2.2063

Problem: Vim9: only one level of indexing supported.
Solution: Handle more than one index in an assignment.
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_assign.vim

Patch 8.2.2064

Problem: terminal: cursor is on while redrawing, causing flicker.
Solution: Switch the cursor off while redrawing. Always add the top and left offset to the cursor position when not done already. (closes #5943)
Files: src/terminal.c, src/popupwin.c

Patch 8.2.2065

Problem: Using map() and filter() on a range() is inefficient.
Solution: Do not materialize the range. (closes #7388)

Files: src/list.c, src/testdir/test_functions.vim

Patch 8.2.2066

Problem: Vim9: assignment with += doesn't work.

Solution: Do not see the "+" as an addition operator.

Files: src/eval.c, src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2067 (after 8.2.2064)

Problem: Cursor position in popup terminal is wrong.

Solution: Don't check the flags.

Files: src/terminal.c, src/testdir/test_popupwin.vim

Patch 8.2.2068

Problem: Transparent syntax item uses start/end of containing region.

Solution: Do not change the startpos and endpos of a transparent region to that of its containing region. (Adrian Ghizaru, closes #7349, closes #7391)

Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.2069

Problem: The quickfix window is not updated after setqflist().

Solution: Update the quickfix buffer. (Yegappan Lakshmanan, closes #7390, closes #7385)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.2070

Problem: Can't get the exit value in VimLeave or VimLeavePre autocommands.

Solution: Add v:exiting like in Neovim. (Yegappan Lakshmanan, closes #7395)

Files: runtime/doc/autocmd.txt, runtime/doc/eval.txt, src/evalvars.c, src/main.c, src/testdir/test_exit.vim, src/vim.h

Patch 8.2.2071

Problem: Vim9: list assign doesn't accept an empty remainder list.

Solution: Recognize list assignment with ";".

Files: src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2072

Problem: Vim9: list assign not well tested.

Solution: Test with different destinations. Fix white space error.

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2073

Problem: Vim9: for with unpack only works for local variables.

Solution: Recognize different destinations.

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2074

Problem: Vim9: using :normal from Vim9 script can't handle range.

Solution: Execute a :normal command in legacy script context. (closes #7401)

Files: src/structs.h, src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2075

Problem: Error for const argument to mapnew().

Solution: Don't give an error. (closes #7400)

Files: src/list.c, src/testdir/test_filter_map.vim

Patch 8.2.2076

Problem: MS-Windows console: sometimes drops typed characters.
Solution: Do not wait longer than 10 msec for input. (issue #7164)
Files: src/os_win32.c

Patch 8.2.2077

Problem: Build failure with small features.
Solution: Add #ifdef.
Files: src/structs.h, src/ex_docmd.c

Patch 8.2.2078

Problem: Illegal memory access when using :print on invalid text. (Dhiraj Mishra)
Solution: Check for more composing characters than supported. (closes #7399)
Files: src/message.c, src/testdir/test_utf8.vim

Patch 8.2.2079

Problem: Vim9: cannot put a linebreak before or after "in" of ":for".
Solution: Skip over linebreak.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2080

Problem: Vim9: no proper error message for using s:var in for loop.
Solution: Give a specific error.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2081

Problem: Vim9: cannot handle a linebreak after "=" in assignment.
Solution: Skip over linebreak. (closes #7407)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.2082

Problem: Vim9: can still use the deprecated #{} dict syntax.
Solution: Remove support for #{} in Vim9 script. (closes #7406, closes #7405)
Files: src/dict.c, src/proto/dict.pro, src/eval.c, src/vim9compile.c,
src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_popupwin.vim,
src/testdir/test_textprop.vim

Patch 8.2.2083

Problem: Vim9: crash when using ":silent!" and getting member fails.
Solution: Jump to on_fatal_error. (closes #7412)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2084

Problem: **CTRL-V** U doesn't work to enter a Unicode character when modifyOtherKeys is effective. (Ken Takata)
Solution: Add a flag to get_literal() for the shift key. (closes #7413)

Files: src/edit.c, src/proto/edit.pro, src/ex_getln.c, src/getchar.c,
src/normal.c, src/testdir/test_termcodes.vim

Patch 8.2.2085

Problem: Qt translation file is recognized as typescript.
Solution: Check the first line for "<?xml". (closes #7418)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2086

Problem: Libvterm tests are only run on Linux.
Solution: Use static libraries. (Ozaki Kiichi, closes #7419)
Files: .travis.yml, src/Makefile, src/libvterm/Makefile,
src/libvterm/t/run-test.pl

Patch 8.2.2087

Problem: Vim9: memory leak when statement is truncated.
Solution: Increment the number of local variables.
Files: src/vim9compile.c

Patch 8.2.2088

Problem: Vim9: script test sometimes fails.
Solution: Unlet variables.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2089

Problem: Libvterm test fails to build on Mac.
Solution: Adjust configure to remove a space between -L and the path that
follows.
Files: src/configure.ac, src/auto/configure

Patch 8.2.2090

Problem: Vim9: dict does not accept a key in quotes.
Solution: Recognize a key in single or double quotes.
Files: runtime/doc/vim9.txt, src/dict.c, src/proto/dict.pro,
src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2091

Problem: MS-Windows: build warnings.
Solution: Add a #pragma to suppress the deprecation warning. (Ken Takata)
Avoid using a non-ASCII character. (closes #7421)
Files: src/message.c, src/os_win32.c

Patch 8.2.2092

Problem: Vim9: unpredictable errors for script tests.
Solution: Use a different script file name for each run.
Files: src/testdir/vim9.vim, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_quickfix.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.2093

Problem: Vim9: script test sometimes fails.
Solution: Do not find a script variable by its typval if the name was
cleared.
Files: src/vim9script.c

Patch 8.2.2094

Problem: When an expression fails getting the next command may be wrong.
Solution: Do not check for a next command after :eval fails. (closes #7415)
Files: src/eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2095

Problem: Vim9: crash when failed dict member is followed by concatenation.
Solution: Remove the dict from the stack. (closes #7416)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2096

Problem: Vim9: command modifiers not restored after assignment.
Solution: Jump to nextline instead of using continue.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2097

Problem: Vim9: using :silent! when calling a function prevents aborting that function.
Solution: Add emsg_silent_def and did_emsg_def.
Files: src/globals.h, src/message.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2098

Problem: Vim9: function argument of sort() and map() not tested.
Solution: Add a couple of tests.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.2099

Problem: Vim9: some checks are not tested.
Solution: Add a few more tests. Give better error messages.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.2100

Problem: Insufficient testing for function range and dict.
Solution: Add a few tests. (Dominique Pellé, closes #7428)
Files: src/testdir/test_functions.vim, src/testdir/test_lambda.vim,
src/testdir/test_signals.vim, src/testdir/test_user_func.vim

Patch 8.2.2101

Problem: Vim9: memory leak when literal dict has an error and when an expression is not complete.
Solution: Clear the typval and the growarray.
Files: src/dict.c, src/vim9compile.c

Patch 8.2.2102

Problem: Vim9: not all error messages tested.
Solution: Add a few test cases.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.2103

Problem: Vim9: unreachable code.

Solution: Remove the code to prepend s: to the variable name
Files: src/vim9compile.c

Patch 8.2.2104

Problem: Build problem with Ruby 2.7.
Solution: Adjust function declarations. (Ozaki Kiichi, closes #7430)
Files: src/configure.ac, src/auto/configure, src/if_ruby.c

Patch 8.2.2105

Problem: Sound test is a bit flaky.
Solution: Use WaitForAssert(). (Dominique Pellé, closes #7429)
Files: src/testdir/test_sound.vim

Patch 8.2.2106

Problem: TOML files are not recognized.
Solution: Match *.toml. (issue #7432)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2107

Problem: Vim9: some errors not tested.
Solution: Add tests. Fix getting the right error.
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.2108

Problem: Vim9: no test to check for :let error.
Solution: Add a test. Rename tests from _let_ to _var_.
Files: src/testdir/test_vim9_assign.vim

Patch 8.2.2109

Problem: "vim -" does not work well when modifyOtherKeys is enabled and a shell command is executed on startup.
Solution: Only change modifyOtherKeys when executing a shell command in raw mode.
Files: src/os_unix.c

Patch 8.2.2110

Problem: Cannot use ":shell" when reading from stdin. (Gary Johnson)
Solution: Revert patch 8.2.1833.
Files: src/main.c

Patch 8.2.2111

Problem: GTK: Menu background is the same color as the main window.
Solution: Fix white space around the text in another way. (closes #7437, closes #7427)
Files: src/gui_gtk_x11.c

Patch 8.2.2112

Problem: Running tests may leave some files behind.
Solution: Delete the right files. Fix a few typos. (Dominique Pellé, closes #7436)
Files: src/testdir/test_filetype.vim, src/testdir/test_messages.vim,
src/testdir/test_mksession.vim

Patch 8.2.2113

Problem: MS-Windows GUI: crash after using ":set guifont=" four times.
Solution: Check for NULL pointer. (Ken Takata, closes #7434)
Files: src/gui_dwrite.cpp, src/testdir/test_gui.vim

Patch 8.2.2114

Problem: Vim9: unreachable code in assignment.
Solution: Remove impossible condition and code.
Files: src/vim9compile.c

Patch 8.2.2115

Problem: Vim9: some errors not tested for; dead code.
Solution: Add a test. Remove dead code.
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2116

Problem: MS-Windows GUI: test for 'guifont' is incomplete.
Solution: Set 'renderoptions'. (Christian Brabandt)
Files: src/testdir/test_gui.vim

Patch 8.2.2117

Problem: Some functions use any value as a string.
Solution: Check that the value is a non-empty string.
Files: src/typval.c, src/proto/typval.pro, src/mbyte.c, src/filepath.c,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.2118

Problem: Dead code in the job support. (Dominique Pellé)
Solution: Define USE_ARGV before checking for it.
Files: src/job.c

Patch 8.2.2119

Problem: GTK3: status line background color is wrong.
Solution: Don't change the code for earlier GTK3 versions. (closes #7444)
Files: src/gui_gtk_x11.c

Patch 8.2.2120

Problem: Not all Perl functionality is tested.
Solution: Add a few more test cases. (Dominique Pellé, closes #7440)
Files: src/testdir/test_perl.vim

Patch 8.2.2121

Problem: Internal error when using \ze before \zs in a pattern.
Solution: Check the end is never before the start. (closes #7442)
Files: src/regexp_bt.c, src/regexp_nfa.c,
src/testdir/test_regexp_latin.vim

Patch 8.2.2122

Problem: Vim9: crash when sourcing vim9script early.
Solution: Use set_option_value() instead of setting p_cpo directly.
(closes #7441)
Files: src/scriptfile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2123

Problem: After using a complete popup the buffer is listed. (Boris Staletic)
Solution: Make the buffer unlisted.
Files: src/popupmenu.c, src/testdir/test_popupwin.vim

Patch 8.2.2124

Problem: Vim9: a range cannot be computed at runtime.
Solution: Add the ISN_RANGE instruction.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2125 (after 8.2.2122)

Problem: Vim9: leaking memory.
Solution: Free the saved 'cpo' value.
Files: src/scriptfile.c

Patch 8.2.2126

Problem: Ruby: missing function prototype.
Solution: Add the prototype.
Files: src/if_ruby.c

Patch 8.2.2127

Problem: Vim9: executing user command defined in Vim9 script not tested.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2128

Problem: There is no way to do something on **CTRL-Z**.
Solution: Add VimSuspend and VimResume autocommand events. (closes #7450)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/ex_docmd.c,
src/normal.c, src/testdir/test_suspend.vim, src/vim.h

Patch 8.2.2129

Problem: MS-Windows: Checking if a file name is absolute is slow.
Solution: Do not use mch_FullName(). (closes #7033)
Files: src/os_mswin.c

Patch 8.2.2130

Problem: Insert mode completion messages end up in message history.
Solution: Set msg_hist_off. (closes #7452)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.2131

Problem: Vim9: crash when lambda uses same var as assignment.
Solution: Do not let lookup_local change lv_from_outer, make a copy. (closes #7461)
Files: src/vim9compile.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/evalvars.c, src/proto/evalvars.pro,
src/testdir/test_vim9_func.vim

Patch 8.2.2132

Problem: Padding not drawn properly for popup window with title.
Solution: Draw the padding below the title. (closes #7460)

Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_longtitle_3.dump,
src/testdir/dumps/Test_popupwin_longtitle_4.dump

Patch 8.2.2133

Problem: Vim9: checking for a non-empty string is too strict.
Solution: Check for any string. (closes #7447)
Files: src/typval.c, src/proto/typval.pro, src/errors.h, src/filepath.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2134

Problem: Vim9: get E1099 when autocmd triggered in builtin function.
Solution: Check that did_emsg increased instead of checking that it changed.
(closes #7448)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2135

Problem: Vim9: #{ still seen as start of dict in some places.
Solution: Remove check for { after #. (closes #7456)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2136

Problem: Vim9: Using uninitialized variable.
Solution: Initialize "len" to zero. Clean up fnamemodify().
Files: src/filepath.c

Patch 8.2.2137

Problem: Vim9: :echo and :execute give error for empty argument.
Solution: Ignore an empty argument. (closes #7468)
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2138

Problem: Vim9: "exit_cb" causes Vim to exit.
Solution: Require white space after a command in Vim9 script. (closes #7467)
Also fix that Vim9 style heredoc was not always recognized.
Files: src/ex_cmds.h, src/ex_docmd.c, src/errors.h, src/userfunc.c,
src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_let.vim

Patch 8.2.2139

Problem: Vim9: unreachable code in assignment.
Solution: Don't check "new_local" when "has_index" is set. Add test for
wrong type of list index.
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2140

Problem: Build failure with tiny features.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.2141

Problem: A user command with try/catch may not catch an expression error.
Solution: When an expression fails check for following "|". (closes #7469)

Files: src/eval.c, src/testdir/test_trycatch.vim,
src/testdir/test_vimscript.vim

Patch 8.2.2142

Problem: Memory leak when heredoc is not terminated.
Solution: Free heredoc_trimmed.
Files: src/userfunc.c

Patch 8.2.2143

Problem: Vim9: dead code in compiling :unlet.
Solution: Don't check for "!" a second time.
Files: src/vim9compile.c

Patch 8.2.2144

Problem: Vim9: some corner cases not tested.
Solution: Add a few tests.
Files: src/testdir/test_vim9_script.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.2145

Problem: Vim9: concatenating lists does not adjust type of result.
Solution: When list member types differ use "any" member type.
(closes #7473)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2146

Problem: Vim9: automatic conversion of number to string for dict key.
Solution: Do not convert number to string. (closes #7474)
Files: src/dict.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2147

Problem: Quickfix window title not updated in all tab pages.
Solution: Update the quickfix window title in all tab pages. (Yegappan
Lakshmanan, closes #7481, closes #7466)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.2148

Problem: Vim9: crash when user command doesn't match.
Solution: Adjust command index. (closes #7479)
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2149

Problem: Popupwin test for latin1 sometimes fails.
Solution: Wait for the script to finish.
Files: src/testdir/test_popupwin.vim

Patch 8.2.2150

Problem: Github actions CI isn't used for all available platforms.
Solution: Update the github workflows. (Ozaki Kiichi, closes #7433)
Files: .coveralls.yml, .github/workflows/ci-windows.yaml,
.github/workflows/ci.yml, .travis.yml, README.md,
ci/build-snd-dummy.sh, ci/setup-xvfb.sh

Patch 8.2.2151

Problem: \$dir not expanded when configure checks for moonjit.

Solution: Use double quotes instead of single quotes. (closes #7478)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2152

Problem: screenpos() does not include the WinBar offset.
Solution: Use W_WINROW() instead of directly using w_window. (closes #7487)
Files: src/move.c, src/testdir/test_cursor_func.vim

Patch 8.2.2153

Problem: Popupwin test for latin1 still fails sometimes.
Solution: Wait for the "cat" command to finish.
Files: src/testdir/test_popupwin.vim

Patch 8.2.2154

Problem: Popupwin test for terminal buffer fails sometimes.
Solution: Wait for the prompt to appear.
Files: src/testdir/test_popupwin.vim

Patch 8.2.2155

Problem: Warning from Github actions for code analysis.
Solution: Remove the "git checkout HEAD^2" block.
Files: .github/workflows/codeql-analysis.yml

Patch 8.2.2156

Problem: Github actions run on pushing a tag.
Solution: Don't run CI on tag push. Omit coveralls on pull-request.
(Ozaki Kiichi, closes #7489)
Files: .github/workflows/ci.yml, .github/workflows/codeql-analysis.yml

Patch 8.2.2157

Problem: Vim9: can delete a Vim9 script variable from a function.
Solution: Check the variable is defined in Vim9 script. (closes #7483)
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2158

Problem: CI on cirrus times out, coveralls doesn't always run.
Solution: Set timeout to 20 minutes. Adjust condition. (closes #7493)
Files: .cirrus.yml, .github/workflows/ci.yml

Patch 8.2.2159

Problem: Vim9: when declaring a list it is not allocated yet, causing a following extend() to fail.
Solution: When fetching a variable value for a list or dict that is null allocate the list or dict, so it can be used. (closes #7491)
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2160

Problem: Various typos.
Solution: Fix spelling mistakes. (closes #7494)
Files: src/bufwrite.c, src/cindent.c, src/cmdexpand.c, src/eval.c, src/ex_cmds.c, src/feature.h, src/getchar.c, src/gui_haiku.cc, src/gui_xmdlg.c, src/help.c, src/if_ole.cpp, src/insexpand.c, src/list.c, src/map.c, src/memline.c, src/normal.c, src/os_win32.c, src/search.c, src/term.c,

src/testdir/test_arglist.vim, src/testdir/test_autocmd.vim,
src/testdir/test_debugger.vim, src/testdir/test_increment.vim,
src/testdir/test_menu.vim, src/testdir/test_netbeans.vim,
src/testdir/test_popupwin.vim, src/testdir/test_python2.vim,
src/testdir/test_python3.vim, src/testdir/test_sort.vim,
src/testdir/test_terminal2.vim, src/testdir/test_terminal3.vim,
src/testdir/test_vartabs.vim, src/testdir/test_vimscript.vim,
src/textprop.c, src/userfunc.c, src/vim9.h, src/vim9compile.c,
src/vim9execute.c

Patch 8.2.2161

Problem: Arguments -T and -x not tested yet.
Solution: Add a test. (Dominique Pellé, closes #7490)
Files: src/testdir/test_startup.vim

Patch 8.2.2162

Problem: Vim9: Cannot load or store autoload variables.
Solution: Add ISN_LOADAUTO and ISN_STOREAUTO. (closes #7485)
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/dict.c,
src/eval.c, src/evalvars.c, src/proto/evalvars.pro,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2163

Problem: Crash when discarded exception is the current exception.
Solution: Compare the exception with current_exception. (closes #7499)
Files: src/ex_eval.c

Patch 8.2.2164

Problem: Vim9: autoload function doesn't work in script that starts with
an upper case letter.
Solution: Check for the autoload character. (closes #7502)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.2165

Problem: Vim9: assignment to dict member does not work.
Solution: Fix recognizing dict member. (closes #7484)
Files: src/ex_docmd.c, src/eval.c, src/evalvars.c, src/vim.h

Patch 8.2.2166

Problem: Auto format doesn't work when deleting text.
Solution: Make "x" trigger auto format. (closes #7504)
Files: src/ops.c, src/testdir/test_textformat.vim

Patch 8.2.2167

Problem: Vim9: assign test fails. (Elimar Riesebieter)
Solution: Adjust the test for dict assignment.
Files: src/testdir/test_vim9_assign.vim

Patch 8.2.2168

Problem: Vim9: error for assigning to dict of dict.
Solution: Remember the destination type. (closes #7506)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2169

Problem: Vim9: test leaves file behind.
Solution: Rename script files. (Dominique Pellé, closes #7511)
Use try/finally.
Files: src/testdir/test_vim9_script.vim, src/testdir/vim9.vim

Patch 8.2.2170

Problem: Vim9: a global function defined in a :def function fails if it uses the context.
Solution: Create a partial to store the closure context. (see #7410)
Files: src/userfunc.c, src/proto/userfunc.pro, src/vim9execute.c, src/structs.h, src/testdir/test_vim9_func.vim

Patch 8.2.2171

Problem: Valgrind warning for using uninitialized value.
Solution: Do not use "startp" or "endp" unless there is a match.
Files: src/regexp_nfa.c

Patch 8.2.2172

Problem: Vim9: number of arguments is not always checked. (Yegappan Lakshmanan)
Solution: Check number of arguments when calling function by name.
Files: src/userfunc.c, src/proto/userfunc.pro, src/vim9execute.c, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.2173

Problem: Vim9: get internal error when assigning to undefined variable.
Solution: Add error message. (closes #7475)
Files: src/vim9compile.c, src/vim9execute.c, src/errors.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.2174

Problem: Mac version doesn't specify the CPU architecture.
Solution: Add "arm64" or "x86_64". (Yee Cheng Chin, closes #7519)
Files: src/version.c

Patch 8.2.2175

Problem: Github actions: clang-11 handling suboptimal.
Solution: Separate step of installing clang-11. Get ubuntu release name dynamically. (Ozaki Kiichi, closes #7514)
Files: .github/workflows/ci.yml

Patch 8.2.2176

Problem: Crash with a sequence of fold commands.
Solution: Bail out when there are no folds at all. Add a test (Dominique Pellé) (closes #7515)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.2177

Problem: Pattern "^" does not match if the first character in the line is combining. (Rene Kita)
Solution: Do accept a match at the start of the line. (closes #6963)
Files: src/regexp_nfa.c, src/testdir/test_regexp_utf8.vim

Patch 8.2.2178

Problem: Python 3: non-utf8 character cannot be handled.
Solution: Change the string decode. (Björn Linse, closes #1053)
Files: src/if_py_both.h, src/if_python.c, src/if_python3.c,
src/testdir/test_python3.vim, src/testdir/test_python2.vim

Patch 8.2.2179

Problem: Vim9: crash when indexing a dict with a number.
Solution: Add ISN_STOREINDEX. (closes #7513)
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h,
src/errors.h, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2180

Problem: Vim9: test for error after error is flaky.
Solution: Wait for job to finish instead of a fixed delay.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2181

Problem: Valgrind warnings for using uninitialized value.
Solution: Do not use "start" or "end" unless there is a match.
Files: src/regexp_nfa.c, src/regexp_bt.c

Patch 8.2.2182

Problem: Vim9: value of **'magic'** is still relevant.
Solution: Always behave like **'magic'** is on in Vim9 script (closes #7509)
Files: src/option.c, src/proto/option.pro, src/arglist.c, src/buffer.c,
src/cmdexpand.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c,
src/insexpand.c, src/normal.c, src/search.c, src/tag.c,
src/structs.h, src/globals.h, src/ex_cmds.h,
src/testdir/test_vim9_cmd.vim

Patch 8.2.2183

Problem: Vim9: value of **'edcompatible'** and **'gdefault'** are used.
Solution: Ignore these deprecated options in Vim9 script. (closes #7508)
Files: src/ex_cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2184

Problem: Vim9: no error when using "2" for a line number.
Solution: Give an error message if the line number is invalid. (closes #7492)
Files: src/typval.c, src/evalfunc.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_cursor_func.vim

Patch 8.2.2185

Problem: BufUnload is not triggered for the quickfix dummy buffer.
Solution: Do trigger BufUnload. (Pontus Leitzler, closes #7518, closes #7517)
Fix white space around "=".
Files: src/quickfix.c, src/testdir/test_autocmd.vim

Patch 8.2.2186

Problem: Vim9: error when using **'opfunc'**.
Solution: Do not expect a return value from **'opfunc'**. (closes #7510)
Files: src/eval.c, src/proto/eval.pro, src/ops.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2187

Problem: Python 3 test fails sometimes. (Christian Brabandt)
Solution: Accept two SystemError messages.
Files: src/testdir/test_python3.vim

Patch 8.2.2188

Problem: Vim9: crash when calling global function from :def function.
Solution: Set the outer context. Define the partial for the context on the original function. Use a refcount to keep track of which ufunc is using a dfunc. (closes #7525)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9execute.c, src/proto/vim9execute.pro, src/userfunc.c, src/proto/userfunc.pro, src/structs.h, src/vim9.h, src/testdir/test_vim9_func.vim

Patch 8.2.2189

Problem: Cannot repeat a command that uses the small delete register.
Solution: Store the register name instead of the contents. (Christian Brabandt, closes #7527)
Files: src/ops.c, src/register.c, src/testdir/test_registers.vim

Patch 8.2.2190

Problem: Vim9: crash when compiled with EXITFREE.
Solution: Check that df_ufunc is not NULL.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2191

Problem: Vim9: using wrong name with lambda in nested function.
Solution: Copy the lambda name earlier. (closes #7525)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2192

Problem: Codecov on github actions fails.
Solution: Revert to codecov script. (Ozaki Kiichi, closes #7529)
Files: Filelist, .github/workflows/ci.yml

Patch 8.2.2193

Problem: Vim9: can change constant in :def function.
Solution: Check if a variable is locked. (issue #7526)
Files: src/evalvars.c, src/proto/evalvars.pro, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2194

Problem: Vim9: cannot use :const or :final at the script level.
Solution: Support using :const and :final. (closes #7526)
Files: src/vim.h, src/evalvars.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2195

Problem: Failing tests for :const.
Solution: Add missing check for ASSIGN_FINAL.
Files: src/eval.c, src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.2196

Problem: :version output has extra spaces in compile and link command.
Solution: Adjust QUOTESED. (closes #7505)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2197

Problem: Assert arguments order reversed.
Solution: Swap the arguments. (Christian Brabandt, closes #7531)
Files: src/testdir/test_registers.vim

Patch 8.2.2198

Problem: ml_get error when resizing window and using text property.
Solution: Validate botline of the right window. (closes #7528)
Files: src/move.c, src/proto/move.pro, src/textprop.c,
src/testdir/test_textprop.vim

Patch 8.2.2199

Problem: First write after setting 'eol' does not have NL added. (Tomáš Janoušek)
Solution: Only use b_no_eol_lnum when doing a binary write. (closes #7535)
Files: src/bufwrite.c, src/testdir/test_writefile.vim

Patch 8.2.2200

Problem: Vim9: lambda without white space around -> is confusing.
Solution: Require white space in a :def function. (issue #7503)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2201

Problem: Write file test fails on MS-Windows.
Solution: Force edit after setting 'fileformat'.
Files: src/testdir/test_writefile.vim

Patch 8.2.2202

Problem: Write file test still fails on MS-Windows.
Solution: Set fileformat with the :edit command
Files: src/testdir/test_writefile.vim

Patch 8.2.2203

Problem: Moodle gift files are not recognized.
Solution: Add a filetype pattern. (Delim Temizer)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2204

Problem: Vim9: using -> both for method and lambda is confusing.
Solution: Use => for lambda in :def function.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/userfunc.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.2205

Problem: Vim9: memory leak when parsing lambda fails.
Solution: Clear growarrays.
Files: src/userfunc.c

Patch 8.2.2206

Problem: :exe command line completion only works for first argument.
Solution: Skip over text if more is following. (closes #7546)
Files: src/eval.c, src/testdir/test_cmdline.vim

Patch 8.2.2207

Problem: Illegal memory access if popup menu items are changed while the menu is visible. (Tomáš Janoušek)
Solution: Make a copy of the text. (closes #7537)
Files: src/popupmenu.c, src/testdir/test_popup.vim, src/testdir/dumps/Test_popup_command_04.dump, src/testdir/dumps/Test_popup_command_05.dump

Patch 8.2.2208

Problem: Vim9: after reloading a script variable index may be invalid.
Solution: When the sequence number doesn't match give an error for using a script-local variable from a compiled function. (closes #7547)
Files: src/vim9.h, src/structs.h, src/errors.h, src/vim9compile.c, src/vim9execute.c, src/scriptfile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2209

Problem: Vim9: return type of => lambda not parsed.
Solution: Parse and use the return type.
Files: src/vim9compile.c, src/userfunc.c, src/vim9type.c, src/proto/vim9type.pro, src/vim9script.c, src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2210

Problem: Vim9: allocating a type to set TTFLAG_BOOL_OK.
Solution: Add t_number_bool.
Files: src/globals.h, src/vim9type.c, src/vim9compile.c

Patch 8.2.2211

Problem: MS-Windows: can't load Python dll if not in the path.
Solution: Use the InstallPath registry entry. (Kelvin Lee, closes #7540)
Files: src/if_python3.c

Patch 8.2.2212

Problem: Vim9: lambda with => does not work at the script level.
Solution: Make it work.
Files: src/eval.c, src/vim9type.c, src/userfunc.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.2213

Problem: Checking white space around -> is not backwards compatible.
Solution: Only check white space around =>.
Files: src/userfunc.c

Patch 8.2.2214

Problem: ":e#" does not give a warning for missing white space.
Solution: Adjust the check for white space. (closes #7545)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2215

Problem: Vim9: `=expr` not recognized in global command.
Solution: Skip over pattern. (issue #7541)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2216

Problem: Vim9: range with missing colon can be hard to spot.
Solution: Include the start of the range in the error. (closes #7543)
Files: src/errors.h, src/ex_docmd.c, src/vim9compile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.2217

Problem: Vim9: command modifiers not restored in catch block.
Solution: Restore command modifiers. (closes #7542)
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2218

Problem: Vim9: failure if passing more arguments to a lambda than expected.
Solution: Only put expected arguments on the stack. (closes #7548)
Files: src/vim9execute.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2219

Problem: Vim9: method call with expression not supported.
Solution: Implement expr->(expr)().
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2220

Problem: Vim9: memory leak when parsing nested parenthesis.
Solution: Clear newargs.
Files: src/userfunc.c

Patch 8.2.2221

Problem: If [<Down>](#) is mapped on the command line 'wildchar' is inserted.
Solution: Set KeyTyped when using 'wildchar'. (closes #7552)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.2222

Problem: Vim9: cannot keep script variables when reloading.
Solution: Add the "noclear" argument to :vim9script.
Files: runtime/doc/vim9.txt, src/structs.h, src/scriptfile.c,
src/vim9script.c, src/ex_cmds.h, src/ex_docmd.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2223

Problem: Vim9: Reloading marks a :def function as deleted.
Solution: Clear the function contents but keep the index.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/userfunc.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2224

Problem: Vim9: crash if script reloaded with different variable type.
Solution: Check the type when accessing the variable.
Files: src/vim9execute.c, src/vim9compile.c, src/vim9.h, src/vim9type.c,
src/proto/vim9type.pro, src/errors.h, src/evalvars.c,
src/vim9script.c, src/proto/vim9script.pro,

src/testdir/test_vim9_script.vim

Patch 8.2.2225

Problem: Vim9: error when using :import in legacy script twice.
Solution: Make it possible to redefine an import when reloading.
Files: src/vim9script.c, src/proto/vim9script.pro, src/structs.h,
src/evalvars.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2226

Problem: Vim9: script test fails.
Solution: Add missing change.
Files: src/scriptfile.c

Patch 8.2.2227

Problem: Vim9: recognizing lambda is too complicated.
Solution: Call compile_lambda() and check for NOTDONE.
Files: src/vim9compile.c, src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2228

Problem: Vim9: cannot use ":e #" because # starts a comment.
Solution: Support using %% instead of #.
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2229

Problem: build failure without the +eval feature.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.2230

Problem: Vim9: insert completion runs into error.
Solution: Insert colon before range. (closes #7556)
Files: src/insexpand.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2231

Problem: When "--remote file" is used "file" is not reloaded.
Solution: When a :drop command is used for a file that is already displayed in a window and it has not been changed, check if it needs to be reloaded. (closes #7560)
Files: src/ex_cmds.c, src/testdir/test_clientserver.vim

Patch 8.2.2232

Problem: Compiler error for falling through into next case.
Solution: Move FALLTHROUGH below the #endif
Files: src/ex_docmd.c

Patch 8.2.2233

Problem: Cannot convert a byte index into a character index.
Solution: Add charidx(). (Yegappan Lakshmanan, closes #7561)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/testdir/test_functions.vim

Patch 8.2.2234

Problem: Command line wildmenu test often fails with Unix GUI.

Solution: Skip the test where it is expected to fail.
Files: src/testdir/test_cmdline.vim

Patch 8.2.2235

Problem: Build failure with some Ruby versions.
Solution: Adjust the code for Ruby 3.0. (Ozaki Kiichi, closes #7564)
Files: ci/config.mk.clang.sed, src/if_ruby.c

Patch 8.2.2236

Problem: **'scroll'** option can change when setting the statusline or tabline but the option context is not updated.
Solution: Update the script context when the scroll option is changed as a side effect. (Christian Brabandt, closes #7533)
Files: runtime/doc/options.txt, src/scriptfile.c, src/testdir/test_options.vim, src/vim.h, src/window.c

Patch 8.2.2237

Problem: CI on Mac fails in sed command.
Solution: Set LC_ALL to "C". (Ozaki Kiichi, closes #7565)
Files: .github/workflows/ci.yml

Patch 8.2.2238

Problem: Vim9: cannot load a Vim9 script without the +eval feature.
Solution: Support Vim9 script syntax without the +eval feature.
Files: src/ex_docmd.c, src/vim9script.c, src/globals.h, src/main.c, src/autocmd.c, src/buffer.c, src/structs.h, src/menu.c, src/scriptfile.c, src/usercmd.c, src/proto.h, src/errors.h

Patch 8.2.2239

Problem: Vim9: concatenating lines with backslash is inconvenient.
Solution: Support concatenating lines starting with '|', useful for :autocmd, :command, etc. (closes #6702)
Files: runtime/doc/vim9.txt, src/scriptfile.c, src/vim9script.c, src/proto/vim9script.pro, src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/structs.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.2240

Problem: Clientserver test fails if full path is used.
Solution: Ignore the path preceding the file name.
Files: src/testdir/test_clientserver.vim

Patch 8.2.2241

Problem: Build with Ruby and clang may fail.
Solution: Adjust configure and sed script. (Ozaki Kiichi, closes #7566)
Files: ci/config.mk.clang.sed, src/auto/configure, src/configure.ac

Patch 8.2.2242

Problem: Vim9: line continuation with bar does not work at script level.
Solution: Check for Vim9 script.
Files: src/structs.h, src/ex_docmd.c, src/userfunc.c, src/scriptfile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2243

Problem: Crash when popup mask contains zeroes.
Solution: Check boundaries properly. (closes #7569)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.2244

Problem: Crash when making the window width of the not-current window negative.
Solution: Make sure the window width is not negative. (closes #7568)
Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.2.2245

Problem: Vim9: return value of winrestcmd() cannot be executed.
Solution: Put colons before each range. (closes #7571)
Files: src/evalwindow.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2246

Problem: Cursor keys not recognized at the hit-Enter prompt after executing an external command.
Solution: Change the codes for the extra cursor keys. (closes #7562)
Tune the delays to avoid test flakiness.
Files: runtime/doc/term.txt, src/term.c, src/testdir/test_terminal3.vim

Patch 8.2.2247

Problem: VMS: various smaller problems.
Solution: Fix VMS building and other problems. (Zoltan Arpadffy)
Files: src/term.c, src/gui_gtk_vms.h, src/os_vms_conf.h, src/gui_x11.c, src/Make_vms.mms, src/macros.h, src/gui.h, src/os_unix.h

Patch 8.2.2248

Problem: ASAN error on exit with GUI.
Solution: Check the window still has lines. (Christian Brabandt, closes #7573)
Files: src/term.c

Patch 8.2.2249

Problem: Termcodes test is flaky when used over ssh with X forwarding.
Solution: Set **'mousetime'** to a larger value. (Dominique Pellé, closes #7576, closes #7563)
Files: src/testdir/test_termcodes.vim

Patch 8.2.2250

Problem: Vim9: sublist is ambiguous.
Solution: Require white space around the colon. (closes #7409)
Files: src/vim9compile.c, src/eval.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2251

Problem: Test failures in legacy script.
Solution: Check for Vim9 script.
Files: src/eval.c

Patch 8.2.2252

Problem: Vim9: crash when using lambda without return type in dict.
Solution: Without a return type use t_unknown. (closes #7587)

Files: src/vim9type.c, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2253

Problem: Vim9: expr test fails.
Solution: Add missing assignment.
Files: src/userfunc.c

Patch 8.2.2254

Problem: Vim9: bool option type is number.
Solution: Have get_option_value() return a different value for bool and number options. (closes #7583)
Files: src/option.h, src/option.c, src/proto/option.pro, src/evalvars.c, src/if_mzsch.c, src/if_ruby.c, src/spell.c, src/typval.c, src/vim9compile.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.2255 (after 8.2.2254)

Problem: Tcl test fails.
Solution: Change option handling.
Files: src/if_tcl.c

Patch 8.2.2256

Problem: Vim9: cannot use function(after line break in :def function.
Solution: Check for "(" after "function". (closes #7581)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2257

Problem: Vim9: using -> for lambda is ambiguous.
Solution: Stop supporting ->, must use =>.
Files: src/eval.c, src/vim9compile.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.2258

Problem: Not all OCaml related files are detected.
Solution: Update OCaml file type detection. (Markus Mottl, closes #7590)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2259

Problem: Test_Executable() fails when using chroot.
Solution: Ignore the difference between "sbin" and "bin".
Files: src/testdir/test_functions.vim

Patch 8.2.2260

Problem: Window resize test fails in very wide terminal.
Solution: Resize using the '**columns**' option. (Vladimir Lomov, closes #7592)
Files: src/testdir/test_window_cmd.vim

Patch 8.2.2261

Problem: Vim9: boolean option gets string type.
Solution: Check for VAR_BOOL. (closes #7588)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2262

Problem: Vim9: converting bool to string prefixes v:.
Solution: Do not use the v: prefix.
Files: src/evalvars.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2263

Problem: Vim9: compilation error with try-catch in skipped block.
Solution: Do not bail out when generate_instr() returns NULL. (closes #7584)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2264

Problem: Vim9: no error for mismatched :endfunc or :enddef.
Solution: Check for the mismatch. (issue #7582)
Files: src/errors.h, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2265

Problem: Error message for missing endfunc/enddef is last line.
Solution: Report the line where the function starts. (closes #7582)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2266

Problem: Vim9: it can be hard to see where white space is missing.
Solution: Mention the text where the error was seen. (closes #7580)
Files: src/errors.h, src/eval.c, src/evalvars.c, src/userfunc.c,
src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2267

Problem: Vim9: cannot use unlet for a dict member.
Solution: Pass GLV_NO_DECL to get_lval(). (closes #7585)
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2268

Problem: Vim9: list unpack seen as declaration.
Solution: Check for "var". (closes #7594)
Files: src/vim9compile.c, src/evalvars.c, src/eval.c, src/vim.h,
src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2269

Problem: Not all :hardcopy code covered by tests.
Solution: Test more combinations. (Dominique Pellé, closes #7595)
Files: src/testdir/test_hardcopy.vim

Patch 8.2.2270

Problem: Warning for size_t to int conversion. (Randall W. Morris)
Solution: Add a type cast.
Files: src/vim9execute.c

Patch 8.2.2271

Problem: ml_get error when changing hidden buffer in Python.
Solution: Block updating folds. (closes #7598)
Files: src/evalbuffer.c, src/testdir/test_python3.vim

Patch 8.2.2272

Problem: Vim9: extend() can violate the type of a variable.
Solution: Add the type to the dictionary or list and check items against it.
(closes #7593)
Files: src/structs.h, src/evalvars.c, src/dict.c, src/list.c,
src/vim9script.c, src/proto/vim9script.pro, src/vim9compile.c,
src/vim9execute.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2273

Problem: Build failure.
Solution: Add missing changes to header file.
Files: src/vim9.h

Patch 8.2.2274

Problem: badge for Travis is outdated.
Solution: Update badge for move from travis-ci.org to travis-ci.com.
Files: README.md

Patch 8.2.2275

Problem: **CTRL-C** not recognized in Mintty.
Solution: Recognize the modifyOtherKeys code ending in "u". (Christian Brabandt, closes #7575)
Files: src/ui.c

Patch 8.2.2276

Problem: List of distributed files is outdated.
Solution: Update the file list. Minor comment updates.
Files: Filelist, src/clipboard.c, src/fileio.c, src/option.c,
src/screen.c, src/testdir/test_signals.vim,
src/testdir/Make_vms.mms

Patch 8.2.2277

Problem: Missing backslash.
Solution: Add backslash.
Files: Filelist

Patch 8.2.2278

Problem: Falling back to old regexp engine can some patterns.
Solution: Do not fall back once [[:lower:]] or [[:upper:]] is used.
(Christian Brabandt, closes #7572)
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test_regexp_utf8.vim

Patch 8.2.2279

Problem: Vim9: memory leak with catch in skipped block.
Solution: Free the pattern if not used.
Files: src/vim9compile.c

Patch 8.2.2280

Problem: Fuzzy matching doesn't give access to the scores.
Solution: Return the scores with a third list. (Yegappan Lakshmanan, closes #7596)
Files: runtime/doc/eval.txt, src/search.c,
src/testdir/test_matchfuzzy.vim

Patch 8.2.2281

Problem: Vim9: compiled "wincmd" cannot be followed by bar.
Solution: Check for bar after "wincmd". (closes #7599)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2282

Problem: Length check mismatch with argument of strncmp(). (Christian Brabandt)
Solution: Adjust length check.
Files: src/ui.c

Patch 8.2.2283

Problem: Vim9: crash when lambda has fewer arguments than expected.
Solution: Don't check arguments when already failed. (closes #7606)
Files: src/vim9type.c, src/testdir/test_vim9_func.vim

Patch 8.2.2284

Problem: Vim9: cannot set an option to a boolean value.
Solution: Check for VAR_BOOL. (closes #7603)
Files: src/evalvars.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2285

Problem: Vim9: cannot set an option to a false.
Solution: For VAR_BOOL use string "0". (closes #7603)
Files: src/evalvars.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2286

Problem: Sort test fails when locale is Canadian English. (Neil H Watson)
Solution: Expect a different sort order. (closes #7609)
Files: src/testdir/test_sort.vim

Patch 8.2.2287

Problem: Sort test fails when locale is French Canadian.
Solution: Expect a different sort order. (Dominique Pellé, closes #7609)
Files: src/testdir/test_sort.vim

Patch 8.2.2288

Problem: Vim9: line break and comment not always skipped.
Solution: Skip over white space and then line break more consistently. (closes #7610)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2289

Problem: Vim9: **'cpo'** can become empty.
Solution: Use empty_option instead of an empty string. Update quickfix buffer after restoring **'cpo'**. (closes #7608)
Files: src/evalfunc.c, src/syntax.c, src/eval.c, src/quickfix.c, src/evalbuffer.c, src/ex_eval.c, src/gui_motif.c, src/map.c, src/testdir/test_quickfix.vim

Patch 8.2.2290

Problem: Vim9: unlet of global variable cannot be compiled.
Solution: Skip over variables that might be defined later. Give an error if

a subscript is found. (closes #7585)
Files: src/eval.c, src/vim9compile.c, src/vim.h,
src/testdir/test_vim9_assign.vim

Patch 8.2.2291

Problem: Vim9: cannot use "null" for v:null.
Solution: Support "null" like "true" and "false". (closes #7495)
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/evalvars.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.2292

Problem: Vim: expr test fails.
Solution: Add missing part of "null" support.
Files: src/eval.c

Patch 8.2.2293

Problem: Build failure with Motif. (Tony Mechelynck)
Solution: Use empty_option instead of empty_options.
Files: src/gui_motif.c

Patch 8.2.2294

Problem: VMS: a few remaining problems.
Solution: Add VMS specific changes. Add Lua support. (Zoltan Arpadffy)
Files: src/fileio.c, src/os_vms_conf.h, src/Make_vms.mms, src/macros.h,
src/os_vms.c, src/vim9execute.c, src/gui_xmew.c, src/os_unix.h

Patch 8.2.2295

Problem: Incsearch does not detect empty pattern properly.
Solution: Return magic state when skipping over a pattern. (Christian
Brabandt, closes #7612, closes #6420)
Files: src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/globals.h,
src/option.c, src/tag.c, src/proto/regexp.pro, src/regexp.c,
src/search.c, src/structs.h, src/vim9compile.c,
src/testdir/dumps/Test_incsearch_sub_01.dump,
src/testdir/dumps/Test_incsearch_sub_02.dump,
src/testdir/test_search.vim

Patch 8.2.2296

Problem: Cannot use **CTRL-N** and **CTRL-P** in a popup menu.
Solution: Use **CTRL-N** like <Down> and **CTRL-P** like <Up>. (closes #7614)
Files: runtime/doc/popup.txt, src/popupwin.c,
src/testdir/test_popupwin.vim

Patch 8.2.2297

Problem: Vim9: cannot set 'number' to a boolean value.
Solution: Use tv_get_bool(). (closes #7615)
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2298

Problem: Vim9: comment right after "(" of function not recognized.
Solution: Do not skip over white space before calling get_function_args().
(closes #7613)
Files: src/userfunc.c, src/proto/userfunc.pro,
src/testdir/test_vim9_func.vim

Patch 8.2.2299

Problem: Vim9: invalid memory access making error message flaky.
Solution: Do not check cmd_argt for CMD_USER. (issue #7467)
Files: src/ex_docmd.c, src/vim9execute.c, src/errors.h,
src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2300

Problem: Vim9: wrong order on type stack when using dict.
Solution: Generate constants before a dict. (closes #7619)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2301

Problem: Vim9: cannot unlet a dict or list item.
Solution: Add ISN_UNLETINDEX. Refactor assignment code to use for unlet.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.2302

Problem: Vim9: using an option value may use uninitialized memory.
Solution: Clear v_lock. (closes #7620)
Files: src/typval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2303

Problem: Vim9: backtick expansion doesn't work for :foldopen.
Solution: Do recognize backtick expansion. (closes #7621)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2304

Problem: Vim9: no test for unletting an imported variable.
Solution: Add a test. Fix line number in error.
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2305

Problem: Vim9: "++var" and "--var" are silently accepted.
Solution: Give an error message.
Files: src/vim9compile.c, src/eval.c, src/proto/eval.pro,
src/testdir/test_vim9_expr.vim

Patch 8.2.2306

Problem: Vim9: when using function reference type is not checked.
Solution: When using a function reference lookup the type and check the
argument types. (issue #7629)
Files: src/userfunc.c, src/proto/userfunc.pro, src/eval.c, src/structs.h,
src/vim9type.c, src/proto/vim9type.pro, src/vim9compile.c,
src/vim9execute.c, src/evalvars.c, src/evalfunc.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2307

Problem: A shell command in the vimrc causes terminal output.
Solution: Do not call starttermcap() after a shell command if the termcap
wasn't active before.
Files: src/ex_cmds.c

Patch 8.2.2308

Problem: Vim9: no error when assigning lambda to funcref without return value.

Solution: Default return value to "any". (closes #7629)

Files: src/userfunc.c, src/vim9compile.c,
src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2309

Problem: 0o777 not recognized as octal.

Solution: Use vim_isodigit(). (Ken Takata, closes #7633, closes #7631)

Files: src/charset.c, src/testdir/test_eval_stuff.vim

Patch 8.2.2310

Problem: Vim9: winsaveview() return type is too generic.

Solution: use dict<number> instead of dict<any>. (closes #7626)

Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2311

Problem: Vim9: cannot assign to a variable that shadows a command modifier.

Solution: Check for assignment after possible command modifier.
(closes #7632)

Files: src/vim9compile.c, src/ex_docmd.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.2312

Problem: Build failure with Ruby 3.0 and 32 bits.

Solution: Add #ifdef. (closes #7638)

Files: src/if_ruby.c

Patch 8.2.2313

Problem: Vim9: using uninitialized field when parsing range. ":silent!" not respected when parsing range fails.

Solution: Initialize ea.skip. On pattern failure handle it like an error.
(closes #7636)

Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2314

Problem: Vim9: returning zero takes two instructions.

Solution: Add ISN_RETURN_ZERO.

Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2315

Problem: Vim9: "endif" as dict key misinterpreted as function end.

Solution: Check for following colon. (closes #7640)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2316

Problem: Vim9: cannot list a lambda function.

Solution: Support the <lambda>9 notation, like :disassemble. (closes #7634)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2317

Problem: Vim9: command modifier before list unpack doesn't work.

Solution: Only recognize "[" directly after the name. (closes #7641)
Files: src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2318

Problem: Vim9: string and list index work differently.
Solution: Make string index work like list index. (closes #7643)
Files: src/eval.c, src/proto/eval.pro, src/vim9execute.c, src/list.c,
src/proto/vim9execute.pro, src/testdir/test_vim9_expr.vim

Patch 8.2.2319

Problem: "exptype_T" can be read as "expected type".
Solution: Rename to "exprtype_T", expression type.
Files: src/eval.c, src/typval.c, src/proto/typval.pro, src/vim9compile.c,
src/proto/vim9compile.pro, src/vim9execute.c, src/structs.h,
src/vim9.h

Patch 8.2.2320

Problem: Vim9: no error for comparing bool with string.
Solution: Check for wrong types when comparing. (closes #7639)
Files: src/typval.c, src/errors.h, src/testdir/test_vim9_expr.vim

Patch 8.2.2321

Problem: Vim9: cannot nest closures.
Solution: Add the nesting level to ISN_LOADOUTER and ISN_STOREOUTER.
(closes #7150, closes #7635)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/structs.h,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.2322

Problem: Vim9: closure nested limiting to one level.
Solution: Add outer_T. Also make STOREOUTER work.
Files: src/vim9execute.c, src/vim9.h, src/structs.h,
src/testdir/test_vim9_func.vim

Patch 8.2.2323

Problem: Vim9: error when inferring type from empty dict/list.
Solution: When the member is t_unknown use t_any. (closes #7009)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2324

Problem: Not easy to get mark en cursor position by character count.
Solution: Add functions that use character index. (Yegappan Lakshmanan,
closes #7648)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/eval.c,
src/evalfunc.c, src/proto/eval.pro, src/tag.c,
src/testdir/test_cursor_func.vim, src/typval.c

Patch 8.2.2325

Problem: Vim9: crash if map() changes the item type.
Solution: Check that the item type is still OK. (closes #7652)
Fix problem with mapnew() on range list.
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9compile.c,
src/list.c, src/testdir/test_vim9_builtin.vim,

src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2326

Problem: Build error with +eval feature but without +spell.
Solution: Adjust #ifdef. (John Marriott)
Files: src/mbyte.c

Patch 8.2.2327

Problem: Debugging code included.
Solution: Remove the debugging code.
Files: src/vim9execute.c

Patch 8.2.2328

Problem: Some test files may not be deleted.
Solution: Add a delete() call, correct name. (Dominique Pellé, closes #7654)
Files: src/testdir/test_clientserver.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2329

Problem: Not all ways Vim can be started are tested.
Solution: Add a test for different program names. (Dominique Pellé,
closes #7651)
Files: src/testdir/test_startup.vim

Patch 8.2.2330

Problem: Vim9: crash when using :throw in a not executed block.
Solution: Don't generate the instruction when skipping. (closes #7659)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2331

Problem: Vim9: wrong error when modifying dict declared with :final.
Solution: Do not check for writable variable when an index follows.
(closes #7657)
Files: src/vim9compile.c, src/structs.h, src/vim9script.c,
src/proto/vim9script.pro, src/evalvars.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.2332

Problem: Vim9: missing :endif not reported when using :windo.
Solution: Pass a getline function to do_cmdline(). (closes #7650)
Files: src/vim9execute.c, src/structs.h, src/scriptfile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.2333

Problem: Vim9: warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize "res".
Files: src/vim9execute.c

Patch 8.2.2334

Problem: Pascal-like filetypes not always detected.
Solution: Improved Puppet, InstantFPC and Pascal detection. (Doug Kearns,
closes #7662)
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim,
runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.2335

Problem: Vim9: "silent return" does not restore command modifiers.
Solution: Restore command modifiers before returning. (closes #7649)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2336

Problem: Vim9: it is not possible to extend a dictionary with different item types.
Solution: Add extendnew(). (closes #7666)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/list.c, src/proto/list.pro, src/testdir/test_listdict.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2337

Problem: Configure test for GTK only says "no". (Harm te Hennepe)
Solution: Hint that a -def package is needed. (closes #5229)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2338

Problem: Vim9: no error if using job_info() result wrongly.
Solution: Adjust return type on number of arguments. (closes #7667)
Files: src/evalfunc.c, src/globals.h, src/testdir/test_vim9_builtin.vim

Patch 8.2.2339

Problem: Cannot get the type of a value as a string.
Solution: Add typename().
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/vim9type.c, src/proto/vim9type.pro, src/testdir/test_vimscript.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2340

Problem: win_execute() unexpectedly returns number zero when failing.
Solution: Return an empty string. (closes #7665)
Files: src/evalwindow.c, src/testdir/test_vim9_builtin.vim, src/testdir/test_execute_func.vim

Patch 8.2.2341

Problem: Expression command line completion shows variables but not functions after "g:". (Gary Johnson)
Solution: Prefix "g:" when needed to a global function.
Files: src/evalfunc.c, src/evalvars.c, src/proto/evalvars.pro, src/testdir/test_cmdline.vim

Patch 8.2.2342

Problem: "char" functions return the wrong column in Insert mode when the cursor is beyond the end of the line.
Solution: Compute the column correctly. (Yegappan Lakshmanan, closes #7669)
Files: src/eval.c, src/evalfunc.c, src/testdir/test_cursor_func.vim

Patch 8.2.2343

Problem: Vim9: return type of readfile() is any.
Solution: Add readblob() so that readfile() can be expected to always return a list of strings. (closes #7671)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/filepath.c, src/proto/filepath.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2344

Problem: Using inclusive index for slice is not always desired.
Solution: Add the slice() method, which has an exclusive index. (closes
#7408)

Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/eval.c, src/proto/eval.pro, src/vim9execute.c,
src/proto/vim9execute.pro, src/list.c, src/proto/list.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2345

Problem: No focus events in a terminal.
Solution: Add the t_fd and t_fe termcap entries and implement detecting
focus events. (Hayaki Saito, Magnus Groß, closes #7673,
closes #609, closes #5526)

Files: runtime/doc/term.txt, src/optiondefs.h, src/term.c, src/term.h

Patch 8.2.2346

Problem: Codecov reports every little coverage drop.
Solution: Tolerate a 0.05% drop. Hide the appveyor config file. (Ozaki
Kiichi, closes #7678)

Files: .appveyor.yml, appveyor.yml, .codecov.yml

Patch 8.2.2347

Problem: Build failure without GUI.

Solution: Add #ifdef.

Files: src/term.c

Patch 8.2.2348 (after 8.2.2345)

Problem: No check for modified files after focus gained. (Mathias Stearn)

Solution: Call ui_focus_change().

Files: src/term.c, src/ui.c

Patch 8.2.2349

Problem: Vim9: cannot handle line break after parenthesis at line end.

Solution: Skip over line break. (closes #7677)

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2350

Problem: Using "void" for no reason.

Solution: Use "char *".

Files: src/ex_docmd.c

Patch 8.2.2351

Problem: Vim9: error message for "throw" in function that was called with
"silent!".

Solution: Do not throw the exception when not caught or displayed.
(closes #7672)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2352

Problem: If the focus lost/gained escape sequence is received twice it is not ignored. (Christ van Willegen)
Solution: Adjust the logic to ignore the escape code.
Files: src/term.c

Patch 8.2.2353

Problem: Sparql files are not detected.
Solution: Add the sparql filetype. (closes #7679)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2354

Problem: Crash with a weird combination of autocommands.
Solution: Increment b_nwindows when needed. (closes #7674)
Files: src/ex_cmds.c, src/buffer.c, src/proto/buffer.pro, src/testdir/test_autocmd.vim

Patch 8.2.2355

Problem: Stray test failure on Appveyor.
Solution: Finish insert command.
Files: src/testdir/test_autocmd.vim

Patch 8.2.2356

Problem: Vim9: ":put =expr" does not handle a list properly.
Solution: Use the same logic as eval_to_string_eap(). (closes #7684)
Files: src/vim9execute.c, src/eval.c, src/proto/eval.pro, src/testdir/test_vim9_cmd.vim

Patch 8.2.2357

Problem: Vim9: crash when parsing function return type fails.
Solution: Bail out and set return type to "unknown". (closes #7685)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2358

Problem: Wrong #ifdef for use_xterm_like_mouse().
Solution: Use FEAT_MOUSE_XTERM.
Files: src/term.c

Patch 8.2.2359

Problem: Strange test failure with MS-Windows.
Solution: Skip the system() call for now.
Files: src/testdir/test_autocmd.vim

Patch 8.2.2360

Problem: Test leaves file behind.
Solution: Delete the right file. (Dominique Pellé, closes #7689)
Files: src/testdir/test_filetype.vim

Patch 8.2.2361

Problem: Vim9: no highlight for "s//gc" when using 'opfunc'.
Solution: Reset 'lazyredraw' temporarily. (closes #7687)
Files: src/ex_cmds.c

Patch 8.2.2362

Problem: Vim9: check of builtin function argument type is incomplete.

Solution: Use need_type() instead of check_arg_type().
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/evalfunc.c,
src/proto/evalfunc.pro, src/vim9type.c, src/proto/vim9type.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2363

Problem: curpos() does not accept a string argument as before.
solution: Make a string argument work again. (Yegappan Lakshmanan,
closes #7690)
Files: src/evalfunc.c, src/testdir/test_cursor_func.vim

Patch 8.2.2364

Problem: Vim9: line break in lambda accesses freed memory.
Solution: Make a copy of the return type. (closes #7664)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2365

Problem: Vim9: no check for map() changing item type at script level.
Solution: Check the new value type.
Files: src/list.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.2366

Problem: When using ":sleep" the cursor is always displayed.
Solution: Do not display the cursor when using ":sleep!". (Jeremy Lerner,
closes #7688)
Files: runtime/doc/index.txt, runtime/doc/various.txt, src/ex_cmds.h,
src/ex_docmd.c, src/normal.c, src/proto/ex_docmd.pro, src/term.c,
src/testdir/Make_all.mak, src/testdir/test_sleep.vim

Patch 8.2.2367

Problem: Test failures on some less often used systems.
Solution: Adjust printf formats and types. (James McCoy, closes #7691)
Files: src/errors.h, src/evalfunc.c, src/list.c, src/vim9execute.c

Patch 8.2.2368

Problem: Insufficient tests for setting options.
Solution: Add a few tests. (Dominique Pellé, closes #7695)
Files: src/testdir/test_options.vim

Patch 8.2.2369

Problem: Vim9: functions return true/false but can't be used as bool.
Solution: Add ret_number_bool(). (closes #7693)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2370

Problem: Vim9: command fails in catch block.
Solution: Reset force_abort and need_rethrow. (closes #7692)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2371

Problem: Vim9: crash when using types in :for with unpack.
Solution: Check for skip_var_list() failing. Pass include_type to

Files: skip_var_one(). Skip type when compiling. (closes #7694)
src/vim9compile.c, src/evalvars.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2372 (after 8.2.2371)

Problem: Confusing error message for wrong :let command.
Solution: Only check for type in Vim9 script.
Files: src/evalvars.c

Patch 8.2.2373

Problem: Vim9: list assignment only accepts a number index.
Solution: Accept "any" and do a runtime type check. (closes #7694)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2374

Problem: Accessing uninitialized memory in test_undo.
Solution: Do not look in typebuf.tb_buf if it is empty. (Dominique Pellé,
closes #7697)
Files: src/edit.c

Patch 8.2.2375

Problem: Test for RGB color skipped in the terminal.
Solution: Run the GUI if possible.
Files: src/testdir/test_highlight.vim

Patch 8.2.2376

Problem: Vim9: crash when dividing by zero in compiled code using
constants.
Solution: Call num_divide() and num_modulus(). (closes #7704)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2377

Problem: Vim9: crash when using a range after another expression.
Solution: Set the variable type to number. Fix using :put with a range and
the "=" register. (closes #7706)
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2378

Problem: Vim9: no error message for dividing by zero.
Solution: Give an error message. (issue #7704)
Files: src/errors.h, src/eval.c, src/vim9execute.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.2379

Problem: Finding spell suggestions twice if '**spellsuggest**' contains number.
Solution: Only do internal suggestions once. (closes #7713)
Files: src/spellsuggest.c

Patch 8.2.2380

Problem: Vim9: occasional crash when using try/catch and a timer.
Solution: Save and restore "need_rethrow" when invoking a timer callback.
(closes #7708)
Files: src/time.c

Patch 8.2.2381

Problem: Vim9: divide by zero does not abort expression execution.
Solution: Use a "failed" flag. (issue #7704)
Files: src/eval.c, src/proto/eval.pro, src/evalvars.c, src/vim9compile.c,
src/testdir/vim9.vim, src/testdir/test_vim9_assign.vim

Patch 8.2.2382 (after 8.2.2381)

Problem: Build failure.
Solution: Add missing changes.
Files: src/vim9execute.c

Patch 8.2.2383

Problem: Focus escape sequences are not named in ":set termcap" output.
Solution: Add the names to the list. (closes #7718)
Files: src/misc2.c

Patch 8.2.2384

Problem: Turtle filetype not recognized.
Solution: Add a rule to detect turtle files. (closes #7722)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2385

Problem: "gj" and "gk" do not work correctly when inside a fold.
Solution: Move check for folding. (closes #7724, closes #4095)
Files: src/normal.c, src/testdir/test_fold.vim

Patch 8.2.2386

Problem: Vim9: crash when using ":silent! put".
Solution: When ignoring an error for ":silent!" rewind the stack and skip ahead to restoring the cmdmod. (closes #7717)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2387

Problem: Runtime type check does not mention argument index.
Solution: Add ct_arg_idx. (closes #7720)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.2388

Problem: No easy way to get the maximum or minimum number value.
Solution: Add v:numbermax and v:numbermin.
Files: src/evalvars.c, src/vim.h, src/testdir/test_eval_stuff.vim,
runtime/doc/eval.txt

Patch 8.2.2389

Problem: Test failure on a few systems.
Solution: Avoid that "char" value is negative.
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h

Patch 8.2.2390

Problem: Vim9: using positive offset is unexpected.
Solution: Use int8_T instead of char. (James McCoy)

Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c

Patch 8.2.2391

Problem: Memory leak when creating a global function with closure.
Solution: Create a separate partial for every instantiated function.
Files: src/userfunc.c, src/vim9execute.c

Patch 8.2.2392

Problem: Fennel filetype not recognized.
Solution: Detect with pattern and hashbang. (Chinmay Dalal, closes #7729)
Files: runtime/filetype.vim, runtime/scripts.vim,
src/testdir/test_filetype.vim

Patch 8.2.2393

Problem: Vim9: error message when script line starts with "[{".
Solution: Do not give an error for checking for end of list.
Files: src/dict.c, src/testdir/test_vim9_script.vim

Patch 8.2.2394

Problem: Vim9: min() and max() return type is "any".
Solution: Use return type "number". (closes #7728)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2395

Problem: Vim9: error for wrong type may report wrong line number.
Solution: Save and restore the line number when evaluating the expression.
(closes #7727)
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2396

Problem: Vim9: no white space allowed before "->".
Solution: Allow for white space. (closes #7725)
Files: src/ex_docmd.c, src/eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2397

Problem: Vim9: "%%" not seen as alternate file name for commands with a
buffer name argument.
Solution: Recognize "%%" like "#". (closes #7732)
Files: src/buffer.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2398 (after 8.2.2396)

Problem: Method test fails.
Solution: Adjust test for allowed white space.
Files: src/testdir/test_method.vim

Patch 8.2.2399 (after 8.2.2385)

Problem: Fold test fails in wide terminal.
Solution: Adjust the test. (Dominique Pelle, closes #7731, closes #7739)
Files: src/testdir/test_fold.vim

Patch 8.2.2400

Problem: Vim9: compiled functions are not profiled.
Solution: Add initial changes to profile compiled functions. Fix that a
script-local function was hard to debug.

Files: runtime/doc/repeat.txt, src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/userfunc.c, src/proto/vim9compile.pro, src/structs.h, src/vim9type.c, src/debugger.c, src/ex_cmds.h, src/ex_docmd.c, src/profiler.c, src/proto/profiler.pro, src/testdir/test_vim9_disassemble.vim, src/testdir/test_profile.vim

Patch 8.2.2401

Problem: Build fails without +profiling feature.

Solution: Add #ifdefs.

Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/structs.h, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2402

Problem: Some filetypes not detected.

Solution: Detect Ruby Signature and Puppet related files. (Doug Kearns)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2403

Problem: Vim9: profiling if/elseif/endif not correct.

Solution: Add profile instructions. Fix that "elseif" was wrong.

Files: src/vim9compile.c, src/testdir/test_profile.vim, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2404

Problem: Vim9: profiling try/catch not correct.

Solution: Add profile instructions. Fix that "entry" did not rethrow an exception.

Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_profile.vim

Patch 8.2.2405

Problem: Vim9: no need to allow white space before "(" for :def.

Solution: Give an error for stray white space. (issue #7734)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2406

Problem: Vim9: profiled :def function leaks memory.

Solution: Delete the profiled instructions.

Files: src/vim9compile.c

Patch 8.2.2407

Problem: Old jumplist code is never used.

Solution: Delete the dead code. (Yegappan Lakshmanan, closes #7740)

Files: src/mark.c

Patch 8.2.2408

Problem: MinGW: "--preprocessor" flag no longer supported.

Solution: Remove the flag, use the defaults. (Christopher Wellons, closes #7741)

Files: src/GvimExt/Make_ming.mak, src/Make_cyg_ming.mak

Patch 8.2.2409

Problem: Vim9: profiling only works for one function.

Solution: Select the right instructions when calling and returning.
(closes #7743)
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h,
src/testdir/test_profile.vim

Patch 8.2.2410

Problem: Build failure without the +profiling feature.
Solution: Add dummy argument to macro.
Files: src/vim9.h

Patch 8.2.2411

Problem: Profile test fails on MS-Windows.
Solution: Do the profiling in a separate Vim command.
Files: src/testdir/test_profile.vim

Patch 8.2.2412

Problem: Not all fields in "cstack" are initialized which might cause a
crash.
Solution: Use CLEAR_FIELD().
Files: src/ex_docmd.c

Patch 8.2.2413

Problem: Crash when using :all while using a cmdline window. (Zdenek Dohnal)
Solution: Disallow :all from the cmdline window.
Files: src/arglist.c, src/ex_getln.c, src/testdir/test_arglist.vim

Patch 8.2.2414

Problem: Using freed memory when closing the cmdline window.
Solution: Check the window is still valid.
Files: src/ex_getln.c

Patch 8.2.2415

Problem: No way to check for the cmdwin feature, cmdline_hist is now always
enabled.
Solution: Add has('cmdwin') support. Skip arglist test on Windows
temporarily.
Files: runtime/doc/cmdline.txt, src/evalfunc.c,
src/testdir/test_autocmd.vim, src/testdir/test_arglist.vim,
src/testdir/test_cmdline.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_normal.vim, src/testdir/test_tabpage.vim,
src/testdir/test_termcodes.vim, src/testdir/test_window_cmd.vim

Patch 8.2.2416

Problem: May get stuck in command line window state.
Solution: Reset "cmdwin_type" when editing buffer fails. Make arglist test
pass on MS-Windows.
Files: src/ex_getln.c, src/testdir/test_arglist.vim

Patch 8.2.2417

Problem: Condition stack values may be used when not set.
Solution: Clear cs_script_var_len and cs_block_id just in case they get used
later. (issue #7733)
Files: src/ex_eval.c

Patch 8.2.2418

Problem: Color not changed if ModeMsg highlight is set in InsertEnter autocmd event. (Paul Swanson)
Solution: Call highlight_changed() after triggering InsertEnter. (closes #7751)
Files: src/edit.c

Patch 8.2.2419

Problem: Autocmd test was failing on MS-Windows with GUI.
Solution: Remove stray feedkeys().
Files: src/testdir/test_autocmd.vim

Patch 8.2.2420

Problem: Too many problems with using all autocommand events.
Solution: Disallow defining an autocommand for all events.
Files: src/autocmd.c, src/errors.h, src/testdir/test_autocmd.vim, src/testdir/test_quickfix.vim, src/testdir/test_window_cmd.vim

Patch 8.2.2421

Problem: Double free when using autocommand with "argdel". (Houyunsong)
Solution: Add the arglist_locked flag.
Files: src/arglist.c, src/testdir/test_autocmd.vim

Patch 8.2.2422

Problem: Crash when deleting with line number out of range. (Houyunsong)
Solution: Avoid using a negative line number.
Files: src/normal.c, src/testdir/test_ex_mode.vim

Patch 8.2.2423 (after 8.2.2422)

Problem: Missing error message.
Solution: Add the error message.
Files: src/errors.h

Patch 8.2.2424

Problem: Some tests are known to cause an error with ASAN.
Solution: Add CheckNotAsan.
Files: src/testdir/check.vim, src/testdir/test_ins_complete.vim, src/testdir/test_memory_usage.vim, src/testdir/test_ex_mode.vim

Patch 8.2.2425

Problem: Cursor on invalid line with range and :substitute.
Solution: Do not move the cursor when skipping commands. (closes #3434)
Files: src/ex_cmds.c, src/testdir/test_eval_stuff.vim

Patch 8.2.2426

Problem: Allowing **'completefunc'** to switch windows causes trouble.
Solution: use "textwinlock" instead of "textlock".
Files: src/insexpand.c, src/testdir/test_ins_complete.vim, src/testdir/test_popup.vim

Patch 8.2.2427

Problem: Can still switch windows for **'completfunc'**.
Solution: Also disallow switching windows for other completions.
Files: src/insexpand.c, src/testdir/test_ins_complete.vim,

src/testdir/test_popup.vim

Patch 8.2.2428

Problem: FocusGained does not work when 'ttymouse' is empty.
Solution: Don't use the short mouse code if there is a longer matching code.
(closes #7755) Add a test.
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.2429

Problem: :goto does not work correctly with text properties. (Sam McCall)
Solution: Add a test. (Andrew Radev) Also use the text property size when
computing the remaining offset. (closes #5930)
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.2.2430

Problem: :vimgrep expands wildcards twice.
Solution: Do not expand wildcards a second time.
Files: src/quickfix.c, src/arglist.c, src/testdir/test_quickfix.vim

Patch 8.2.2431

Problem: Warning for -fno-strength-reduce with Clang 11.
Solution: Adjust check for clang version number.
Files: src/configure.ac, src/auto/configure

Patch 8.2.2432

Problem: Libvterm tests are executed even when libtool doesn't work.
Solution: Only run libvterm tests if /usr/bin/gcc exists.
Files: src/Makefile

Patch 8.2.2433

Problem: Opening cmdline window gives error in BufLeave autocommand.
Solution: Reset cmdwin_type when triggering the autocommand.
Files: src/ex_cmds.c, src/testdir/test_cmdline.vim

Patch 8.2.2434

Problem: Vim9: no error when compiling str2nr() with a number.
Solution: Add argument type checks. (closes #7759)
Files: src/evalfunc.c, src/typval.c, src/proto/typval.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2435

Problem: setline() gives an error for some types.
Solution: Allow any type, convert each item to a string.
Files: runtime/doc/eval.txt, src/evalbuffer.c, src/typval.c,
src/proto/typval.pro, src/debugger.c, src/vim9execute.c,
src/testdir/test_buflines.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2436

Problem: Vim9 script test is a bit flaky.
Solution: Wait longer for exit callback.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2437

Problem: Deprecation warnings with default configuration.

Solution: Add -Wno-deprecated-declarations.
Files: src/configure.ac, src/auto/configure

Patch 8.2.2438

Problem: Out of bounds compiler warning.
Solution: Increase the size of uf_name.
Files: src/structs.h

Patch 8.2.2439

Problem: Not easy to figure out what packages to get when installing Vim on a new Ubuntu system.
Solution: Mention explicit commands that are easy to follow.
Files: src/INSTALL

Patch 8.2.2440

Problem: Documentation based on patches is outdated.
Solution: Add changes to documentation in a patch.
Files: runtime/doc/arabic.txt, runtime/doc/autocmd.txt,
runtime/doc/change.txt, runtime/doc/channel.txt,
runtime/doc/cmdline.txt, runtime/doc/debugger.txt,
runtime/doc/develop.txt, runtime/doc/digraph.txt,
runtime/doc/editing.txt, runtime/doc/eval.txt,
runtime/doc/filetype.txt, runtime/doc/ft_sql.txt,
runtime/doc/gui.txt, runtime/doc/gui_w32.txt,
runtime/doc/gui_x11.txt, runtime/doc/hangulin.txt,
runtime/doc/helphelp.txt, runtime/doc/help.txt,
runtime/doc/if_lua.txt, runtime/doc/if_mzsch.txt,
runtime/doc/if_tcl.txt, runtime/doc/indent.txt,
runtime/doc/index.txt, runtime/doc/insert.txt,
runtime/doc/intro.txt, runtime/doc/map.txt, runtime/doc/mbyte.txt,
runtime/doc/message.txt, runtime/doc/mlang.txt,
runtime/doc/motion.txt, runtime/doc/netbeans.txt,
runtime/doc/options.txt, runtime/doc/os_dos.txt,
runtime/doc/os_haiku.txt, runtime/doc/os_unix.txt,
runtime/doc/os_vms.txt, runtime/doc/os_win32.txt,
runtime/doc/pattern.txt, runtime/doc/pi_getscript.txt,
runtime/doc/pi_logipat.txt, runtime/doc/pi_netrw.txt,
runtime/doc/pi_tar.txt, runtime/doc/pi_vimball.txt,
runtime/doc/pi_zip.txt, runtime/doc/popup.txt,
runtime/doc/print.txt, runtime/doc/quickfix.txt,
runtime/doc/quickref.txt, runtime/doc/recover.txt,
runtime/doc/remote.txt, runtime/doc/repeat.txt,
runtime/doc/rileft.txt, runtime/doc/sign.txt,
runtime/doc/spell.txt, runtime/doc/starting.txt,
runtime/doc/syntax.txt, runtime/doc/tabpage.txt,
runtime/doc/tagsrch.txt, runtime/doc/terminal.txt,
runtime/doc/term.txt, runtime/doc/testing.txt,
runtime/doc/textprop.txt, runtime/doc/tips.txt,
runtime/doc/todo.txt, runtime/doc/uganda.txt,
runtime/doc/undo.txt, runtime/doc/usr_02.txt,
runtime/doc/usr_03.txt, runtime/doc/usr_04.txt,
runtime/doc/usr_05.txt, runtime/doc/usr_07.txt,
runtime/doc/usr_08.txt, runtime/doc/usr_10.txt,
runtime/doc/usr_11.txt, runtime/doc/usr_20.txt,

runtime/doc/usr_22.txt, runtime/doc/usr_23.txt,
runtime/doc/usr_24.txt, runtime/doc/usr_27.txt,
runtime/doc/usr_30.txt, runtime/doc/usr_31.txt,
runtime/doc/usr_40.txt, runtime/doc/usr_41.txt,
runtime/doc/usr_42.txt, runtime/doc/usr_44.txt,
runtime/doc/usr_45.txt, runtime/doc/usr_46.txt,
runtime/doc/usr_90.txt, runtime/doc/usr_toc.txt,
runtime/doc/varioues.txt, runtime/doc/version5.txt,
runtime/doc/version6.txt, runtime/doc/version7.txt,
runtime/doc/version8.txt, runtime/doc/vi_diff.txt,
runtime/doc/vim9.txt, runtime/doc/visual.txt,
runtime/doc/windows.txt

Patch 8.2.2441

Problem: Vim9: extend() does not give an error for a type mismatch.
Solution: Check the type of the second argument. (closes #7760)
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2442

Problem: Automatic GUI selection does not check for GTK 3.
Solution: Make SKIP_GTK3 empty for automatic GUI support. Set SKIP_GTK3 to YES when checking for GTK2.
Files: src/configure.ac, src/auto/configure

Patch 8.2.2443

Problem: Vim9: no compile time error for wrong str2float argument.
Solution: Check argument type. (closes #7759)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2444

Problem: Vim9: compile error with combination of operator and list.
Solution: Generate constants before parsing a list or dict. (closes #7757)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2445

Problem: Vim9: no proper error for lambda missing return type.
Solution: Check for this error. (closes #7758)
Files: src/errors.h, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2446

Problem: Setting **'term'** empty has different error if compiled with GUI.
Solution: Insert "else". (closes #7766)
Files: src/optionstr.c, src/testdir/test_options.vim

Patch 8.2.2447

Problem: **'foldlevel'** not applied to folds restored from session.
Solution: Set **'foldlevel'** after creating the folds. (closes #7767)
Files: src/fold.c, src/testdir/test_mksession.vim

Patch 8.2.2448

Problem: Compilation error with Ruby 3.0.
Solution: Adjust #ifdefs and declaration. (Ken Takata, closes #7761)
Files: src/if_ruby.c

Patch 8.2.2449

Problem: Vim9: flatten() always changes the list type.
Solution: Disallow using flatten() and add flattennew().
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/list.c, src/proto/list.pro, src/errors.h, src/vim9compile.c,
src/testdir/test_flatten.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2450

Problem: MS-Windows: ADS was not preserved if **'backupcopy'** is "yes".
Solution: Copy ADS before truncating the file. (Ken Takata, closes #7762)
Files: src/bufwrite.c

Patch 8.2.2451

Problem: MS-Windows: Extended Attributes not preserved.
Solution: Preserve Extended Attributes when writing a file. (Ken Takata,
closes #7765)
Files: src/os_win32.c

Patch 8.2.2452

Problem: No completion for the **'filetype'** option.
Solution: Add filetype completion. (Martin Tournoij, closes #7747)
Files: src/option.c, src/optiondefs.h, src/testdir/test_options.vim

Patch 8.2.2453

Problem: Vim9: a variable name with "->" in the next line doesn't work.
Solution: Recognize a variable name by itself. (closes #7770)
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2454

Problem: Leading space can not be made visible.
Solution: Add "lead:" to **'listchars'**. (closes #7772)
Files: runtime/doc/options.txt, src/drawline.c, src/globals.h,
src/message.c, src/screen.c, src/testdir/test_listchars.vim

Patch 8.2.2455

Problem: Vim9: key type that can be used for literal dict and indexing is
inconsistent.
Solution: Allow using number and bool as key for a literal dict. (#7771)
Files: runtime/doc/vim9.txt, src/dict.c, src/eval.c, src/vim9compile.c,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2456

Problem: Coverity warning for strcpy() into fixed size array.
Solution: Add a type cast to hopefully silence the bogus warning.
Files: src/userfunc.c

Patch 8.2.2457

Problem: Coverity warns for memory leak.
Solution: Free memory when out of memory.
Files: src/if_cscope.c

Patch 8.2.2458

Problem: Coverity warns for :retab using freed memory.

Solution: Use the updated line pointer when moving text properties.
Files: src/indent.c

Patch 8.2.2459

Problem: Coverity reports dead code.
Solution: Remove the dead code.
Files: src/eval.c

Patch 8.2.2460

Problem: Coverity warns for unused value.
Solution: Do not reset the return value to OK.
Files: src/vim9compile.c

Patch 8.2.2461

Problem: Coverity warns for unchecked return value.
Solution: Add "(void)" to avoid the warning.
Files: src/vim9execute.c

Patch 8.2.2462

Problem: Coverity warns for not checking for fseek() error.
Solution: Give an error message if fseek() fails.
Files: src/spellfile.c

Patch 8.2.2463

Problem: Using :arglocal in an autocommand may use freed memory.
(houyunsong)
Solution: Check if the arglist is locked.
Files: src/arglist.c, src/testdir/test_autocmd.vim

Patch 8.2.2464

Problem: Using freed memory if window closed in autocommand. (houyunsong)
Solution: Check the window still exists.
Files: src/ex_cmds.c, src/testdir/test_autocmd.vim

Patch 8.2.2465

Problem: Using freed memory in :psearch. (houyunsong)
Solution: Check the current window is still valid. Fix flaky test.
Files: src/search.c, src/testdir/test_autocmd.vim

Patch 8.2.2466

Problem: Max() and min() can give many error messages.
Solution: Bail out at the first error. (closes #1039, closes #7778)
Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.2.2467

Problem: Script generated by :mkview changes alternate file.
Solution: Only write :balt in the session file. (Harish Rajagopal,
closes #7779)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.2468

Problem: Not easy to get the full command name from a shortened one.
Solution: Add fullcommand(). (Martin Tournoij, closes #7777)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,

src/ex_docmd.c, src/proto/evalfunc.pro,
src/testdir/test_cmdline.vim

Patch 8.2.2469

Problem: Confusing error if :winsize has a wrong argument.
Solution: Quote the argument in the error. (closes #2523)
Files: src/ex_docmd.c, src/testdir/test_excmd.vim

Patch 8.2.2470

Problem: Popup_getoptions() does not get textprop from other tab.
Solution: use win_valid_any_tab(). (closes #7786)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.2471

Problem: Popup_setoptions() does not set textprop in other tab.
Solution: use win_valid_any_tab(). (closes #7788)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.2472

Problem: Crash when using command line window in an autocommand.
(houyunsong)
Solution: Save and restore au_new_curbuf.
Files: src/ex_cmds.c, src/testdir/test_autocmd.vim

Patch 8.2.2473

Problem: Crash when leaving command line window triggers autocommand.
(houyunsong)
Solution: Make sure not to close the current window or buffer.
Files: src/ex_getln.c, src/testdir/test_autocmd.vim

Patch 8.2.2474

Problem: Using freed memory when window is closed by autocommand.
(houyunsong)
Solution: Check the window pointer is still valid.
Files: src/quickfix.c, src/testdir/test_autocmd.vim

Patch 8.2.2475

Problem: Autocommand tests hangs on MS-Windows.
Solution: Skip one test.
Files: src/testdir/test_autocmd.vim

Patch 8.2.2476

Problem: Using freed memory when using an autocommand to split a window
while a buffer is being closed.
Solution: Disallow splitting when the buffer has b_locked_split set.
Files: src/buffer.c, src/window.c, src/errors.h, src/structs.h,
src/popupwin.c, src/testdir/test_autocmd.vim

Patch 8.2.2477

Problem: Autocommand tests hang on MS-Windows.
Solution: Skip a couple of tests. Fix file name.
Files: src/testdir/test_autocmd.vim

Patch 8.2.2478

Problem: MS-Windows: backup files for plugins are loaded.
Solution: Do not use the alternate file name for files ending in "~".
Files: src/filepath.c

Patch 8.2.2479

Problem: set/getbuflin test fails without the job feature.
Solution: Check whether the job feature is supported. (Dominique Pellé,
closes #7790)
Files: src/testdir/test_buflin.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2480

Problem: Vim9: some errors for white space do not show context.
Solution: Include the text at the error.
Files: src/errors.h, src/dict.c, src/list.c, src/userfunc.c,
src/vim9compile.c, src/vim9script.c, src/vim9type.c

Patch 8.2.2481

Problem: Vim9: confusing error when variable arguments have a default
value.
Solution: Give a specific error message. (closes #7793)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2482

Problem: Build error.
Solution: Add new error message.
Files: src/errors.h

Patch 8.2.2483

Problem: Vim9: type error for malformed expression.
Solution: Check for end of command before checking type. (closes #7795)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2484

Problem: Vim9: Cannot use a comment starting with #{ after an expression.
Solution: Remove the check for "{" since #{ dictionaries are not supported.
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2485

Problem: When sourcing a script again the script version isn't reset.
Solution: Set sn_version to one when sourcing a script again. Clear
sn_save_cpo properly. (closes #7608)
Files: src/scriptfile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2486

Problem: Vim9: some errors for white space do not show context.
Solution: Include the text at the error.
Files: src/errors.h, src/dict.c, src/list.c, src/userfunc.c,
src/vim9compile.c, src/vim9type.c

Patch 8.2.2487

Problem: Terminal shows garbage after double-wide character with a
combining character. (Kyoichiro Yamada)
Solution: Libvterm: do not add the width of the combining character to the
glyph width. (closes #7801)

Files: src/libvterm/src/state.c, src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_combining.dump

Patch 8.2.2488

Problem: json_encode() gives generic argument error.
Solution: Mention the type that can't be encoded. (issue #7802)
Files: src/json.c, src/errors.h, src/testdir/test_json.vim

Patch 8.2.2489

Problem: current buffer is wrong after deletebuflin() fails to delete a
line in another buffer.
Solution: Restore the current buffer.
Files: src/evalbuffer.c, src/testdir/test_buflin.vim

Patch 8.2.2490

Problem: **'wrap'** option is always reset when starting diff mode.
Solution: Add the "followwrap" item in **'diffopt'**. (Rick Howe, closes #7797)
Files: runtime/doc/diff.txt, runtime/doc/options.txt, src/diff.c,
src/testdir/test_diffmode.vim

Patch 8.2.2491

Problem: Popup window for text property may show in first screen line.
Solution: If the text position is invisible do not show the popup window.
(closes #7807)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popup_prop_not_visible_01.dump,
src/testdir/dumps/Test_popup_prop_not_visible_02.dump,
src/testdir/dumps/Test_popup_prop_not_visible_03.dump

Patch 8.2.2492

Problem: Command line buffer name cannot be translated.
Solution: Add _(). (Gabriel Dupras, closes #7812)
Files: src/ex_getln.c

Patch 8.2.2493

Problem: Text property for text left of window shows up.
Solution: Check if the text property ends before the current column.
(closes #7806)
Files: src/drawline.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_textprop_nowrap_01.dump,
src/testdir/dumps/Test_textprop_nowrap_02.dump

Patch 8.2.2494

Problem: ":rviminfo!" clears most of oldfiles.
Solution: Add VIF_ONLY_CURBUF to read_viminfo(). (closes #1781)
Files: src/viminfo.c, src/vim.h

Patch 8.2.2495

Problem: Text jumps up and down when moving the cursor in a small window
with wrapping text and **'scrolloff'** set.
Solution: Adjust the computation of w_skipcol. (partly by Ghjuvan Lacambre,
closes #7813)
Files: src/move.c, src/testdir/test_breakindent.vim

Patch 8.2.2496 (after 8.2.2495)

Problem: Insufficient testing for text jumping fix.

Solution: Add another test case.

Files: src/testdir/test_breakindent.vim

Patch 8.2.2497

Problem: No error when using more than one character for a register name.

Solution: In Vim9 script check for a single character string. (closes #7814)

Fix that VAR_BOOL and VAR_SPECIAL are not considered equal.

Files: src/errors.h, src/evalfunc.c, src/typval.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2498

Problem: No test for what 8.2.2494 fixes.

Solution: Add a simple change to test the fix. (closes #7818)

Files: src/testdir/test_viminfo.vim

Patch 8.2.2499

Problem: "vim -g --version" does not redirect output.

Solution: Reset gui.starting when showing version info. (closes #7815)

Files: src/main.c, src/testdir/test_version.vim

Patch 8.2.2500 (after 8.2.2499)

Problem: Build fails without the GUI feature.

Solution: Add #ifdef.

Files: src/main.c

Patch 8.2.2501

Problem: Not always clear where an error is reported.

Solution: Add the where_T structure and pass it around. (closes #7796)

Files: src/structs.h, src/vim9type.c, src/proto/vim9type.pro,
src/errors.h, src/evalvars.c, src/proto/evalvars.pro, src/eval.c,
src/proto/eval.pro, src/vim9execute.c, src/vim9script.c,
src/proto/vim9script.pro, src/dict.c, src/list.c,
src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2502

Problem: A few github actions are failing.

Solution: Install setuptools-rust. (closes #7823)

Files: .github/workflows/ci.yml

Patch 8.2.2503

Problem: Vim9: a caught error may leave something on the stack.

Solution: Drop items from the stack if needed. (closes #7826)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2504

Problem: Vim9: crash when using an argument from a closure.

Solution: Check if gen_load_outer is NULL. (closes #7821)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2505

Problem: Vim9: crash after defining function with invalid return type.

Solution: Clear function growarrays. Fix memory leak.

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2506

Problem: Vim9: :continue does not work correctly in a :try block

Solution: Add the TRYCLEANUP instruction. (closes #7827)

Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2507

Problem: Github build may fail if Ubuntu 20.04 is used. Installing rust is not needed.

Solution: Specify ubuntu-18.04 instead of latest. Update "pip" instead of installing rust. (Ozaki Kiichi, closes #7820)

Files: .github/workflows/ci.yml

Patch 8.2.2508

Problem: Cannot change the character displayed in non existing lines.

Solution: Add the "eob" item to 'fillchars'. (closes #7832, closes #3820)

Files: runtime/doc/options.txt, runtime/doc/todo.txt,
runtime/doc/windows.txt, src/drawscreen.c, src/globals.h,
src/optiondefs.h, src/screen.c, src/testdir/test_display.vim

Patch 8.2.2509

Problem: Tests fail on s390 build.

Solution: Initialize trycmd_T.

Files: src/vim9execute.c

Patch 8.2.2510

Problem: Internal error when popup with mask is zero height or width.

Solution: Bail out if width or height is zero. (closes #7831)

Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.2511

Problem: Vim9: cannot use Vim9 script syntax in some places.

Solution: Add the :vim9cmd command modifier. Incompatible: Makes ":vim9" mean ":vim9cmd" instead of ":vim9script".

Files: runtime/doc/vim9.txt, runtime/doc/repeat.txt, src/ex_docmd.c,
src/ex_cmds.h, src/structs.h, src/ex_cmdidxs.h, src/errors.h,
src/testdir/test_vim9_cmd.vim, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_1.dump,
src/testdir/dumps/Test_wildmenu_2.dump,
src/testdir/dumps/Test_wildmenu_3.dump,
src/testdir/dumps/Test_wildmenu_4.dump,
src/testdir/test_quickfix.vim

Patch 8.2.2512

Problem: Vim9: compiling error test sometimes fails.

Solution: use WaitForAssert() instead of sleeping for a bit. (Dominique Pellé, closes #7837)

Files: src/testdir/term_util.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2513 (after 8.2.2511)

Problem: Vim9: missing part of :vim9cmd change.
Solution: Use command modifier in in_vim9script().
Files: src/vim9script.c

Patch 8.2.2514 (after 8.2.2511)

Problem: Vim9: build error in tiny version.
Solution: Add #ifdef.
Files: src/ex_docmd.c

Patch 8.2.2515

Problem: Memory access error when truncating an empty message.
Solution: Check for an empty string. (Dominique Pellé, closes #7841)
Files: src/message.c, src/message_test.c

Patch 8.2.2516

Problem: Test failure on s390. (analyses by James McCoy)
Solution: Only set the try_finally label when not skipping.
Files: src/vim9compile.c

Patch 8.2.2517

Problem: Vim9: fix for s390 not tested on other systems.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2518

Problem: **'listchars'** should be window-local.
Solution: Make **'listchars'** global-local. (Yegappan Lakshmanan, Marco Hinz, closes #5206, closes #7850)
Files: runtime/doc/options.txt, src/buffer.c, src/charset.c, src/drawline.c, src/drawscreen.c, src/evalfunc.c, src/globals.h, src/indent.c, src/message.c, src/misc1.c, src/option.c, src/option.h, src/optiondefs.h, src/optionstr.c, src/proto/screen.pro, src/screen.c, src/structs.h, src/testdir/test_listchars.vim, src/testdir/test_listlbr.vim

Patch 8.2.2519

Problem: Vim9: no reason to keep strange Vi behavior.
Solution: ":3" and ":3|" both go to line 3. ":|" does not print the line. (closes #7840)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2520

Problem: Missing tests for **'listchars'**.
Solution: Add a few more checks. (Yegappan Lakshmanan, closes #7854)
Files: src/testdir/test_listchars.vim

Patch 8.2.2521

Problem: Some compilers can't handle pointer initialization. (John Marriott)
Solution: Use a local struct and assign it afterwards.
Files: src/screen.c

Patch 8.2.2522

Problem: Beancount filetype not recognized.

Solution: Add a detection rule. (Brian Ryall, closes #7859)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2523

Problem: Svelte filetype not recognized.
Solution: Add a detection rule. (Brian Ryall, closes #7858)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2524

Problem: Cannot change the characters displayed in the foldcolumn.
Solution: Add fields to '**fillchars**'. (Yegappan Lakshmanan, Matthieu Coudron, closes #7860)
Files: runtime/doc/options.txt, src/globals.h, src/mouse.c, src/screen.c, src/testdir/test_display.vim

Patch 8.2.2525

Problem: Vim9: only local variables checked for a name.
Solution: Also check arguments and script variables. (closes #7838)
Files: src/vim9compile.c, src/ex_docmd.c, src/proto/ex_docmd.pro, src/testdir/test_vim9_cmd.vim

Patch 8.2.2526 (after 8.2.2525)

Problem: Build failure.
Solution: Change lookup_scriptvar() arguments.
Files: src/evalvars.c, src/proto/evalvars.pro

Patch 8.2.2527

Problem: Vim9: lambda return type is not determined at script level.
Solution: Compile the lambda to get the return type. (closes #7843)
Files: src/eval.c, src/vim.h, src/vim9.h, src/testdir/test_vim9_assign.vim

Patch 8.2.2528

Problem: Vim9: crash when compiling lambda fails.
Solution: Bail out after compilation fails. (closes #7862)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2529

Problem: Vim9: Not possible to use legacy and Vim9 script in one file.
Solution: Vim9: allow for "if false" before :vim9script. (closes #7851)
Files: runtime/doc/vim9.txt, src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.2530

Problem: Vim9: not enough testing for profiling.
Solution: Add a test with nested functions and a lambda. Fix profiling for calling a compiled function.
Files: src/profiler.c, src/proto/profiler.pro, src/userfunc.c, src/vim9execute.c, src/testdir/test_profile.vim

Patch 8.2.2531

Problem: Vim9: the :k command is obscure.
Solution: Disallow using :k, can use :mark instead. (closes #7874)
Files: runtime/doc/vim9.txt, src/ex_docmd.c, src/vim9script.c,

src/vim9compile.c, src/ex_cmds.h, src/testdir/test_vim9_script.vim

Patch 8.2.2532

Problem: Vim9: confusing error if :k is used with a range.
Solution: Give an error about the range. (issue #7874)
Files: src/vim9script.c, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2533

Problem: Vim9: cannot use a range with :unlet.
Solution: Implement ISN_UNLETRANGE.
Files: src/errors.h, src/eval.c, src/evalvars.c, src/list.c,
src/proto/evalvars.pro, src/proto/list.pro, src/vim9.h,
src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.2534

Problem: Missing test coverage.
Solution: Improve test coverage for completion with different encodings,
mapset(), and term function failures. (Dominique Pellé,
closes #7877)
Files: src/testdir/test_edit.vim, src/testdir/test_maparg.vim,
src/testdir/test_terminal3.vim

Patch 8.2.2535

Problem: MS-Windows: cannot run all vim9 tests.
Solution: Make test_vim9 target work.
Files: src/Make_mvc.mak

Patch 8.2.2536

Problem: Coverity complains about unchecked return value.
Solution: Add (void).
Files: src/userfunc.c

Patch 8.2.2537

Problem: Vim9: crash when map() fails.
Solution: Clear typval before using it. (closes #7884)
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2538

Problem: Crash when using Python list iterator.
Solution: Increment the list reference count. (closes #7886)
Files: src/if_py_both.h, src/testdir/test_python3.vim

Patch 8.2.2539

Problem: Vim9: return from finally block causes a hang.
Solution: Store both the finally and endtry indexes. (closes #7885)
Files: src/vim9execute.c, src/vim9compile.c, src/vim9.h,
src/testdir/test_vim9_script.vim,

Patch 8.2.2540

Problem: Vim9: no error for using script var name for argument.
Solution: Check for this error. (closes #7868)
Files: src/userfunc.c, src/vim9compile.c, src/proto/vim9compile.pro,

src/testdir/test_vim9_func.vim

Patch 8.2.2541

Problem: Popup_create() does not allow boolean for "cursorline".
Solution: Use dict_get_bool(). (issue #7869)
Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.2542

Problem: Highlight of char beyond line end is not correct. (Chuan Wei Foo)
Solution: Fix counting NUL as one cell. Draw one more character if the EOL is part of the match. (closes #7883)
Files: src/match.c, src/testdir/test_search.vim, src/testdir/dumps/Test_hlsearch_1.dump, src/testdir/dumps/Test_hlsearch_2.dump

Patch 8.2.2543

Problem: Vim9: a return inside try/catch does not restore exception state properly.
Solution: When there is no ":finally" jump to ":endtry". (closes #7882)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2544

Problem: Vim9: error for argument when checking for lambda.
Solution: Respect the skip flag. (closes #7887)
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2545

Problem: Errors and crash when terminal window is zero height. (Leonid V. Fedorenchik)
Solution: Do not resize when width or height is zero. (closes #7890)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.2546

Problem: Typo in mouse key name.
Solution: Fix the typo. (issue #4725)
Files: src/misc2.c

Patch 8.2.2547

Problem: "%" command not accurate for big files.
Solution: Make it more accurate for files up to 21M lines. (Dominique Pellé, closes #7889)
Files: src/normal.c

Patch 8.2.2548

Problem: May get stuck in the cmdline window using :normal.
Solution: Have nv_esc() return K_IGNORE.
Files: src/normal.c

Patch 8.2.2549

Problem: Crash after using "g:" in a for loop.
Solution: Increment the reference count. (closes #7892)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2550

Problem: Signal stack size is wrong with latest glibc 2.34.
Solution: Use sysconf(_SC_SIGSTKSZ) if available. (Zdenek Dohnal, closes #7895)
Files: src/config.h.in, src/configure.ac, src/os_unix.c, src/auto/configure

Patch 8.2.2551

Problem: MS-Windows: colors test file is not installed.
Solution: Also copy runtime/colors/tools. (Ken Takata, closes #7902)
Files: nsis/gvim.nsi

Patch 8.2.2552

Problem: Vim9: no reason to consider "{{{{{{{{{{" a command.
Solution: Just use "{". (issue #7904)
Files: src/ex_cmds.h

Patch 8.2.2553

Problem: Vim9: Cannot put "|" after "{".
Solution: Add the EX_TRLBAR flag. (issue #7904)
Files: src/ex_cmds.h, src/ex_eval.c

Patch 8.2.2554

Problem: Vim9: exporting a final is not tested.
Solution: Add a test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2555

Problem: Vim9: missing test for 8.2.2553.
Solution: Add a simple test.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2556

Problem: Vim9: :import with "as" not fully supported.
Solution: Implement "as" for more cases.
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.2557

Problem: Compiler warning for shadowed variable.
Solution: Declare "p" only once.
Files: src/vim9script.c

Patch 8.2.2558

Problem: No error if a lambda argument shadows a variable.
Solution: Check that the argument name shadows a local, argument or script variable. (closes #7898)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/userfunc.c, src/vim9script.c, src/errors.h, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_script.vim

Patch 8.2.2559

Problem: MS-Windows: guifont test fails on Windows XP.
Solution: Check windowsversion().
Files: src/testdir/test_gui.vim

Patch 8.2.2560

Problem: Setting **'winminheight'** does not take tabline into account.
Solution: Subtract the tabline from the available height. (closes #7899)
Files: src/window.c, src/testdir/test_options.vim

Patch 8.2.2561

Problem: Not all textprop code is covered by tests.
Solution: Add a few more test cases. (Dominique Pellé, closes #7908)
Files: src/testdir/test_textprop.vim

Patch 8.2.2562

Problem: GUI: star register changed when **'clipboard'** is "unnamedplus". (Ingo Karkat)
Solution: Do not change the star register when **'clipboard'** contains "unnamedplus" and not "unnamed". (closes #1516)
Files: src/register.c

Patch 8.2.2563

Problem: Cannot use multibyte characters for folding in **'fillchars'**.
Solution: Port pull request 11568 to Vim. (Yegappan Lakshmanan, closes #7924)
Files: src/drawline.c, src/drawscreen.c, src/macros.h, src/proto/screen.pro, src/screen.c, src/testdir/test_fold.vim, src/testdir/test_profile.vim

Patch 8.2.2564

Problem: Focus events end Insert mode if **'esckey'** is not set.
Solution: Do not enable focus events when **'esckey'** is off. (closes #7926)
Files: src/term.c

Patch 8.2.2565

Problem: Vim9: "..=" not always recognized.
Solution: Do not consider "..=" to be string concatenation. (closes #7905)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2566

Problem: Vim9: Function name is not recognized.
Solution: Change lookup_scriptvar() to also find function names. (closes #7770)
Files: src/vim9script.c, src/evalvars.c, src/proto/evalvars.pro, src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2567

Problem: Vim9: no error if variable is defined for existing function.
Solution: Check if name isn't already in use. (closes #7897)
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.2568

Problem: Second time a preview popup is opened highlight is not set. (Gabriel Dupras)
Solution: Apply **'previewpopup'** after getting the file. (closes #7928)
Files: src/tag.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_previewpopup_2.dump, src/testdir/dumps/Test_popupwin_previewpopup_3.dump,

src/testdir/dumps/Test_popupwin_previewpopup_4.dump,
src/testdir/dumps/Test_popupwin_previewpopup_5.dump

Patch 8.2.2569

Problem: **'fillchars'** "stl" and "stlnc" items must be single byte.
Solution: Accept multi-byte characters. (Christian Wellenbrock, Yegappan Lakshmanan, closes #7927)
Files: runtime/doc/options.txt, src/buffer.c, src/macros.h, src/screen.c,
src/testdir/test_fold.vim, src/testdir/test_statusline.vim

Patch 8.2.2570

Problem: Tests fail when run as root.
Solution: Add a comment mentioning the expected failure. (issue #7919)
Files: src/testdir/test_edit.vim, src/testdir/test_excmd.vim,
src/testdir/test_help.vim, src/testdir/test_writefile.vim

Patch 8.2.2571

Problem: Test may leave file behind.
Solution: Delete the temporary file. Don't profile in the running Vim instance.
Files: src/testdir/test_quickfix.vim, src/testdir/test_profile.vim

Patch 8.2.2572

Problem: Vim9: crash when getting the types for a legacy function.
Solution: Initialize the type list growarray. (closes #7929)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2573

Problem: Vim9: using invalid pointer for error message.
Solution: Use the right pointer. (closes #7921)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2574

Problem: Vim9: crash when calling partial with wrong function.
Solution: Check argument types of called function. (closes #7912)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2575

Problem: Vim9: a function name with "->" in the next line doesn't work.
Solution: Recognize a function name by itself. (closes #7770)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2576

Problem: Vim9: defining a :func function checks for white space after a comma in the arguments.
Solution: Only check for white space in a :def function. (closes #7930)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2577

Problem: Compiler warning for type conversion.
Solution: Add a typecast. (Mike Williams)
Files: src/drawline.c

Patch 8.2.2578

Problem: Lua cannot handle a passed in lambda.
Solution: Handle VAR_PARTIAL. (Prabir Shrestha, closes #7937, closes #7936)
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.2579

Problem: Vim9: crash in garbagecollect after for loop.
Solution: Do not set a reference in script item when the name was cleared.
(closes #7935)
Files: src/evalvars.c

Patch 8.2.2580

Problem: Vim9: checking vararg type is wrong when function is auto-loaded.
Solution: Use the member type. (closes #7933)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2581

Problem: Vim9: sourcing Vim9 script triggers a redraw.
Solution: Do not let setting/restoring **'cptions'** cause a redraw.
(closes #7920)
Files: src/vim.h, src/option.c, src/optionstr.c, src/scriptfile.c,
src/vim9script.c, src/testdir/test_vim9_script.vim,
src/testdir/dumps/Test_vim9_no_redraw.dump

Patch 8.2.2582 (after 8.2.2581)

Problem: Vim9: screendump test fails on MS-Windows.
Solution: Use :function instead of :def.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2583

Problem: Vim9: cannot compare result of getenv() with null.
Solution: Make the return type of getenv() "any". (closes #7943)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2584

Problem: Vim9: type error for assigning the result of list concatenation to a list.
Solution: Do not consider concatenation result in a constant. (closes #7942)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2585

Problem: Vim9: illegal memory access.
Solution: Check byte right after "null", not one more.
Files: src/vim9compile.c

Patch 8.2.2586

Problem: Process id may be invalid.
Solution: Use sysinfo.uptime to check for recent reboot. (suggested by Hugo van der Sanden, closes #7947)
Files: src/configure.ac, src/auto/configure, src/config.h.in,
src/memline.c, src/testing.c, src/globals.h,
src/testdir/test_recover.vim

Patch 8.2.2587 (after 8.2.2586)

Problem: Recover test fails on FreeBSD.

Solution: Check for Linux.
Files: src/testdir/check.vim, src/testdir/test_recover.vim

Patch 8.2.2588 (after 8.2.2586)
Problem: Build failure with tiny features.
Solution: Add #ifdef. Run recover test separately.
Files: src/memline.c, src/testdir/Make_all.mak, src/testdir/test_alot.vim

Patch 8.2.2589 (after 8.2.2586)
Problem: Recover test hangs in the GUI.
Solution: Add g:skipped_reason to skip a _nocatch_ test.
Files: src/testdir/runtest.vim, src/testdir/test_recover.vim

Patch 8.2.2590
Problem: Vim9: default argument value may cause internal error.
Solution: Hide later function arguments when compiling the expression.
(closes #7948)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2591
Problem: Poke files are not recognized.
Solution: Add a filetype entry. (Matt Ihlenfield)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2592
Problem: Code coverage could be improved.
Solution: Add a few more tests. (Dominique Pellé, closes #7957)
Files: src/testdir/test_fileformat.vim, src/testdir/test_normal.vim,
src/testdir/test_sleep.vim, src/testdir/test_textformat.vim,
src/testdir/test_viminfo.vim

Patch 8.2.2593
Problem: List of distributed files is incomplete.
Solution: Add a file and rename another.
Files: Filelist

Patch 8.2.2594
Problem: Alternate buffer added to session file even when it's hidden.
Solution: Check the **'buflisted'** option. (closes #7951)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.2595
Problem: Setting **'winminheight'** may cause **'lines'** to change.
Solution: Also take minimal height of other tabpages into account. (#7899)
Files: src/window.c, src/testdir/test_options.vim

Patch 8.2.2596
Problem: :doautocmd may confuse scripts listening to WinEnter.
Solution: Do the current buffer last. (closes #7958)
Files: src/autocmd.c, src/testdir/test_autocmd.vim

Patch 8.2.2597
Problem: Vim9: "import * as" does not work at script level.
Solution: Implement using an imported namespace.

Files: src/vim.h, src/eval.c, src/evalvars.c, src/proto/evalvars.pro,
src/vim9execute.c, src/errors.h, src/vim9script.c,
src/proto/vim9script.pro, src/testdir/test_vim9_script.vim

Patch 8.2.2598

Problem: Vim9: :open does not need to be supported.
Solution: Do not support :open in Vim9 script.
Files: src/ex_docmd.c, src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.2599 (after 8.2.2597)

Problem: Build failure.
Solution: Add missing change.
Files: src/vim9compile.c

Patch 8.2.2600

Problem: Vim9: crash when putting an unknown type in a dictionary.
(Yegappan Lakshmanan)
Solution: Handle a NULL type pointer.
Files: src/vim9type.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2601

Problem: Memory usage test often fails on FreeBSD.
Solution: Increase multiplier for upper limit.
Files: src/testdir/test_memory_usage.vim

Patch 8.2.2602

Problem: Vim9: continue doesn't work if :while is very first command.
(Yegappan Lakshmanan)
Solution: Add one to the continue instruction index.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.2603

Problem: Vim9: no effect if user command is also a function.
Solution: Check for paren following. (closes #7960)
Files: src/evalvars.c, src/proto/evalvars.pro, src/ex_docmd.c,
src/proto/ex_docmd.pro, src/vim9compile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.2604

Problem: GUI-specific command line arguments not tested.
Solution: Add tests for several arguments. (Dominique Pellé, closes #7962)
Files: src/testdir/test_startup.vim

Patch 8.2.2605

Problem: Vim9: string index and slice does not include composing chars.
Solution: Include composing characters. (issue #6563)
Files: runtime/doc/vim9.txt, src/vim9execute.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.2606

Problem: strchars() defaults to counting composing characters.
Solution: Add strcharlen() which ignores composing characters.
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/testdir/test_utf8.vim

Patch 8.2.2607

Problem: strcharpart() cannot include composing characters.
Solution: Add the `{skipcc}` argument.
Files: runtime/doc/eval.txt, src/evalfunc.c,
src/testdir/test_expr_utf8.vim

Patch 8.2.2608

Problem: Character input not fully tested.
Solution: Add more tests. (Yegappan Lakshmanan, closes #7963)
Files: src/testdir/test_functions.vim, src/testdir/test_messages.vim,
src/testdir/test_paste.vim, src/testdir/test_registers.vim,
src/testdir/test_undo.vim

Patch 8.2.2609

Problem: Test disabled on MS-Windows even though it should work.
Solution: Restore the condition for skipping the test. (Ken Takata,
closes #7970)
Files: src/testdir/test_startup.vim

Patch 8.2.2610

Problem: Mouse click test fails when using remote connection.
Solution: Use a larger `'mousetime'`. (Dominique Pellé, closes #7968)
Files: src/testdir/test_selectmode.vim

Patch 8.2.2611

Problem: Conditions for startup tests are not exactly right.
Solution: Check for type of GUI instead of MS-Windows. (Ozaki Kiichi,
closes #7976)
Files: src/main.c, src/testdir/check.vim, src/testdir/test_startup.vim

Patch 8.2.2612

Problem: col('.') may get outdated column value.
Solution: Add a [note](#) to the help how to make this work and add a test for
it. (closes #7971)
Files: runtime/doc/map.txt, src/testdir/test_mapping.vim

Patch 8.2.2613 (after 8.2.2612)

Problem: New test throws exception.
Solution: Adjust the function cleanup.
Files: src/testdir/test_mapping.vim

Patch 8.2.2614

Problem: Vim9: function is deleted while executing.
Solution: increment the call count, when more than zero do not delete the
function but mark it as dead. (closes #7977)
Files: src/vim9execute.c, src/userfunc.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2615 (after 8.2.2614)

Problem: Test is sourcing the wrong file.
Solution: Correct the file name.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2616

Problem: Vim9: if 'cpo' is changed in Vim9 script it may be restored.
Solution: Apply the changes to 'cpo' to the restored value.
Files: runtime/doc/vim9.txt, src/scriptfile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2617

Problem: Vim9: script variable in a block scope not found by a nested function.
Solution: Copy the block scope IDs before compiling the function.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2618

Problem: Vim9: cannot use a normal list name to store function refs.
Solution: Allow a lower case name if it is indexed.
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2619

Problem: Vim9: no test for return type of lambda.
Solution: Add a test.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.2620

Problem: Vim9: Using #{ for a dictionary gives strange errors.
Solution: Give an error when using #{ for a comment after a command.
Files: src/vim9compile.c, src/vim9script.c, src/proto/vim9script.pro,
src/errors.h, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2621

Problem: typval2type() cannot handle recursive structures.
Solution: Use copyID. (closes #7979)
Files: src/list.c, src/vim9script.c, src/vim9type.c,
src/proto/vim9type.pro, src/testdir/test_vimscript.vim

Patch 8.2.2622

Problem: GTK: error when starting up and -geometry is given. (Dominique Pellé)
Solution: Use another function to get the monitor if the window has not been created yet. (closes #7978)
Files: src/gui_gtk_x11.c, src/proto/gui_gtk_x11.pro, src/gui_beval.c,
src/gui_xim.c

Patch 8.2.2623

Problem: Some tests fail when run as root.
Solution: Use CheckNotRoot.
Files: src/testdir/test_edit.vim, src/testdir/test_excmd.vim,
src/testdir/test_help.vim, src/testdir/test_writefile.vim

Patch 8.2.2624

Problem: Atom files not recognized.
Solution: Recognize .atom as XML. (Kivin Locke, closes #7986)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2625

Problem: Rss files not recognized.
Solution: Recognize .rss as XML. (Kivin Locke, closes #7987)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2626

Problem: GTK3: error when starting up and -geometry is given. (Dominique Pellé)
Solution: Use another function to get the monitor if the window has not been created yet. (closes #7978)
Files: src/gui_gtk_x11.c

Patch 8.2.2627

Problem: No need to check for BSD after checking for not root.
Solution: Remove CheckNotBSD. (Ozaki Kiichi, closes #7989)
Files: src/testdir/test_excmd.vim, src/testdir/test_help.vim, src/testdir/check.vim

Patch 8.2.2628

Problem: Vim9: #{ can still be used at the script level.
Solution: Give an error for #{ like in a :def function.
Files: src/eval.c, src/ex_docmd.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2629

Problem: Vim9: error for #{{ is not desired.
Solution: Adjust the checks. (closes #7990)
Files: src/errors.h, src/vim9script.c, src/ex_docmd.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2630

Problem: Hard to see where a test gets stuck.
Solution: Print the executed test function. (Dominique Pellé, closes #7975)
Files: src/testdir/Makefile

Patch 8.2.2631

Problem: Commands from winrestcmd() do not always work properly. (Leonid V. Fedorenchik)
Solution: Repeat the size commands twice. (closes #7988)
Files: src/evalwindow.c, src/testdir/test_window_cmd.vim

Patch 8.2.2632

Problem: Not all command line arguments are tested.
Solution: Add tests for -D and -serverlist. (Dominique Pellé, closes #7992)
Files: src/testdir/test_clientserver.vim, src/testdir/test_startup.vim

Patch 8.2.2633

Problem: Multi-byte 'fillchars' for folding do not show properly.
Solution: Handle multi-byte characters correctly. (Yegappan Lakshmanan, closes #7983, closes #7955)
Files: src/screen.c, src/testdir/test_fold.vim

Patch 8.2.2634

Problem: 'tagfunc' does not indicate using a pattern.

Solution: Add the "r" flag. (Andy Massimino, closes #7982)
Files: runtime/doc/tagsrch.txt, src/tag.c, src/testdir/test_tagfunc.vim

Patch 8.2.2635

Problem: Vim9: cannot define an inline function.
Solution: Make an inline function mostly work.
Files: src/userfunc.c, src/errors.h, src/vim9compile.c, src/misc2.c,
src/proto/vim9compile.pro, src/testdir/test_vim9_expr.vim

Patch 8.2.2636 (after 8.2.2635)

Problem: Memory leak when compiling inline function.
Solution: Free the prefetched line.
Files: src/userfunc.c, src/vim9compile.c, src/structs.h, src/globals.h,
src/eval.c

Patch 8.2.2637

Problem: prop_remove() causes a redraw even when nothing changed.
Solution: Only redraw if a property was removed. (Dominique Pellé)
Files: src/textprop.c

Patch 8.2.2638

Problem: Cannot write a message to the terminal from the GUI.
Solution: Add :echoconsole and use it in the test runner. (issue #7975)
Files: runtime/doc/eval.txt, runtime/doc/index.txt, src/ex_cmds.h,
src/ex_cmdidxs.h, src/eval.c, src/ui.c, src/proto/ui.pro,
src/term.c, src/testdir/runtest.vim

Patch 8.2.2639 (after 8.2.2638)

Problem: Build failure when fsync() is not available.
Solution: Add #ifdef.
Files: src/ui.c

Patch 8.2.2640

Problem: screenstring() returns non-existing composing characters.
Solution: Only use composing characters if there is a base character.
Files: src/evalfunc.c, src/testdir/test_listchars.vim

Patch 8.2.2641

Problem: Display test fails because of lacking redraw.
Solution: Add a redraw command.
Files: src/testdir/test_display.vim

Patch 8.2.2642

Problem: Vim9: no clear error for wrong inline function.
Solution: Check for something following the "{".
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2643

Problem: Various code not covered by tests.
Solution: Add a few more test. (Yegappan Lakshmanan, closes #7995)
Files: src/testdir/test_edit.vim, src/testdir/test_functions.vim,
src/testdir/test_mapping.vim, src/testdir/test_termcodes.vim,
src/testdir/test_undo.vim

Patch 8.2.2644

Problem: prop_clear() causes a screen update even when nothing changed.
Solution: Only redraw when a property was cleared. (Dominique Pellé)
Files: src/textprop.c

Patch 8.2.2645

Problem: Using inline function is not properly tested.
Solution: Add test cases, esp. for errors. Minor code improvements.
Files: src/userfunc.c, src/errors.h, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.2646

Problem: Vim9: error for not using string doesn't mention argument.
Solution: Add argument number.
Files: src/filepath.c, src/typval.c, src/proto/typval.pro, src/errors.h,
src/mbyte.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2647

Problem: Terminal test sometimes hangs.
Solution: Wait for the shell to display a prompt.
Files: src/testdir/test_terminal.vim

Patch 8.2.2648

Problem: Terminal resize test sometimes hangs.
Solution: Wait for the shell to display a prompt and other output.
Files: src/testdir/test_terminal2.vim

Patch 8.2.2649

Problem: Vim9: some wincmd arguments cause a white space error.
Solution: Insert a space before the count. (closes #8001)
Files: src/window.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2650

Problem: Vim9: command modifiers not handled in nested function.
Solution: Keep function-local info in a structure and save it on the stack.
Files: src/vim9execute.c, src/vim9.h, src/testdir/test_vim9_func.vim

Patch 8.2.2651

Problem: Vim9: restoring command modifiers happens after jump.
Solution: Move the restore instruction to before the jump. (closes #8006)
Also handle for and while.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2652

Problem: Vim9: can use command modifier without an effect.
Solution: Give an error for a misplaced command modifier. Fix error message
number.
Files: src/vim9compile.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/ex_eval.c, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2653

Problem: Build failure.
Solution: Add missing changes.
Files: src/errors.h

Patch 8.2.2654

Problem: Vim9: getting a character from a string can be slow.
Solution: Avoid a function call to get the character byte size. (#8000)
Files: src/vim9execute.vim

Patch 8.2.2655

Problem: The `-w` command line argument doesn't work.
Solution: Don't set `'window'` when set with the `-w` argument. (closes #8011)
Files: src/term.c, src/testdir/test_startup.vim

Patch 8.2.2656

Problem: Some command line arguments and regexp errors not tested.
Solution: Add a few test cases. (Dominique Pellé, closes #8013)
Files: src/testdir/test_regexp_latin.vim, src/testdir/test_startup.vim

Patch 8.2.2657

Problem: Vim9: error message for declaring variable in for loop.
Solution: Clear variables when entering block again. (closes #8012)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.2658

Problem: `:for` cannot loop over a string.
Solution: Accept a string argument and iterate over its characters.
Files: runtime/doc/eval.txt, src/eval.c, src/vim9compile.c,
src/vim9execute.c, src/errors.h, src/testdir/test_vimscript.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2659 (after 8.2.2658)

Problem: Eval test fails because for loop on string works.
Solution: Check looping over function reference fails.
Files: src/testdir/test_eval_stuff.vim

Patch 8.2.2660

Problem: Vim9: no error for declaration with trailing text.
Solution: Give an error. (closes #8014)
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2661

Problem: Leaking memory when looping over a string.
Solution: Free the memory.
Files: src/eval.c

Patch 8.2.2662

Problem: There is no way to avoid some escape sequences.
Solution: Suppress escape sequences when the `--not-a-term` argument is used.
(Gary Johnson)
Files: src/main.c, src/os_unix.c, src/testdir/test_startup.vim

Patch 8.2.2663

Problem: Vim9: leaking memory when inline function has an error.
Solution: Free the partially allocated function.
Files: src/userfunc.c

Patch 8.2.2664

Problem: Vim9: not enough function arguments checked for string.
Solution: Check in balloon functions. Refactor function arguments.
Files: src/typval.c, src/proto/typval.pro, src/filepath.c,
src/evalfunc.c, src/mbyte.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2665 (after 8.2.2664)

Problem: Test failures.
Solution: Check more specific feature. Add missing change.
Files: src/testdir/test_vim9_builtin.vim, src/evalbuffer.c

Patch 8.2.2666

Problem: Vim9: not enough function arguments checked for string.
Solution: Check in ch_logfile(), char2nr() and others.
Files: src/channel.c, src/evalfunc.c, src/filepath.c, src/eval.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2667

Problem: prop_find() cannot find item matching both id and type.
Solution: Add the "both" argument. (Naohiro Ono, closes #8019)
Files: runtime/doc/textprop.txt, src/testdir/test_textprop.vim,
src/textprop.c

Patch 8.2.2668

Problem: Vim9: omitting "call" for "confirm()" does not give an error.
Solution: Do not recognize a modifier followed by "(".
Files: src/ex_docmd.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2669

Problem: Command line completion does not work after "vim9".
Solution: Include the "9". (Naohiro Ono, closes #8025)
Files: src/cmdexpand.c, src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.2670

Problem: Vim9: error for append(0, text).
Solution: Check for negative number. (closes #8022)
Files: src/typval.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2671 (after 8.2.2670)

Problem: Error for line number in legacy script.
Solution: Check for number type.
Files: src/typval.c

Patch 8.2.2672

Problem: Vim9: cannot use :lockvar and :unlockvar in compiled script.
Solution: Implement locking support.
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.2673

Problem: Vim9: script-local funcref can have lower case name.

Solution: Require an upper case name.
Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2674

Problem: Motif: cancelling the font dialog resets the font.
Solution: When no font is selected to not change the font. (closes #7825, closes #8035) Fix compiler warnings.
Files: src/gui_x11.c, src/gui_motif.c

Patch 8.2.2675

Problem: Directory change in a terminal window shell is not followed.
Solution: Add the **'autoshelldir'** option. (closes #6290)
Files: runtime/doc/options.txt, runtime/doc/quickref.txt, runtime/optwin.vim, src/charset.c, src/feature.h, src/option.h, src/optiondefs.h, src/terminal.c, src/testdir/check.vim, src/testdir/test_terminal3.vim

Patch 8.2.2676

Problem: Missing error message.
Solution: Add new error message.
Files: src/errors.h

Patch 8.2.2677

Problem: Vim9: cannot use only some of the default arguments.
Solution: Use v:none to use default argument value. Remove uf_def_arg_idx[], use JUMP_IF_ARG_SET. (closes #6504)
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/vim9execute.c, src/userfunc.c, src/structs.h, src/vim9.h, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2678

Problem: Test for **'autoshelldir'** does not reset the option.
Solution: Reset the option after testing.
Files: src/testdir/test_terminal3.vim

Patch 8.2.2679

Problem: Winbar drawn over status line for non-current window with winbar if frame is zero height. (Leonid V. Fedorenchik)
Solution: Do not draw the window if the frame height is zero. (closes #8037)
Files: src/drawscreen.c, src/testdir/test_winbar.vim, src/testdir/dumps/Test_winbar_not_visible.dump

Patch 8.2.2680

Problem: Vim9: problem defining a script variable from legacy function.
Solution: Check if the script is Vim9, not the current syntax. (closes #8032)
Files: src/vim9script.c, src/proto/vim9script.pro, src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.2681

Problem: Vim9: test fails for redeclaring script variable.
Solution: It's OK to assign to an existing script variable in legacy.
Files: src/evalvars.c

Patch 8.2.2682

Problem: Vim9: cannot find Name.Func from "import * as Name". (Alexander Goussas)

Solution: When no variable found try finding a function. (closes #8045)
Check that the function was exported.

Files: src/vim9compile.c, src/vim9script.c,
src/testdir/test_vim9_script.vim

Patch 8.2.2683

Problem: Build failure without the +eval feature.

Solution: Add #ifdef.

Files: src/vim9script.c

Patch 8.2.2684

Problem: Not enough folding code is tested.

Solution: Add more test cases. (Yegappan Lakshmanan, closes #8046)

Files: src/testdir/test_fold.vim, src/testdir/test_mksession.vim,
src/testdir/test_source.vim

Patch 8.2.2685 (after 8.2.2152)

Problem: Custom statusline not drawn correctly with WinBar.

Solution: Also adjust the column for the custom status line. (Yee Cheng Chin, closes #8047)

Files: src/drawscreen.c, src/proto/drawscreen.pro, src/screen.c,
src/testdir/dumps/Test_winbar_not_visible_custom_statusline.dump,
src/testdir/test_winbar.vim

Patch 8.2.2686

Problem: Status line is not updated when going to cmdline mode.

Solution: Redraw status lines if '**statusline**' is set and going to status line mode. (based on patch from Justin M. Keyes et al., closes #8044)

Files: src/ex_getln.c, src/testdir/test_statusline.vim,
src/testdir/dumps/Test_statusline_mode_1.dump,
src/testdir/dumps/Test_statusline_mode_2.dump

Patch 8.2.2687

Problem: Vim9: cannot use "const" for global variable in :def function.

Solution: Do allow using :const for a global variable. (closes #8030)

Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.2688

Problem: Vim9: crash when using s: for script variable.

Solution: Pass the end pointer. (closes #8045)

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2689

Problem: Tiny build fails.

Solution: Add #ifdef around use of p_stl.

Files: src/ex_getln.c

Patch 8.2.2690

Problem: PowerShell files are not recognized.
Solution: Recognize several PowerShell extension. (Heath Stewart, closes #8051)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2691

Problem: Autoconf may mess up compiler flags.
Solution: Handle removing FORTIFY_SOURCE a bit better. (Vladimir Lomov, closes #8049)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2692

Problem: Vim9: locked script variable can be changed.
Solution: Check for locked value. (closes #8031)
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2693

Problem: Vim9: locked script variable can be changed.
Solution: Check legacy script variable for being locked. (issue #8031)
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2694

Problem: When **'matchpairs'** is empty every character beeps. (Marco Hinz)
Solution: Bail out when no character in **'matchpairs'** was found. (closes #8053) Add assert_nobeep().
Files: runtime/doc/testing.txt, runtime/doc/eval.txt, src/search.c, src/testing.c, src/proto/testing.pro, src/evalfunc.c, src/testdir/test_textformat.vim

Patch 8.2.2695

Problem: Cursor position reset with nested autocommands.
Solution: Only check and reset line numbers for not nested autocommands. (closes #5820)
Files: src/autocmd.c, src/testdir/test_terminal.vim

Patch 8.2.2696

Problem: Lua test fails with Lua 5.4.3 and later.
Solution: Check for different error messages. (Yegappan Lakshmanan, closes #8050)
Files: src/testdir/test_lua.vim

Patch 8.2.2697

Problem: Function list test fails.
Solution: Add missing function. (Yegappan Lakshmanan)
Files: runtime/doc/usr_41.txt

Patch 8.2.2698 (after 8.2.2696)

Problem: Lua test fails on MS-Windows.
Solution: Fall back to old method if "lua -v" doesn't work.
Files: src/testdir/test_lua.vim

Patch 8.2.2699

Problem: Lua test fails.
Solution: Fix condition. (Yegappan Lakshmanan, closes #8060)

Files: src/testdir/test_lua.vim

Patch 8.2.2700

Problem: Nested autocmd test fails sometimes.

Solution: Wait for the job to finish.

Files: src/testdir/test_terminal.vim

Patch 8.2.2701

Problem: Order of removing FORTIFY_SOURCE is wrong.

Solution: Use the more specific pattern first.

Files: src/configure.ac, src/auto/configure

Patch 8.2.2702

Problem: Compiler completion test fails when more scripts are added.

Solution: Add a more generic pattern.

Files: src/testdir/test_compiler.vim

Patch 8.2.2703

Problem: Vim9: memory leak when failing on locked variable.

Solution: Free the memory.

Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2704

Problem: Adding a lot of completions can be a bit slow.

Solution: Use fast_breakcheck() instead of ui_breakcheck() when adding a list of completions. (Ben Jackson, closes #8061)

Files: src/insexpand.c

Patch 8.2.2705

Problem: Vim9: misleading reported line number for wrong type.

Solution: Remember and use the line number at the start. (closes #8059)

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2706

Problem: Vim9: wrong line number reported for boolean operator.

Solution: Use the line number before skipping over line break. (closes #8058)

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2707 (after 8.2.2704)

Problem: Adding a lot of completions can still be a bit slow.

Solution: Add the check for CP_FAST. (Ben Jackson)

Files: src/insexpand.c

Patch 8.2.2708

Problem: Test sometimes fails waiting for shell in terminal.

Solution: Use WaitForAssert() so we can see the actual job status. Use Run_shell_in_terminal().

Files: src/testdir/term_util.vim, src/testdir/test_mksession.vim

Patch 8.2.2709

Problem: The GTK GUI has a gap next to the scrollbar.

Solution: Calculate the scrollbar padding for GTK. (closes #8027)

Files: src/gui_gtk.c

Patch 8.2.2710

Problem: Vim9: not all tests cover script and :def function.
Solution: Run tests in both if possible. Fix differences.
Files: src/eval.c, src/vim9compile.c, src/vim9execute.c,
src/testdir/vim9.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.2711

Problem: "gj" in a closed fold does not move out of the fold. (Marco Hinz)
Solution: Add a check for being in a closed fold. (closes #8062)
Files: src/normal.c, src/testdir/test_fold.vim

Patch 8.2.2712

Problem: Memory leak when adding to a blob fails.
Solution: Clear the second typval before returning.
Files: src/eval.c

Patch 8.2.2713

Problem: Folding code not sufficiently tested.
Solution: Add a few more test cases. (Yegappan Lakshmanan, closes #8064)
Files: src/testdir/test_fold.vim

Patch 8.2.2714

Problem: Filetype pattern ending in star is too far up.
Solution: Move down to where patterns ending in star belong. (closes #8065)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2715

Problem: Vim9: tests fail without the channel feature. (Dominique Pellé)
Solution: Check for the channel feature. (closes #8063)
Files: src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.2716

Problem: The equivalent class regexp is missing some characters.
Solution: Update the list of equivalent characters. (Dominique Pellé,
closes #8029)
Files: src/regexp_bt.c, src/regexp_nfa.c,
src/testdir/test_regexp_utf8.vim

Patch 8.2.2717

Problem: GTK menu items don't show a tooltip.
Solution: Add a callback to show the tooltip. (Leonid V. Fedorenchik,
closes #8067, closes #7810)
Files: src/gui_gtk.c

Patch 8.2.2718

Problem: Vim9: no explicit test for using a global function without the g:
prefix.
Solution: Add a test case.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.2719

Problem: Vim9: appending to dict item doesn't work in a :def function.
Solution: Implement assignment with operator on indexed item.

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2720

Problem: GTK menu tooltip moves the cursor.

Solution: Position the cursor after displaying the tooltip. Do not show the tooltip when editing the command line.

Files: src/gui_gtk.c

Patch 8.2.2721

Problem: Vim9: cannot have a linebreak inside a lambda.

Solution: Compile the expression before the arguments.

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2722

Problem: Vim9: crash when using LHS with double index.

Solution: Handle lhs_dest which is "dest_expr". (closes #8068)
Fix confusing error message for missing dict item.

Files: src/vim9compile.c, src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2723 (after 8.2.2722)

Problem: Assignment test fails.

Solution: Adjust error number.

Files: src/testdir/test_let.vim

Patch 8.2.2724 (after 8.2.2722)

Problem: Vim9: concatenating to list in dict not tested.

Solution: Add a test. (issue #8068)

Files: src/testdir/test_vim9_assign.vim

Patch 8.2.2725

Problem: Vim9: message about compiling is wrong when using try/catch.

Solution: Store the compiling flag with the message. (closes #8071)

Files: src/ex_docmd.c, src/ex_eval.c, src/structs.h,
src/testdir/test_vim9_func.vim

Patch 8.2.2726

Problem: Confusing error message with white space before comma in the arguments of a function declaration.

Solution: Give a specific error message. (closes #2235)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2727 (after 8.2.2726)

Problem: Function test fails.

Solution: Adjust expected error number.

Files: src/testdir/test_user_func.vim

Patch 8.2.2728

Problem: Special key names don't work if 'isident' is cleared.

Solution: Add vim_isNormalIDc() and use it for special key names.
(closes #2389)

Files: src/charset.c, src/proto/charset.pro, src/misc2.c,
src/testdir/test_mapping.vim

Patch 8.2.2729

Problem: Vim9: wrong error message for referring to legacy script variable.
Solution: Do allow referring to a variable in legacy script without "s:" if it exists at compile time. (closes #8076)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2730

Problem: Coverity complains about not restoring character.
Solution: Also restore the character in case of an error.
Files: src/vim9compile.c

Patch 8.2.2731

Problem: Mac: SF symbols are not displayed properly.
Solution: Add custom range to list of double-width characters. (Yee Cheng Chin, closes #8077)
Files: src/mbyte.c

Patch 8.2.2732

Problem: Prompt for s///c in Ex mode can be wrong.
Solution: Position the cursor before showing the prompt. (closes #8073)
Files: src/ex_cmds.c, src/testdir/test_ex_mode.vim

Patch 8.2.2733

Problem: Detecting Lua version is not reliable.
Solution: Add "vim.lua_version". (Ozaki Kiichi, closes #8080)
Files: runtime/doc/if_lua.txt, ci/if_ver-1.vim, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.2734

Problem: Vim9: cannot use legacy script-local var from :def function.
Solution: Do not insist on using "s:" prefix. (closes #8076)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/testdir/test_vim9_expr.vim

Patch 8.2.2735

Problem: Vim9: function reference found with prefix, not without.
Solution: Also find function reference without prefix.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2736

Problem: Vim9: for loop over string is a bit slow.
Solution: Avoid using strlen().
Files: src/vim9execute.c

Patch 8.2.2737

Problem: Status line not updated when local **'statusline'** option set.
Solution: Check the **'statusline'** option of each window.
Files: src/ex_getln.c, src/testdir/test_statusline.vim, src/testdir/dumps/Test_statusline_mode_1.dump, src/testdir/dumps/Test_statusline_mode_2.dump

Patch 8.2.2738

Problem: Extending a list with itself can give wrong result.
Solution: Remember the item before where the insertion happens and skip to after the already inserted items. (closes #1112)

Files: src/list.c, src/testdir/test_listdict.vim

Patch 8.2.2739

Problem: Vim9: a lambda accepts too many arguments at the script level.

Solution: Do not set uf_varargs in Vim9 script.

Files: src/userfunc.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2740

Problem: Vim9: lambda with varargs doesn't work.

Solution: Make "...name" work. Require type to be a list.

Files: src/userfunc.c, src/vim9compile.c, src/vim9execute.c,
src/errors.h, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2741

Problem: Vim9: Partial call does not check right arguments.

Solution: Adjust the offset for whether the partial is before or after the arguments. (closes #8091)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2742

Problem: Vim9: when compiling a function fails it is cleared.

Solution: Keep the function lines, prevent execution with a different status. (closes #8093)

Files: src/vim9compile.c, src/structs.h, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2743

Problem: Vim9: function state stuck when compiling with ":silent!".

Solution: Check for uf_def_status to be UF_COMPILING.

Files: src/vim9compile.c, src/message.c, src/globals.h,
src/testdir/test_vim9_func.vim

Patch 8.2.2744

Problem: Vim9: no way to explicitly ignore an argument.

Solution: Use the underscore as the name for an ignored argument.

Files: runtime/doc/vim9.txt, src/vim9compile.c, src/eval.c,
src/evalvars.c, src/errors.h, src/testdir/test_vim9_func.vim

Patch 8.2.2745 (after 8.2.2744)

Problem: Vim9: missing part of the argument change.

Solution: Add missing changes.

Files: src/userfunc.c

Patch 8.2.2746 (after 8.2.2745)

Problem: Check for duplicate arguments does not work.

Solution: Correct condition.

Files: src/userfunc.c

Patch 8.2.2747

Problem: Vim9: not always an error for too many function arguments.

Solution: Check for getting too many arguments.

Files: src/vim9execute.c, src/testdir/test_vim9_func.vim,

src/testdir/test_vim9_builtin.vim

Patch 8.2.2748

Problem: Vim9: memory leak when calling :def function fails.
Solution: Jump to failed_early instead of returning.
Files: src/vim9execute.c

Patch 8.2.2749

Problem: Vim9: test for error can be a bit flaky.
Solution: Increase the wait time a bit.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.2750

Problem: Vim9: error for using underscore in nested function.
Solution: Do not consider "_" already defined. (closes #8096)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2751

Problem: Coverity warns for using NULL pointer.
Solution: Check for NULL in calling function.
Files: src/userfunc.c

Patch 8.2.2752

problem: coverity reports unreachable code.
Solution: Remove check for positive index.
Files: src/typval.c

Patch 8.2.2753

Problem: Vim9: cannot ignore an item in assignment unpack.
Solution: Allow using an underscore.
Files: runtime/doc/vim9.txt, src/vim.h, src/evalvars.c, src/eval.c,
src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2754

Problem: :sleep! does not always hide the cursor.
Solution: Add the cursor_is_asleep flag. (Jeremy Lerner, closes #8097,
closes #7998)
Files: src/drawscreen.c, src/ex_docmd.c, src/gui.c, src/proto/term.pro,
src/term.c

Patch 8.2.2755

Problem: Vim9: no error for using a number in a condition.
Solution: Also use ISN_COND2BOOL if the type is t_number_bool.
(closes #7644)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2756

Problem: Vim9: blob index and slice not implemented yet.
Solution: Implement blob index and slice.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/eval.c,
src/blob.c, src/proto/blob.pro, src/testdir/test_vim9_expr.vim

Patch 8.2.2757

Problem: Vim9: blob tests for legacy and Vim9 script are separate.
Solution: Add CheckLegacyAndVim9Success(). Make blob index assign work.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/errors.h,
src/blob.c, src/proto/blob.pro, src/eval.c, src/ex_docmd.c,
src/testdir/vim9.vim, src/testdir/test_blob.vim

Patch 8.2.2758

Problem: Vim9: wrong line number for autoload function with wrong name.
Solution: Set and restore SOURCING_LNUM. (closes #8100)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2759

Problem: Vim9: for loop infers type of loop variable.
Solution: Do not get the member type. (closes #8102)
Files: src/vim9type.c, src/proto/vim9type.pro, src/list.c,
src/vim9script.c, src/proto/vim9script.pro, src/vim.h,
src/testdir/test_vim9_script.vim

Patch 8.2.2760

Problem: Vim9: no error for changing a for loop variable.
Solution: Make the loop variable read-only. (issue #8102)
Files: src/eval.c, src/evalvars.c, src/vim9compile.c, src/vim.h,
src/testdir/test_vim9_script.vim

Patch 8.2.2761

Problem: Using "syn include" does not work properly.
Solution: Don't add current_syn_inc_tag to topgrp. (Jaehwang Jerry Jung,
closes #8104)
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.2762

Problem: Vim9: function line truncated when compiling.
Solution: Copy the line before processing it. (closes #8101)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2763

Problem: Vim9: cannot use type in for loop unpack at script level.
Solution: Advance over the type name.
Files: src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.2764

Problem: Memory leak when default function argument is allocated.
Solution: Free the expression result.
Files: src/userfunc.c, src/testdir/test_functions.vim

Patch 8.2.2765

Problem: Vim9: not all blob operations work.
Solution: Run more tests also with Vim9 script and :def functions. Fix what
doesn't work.
Files: src/eval.c, src/blob.c, src/proto/blob.pro, src/vim9execute.c,
src/errors.h, src/testdir/vim9.vim, src/testdir/test_blob.vim

Patch 8.2.2766 (after 8.2.2765)

Problem: Test failure.

Solution: Add change to Vim9 compilation error message.
Files: src/vim9compile.c

Patch 8.2.2767 (after 8.2.2765)

Problem: Compiler warning for unused argument.
Solution: Remove the argument.
Files: src/blob.c, src/proto/blob.pro, src/vim9execute.c, src/eval.c

Patch 8.2.2768

Problem: Vim9: memory leak with blob range error.
Solution: Jump to end instead of returning.
Files: src/vim9compile.c

Patch 8.2.2769

Problem: Modula-3 config files are not recognized.
Solution: Add filetype patterns. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2770

Problem: Vim9: type of loop variable is not used.
Solution: Parse and check the variable type. (closes #8107)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2771

Problem: Vim9: assignment not recognized if declaration was skipped.
Solution: Also recognized an assignment if the variable does not exist.
(closes #8108)
Files: src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2772

Problem: Problems when restoring **'runtimepath'** from a session file.
Solution: Add the "skiprtp" item in **'sessionoptions'**.
Files: runtime/doc/options.txt, src/session.c, src/optionstr.c,
src/option.h, src/vim.h, src/option.c,
src/testdir/test_mksession.vim

Patch 8.2.2773

Problem: PSL filetype not recognized.
Solution: Add a filetype pattern. (Daniel Kho, closes #8117)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2774

Problem: Vim9: cannot import an existing name even when using "as".
Solution: Do not check for an existing name when using "as". (closes #8113)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.2775

Problem: Vim9: wrong line number used for some commands.
Solution: For :exe, :echo and the like use the line number of the start of
the command. When calling a function set the line number in the
script context.
Files: src/vim9compile.c, src/vim9execute.c, src/structs.h,
src/testdir/test_vim9_script.vim

Patch 8.2.2776

Problem: :mksession uses current value of **'splitbelow'** and **'splitright'** even though "options" is not in **'sessionoptions'**. (Maxim Kim)
Solution: Save and restore the values, instead of setting to the current value. (closes #8119)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.2777

Problem: Vim9: blob operations not tested in all ways.
Solution: Run tests with CheckLegacyAndVim9Success(). Make blob assign with index work.
Files: src/vim9compile.c, src/vim9execute.c, src/errors.h, src/blob.c, src/proto/blob.pro, src/testdir/test_blob.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2778

Problem: Problem restoring **'packpath'** in session.
Solution: Let "skiprtp" also apply to **'packpath'**.
Files: runtime/doc/options.txt, src/option.c, src/testdir/test_mksession.vim

Patch 8.2.2779

Problem: Memory access error in remove() for blob.
Solution: Adjust length for memmove().
Files: src/blob.c

Patch 8.2.2780

Problem: Vim9: for loop over blob doesn't work.
Solution: Make it work.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_blob.vim

Patch 8.2.2781

Problem: Add() silently skips when adding to null list or blob.
Solution: Give an error in Vim9 script. Allocate blob when it is NULL like with list and dict.
Files: src/list.c, src/evalvars.c, src/vim9execute.c, src/testdir/test_blob.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2782

Problem: Vim9: blob operations not fully tested.
Solution: Make more blob tests run in Vim9 script. Fix filter(). Make insert() give an error for a null blob, like add().
Files: src/list.c, src/testdir/test_blob.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.2783

Problem: Duplicate code for setting byte in blob, blob test may fail.
Solution: Call blob_set_append(). Test sort failure with "N".
Files: src/eval.c, src/testdir/test_blob.vim

Patch 8.2.2784

Problem: Vim9: cannot use \=expr in :substitute.
Solution: Compile the expression into instructions and execute them when invoked.

Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/proto/vim9execute.pro, src/regexp.c, src/ex_cmds.c,
src/proto/ex_cmds.pro, src/globals.h,
src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2785

Problem: Vim9: cannot redirect to local variable.
Solution: Compile :redir when redirecting to a variable.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/errors.h,
src/evalvars.c, src/proto/evalvars.pro,
src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.2786

Problem: Vim9: memory leak when using :s with expression.
Solution: Clean up the instruction list.
Files: src/vim9compile.c

Patch 8.2.2787

Problem: MS-Windows: crash when using :echoconsole.
Solution: Do not write a NUL when it's already there.
Files: src/os_win32.c

Patch 8.2.2788

Problem: Raku is now the only name what once was called perl6.
Solution: Adjust the filetype detection. (closes #8120)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2789

Problem: Vim9: using \=expr in :substitute does not handle jumps.
Solution: Start with instruction count zero. (closes #8128)
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2790 (after 8.2.2788)

Problem: filetype test fails
Solution: Also update the scripts detection
Files: runtime/scripts.vim

Patch 8.2.2791

Problem: Vim9: memory leak when using \=expr in :substitute.
Solution: Do not allocate a new instruction list.
Files: src/vim9compile.c

Patch 8.2.2792

Problem: Vim9: :disas shows instructions for default args but no text.
Solution: Show the expression test above the default argument instructions.
(closes #8129)
Files: src/vim9execute.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2793

Problem: MS-Windows: string literals are writable with MSVC.
Solution: Add the /GF compiler flag. Make mch_write() safer. (Ken Takata,
closes #8133)

Files: src/Make_mvc.mak, src/os_win32.c

Patch 8.2.2794

Problem: Linux users don't know how to get ncurses.
Solution: Add the name of the package. (closes #8132)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2795

Problem: Coverity warns for not using return value.
Solution: Check the return value of compiling the substitute expression.
Files: src/vim9compile.c

Patch 8.2.2796

Problem: Vim9: redir to variable does not accept an index.
Solution: Make the index work.
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2797

Problem: Search highlight disappears in the Visual area.
Solution: Combine the search attributes. (closes #8134)
Files: src/drawline.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_hlsearch_visual_1.dump

Patch 8.2.2798

Problem: Vim9: redir to variable with append does not accept an index.
Solution: Make the appending work.
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2799

Problem: Vim9: type casts don't fully work at the script level.
Solution: Implement the missing piece.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2800

Problem: After a timer displays text a hit-enter prompt is given.
Solution: Reset msg_didany and need_wait_return. (closes #8136)
Files: src/drawscreen.c, src/testdir/test_timers.vim

Patch 8.2.2801

Problem: Free Pascal makefile not recognized.
Solution: Add the fpcmake filetype. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2802

Problem: Vim9: illegal memory access.
Solution: Check for comment before checking for white space. (closes #8142)
Files: src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.2803

Problem: Flicker when the popup menu has an info popup.
Solution: Avoid drawing over the popup when it's going to be redrawn in the same position. (closes #8131) Also avoid redrawing the scrollbar.
Files: src/popupmenu.c, src/proto/popupmenu.pro, src/drawscreen.c,
src/globals.h

Patch 8.2.2804

Problem: Setting buffer local mapping with mapset() changes global mapping.
Solution: Only set the local mapping. (closes #8143)
Files: src/map.c, src/testdir/test_maparg.vim

Patch 8.2.2805

Problem: Vim9: cannot use legacy syntax in Vim9 script.
Solution: Add the :legacy command.
Files: src/ex_cmds.h, runtime/doc/vim9.txt, runtime/doc/index.txt
src/ex_cmdidxs.h, src/ex_docmd.c, src/structs.h, src/vim9script.c,
src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.2806

Problem: Vim9: using "++nr" as a command might not work.
Solution: Do not recognize "++" and "--" in a following line as addition or subtraction.
Files: src/vim9compile.c, src/ex_docmd.c, src/ex_cmds.h, src/ex_cmdidxs.h,
src/vim9script.c, src/proto/vim9script.pro, src/eval.c,
src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.2807

Problem: Build fails with tiny features.
Solution: Use a dummy function for ex_incdec().
Files: src/ex_docmd.c

Patch 8.2.2808

Problem: Vim9: increment and decrement not sufficiently tested.
Solution: Add assertions.
Files: src/testdir/test_vim9_assign.vim

Patch 8.2.2809

Problem: Vim9: :def function compilation fails when using :legacy.
Solution: Reset CMOD_LEGACY when compiling a function. (closes #8137)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2810

Problem: Vim9: crash when calling a function in a substitute expression.
Solution: Set the instructions back to the substitute expression instructions. (closes #8148)
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2811

Problem: Vim9: error for missing white space doesn't say where it is missing
Solution: Mention the command. (closes #8149)
Files: src/ex_docmd.c, src/errors.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.2812

Problem: Vim9: still crash when using substitute expression.
Solution: Put the instruction list in the stack frame. (closes #8154)
Files: src/vim9execute.c, src/vim9.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.2813

Problem: Cannot grep using fuzzy matching.
Solution: Add the "f" flag to :vimgrep. (Yegappan Lakshmanan, closes #8152)
Files: runtime/doc/quickfix.txt, src/ex_cmds.c, src/proto/search.pro,
src/quickfix.c, src/search.c, src/vim.h,
src/testdir/test_quickfix.vim

Patch 8.2.2814 (after 8.2.2812)

Problem: Vim9: unused variable. (John Marriott)
Solution: Adjust #ifdef.
Files: src/vim9execute.c

Patch 8.2.2815

Problem: Status line flickers when redrawing popup menu info.
Solution: Do not redraw the status line when the focus is in the popup
window. (issue #8144)
Files: src/popupmenu.c

Patch 8.2.2816

Problem: Vim9: comment below expression in lambda causes problems.
Solution: Use a single space for empty and comment lines. (closes #8156)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2817

Problem: Vim9: script sourcing continues after an error.
Solution: Make an error in any command in "vim9script" abort sourcing.
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.2818

Problem: No jump added to jumplist when opening terminal in current window.
Solution: Call setpcmark(). (closes #8158)
Files: src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.2819

Problem: Finishing an abbreviation with a multi-byte char may not work.
Solution: Escape K_SPECIAL in the typed character. (closes #8160)
Files: src/map.c, src/testdir/test_mapping.vim

Patch 8.2.2820

Problem: Session file may divide by zero.
Solution: Avoid writing divide by zero. (closes #8162)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.2821

Problem: MS-Windows: unnecessarily loading libraries when registering OLE.
Solution: Skip loading libraries when invoked with "--register".
Files: src/main.c, src/globals.h, src/os_win32.c

Patch 8.2.2822 (after 8.2.2821)

Problem: MS-Windows: unnecessarily loading libraries when unregistering OLE.
Solution: Also skip loading libraries when invoked with "--unregister". Run
Vim for README.txt with user privileges.

Files: src/main.c, nsis/gvim.nsi, nsis/README.txt

Patch 8.2.2823

Problem: MS-Windows: launching Vim from installer doesn't open README.

Solution: Adjust the quotes.

Files: nsis/gvim.nsi

Patch 8.2.2824

Problem: MS-Windows: build failure with MSVC.

Solution: Adjust the list of distributed files. Add hint about python.
Adjust path for reading runtime files.

Files: Filelist, src/testdir/shared.vim,
src/testdir/test_function_lists.vim

Patch 8.2.2825

Problem: Code in checkreadonly() not fully tested.

Solution: Add more tests. (Dominique Pellé, closes #8169)

Files: src/testdir/test_excmod.vim

Patch 8.2.2826

Problem: Compiler warnings for int to size_t conversion. (Randall W. Morris)

Solution: Add type casts.

Files: src/map.c, src/quickfix.c

Patch 8.2.2827

Problem: Test file was not deleted.

Solution: Uncomment the delete() call. (Dominique Pellé, closes #8172)

Files: src/testdir/test_mksession.vim

Patch 8.2.2828

Problem: Coverity complains about not checking the rename() return value.

Solution: Add "(void)", can't do anything in case of a failure.

Files: src/fileio.c

Patch 8.2.2829

Problem: Some comments are not correct or clear.

Solution: Adjust the comments. Add test for cursor position.

Files: src/regexp_bt.c, src/regexp_nfa.c,
src/testdir/test_exec_while_if.vim,
src/testdir/test_substitute.vim

Patch 8.2.2830

Problem: Terminal colors are not updated when **'background'** is set.

Solution: Call term_update_colors() for all terminals. (Marcin Szamotulski, closes #8171, closes #8150)

Files: src/terminal.c, src/proto/terminal.pro, src/optionstr.c

Patch 8.2.2831

Problem: Vim9: expandcmd() not tested.

Solution: Add a test.

Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.2832

Problem: Operator cancelled by moving mouse when using popup. (Sergey Vlasov)
Solution: Do not trigger an operator for a mouse move events. (closes #8176)
Files: src/normal.c

Patch 8.2.2833

Problem: Two key command cancelled by moving mouse when using popup. (Sergey Vlasov)
Solution: Ignore K_MOUSEMOVE in plain_vgetc().
Files: src/getchar.c

Patch 8.2.2834

Problem: Vim9: :cexpr does not work with local variables.
Solution: Compile :cexpr.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/quickfix.c, src/proto/quickfix.pro, src/testdir/test_quickfix.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2835 (after 8.2.2834)

Problem: Vim9: leaking memory in :cexpr.
Solution: Also free the command line copy.
Files: src/vim9compile.c

Patch 8.2.2836 (after 8.2.2834)

Problem: Build failure without the +quickfix feature. (John Marriott)
Solution: Add #ifdef.
Files: src/vim9compile.c, src/vim9execute.c, src/tag.c

Patch 8.2.2837

Problem: Various code lines not covered by tests.
Solution: Add test cases. (Dominique Pellé, closes #8178)
Files: src/testdir/test_excmd.vim, src/testdir/test_functions.vim, src/testdir/test_options.vim, src/testdir/test_startup.vim, src/testdir/test_syntax.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.2838

Problem: File extension .wrap not recognized.
Solution: Use dosini filetype for .wrap files. (Liam Beguin, closes #8177)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2839

Problem: Default redirection missing "ash" and "dash".
Solution: Recognize "ash" and "dash". (Natanael Copa, closes #8180)
Files: runtime/doc/options.txt, src/option.c

Patch 8.2.2840

Problem: Vim9: member operation not fully tested.
Solution: Add a few tests.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2841

Problem: MS-Windows: cursor in wrong position when '**lazyredraw**' and '**statusline**' are set.
Solution: Call compute_cmdrow(). (closes #8170, closes #8184)

Files: src/os_win32.c

Patch 8.2.2842

Problem: Vim9: skip argument to searchpair() is not compiled.

Solution: Add VAR_INSTR.

Files: src/structs.h, src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/proto/vim9execute.pro, src/eval.c, src/evalfunc.c, src/vim.h,
src/evalvars.c, src/typval.c, src/vim9type.c, src/testing.c,
src/viminfo.c, src/if_py_both.h, src/json.c,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2843 (after 8.2.2842)

Problem: Vim9: skip argument to searchpairpos() is not compiled.

Solution: Handle like searchpair(). Also for search() and searchpos().

Files: src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2844

Problem: Vim9: memory leak when using searchpair().

Solution: Free the v_instr field.

Files: src/typval.c

Patch 8.2.2845

Problem: MS-Windows: warning for signed/unsigned comparison.

Solution: Add type cast.

Files: src/terminal.c

Patch 8.2.2846

Problem: Vim9: "echo Func()" does not give an error for a function without a return value.

Solution: Give an error. Be more specific about why a value is invalid.

Files: src/globals.h, src/errors.h, src/eval.c, src/evalfunc.c,
src/typval.c, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.2847

Problem: Perl not tested sufficiently.

Solution: Add test. Also test W17. (Dominique Pellé, closes #8193)

Files: src/testdir/test_arabic.vim, src/testdir/test_perl.vim

Patch 8.2.2848

Problem: Crash when calling partial.

Solution: Check for NULL pointer. (Dominique Pellé, closes #8202)

Files: src/eval.c, src/evalfunc.c, src/testdir/test_functions.vim,
src/testdir/test_listdict.vim

Patch 8.2.2849

Problem: Bufwrite not sufficiently tested.

Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8192)

Files: src/testdir/test_startup.vim, src/testdir/test_writefile.vim

Patch 8.2.2850

Problem: Recalling commands from history is not tested.

Solution: Add tests. (closes #8194)

Files: src/testdir/test_cmdline.vim

Patch 8.2.2851

Problem: Using <Cmd> mapping on the command line triggers CmdlineChanged.
(Naohiro Ono)

Solution: Jump to cmdline_not_changed if the command line didn't change.
(closes #8208)

Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.2852

Problem: Configure can add --as-needed a second time.

Solution: Only add --as-needed if not already there. (Natanael Copa,
closes #8189, closes #8181)

Files: src/configure.ac, src/auto/configure

Patch 8.2.2853 (after 8.2.2851)

Problem: Window is not updated after using <Cmd> mapping.

Solution: So jump to cmdline_changed but skip autocommand.

Files: src/ex_getln.c

Patch 8.2.2854

Problem: Custom statusline cannot contain % items.

Solution: Add "%{ % expr %}". (closes #8190)

Files: runtime/doc/options.txt, src/buffer.c, src/optionstr.c,
src/testdir/test_statusline.vim

Patch 8.2.2855

Problem: White space after "->" does not give E274.

Solution: Do not skip white space in legacy script. (closes #8212)

Files: src/eval.c, src/testdir/test_method.vim

Patch 8.2.2856

Problem: Get readonly error for device that can't be written to.

Solution: Check for being able to write first. (closes #8205)

Files: src/ex_cmds.c, src/testdir/test_writefile.vim

Patch 8.2.2857

Problem: Vim9: exception in ISN_INSTR caught at wrong level.

Solution: Set the starting trylevel in exec_instructions(). (closes #8214)

Files: src/vim9compile.c, src/vim9execute.c, src/globals.h,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2858 (after 8.2.2857)

Problem: Test fails because of changed error message.

Solution: Adjust the expected error message.

Files: src/testdir/test_ex_mode.vim

Patch 8.2.2859 (after 8.2.2857)

Problem: Tcl test fails because of changed error message.

Solution: Adjust the expected error message.

Files: src/testdir/test_tcl.vim

Patch 8.2.2860

Problem: Adding a text property causes the whole window to be redrawn.

Solution: Use `changed_lines_buf()` to only redraw the affected lines.
Files: `src/change.c`, `src/proto/change.pro`, `src/textprop.c`

Patch 8.2.2861

Problem: Vim9: "legacy return" is not recognized as a return statement.
Solution: Specifically check for a return command. (closes #8213)
Files: `src/vim9compile.c`, `src/vim9execute.c`, `src/vim9.h`,
`src/testdir/test_vim9_expr.vim`

Patch 8.2.2862

Problem: Removing a text property causes the whole window to be redrawn.
Solution: Use `changed_lines_buf()` to only redraw the affected lines.
Files: `src/textprop.c`

Patch 8.2.2863 (after 8.2.2862)

Problem: Removing a text property does not redraw optimally.
Solution: Only redraw the lines that mighth actually have been changed.
Files: `src/textprop.c`

Patch 8.2.2864

Problem: Vim9: crash when using inline function.
Solution: Check for NULL pointer. Make using inline function work inside
`lambda`. (closes #8217)
Files: `src/userfunc.c`, `src/testdir/test_vim9_func.vim`

Patch 8.2.2865 (after 8.2.2864)

Problem: Skipping over function body fails.
Solution: Do not define the function when skipping.
Files: `src/userfunc.c`

Patch 8.2.2866

Problem: Vim9: memory leak when using inline function.
Solution: Remember what strings to free.
Files: `src/userfunc.c`, `src/structs.h`, `src/eval.c`

Patch 8.2.2867 (after 8.2.2866)

Problem: Build failure.
Solution: Add missing part of the change.
Files: `src/globals.h`

Patch 8.2.2868

Problem: Vim9: When executing a compiled expression the `trylevel` at start
is changed but not restored. (closes #8214)
Solution: Restore the `trylevel` at start.
Files: `src/vim9execute.c`, `src/testdir/test_vim9_builtin.vim`

Patch 8.2.2869

Problem: Using unified diff is not tested.
Solution: Test all cases also with unified diff. (issue #8197)
Files: `src/testdir/test_diffmode.vim`

Patch 8.2.2870

Problem: `CmdlineChange` event triggered twice for **CTRL-R**.
Solution: Return `CMDLINE_NOT_CHANGED` from `cmdline_insert_reg()`.

(closes #8219)
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.2871

Problem: Unnecessary VIM_ISDIGIT() calls, badly indented code.
Solution: Call skipdigits() on the next character. Improve indenting.
(Dominique Pellé, closes #8227)
Files: src/charset.c, src/evalfunc.c, src/ex_docmd.c, src/json.c,
src/ops.c, src/tag.c, src/vim9compile.c

Patch 8.2.2872

Problem: Python tests fail without the channel feature.
Solution: Add a feature check. (Dominique Pellé, closes #8226)
Files: src/testdir/test_python2.vim, src/testdir/test_python3.vim

Patch 8.2.2873

Problem: Not enough tests for writing buffers.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8229)
Files: src/testdir/test_buffer.vim, src/testdir/test_cmdline.vim,
src/testdir/test_functions.vim, src/testdir/test_writefile.vim

Patch 8.2.2874

Problem: MS-Windows: screen redraws too often.
Solution: Do not redraw when peeking for a character. (closes #8230,
closes #8211)
Files: src/os_win32.c

Patch 8.2.2875

Problem: Cancelling inputlist() after a digit does not return zero.
Solution: Always return zero when cancelling. (closes #8231)
Files: src/misc1.c, src/testdir/test_functions.vim

Patch 8.2.2876

Problem: Configure cannot detect Python 3.10.
Solution: Use sys.version_info. (closes #8233)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2877

Problem: Insufficient tests for popup menu rightleft.
Solution: Add tests. (Yegappan Lakshmanan, closes #8235)
Files: src/testdir/test_popup.vim,
src/testdir/dumps/Test_pum_rightleft_01.dump,
src/testdir/dumps/Test_pum_rightleft_02.dump,
src/testdir/dumps/Test_pum_scrollbar_01.dump,
src/testdir/dumps/Test_pum_scrollbar_02.dump

Patch 8.2.2878

Problem: Vim9: for loop list unpack only allows for one "_".
Solution: Drop the value when the variable is "_". (closes #8232)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2879

Problem: File extension .hsig not recognized.
Solution: Use Haskell filetype for .hsig files. (Marcin Szamotulski,

closes #8236)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2880

Problem: Unified diff fails if actually used.
Solution: Invoke :diffupdate in the test. Fix the check for working external diff. (Ghjuvan Lacambre, Christian Brabandt, closes #8197)
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.2.2881

Problem: Various pieces of code not covered by tests.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8245)
Files: src/testdir/test_const.vim, src/testdir/test_functions.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim, src/testdir/test_user_func.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2882

Problem: Vim9: memory leak when lambda has an error.
Solution: Free the list of argument types on failure.
Files: src/userfunc.c

Patch 8.2.2883

Problem: MS-Windows manifest file name is misleading.
Solution: Rename the file. (closes #8241)
Files: .gitignore, .hgignore, Filelist, Makefile, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/gvim.exe.mnf, src/vim.manifest, src/vim.rc

Patch 8.2.2884

Problem: Not enough cscope code is covered by tests.
Solution: Add a few test cases. (Dominique Pellé, closes #8246)
Files: src/testdir/test_cscope.vim

Patch 8.2.2885

Problem: searching for \%'\> does not match linewise end of line. (Tim Chase)
Solution: Match end of line if column is MAXCOL. (closes #8238)
Files: src/regexp_nfa.c, src/regexp_bt.c, src/testdir/test_search.vim

Patch 8.2.2886

Problem: Various pieces of code not covered by tests.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8255)
Files: src/testdir/test_expr.vim, src/testdir/test_functions.vim, src/testdir/test_listdict.vim, src/testdir/test_registers.vim, src/testdir/test_user_func.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.2887

Problem: Crash when passing null string to fullcommand().
Solution: Check for NULL pointer. (closes #8256)
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.2888

Problem: Vim9: "k" command recognized in Vim9 script.
Solution: Do not recognize "k" or "s" and "d" with flags.

Files: src/ex_docmd.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.2889

Problem: Typo and verbose comment in Makefiles.
Solution: Fix typo. Use @#. (Ken Takata, closes #8252)
Files: Makefile, src/testdir/Makefile

Patch 8.2.2890

Problem: Text property duplicated when data block splits.
Solution: Do not continue text prop from previous line. (closes #8261)
Files: src/memline.c, src/structs.h, src/testdir/test_textprop.vim

Patch 8.2.2891

Problem: Cannot build with Perl 5.34.
Solution: Add Perl_SvTRUE_common(). (Ozaki Kiichi, closes #8266,
closes #8250)
Files: src/if_perl.xs

Patch 8.2.2892

Problem: Error message contains random characters.
Solution: Pass the right pointer to error_white_both(). (closes #8272,
closes #8263)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2893

Problem: Multi-byte text in popup title shows up wrong.
Solution: Use the character width instead of the byte length. (Ralf Schandl,
closes #8267, closes #8264)
Files: src/popupwin.c, src/message_test.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_multibytetitle.dump

Patch 8.2.2894

Problem: MS-Windows: using enc_locale() for strftime() might not work.
Solution: Use wcsftime(). (Ken Takata, closes #8271)
Files: src/time.c

Patch 8.2.2895

Problem: Vim9: random characters appear in some error messages.
Solution: Pass the correct pointer. (closes #8277)
Files: src/eval.c, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2896

Problem: Spellfile functionality not fully tested.
Solution: Add tests for CHECKCOMPOUNDPATTERN and COMMON. (Dominique Pellé,
closes #8270)
Files: src/testdir/test_spellfile.vim

Patch 8.2.2897

Problem: Vim9: can use reserved words at the script level.
Solution: Check variable names for reserved words. (closes #8253)
Files: src/vim9compile.c, src/vim9script.c, src/proto/vim9script.pro,
src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2898

Problem: QuitPre and ExitPre not triggered when GUI window is closed.
Solution: Call before_quit_autocmds(). (closes #8242)
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/gui.c

Patch 8.2.2899

Problem: Appveyor script does not detect nmake failure.
Solution: Explicitly check for executable. (Ken Takata, closes #8281)
Files: ci/appveyor.bat

Patch 8.2.2900

Problem: QuitPre is triggered before :wq writes the file, which is different from other commands.
Solution: Trigger QuitPre after writing the file. (closes #8279)
Files: src/ex_docmd.c, src/testdir/test_writefile.vim

Patch 8.2.2901

Problem: Some operators not fully tested.
Solution: Add a few test cases. (Yegappan Lakshmanan, closes #8282)
Files: src/testdir/test_coptions.vim, src/testdir/test_increment.vim, src/testdir/test_normal.vim, src/testdir/test_virtualedit.vim, src/testdir/test_visual.vim

Patch 8.2.2902

Problem: Spellfile functionality not fully tested.
Solution: Add tests for CIRCUMFIX, NOBREAK and others. (Dominique Pellé, closes #8283)
Files: src/testdir/test_spellfile.vim

Patch 8.2.2903

Problem: Cursor position wrong on wrapped line with **'signcolumn'**.
Solution: Don't add space for showbreak twice. (Christian Brabandt, closes #8262)
Files: src/drawline.c, src/testdir/test_display.vim

Patch 8.2.2904

Problem: "g\$" causes scroll if half a double width char is visible.
Solution: Advance to the last fully visible character. (closes #8254)
Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.2.2905

Problem: No error when defaults.vim cannot be loaded.
Solution: Add an error message. (Christian Brabandt, closes #8248)
Files: runtime/doc/starting.txt, src/errors.h, src/main.c, src/testdir/test_startup.vim

Patch 8.2.2906 (after 8.2.2905)

Problem: ASAN reports errors for test_startup for unknown reasons.
Solution: Temporarily disable the new test.
Files: src/testdir/test_startup.vim

Patch 8.2.2907

Problem: Memory leak when running out of memory.
Solution: Free the allocated memory. (Dominique Pellé, closes #8284)

Files: src/term.c

Patch 8.2.2908

Problem: Crash when using a terminal popup window from the cmdline window.

Solution: Instead of checking cmdwin_type call cmdwin_is_active().
(closes #8286)

Files: src/terminal.c, src/errors.h, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_cmdwin_no_terminal.dump

Patch 8.2.2909

Problem: Build error with non-Unix system.

Solution: Always include limits.h.

Files: src/vim.h

Patch 8.2.2910

Problem: Test for cmdline window and terminal fails on MS-Windows.

Solution: Skip the test on MS-Windows.

Files: src/testdir/test_cmdline.vim

Patch 8.2.2911

Problem: Pattern "\%V" does not match all of block selection. (Rick Howe)

Solution: Use the value of vi_curswant. (closes #8285)

Files: src/regexp.c, src/testdir/test_search.vim,
src/testdir/dumps/Test_hlsearch_block_visual_match.dump

Patch 8.2.2912

Problem: MS-Windows: most users expect using Unicode.

Solution: Default **'encoding'** to utf-8 on MS-Windows. (Ken Takata,
closes #3907)

Files: runtime/doc/options.txt, src/mbyte.c, src/option.c, src/option.h,
src/testdir/test_writefile.vim

Patch 8.2.2913

Problem: MS-Windows conpty supports using mouse events.

Solution: When enabling the mouse enable mouse input and disable quick edit
mode. (Wez Furlong, closes #8280)

Files: src/os_win32.c

Patch 8.2.2914

Problem: Cannot paste a block without adding padding.

Solution: Add "zp" and "zP" which paste without adding padding. (Christian
Brabandt, closes #8289)

Files: runtime/doc/change.txt, runtime/doc/index.txt, src/normal.c,
src/register.c, src/vim.h, src/testdir/test_normal.vim,
src/testdir/test_visual.vim

Patch 8.2.2915

Problem: MS-Windows: when using "default" for encoding utf-8 is used.

Solution: Use the system encoding. (Ken Takata, closes #8300)

Files: src/mbyte.c, runtime/doc/options.txt

Patch 8.2.2916

Problem: Operators are not fully tested.

Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8290)

Files: src/ops.c, src/testdir/test_netbeans.vim,
src/testdir/test_normal.vim, src/testdir/test_visual.vim

Patch 8.2.2917

Problem: Spellfile functionality not fully tested.
Solution: Add tests for SFX with removal of characters, spelling
suggestions with NOBREAK and others. (Dominique Pellé,
closes #8293)
Files: src/testdir/test_spellfile.vim

Patch 8.2.2918

Problem: Builtin function can be shadowed by global variable.
Solution: Check for builtin function before variable. (Yasuhiro Matsumoto,
closes #8302)
Files: src/eval.c, src/testdir/test_functions.vim

Patch 8.2.2919

Problem: Using "!:command" does not work if the command uses posix_spawn().
Solution: Do not call ioctl() with TIOCSCTTY. (Felipe Contreras)
Files: src/os_unix.c

Patch 8.2.2920

Problem: Still a way to shadow a builtin function. (Yasuhiro Matsumoto)
Solution: Check the key when using extend(). (issue #8302)
Files: src/eval.c, src/dict.c, src/proto/dict.pro,
src/testdir/test_functions.vim

Patch 8.2.2921

Problem: E704 for script local variable is not backwards compatible.
(Yasuhiro Matsumoto)
Solution: Only give the error in Vim9 script. Also check for function-local
variable.
Files: src/dict.c, src/testdir/test_functions.vim

Patch 8.2.2922

Problem: Computing array length is done in various ways.
Solution: Use ARRAY_LENGTH everywhere. (Ken Takata, closes #8305)
Files: src/arabic.c, src/blowfish.c, src/cindent.c, src/cmdexpand.c,
src/cmdhist.c, src/dosinst.c, src/eval.c, src/evalfunc.c,
src/ex_docmd.c, src/fileio.c, src/gui_athena.c, src/gui_gtk_x11.c,
src/gui_haiku.cc, src/gui_photon.c, src/gui_w32.c,
src/gui_xmew.c, src/hardcopy.c, src/help.c, src/highlight.c,
src/if_mzsch.c, src/macros.h, src/main.c, src/map.c, src/mbyte.c,
src/memline.c, src/menu.c, src/misc2.c, src/normal.c, src/ops.c,
src/option.c, src/optiondefs.h, src/os_win32.c, src/popupwin.c,
src/quickfix.c, src/regexp.c, src/screen.c, src/search.c,
src/syntax.c, src/term.c, src/terminal.c, src/time.c,
src/usercmd.c, src/version.c

Patch 8.2.2923

Problem: EBCDIC build is broken.
Solution: Move sortFunctions() to evalfunc.c. (Ken Takata, closes #8306)
Files: src/eval.c, src/evalfunc.c, src/proto/evalfunc.pro

Patch 8.2.2924

Problem: Superfluous extern declaration.
Solution: Delete the declaration. (Ken Takata, closes #8307)
Files: src/main.c

Patch 8.2.2925

Problem: Vim9: line continuation comment uses legacy syntax.
Solution: Check for #\ instead of "\. (closes #8295)
Files: src/scriptfile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2926

Problem: Vim9: no good error for using :legacy in a :def function.
Solution: Give an explicit error where :legacy is not working.
(closes #8309)
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_func.vim

Patch 8.2.2927

Problem: Test commented out because it fails with ASAN.
Solution: Only skip the test when running with ASAN.
Files: src/testdir/test_startup.vim

Patch 8.2.2928

Problem: The evalfunc.c file is too big.
Solution: Move float related functionality to a separate file. (Yegappan Lakshmanan, closes #8287)
Files: Filelist, src/Make_ami.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/eval.c, src/evalfunc.c, src/float.c, src/proto.h, src/proto/eval.pro, src/proto/float.pro

Patch 8.2.2929

Problem: Accidentally enable tcl by default.
Solution: Revert change to Makefile
Files: src/Makefile

Patch 8.2.2930

Problem: When a popup is visible a mouse move my restart Visual mode.
Solution: Reset held_button when ending Visual mode. (closes #8318)
Files: src/mouse.c, src/proto/mouse.pro, src/normal.c

Patch 8.2.2931

Problem: Vim9: line continuation comment still uses legacy syntax in one place.
Solution: Check for #\ instead of "\ earlier. (closes #8316)
Files: src/scriptfile.c, src/testdir/test_vim9_script.vim

Patch 8.2.2932 (after 8.2.2930)

Problem: Select mode test fails.
Solution: Do not always reset the held mouse button.
Files: src/mouse.c, src/normal.c, src/proto/normal.pro

Patch 8.2.2933

Problem: When **'clipboard'** is "unnamed" zp and zP do not work correctly.
Solution: Pass -1 to str_to_reg() and fix computing the character width

instead of using the byte length. (Christian Brabandt, closes #8301, closes #8317)
Files: src/clipboard.c, src/mbyte.c, src/register.c

Patch 8.2.2934 (after 8.2.2933)
Problem: ASAN error when using text from the clipboard.
Solution: Get width of each character.
Files: src/register.c

Patch 8.2.2935 (after 8.2.2934)
Problem: Calculating register width is not always needed. (Christian Brabandt)
Solution: Only calculate the width when the type is MBLOCK.
Files: src/register.c

Patch 8.2.2936
Problem: Vim9: converting number to bool uses wrong stack offset. (Salman Halim)
Solution: Include the offset in the 2B00L command.
Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2937
Problem: Popup test fails if rightleft feature not enabled.
Solution: Check that the rightleft feature is available. (Dominique Pellé, closes #8321)
Files: src/testdir/test_popup.vim

Patch 8.2.2938
Problem: After using motion force from feedkeys() it may not be reset.
Solution: Clear motion_force in clearop(). (closes #8323)
Files: src/normal.c, src/testdir/test_visual.vim

Patch 8.2.2939
Problem: GTK: righthand scrollbar does not show with split window.
Solution: Adjust padding when two scrollbars are used. (Matt Wozniski, closes #8324)
Files: src/gui_gtk.c

Patch 8.2.2940
Problem: MS-Windows: cannot see the size of the text area when resizing the gvim window.
Solution: Show a tooltip with the text size. (Ken Takata, closes #8326)
Files: src/gui_w32.c

Patch 8.2.2941
Problem: Vim9: using `=expr` does not handle a list of strings.
Solution: Convert a list to a string and escape each item. (closes #8310)
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2942
Problem: Vim9: internal error when calling function with too few arguments
Solution: Check for argument count to be too few. (closes #8325)

Files: src/errors.h, src/vim9execute.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.2943

Problem: Vim9: check for argument count ignores default values.
Solution: Take default argument values into account.
Files: src/vim9execute.c

Patch 8.2.2944

Problem: Vim9: no error when using job or channel as a string.
Solution: Be more strict about conversion to string. (closes #8312)
Files: src/typval.c, src/job.c, src/proto/job.pro, src/channel.c,
src/proto/channel.pro, src/eval.c, src/vim9execute.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.2945

Problem: Some buffer related code is not tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8320)
Files: src/termlib.c, src/testdir/test_excmod.vim,
src/testdir/test_recover.vim, src/testdir/test_swap.vim,
src/testdir/test_visual.vim

Patch 8.2.2946

Problem: Vim9: substitute expression cannot be a List in a :def function.
Solution: Use typval2string(). (closes #8330)
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.2947

Problem: Build failure without the channel feature.
Solution: Add back #ifdef. (John Marriott)
Files: src/eval.c

Patch 8.2.2948

Problem: Substitute() accepts a number but not a float expression.
Solution: Also accept a float. (closes #8331)
Files: src/typval.c, src/testdir/test_substitute.vim

Patch 8.2.2949 (after 8.2.2948)

Problem: Tests failing because there is no error for float to string conversion.
Solution: Change the check for failure to check for correct result. Make some conversions strict in Vim9 script.
Files: src/evalfunc.c, src/float.c, src/findfile.c, src/json.c,
src/filepath.c, src/testdir/test_eval_stuff.vim,
src/testdir/test_execute_func.vim,
src/testdir/test_float_func.vim, src/testdir/test_functions.vim,
src/testdir/test_listdict.vim, src/testdir/test_glob2regpat.vim

Patch 8.2.2950

Problem: Sound code not fully tested.
Solution: Add more sound tests. (Dominique Pellé, closes #8332)
Files: src/testdir/test_sound.vim

Patch 8.2.2951

Problem: Vim9: cannot use heredoc in :def function for :python, :lua, etc.

Solution: Concatenate the heredoc lines and pass them in the ISN_EXEC_SPLIT instruction.

Files: src/userfunc.c, src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2952

Problem: Recover test fails on big endian systems.

Solution: Disable the failing test on big endian systems. (Yegappan Lakshmanan, closes #8335)

Files: src/testdir/test_recover.vim, src/testdir/test_swap.vim

Patch 8.2.2953 (after 8.2.2951)

Problem: Vim9: leaking memory when using heredoc script.

Solution: Free the first line.

Files: src/vim9execute.c

Patch 8.2.2954

Problem: Short file name extension for Scala not recognized.

Solution: Recognize *.sc. (closes #8337)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2955

Problem: Vim9: using filter in compiled command does not work.

Solution: Generate EXEC including the command modifier.

Files: src/vim9compile.c, src/ex_docmd.c, src/ex_cmds.c, src/proto/ex_cmds.pro, src/testdir/test_vim9_cmd.vim

Patch 8.2.2956

Problem: Vim9: need to plan for future additions.

Solution: Reserve commands for future use: :type, :class, :enum.

Files: src/ex_cmds.h, src/ex_cmdidxs.h

Patch 8.2.2957

Problem: Using getchar() in Vim9 script is problematic.

Solution: Add getcharstr(). (closes #8343)

Files: runtime/doc/eval.txt, src/evalfunc.c, src/getchar.c, src/proto/getchar.pro, src/testdir/test_functions.vim

Patch 8.2.2958 (after 8.2.2957)

Problem: Function list test fails.

Solution: Add newly added function to the list. Fix typo.

Files: runtime/doc/usr_41.txt, src/testdir/test_function_lists.vim

Patch 8.2.2959

Problem: sound_playfile() is not tested on MS-Windows.

Solution: Make it work and enable the test. (Dominique Pellé, closes #8338)

Files: src/sound.c, src/testdir/test_sound.vim

Patch 8.2.2960

Problem: Swap file recovery not sufficiently tested.

Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8339)

Files: src/testdir/test_recover.vim

Patch 8.2.2961

Problem: Keys typed during a :normal command are discarded.
Solution: Concatenate saved typeahead and typed keys. (closes #8340)
Files: src/getchar.c, src/proto/getchar.pro, src/ex_docmd.c,
src/evalfunc.c, src/debugger.c, src/ui.c, src/proto/ui.pro

Patch 8.2.2962

Problem: MS-Windows command line arguments have wrong encoding.
Solution: Always use utf-8 in get_cmd_argsW(). (Ken Takata, closes #8347)
Files: src/os_win32.c

Patch 8.2.2963

Problem: GUI: mouse move may start Visual mode with a popup visible.
Solution: Add special code for mouse move. (closes #8318)
Files: src/vim.h, src/gui.c, src/keymap.h, src/term.c

Patch 8.2.2964

Problem: Vim9: hang when using space after ->. (Naohiro Ono)
Solution: Skip over white space to find the function name. (closes #8341)
Files: src/eval.c, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.2965

Problem: Vim9: crash when calling function that failed to compile.
Solution: Fail when trying to call the function. (closes #8344)
Files: src/errors.h, src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.2966

Problem: ml_get errors after recovering a file. (Yegappan Lakshmanan)
Solution: Fix the cursor position after deleting lines.
Files: src/memline.c

Patch 8.2.2967

Problem: Vim9: crash when using two levels of partials.
Solution: Add outer_ref_T and use it in the execution context.
Files: src/structs.h, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.2968 (after 8.2.2967)

Problem: Vim9: memory leak
Solution: Unreference pt_outer of partial.
Files: src/eval.c

Patch 8.2.2969

Problem: Subtracting from number option fails when result is zero. (Ingo Karkat)
Solution: Reset the string value when using the numeric value.
(closes #8351)
Files: src/evalvars.c, src/testdir/test_vimscript.vim

Patch 8.2.2970

Problem: Python configure check uses deprecated command.
Solution: Use sysconfig instead of distutils if possible. (Zdenek Dohnal,
closes #8354)
Files: src/configure.ac, src/auto/configure

Patch 8.2.2971

Problem: Cannot yank a block without trailing spaces.
Solution: Add the "zy" command. (Christian Brabandt, closes #8292)
Files: runtime/doc/change.txt, runtime/doc/index.txt, src/normal.c,
src/ops.c, src/register.c, src/structs.h,
src/testdir/test_visual.vim

Patch 8.2.2972

Problem: "%bd" tries to delete popup window buffers, which fails. (Ralf Schandl)
Solution: Do not try to delete a popup window buffer. (closes #8349)
Files: src/buffer.c, src/vim.h, src/testdir/test_popupwin.vim

Patch 8.2.2973

Problem: Fix for recovery and diff mode not tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8352)
Files: src/testdir/test_diffmode.vim, src/testdir/test_prompt_buffer.vim,
src/testdir/test_recover.vim

Patch 8.2.2974

Problem: Greek spell checking uses wrong case folding.
Solution: Fold capital sigma depending on whether it is at the end of a word or not. (closes #299)
Files: src/spell.c, src/proto/spell.pro, src/spellfile.c,
src/spellsuggest.c

Patch 8.2.2975

Problem: Vim9: can only use an autoload function name as a string.
Solution: Load the autoload script when encountered. (closes #8124)
Files: src/vim9compile.c, src/evalvars.c, src/scriptfile.c,
src/testdir/test_vim9_func.vim

Patch 8.2.2976 (after 8.2.2975)

Problem: Build failure without the +eval feature.
Solution: Add #ifdefs.
Files: src/scriptfile.c

Patch 8.2.2977

Problem: Crash when using a null function reference. (Naohiro Ono)
Solution: Check for an invalid function name. (closes #8367)
Files: src/eval.c, src/errors.h, src/testdir/test_functions.vim

Patch 8.2.2978 (after 8.2.2977)

Problem: Warning for uninitialized variable.
Solution: Set return value to FAIL.
Files: src/eval.c

Patch 8.2.2979

Problem: Not all options code is covered by tests.
Solution: Add more tests for options. (Yegappan Lakshmanan, closes #8369)
Files: src/testdir/test_edit.vim, src/testdir/test_excmd.vim,
src/testdir/test_help.vim, src/testdir/test_mksession.vim,
src/testdir/test_options.vim, src/testdir/test_vartabs.vim,
src/testdir/test_window_cmd.vim

Patch 8.2.2980

Problem: Popup window test is a bit flaky.
Solution: Add a redraw command.
Files: src/testdir/test_popupwin.vim

Patch 8.2.2981

Problem: Recovery test is not run on big-endian systems.
Solution: Make it work on big-endian systems. (James McCoy, closes #8368)
Files: src/testdir/test_recover.vim

Patch 8.2.2982

Problem: Vim9: future commands are not reserved yet.
Solution: Add commands to be implemented later. Make "this" a reserved name.
Files: runtime/doc/vim9.txt, src/ex_cmds.h, src/ex_cmdidxs.h, src/vim9script.c, src/testdir/test_vim9_assign.vim

Patch 8.2.2983

Problem: Vim9: an inline function requires specifying the return type.
Solution: Make the return type optional.
Files: src/eval.c, src/vim9compile.c, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.2984 (after 8.2.2983)

Problem: Vim9: Test fails because of missing return statement.
Solution: When type is unknown set type to void.
Files: src/vim9compile.c

Patch 8.2.2985

Problem: Vim9: a compiled function cannot be debugged.
Solution: Add initial debugging support.
Files: src/vim9.h, src/vim9compile.c, src/proto/vim9compile.pro, src/vim.h, src/eval.c, src/vim9execute.c, src/userfunc.c, src/vim9type.c, src/testdir/test_debugger.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.2986

Problem: Build failure without the profile feature.
Solution: Add #ifdef.
Files: src/vim9compile.c

Patch 8.2.2987

Problem: Build failure with normal features.
Solution: Remove #define.
Files: src/vim9execute.c

Patch 8.2.2988

Problem: Vim9: debugger test fails.
Solution: Get the debugger instructions when needed.
Files: src/vim.h, src/vim9.h

Patch 8.2.2989

Problem: Vim9: memory leak when debugging a :def function.

Solution: Free the debug instructions.
Files: src/vim9compile.c

Patch 8.2.2990

Problem: Jupyter Notebook files are not recognized.
Solution: Recognize *.ipynb. (closes #8375)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.2991

Problem: Vim9: no completion for :vim9 and :legacy.
Solution: Expand argument as a command. (closes #8377)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.2992

Problem: Vim9: completion for :disassemble is incomplete.
Solution: Recognize the "debug" and "profile" arguments.
Files: src/cmdexpand.c, src/vim9execute.c, src/proto/vim9execute.pro,
src/vim.h, src/testdir/test_cmdline.vim

Patch 8.2.2993

Problem: **'fileencodings'** default value should depend on **'encoding'**. (Gary Johnson)
Solution: When **'encoding'** is "utf-8" use a different default value for **'fileencodings'**.
Files: src/mbyte.c, src/option.c, src/proto/option.pro,
src/testdir/test_options.vim

Patch 8.2.2994

Problem: Various code is not fully tested.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8378)
Files: src/testdir/test_excmd.vim, src/testdir/test_mapping.vim,
src/testdir/test_modeline.vim, src/testdir/test_options.vim,
src/testdir/test_paste.vim, src/vim9compile.c

Patch 8.2.2995

Problem: Linker errors with dynamic Python 3.10.
Solution: Add a couple of library entries. (Zdenek Dohnal, closes #8381,
closes #8356)
Files: src/if_python3.c

Patch 8.2.2996

Problem: Vim9: when debugging cannot inspect local variables.
Solution: Make local variables available when debugging.
Files: src/vim9execute.c, src/proto/vim9execute.pro, src/vim9compile.c,
src/vim9.h, src/debugger.c, src/testdir/test_debugger.vim

Patch 8.2.2997 (after 8.2 2996)

Problem: Vim9: disassemble test fails.
Solution: Adjust expected output.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.2998 (after 8.2 2996)

Problem: Vim9: disassemble test fails.
Solution: Add missing call to lookup_debug_var().

Files: src/evalvars.c

Patch 8.2.2999

Problem: Balloon sometimes does not hide with GTK 3.

Solution: Also listen to GDK_LEAVE_NOTIFY. (Johannes Stezenbach)

Files: src/gui_beval.c

Patch 8.2.3000

Problem: Vim9: warning for uninitialized variable.

Solution: Add initialization. (John Marriott)

Files: src/vim9compile.c

Patch 8.2.3001

Problem: Vim9: memory leak when compilation fails.

Solution: Free the list of variable names.

Files: src/vim9compile.c

Patch 8.2.3002

Problem: Vim doesn't abort on a fatal Tcl error.

Solution: Change emsg() to iemsg(). (Dominique Pellé, closes #8383)

Files: src/if_tcl.c

Patch 8.2.3003

Problem: Vim9: closure compiled with wrong compile type.

Solution: Use COMPILER_TYPE() when calling a function. (closes #8384)

Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3004

Problem: Vim9: error for missing colon given while skipping.

Solution: Do not give the error when skipping. (closes #8385)

Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.3005

Problem: Vim9: using a void value does not give a proper error message.

Solution: Give a clear error message. (closes #8387)

Files: src/typval.c, src/vim9compile.c, src/vim9.h, src/vim9execute.c,
src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3006

Problem: Crash when echoing a value very early. (Naruhiko Nishino)

Solution: Do not use a NUL to truncate the message, make a copy.
(closes #8388)

Files: src/message.c, src/testdir/test_startup.vim

Patch 8.2.3007 (after 8.2.3005)

Problem: Vim9: test for void value fails.

Solution: Adjust expected error. Do not make a copy of void.

Files: src/typval.c, src/testdir/test_functions.vim

Patch 8.2.3008 (after 8.2.3006)

Problem: Startup test may hang.

Solution: Add quit command in the script.

Files: src/testdir/test_startup.vim

Patch 8.2.3009 (after 8.2.3006)

Problem: Startup test may hang.
Solution: Do not run the test in the GUI.
Files: src/testdir/test_startup.vim

Patch 8.2.3010

Problem: Not enough testing for viminfo code.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #8390)
Files: src/register.c, src/testdir/test_fileformat.vim,
src/testdir/test_smartindent.vim, src/testdir/test_viminfo.vim

Patch 8.2.3011

Problem: Vim9: cannot get argument values during debugging.
Solution: Lookup names in the list of arguments. Put debug instruction halfway for command.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_debugger.vim

Patch 8.2.3012

Problem: When **'rightleft'** is set the line number is sometimes drawn reversed.
Solution: Adjust how space is handled. (Christian Brabandt, closes #8389, closes #8391)
Files: src/drawline.c, src/testdir/test_number.vim

Patch 8.2.3013

Problem: Vim: when debugging only the first line of a command using line continuation is displayed.
Solution: Find the next command and concatenate lines until that one. (closes #8392)
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3014

Problem: Coverity warns for freeing static string.
Solution: Do not assign static string to pointer. (Dominique Pellé, closes #8397)
Files: src/vim9execute.c

Patch 8.2.3015

Problem: Vim9: Assigning to @# requires a string. (Naohiro Ono)
Solution: Accent a number or a string. (closes #8396)
Files: src/vim9compile.c, src/vim9execute.c, src/globals.h,
src/testdir/test_vim9_assign.vim

Patch 8.2.3016

Problem: Confusing error when expression is followed by comma.
Solution: Give a different error for trailing text. (closes #8395)
Files: src/eval.c, src/testdir/test_let.vim,
src/testdir/test_eval_stuff.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_viminfo.vim,
src/testdir/test_vimscript.vim

Patch 8.2.3017

Problem: Vim9: debugger shows too many lines.
Solution: Truncate at a comment, "enddef", etc. (closes #8392)
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3018

Problem: Formatting using quickfixtextfunc is lost when updating location lists for different buffers. (Yorick Peterse)
Solution: Use the right window for the location list. (Yegappan Lakshmanan, closes #8400, closes #8403)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.3019

Problem: Location list only has the start position.
Solution: Make it possible to add an end position. (thinca, closes #8393)
Files: runtime/doc/eval.txt, src/quickfix.c,
src/testdir/dumps/Test_quickfix_cwindow_1.dump,
src/testdir/dumps/Test_quickfix_cwindow_2.dump,
src/testdir/test_quickfix.vim, src/testdir/test_tagjump.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.3020

Problem: Unreachable code.
Solution: Remove the code. (closes #8406)
Files: src/ex_docmd.c

Patch 8.2.3021

Problem: Spaces allowed between option name and "!", "?", etc.
Solution: Disallow spaces in Vim9 script, it was not documented. (closes #8408)
Files: src/option.c, src/testdir/test_vim9_script.vim

Patch 8.2.3022

Problem: Available encryption methods are not strong enough.
Solution: Add initial support for xchacha20. (Christian Brabandt, closes #8394)
Files: .github/workflows/ci.yml, runtime/doc/eval.txt,
runtime/doc/options.txt, runtime/doc/various.txt,
src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak,
src/auto/configure, src/blowfish.c, src/bufwrite.c,
src/config.h.in, src/configure.ac, src/crypt.c, src/crypt_zip.c,
src/errors.h, src/evalfunc.c, src/feature.h, src/fileio.c,
src/memline.c, src/option.c, src/optionstr.c,
src/proto/blowfish.pro, src/proto/crypt.pro,
src/proto/crypt_zip.pro, src/structs.h,
src/testdir/samples/crypt_sodium_invalid.txt,
src/testdir/test_crypt.vim, src/undo.c, src/version.c

Patch 8.2.3023

Problem: Vim9: arguments for execute() not checked at compile time.
Solution: Add a function to check the argument types.
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3024 (after 8.2.3023)

Problem: execute() function test fails.

Solution: Adjust test for different error.
Files: src/testdir/test_execute_func.vim

Patch 8.2.3025

Problem: Not enough tests for quickfix end_col and end_lnum.
Solution: Add a few more test cases. (Shane-XB-Qian, closes #8409)
Files: src/testdir/test_quickfix.vim

Patch 8.2.3026

Problem: Vim9: cannot set breakpoint in compiled function.
Solution: Check for breakpoint when calling a function.
Files: src/vim9execute.c, src/structs.h, src/vim.h, src/vim9.h,
src/debugger.c, src/testdir/test_debugger.vim

Patch 8.2.3027

Problem: Vim9: breakpoint in compiled function not always checked.
Solution: Check for breakpoint when calling compiled function from compiled function.
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3028

Problem: GUI mouse events not tested.
Solution: Add test_gui_mouse_event(). Add mouse tests. Also add a few viminfo tests. (Yegappan Lakshmanan, closes #8407)
Files: runtime/doc/eval.txt, runtime/doc/testing.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/testing.pro,
src/testdir/test_gui.vim, src/testdir/test_viminfo.vim,
src/testing.c

Patch 8.2.3029

Problem: Vim9: crash when using operator and list unpack assignment. (Naohiro Ono)
Solution: Get variable value before operation. (closes #8416)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/ex_docmd.c,
src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3030

Problem: Coverity reports a memory leak.
Solution: Fix the leak and a few typos. (Dominique Pellé, closes #8418)
Files: src/crypt.c, src/errors.h

Patch 8.2.3031

Problem: No error if a function name starts with an underscore. (Naohiro Ono)
Solution: In Vim9 script disallow a function name starting with an underscore, as is mentioned in the help. (closes #8414)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3032

Problem: Build problems with MSVC, other crypt issues with libsodium.
Solution: Adjust MSVC makefile. Disable swap file only when 'key' is set. Adjust error message used when key is wrong. Fix Coverity issues. (Christian Brabandt, closes #8420, closes #8411)

Files: src/Make_mvc.mak, src/crypt.c, src/errors.h, src/fileio.c,
src/memline.c, src/proto/crypt.pro

Patch 8.2.3033

Problem: No error when using alpha delimiter with :global.
Solution: Check the delimiter like with :substitute. (closes #8415)
Files: src/ex_cmds.c, src/testdir/test_global.vim

Patch 8.2.3034

Problem: Installing packages on github CI sometimes fails.
Solution: Update package information first. (Christian Brabandt,
closes #8432)
Files: .github/workflows/ci.yml

Patch 8.2.3035

Problem: Vim9: crash when calling :def function with partial and return
type is not set.
Solution: When the return type is not set handle like the return type is
unknown. (closes #8422)
Files: src/vim9type.c, src/testdir/test_vim9_func.vim

Patch 8.2.3036

Problem: Vim9: builtin function arguments not checked at compile time.
Solution: Add more argument type specs. Check arguments to test_setmouse()
and test_gui_mouse_event(). (Yegappan Lakshmanan, closes #8425)
Files: src/evalfunc.c, src/testdir/test_assert.vim,
src/testdir/test_gui.vim, src/testdir/test_popupwin.vim,
src/testdir/test_vim9_builtin.vim, src/testing.c

Patch 8.2.3037

Problem: Configure reports libcanberra when checking for libsodium.
Solution: Adjust the message. (Ozaki Kiichi, closes #8435)
Files: src/configure.ac, src/auto/configure

Patch 8.2.3038

Problem: Amiga built-in version string doesn't include build date.
Solution: Add the build date if available. (Ola Söder, closes #8437)
Files: src/os_amiga.c

Patch 8.2.3039

Problem: Vim9: breakpoint at a comment line does not work.
Solution: Add the comment line number to the debug instruction.
(closes #8429)
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_debugger.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3040

Problem: GUI: dropping files not tested.
Solution: Add test_gui_drop_files() and tests. (Yegappan Lakshmanan,
closes #8434)
Files: runtime/doc/eval.txt, runtime/doc/testing.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/gui.c,
src/proto/testing.pro, src/testdir/test_gui.vim, src/testing.c

Patch 8.2.3041

Problem: Detecting if the process of a swap file is running fails if the process is owned by another user.
Solution: Check for the ESRCH error. (closes #8436)
Files: src/os_unix.c

Patch 8.2.3042 (after 8.2.3041)

Problem: Swap file test fails.
Solution: Check for a very high process ID instead of one, which should be running.
Files: src/testdir/test_swap.vim

Patch 8.2.3043

Problem: Amiga: cannot get the shell size on MorphOS and AROS.
Solution: Use control sequences. (Ola Söder, closes #8438)
Files: src/os_amiga.c

Patch 8.2.3044

Problem: Amiga MorphOS and AROS: process ID is not valid.
Solution: Use FindTask to return something which is unique to all processes. (Ola Söder, closes #8444)
Files: src/os_amiga.c

Patch 8.2.3045

Problem: Minor typos.
Solution: Fix the typos. (Christian Brabandt, closes #8441)
Files: src/VisVim/README_VisVim.txt, src/evalfunc.c, src/testdir/vim9.vim

Patch 8.2.3046

Problem: Amiga MorphOS: Term mode is set using DOS packets.
Solution: Use the same way of setting term mode on all next gen Amiga-like systems. (Ola Söder, closes #8445)
Files: src/os_amiga.c

Patch 8.2.3047

Problem: Increment and decrement don't allow for next command.
Solution: Allow for comment and next command. (closes #8442)
Files: src/ex_cmds.h, src/vim9script.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3048

Problem: Strange error for white space after ++ command.
Solution: Check for white space explicitly. (closes #8440)
Files: src/vim9script.c, src/errors.h, src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3049

Problem: JSON patch file not recognized.
Solution: Recognize json-patch as json. (Kevin Locke, closes #8450)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3050

Problem: Cannot recognize elixir files.
Solution: Recognize Elixir-specific files. Check if an .ex file is Euphoria

Files: or Elixir. (Austin Gatlin, closes #8401, closes #8446)
runtime/autoload/dist/ft.vim, runtime/filetype.vim,
src/testdir/test_filetype.vim

Patch 8.2.3051

Problem: Vim9: for loop with one list variable does not work.
Solution: Use a separate flag for unpacking a list. (closes #8452)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3052

Problem: Vim9: "legacy call" does not work.
Solution: Do not skip "call" after "legacy". (closes #8454)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.3053

Problem: Vim9: cannot assign to @@ in :def function
Solution: Handle '@' like ''. (closes #8456)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3054

Problem: Vim9: unpack assignment using "_" after semicolon fails.
Solution: Drop the expression result. (closes #8453)
Files: src/vim9compile.c, src/errors.h, src/testdir/test_vim9_assign.vim

Patch 8.2.3055

Problem: Strange error for assigning to "x.key" on non-dictionary.
Solution: Add a specific error message. (closes #8451)
Files: src/eval.c, src/errors.h, src/testdir/test_vim9_assign.vim,
src/testdir/test_listdict.vim, src/testdir/test_let.vim

Patch 8.2.3056

Problem: Vim9: using default value in lambda gives confusing error.
Solution: Pass "default_args" on the first pass to get the arguments.
(closes #8455)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3057

Problem: Vim9: debugger test fails with normal features and +terminal.
(Dominique Pellé)
Solution: Adjust the INSTRUCTIONS macro. (closes #8460)
Files: src/vim9.h

Patch 8.2.3058 (after 8.2.3056)

Problem: Vim9: cannot use ternary operator in parentheses.
Solution: Do not use "==" for a default argument value. (closes #8462)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3059 (after 8.2.3056)

Problem: Vim9: memory leak when using lambda.
Solution: Do not store the default value strings when skipping.
Files: src/userfunc.c

Patch 8.2.3060 (after 8.2.3056)

Problem: Vim9: cannot use ternary operator in parentheses.

Solution: Do not use "=~" for a default argument value. (closes #8462)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3061

Problem: Testing the shell option is incomplete and spread out.
Solution: Move shell tests to one file and increase coverage. (Yegappan Lakshmanan, closes #8464)
Files: src/testdir/Make_all.mak, src/testdir/test_functions.vim, src/testdir/test_options.vim, src/testdir/test_shell.vim, src/testdir/test_system.vim

Patch 8.2.3062

Problem: Internal error when adding several text properties.
Solution: Do not handle text properties when deleting a line for splitting a data block. (closes #8466)
Files: src/structs.h, src/memline.c, src/testdir/test_textprop.vim

Patch 8.2.3063

Problem: Crash when switching '**cryptmethod**' to xchacha20 with an existing undo file. (Martin Tournoij)
Solution: Disable reading undo file when decoding can't be done inplace. (issue #8467)
Files: src/fileio.c, src/bufwrite.c

Patch 8.2.3064

Problem: Vim9: in script cannot set item in uninitialized list.
Solution: When a list is NULL allocate an empty one. (closes #8461)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3065

Problem: Vim9: error when sourcing script twice and reusing a function name.
Solution: Check if the function is dead. (closes #8463)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3066

Problem: Vim9: debugging lambda does not work.
Solution: Use the compile type of the function when compiling a lambda. (closes #8412)
Files: src/vim9compile.c, src/testdir/test_debugger.vim

Patch 8.2.3067

Problem: Building fails with Athena. (Elimar Riesebieter)
Solution: Adjust #ifdefs and add the 'drop_file' feature.
Files: src/evalfunc.c, src/testing.c, src/testdir/test_gui.vim

Patch 8.2.3068

Problem: Unicode tables are slightly outdated.
Solution: Update the tables for Unicode release 13. (Christian Brabandt closes #8430)
Files: runtime/tools/unicode.vim, src/mbyte.c

Patch 8.2.3069

Problem: Error messages are spread out.

Solution: Move some error messages to errors.h. Use clearer names.
Files: src/errors.h, src/globals.h, src/arglist.c, src/buffer.c,
src/channel.c, src/eval.c, src/evalfunc.c, src/evalvars.c,
src/evalwindow.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c,
src/filepath.c, src/fold.c, src/getchar.c, src/indent.c,
src/list.c, src/map.c, src/mark.c, src/normal.c, src/ops.c,
src/optionstr.c, src/popupwin.c, src/quickfix.c, src/spellfile.c,
src/textprop.c, src/typval.c, src/undo.c, src/userfunc.c,
src/vim9compile.c, src/window.c

Patch 8.2.3070

Problem: Not enough testing for shell use.
Solution: Add a bit more testing. (Yegappan Lakshmanan, closes #8469)
Files: src/testdir/test_shell.vim, src/testdir/test_startup.vim

Patch 8.2.3071

Problem: Shell options are not set properly for PowerShell.
Solution: Use better option defaults. (Mike Williams, closes #8459)
Files: runtime/doc/eval.txt, runtime/doc/options.txt,
runtime/doc/os_dos.txt, src/fileio.c, src/misc2.c, src/option.c,
src/os_win32.c, src/testdir/test_shell.vim

Patch 8.2.3072

Problem: The "zy" command does not work well when '**virtualedit**' is set to "block". (Johann Höchtl)
Solution: Make endspaces zero. (Christian Brabandt, closes #8468, closes #8448)
Files: src/register.c, src/testdir/test_visual.vim

Patch 8.2.3073

Problem: When cursor is moved for block append wrong text is inserted.
Solution: Calculate an offset. (Christian Brabandt, closes #8433, closes #8288)
Files: src/ops.c, src/testdir/test_blockedit.vim,
src/testdir/test_visual.vim

Patch 8.2.3074

Problem: popup_atcursor() uses wrong position with concealing.
Solution: Keep w_wcol in conceal_check_cursor_line(). (closes #8476)
Files: src/screen.c, src/proto/screen.pro, src/normal.c, src/edit.c,
src/ui.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_atcursor_pos.dump

Patch 8.2.3075

Problem: Xxd always reports an old version string. (Åsmund Ervik)
Solution: Update the version string with the last known change date. (Jürgen Weigert, closes #8475)
Files: src/xxd/xxd.c, src/testdir/test_xxd.vim

Patch 8.2.3076

Problem: Vim9: using try in catch block causes a hang.
Solution: Save and restore the ec_in_catch flag. (closes #8478)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3077

Problem: Vim9: an error in a catch block is not reported.
Solution: Put the "in catch" flag in the try stack. (closes #8478)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3078

Problem: Vim9: profile test fails.
Solution: Make throw in :catch jump to :finally.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim

Patch 8.2.3079

Problem: Powershell core not supported by default.
Solution: Set option defaults for "pwsh". (Mike Williams, closes #8481)
Files: runtime/doc/eval.txt, runtime/doc/options.txt,
runtime/doc/os_dos.txt, runtime/doc/os_win32.txt, src/fileio.c,
src/misc2.c, src/option.c, src/os_win32.c,
src/testdir/test_shell.vim

Patch 8.2.3080

Problem: Recover test fails on 32bit systems. (Ondřej Súpup)
Solution: Detect 32/64 bit systems. (Yegappan Lakshmanan, closes #8485,
closes #8479)
Files: src/testdir/test_recover.vim

Patch 8.2.3081

Problem: Cannot catch errors in a channel command.
Solution: Instead of skipping the error make it silent. (closes #8477)
Files: src/channel.c

Patch 8.2.3082

Problem: A channel command "echoerr" does not show anything.
Solution: Do not use silent errors when using an "echoerr" command.
(closes #8494)
Files: src/channel.c, src/testdir/test_channel.py,
src/testdir/test_channel.vim

Patch 8.2.3083

Problem: Crash when passing null string to charclass().
Solution: Bail out when string pointer is NULL. (Christian Brabandt,
closes #8498, closes #8260)
Files: src/mbyte.c, src/testdir/test_functions.vim

Patch 8.2.3084

Problem: Vim9: builtin function argument types are not checked at compile
time.
Solution: Add argument types. (Yegappan Lakshmanan, closes #8503)
Files: src/evalfunc.c, src/testdir/test_functions.vim,
src/testdir/test_glob2regpat.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.3085

Problem: JSONC files are not recognized.
Solution: Recognize .jsonc files. (Izhak Jakov, closes #8500)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3086

Problem: Vim9: breakpoint on "for" does not work.

Solution: Use the right line number in ISN_DEBUG. (closes #8486)

Files: src/vim9compile.c, src/testdir/test_debugger.vim

Patch 8.2.3087

Problem: Gemtext files are not recognized.

Solution: Recognize .gmi and .gemini files. (closes #8427)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3088

Problem: With '**virtualedit**' set to "block" Visual highlight is wrong after using "\$". (Marco Trosi)

Solution: Do not set w_old_cursor_lcol to MAXCOL. (closes #8495)

Files: src/drawscreen.c, src/testdir/test_visual.vim,
src/testdir/dumps/Test_visual_block_with_virtualedit.dump

Patch 8.2.3089

Problem: Garbage collection has useless code.

Solution: Bail out when aborting. (closes #8504)

Files: src/userfunc.c

Patch 8.2.3090

Problem: With concealing enabled and indirectly closing a fold the cursor may be somewhere in a folded line.

Solution: Recompute the cursor position when the cursor line can be concealed. (closes #8480)

Files: src/drawscreen.c

Patch 8.2.3091

Problem: Vim9: default argument expression cannot use previous argument

Solution: Correct argument index. (closes #8496)

Files: src/vim9compile.c, src/structs.h, src/testdir/test_vim9_func.vim

Patch 8.2.3092

Problem: Vim9: builtin function test fails without the +channel feature.

Solution: Check the +channel feature is supported. (Dominique Pellé,
closes #8507)

Files: runtime/doc/eval.txt, src/testdir/test_vim9_builtin.vim

Patch 8.2.3093

Problem: tablabel_tooltip test fails with Athena. (Dominique Pellé)

Solution: Skip the test when using Athena. (closes #8508)

Files: src/testdir/test_gui.vim, src/testdir/check.vim

Patch 8.2.3094

Problem: Test_popup_atcursor_pos() fails without the conceal feature.

Solution: Add a check for the conceal feature. (Dominique Pellé,
closes #8505)

Files: src/testdir/test_popupwin.vim

Patch 8.2.3095 (after 8.2.3088)

Problem: With **'virtualedit'** set to "block" block selection is wrong after using "\$". (Marco Troisi)
Solution: Compute the longest selected line. (closes #8495)
Files: src/drawscreen.c, src/testdir/test_visual.vim, src/testdir/dumps/Test_visual_block_with_virtualedit2.dump

Patch 8.2.3096

Problem: Temp files remain after running tests.
Solution: Delete the right files. (Dominique Pellé, closes #8509)
Files: src/testdir/test_debugger.vim, src/testdir/test_lambda.vim, src/testdir/test_visual.vim

Patch 8.2.3097

Problem: Crash when using "quit" at recovery prompt and autocommands are triggered.
Solution: Block autocommands when creating an empty buffer to use as the current buffer. (closes #8506)
Files: src/buffer.c, src/testdir/test_swap.vim

Patch 8.2.3098

Problem: Popup window test is flaky on MS-Windows with GUI.
Solution: Skip the check in this situation.
Files: src/testdir/test_popupwin.vim

Patch 8.2.3099

Problem: Vim9: missing catch/finally not reported at script level.
Solution: Give an error. (closes #8487)
Files: src/structs.h, src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3100

Problem: Vim9: no error when using type with unknown number of arguments.
Solution: Do not ignore argument count of -1. (closes #8492)
Files: src/vim9type.c, src/evalfunc.c, src/proto/evalfunc.pro, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.3101

Problem: Missing function prototype for vim_round().
Solution: Add the prototype.
Files: src/proto/float.pro

Patch 8.2.3102 (after 8.2.3097)

Problem: Test for crash fix does not fail without the fix.
Solution: Adjust the test sequence. (closes #8506)
Files: src/testdir/test_swap.vim

Patch 8.2.3103 (after 8.2.3102)

Problem: Swap test may fail on some systems when jobs take longer to exit.
Solution: Use different file names.
Files: src/testdir/test_swap.vim

Patch 8.2.3104

Problem: Vim9: unspecified function type causes type error.

Solution: Don't check type when min_argcount is negative. (issue #8492)
Files: src/globals.h, src/vim9type.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3105

Problem: Vim9: type of partial is wrong when it has arguments.
Solution: Subtract arguments from the count. (issue #8492)
Files: src/vim9type.c, src/userfunc.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3106

Problem: Vim9: confusing line number reported for error.
Solution: Use the start line number for the store instruction.
(closes #8488)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3107

Problem: Vim9: error for arguments while type didn't specify arguments.
Solution: Do not update that type to check when no argument count is specified. (closes #8492)
Files: src/userfunc.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3108

Problem: Test for remote_foreground() fails. (Elimar Riesebieter)
Solution: Check that \$DISPLAY is set. (Christian Brabandt)
Files: src/testdir/check.vim, src/testdir/test_clientserver.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.3109

Problem: Check for \$DISPLAY never fails.
Solution: Use eval().
Files: src/testdir/check.vim

Patch 8.2.3110

Problem: A pattern that matches the cursor position is a bit complicated.
Solution: Use a dot to indicate the cursor line and column. (Christian Brabandt, closes #8497, closes #8179)
Files: runtime/doc/pattern.txt, src/errors.h, src/regexp_bt.c,
src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.2.3111

Problem: Vim9: confusing error with extra whitespace before colon.
Solution: Check for colon after white space. (closes #8513)
Files: src/eval.c, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3112 (after 8.2.3090)

Problem: With concealing enabled and indirectly closing a fold the cursor may be somewhere in a folded line when it is not on the first line of the fold.
Solution: Check if the cursor is somewhere in the folded text.
Files: src/drawscreen.c

Patch 8.2.3113

Problem: No error when for loop variable shadows script variable.
Solution: Check for the error. (closes #8512)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3114

Problem: Amiga-like systems: build error checking for running process
Solution: Only build swapfile_process_running() on systems where it is actually used. (Ola Söder, closes #8519)
Files: src/memline.c

Patch 8.2.3115

Problem: Coverity complains about free_wininfo() use.
Solution: Add a condition that "wip2" is not equal to "wip". (Neovim #14996)
Files: src/window.c

Patch 8.2.3116

Problem: Vim9: crash when debugging a function with line continuation.
Solution: Check for a NULL pointer. (closes #8521)
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3117

Problem: Vim9: type not properly checked in for loop.
Solution: Have items() return a list of lists. Add runtime type checks. (closes #8515)
Files: src/evalfunc.c, src/globals.h, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3118

Problem: Vim9: "any" type not handled correctly in for loop.
Solution: Change compile time check into runtime check. (closes #8516)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3119

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.
Files: src/evalfunc.c

Patch 8.2.3120

Problem: Crypt with sodium test fails on MS-Windows.
Solution: Make the tests pass. (closes #8428)
Files: src/testdir/test_crypt.vim

Patch 8.2.3121

Problem: **'listchars'** "exceeds" character appears in foldcolumn. Window separator is missing. (Leonid V. Fedorenchik)
Solution: Only draw the "exceeds" character in the text area. Break the loop when not drawing the text. (closes #8524)
Files: src/drawline.c, src/testdir/test_listchars.vim, src/testdir/dumps/Test_listchars_01.dump, src/testdir/dumps/Test_listchars_02.dump, src/testdir/dumps/Test_listchars_03.dump, src/testdir/dumps/Test_listchars_04.dump, src/testdir/dumps/Test_listchars_05.dump

Patch 8.2.3122

Problem: With **'nowrap'** cursor position is unexpected in narrow window. (Leonid V. Fedorenchik)

Solution: Put cursor on the last non-empty line. (closes #8525)
Files: src/move.c, src/testdir/test_listchars.vim,
src/testdir/dumps/Test_listchars_06.dump,
src/testdir/dumps/Test_listchars_07.dump

Patch 8.2.3123

Problem: Vim9: confusing error when using white space after option, before one of "!&<".
Solution: Give a specific error. (issue #8408)
Files: src/errors.h, src/option.c, src/testdir/test_vim9_script.vim

Patch 8.2.3124

Problem: Vim9: no error for white space between option and "=9".
Solution: Check for extraneous white space. (issue #8408)
Files: src/option.c, src/testdir/test_vim9_script.vim

Patch 8.2.3125

Problem: Variables are set but not used.
Solution: Move the declarations to the block where they are used. (closes #8527)
Files: src/regexp_nfa.c

Patch 8.2.3126

Problem: Vim9: for loop error reports wrong line number.
Solution: Save and restore the line number when evaluating the expression. (closes #8514)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3127

Problem: Vim9: no error when adding number to list of string.
Solution: Check the value type. (closes #8529)
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3128

Problem: Vim9: uninitialized list does not get type checked.
Solution: Set the type when initializing the variable. (closes #8529)
Files: src/eval.c, src/evalvars.c, src/vim9script.c,
src/userfunc.c, src/proto/vim9script.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.3129

Problem: Vim9: imported uninitialized list does not get type checked.
Solution: Get type from imported variable.
Files: src/eval.c, src/evalvars.c, src/vim9script.c,
src/proto/vim9script.pro, src/userfunc.c,
src/testdir/test_vim9_script.vim

Patch 8.2.3130

Problem: Vim9: import test fails.
Solution: Rename directory back to "import", use "p" to avoid an error when the directory already exists.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3131

Problem: MS-Windows: ipv6 channel test is very flaky in the GUI.
Solution: Skip the test.
Files: src/testdir/test_channel.vim

Patch 8.2.3132

Problem: Compiler warns for size_t to colnr_T conversion. (Randall W. Morris)
Solution: Add a type cast.
Files: src/drawscreen.c

Patch 8.2.3133

Problem: Vim9: memory leak when add() fails.
Solution: Allocate listitem_T after type check.
Files: src/list.c

Patch 8.2.3134

Problem: Crash when using typename() on a function reference. (Naohiro Ono)
Solution: Initialize pointer to NULL. (closes #8531)
Files: src/vim9type.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3135

Problem: Vim9: builtin function arguments not checked at compile time.
Solution: Add more type checks. (Yegappan Lakshmanan, closes #8539)
Files: src/channel.c, src/errors.h, src/evalfunc.c, src/proto/typval.pro, src/terminal.c, src/testdir/test_search.vim, src/testdir/test_textprop.vim, src/testdir/test_vim9_builtin.vim, src/testing.c, src/textprop.c, src/typval.c

Patch 8.2.3136

Problem: No test for E187 and "No swap file".
Solution: Add a test. (Dominique Pellé, closes #8540)
Files: src/testdir/test_cd.vim, src/testdir/test_swap.vim

Patch 8.2.3137

Problem: Vim9: no error when a line only has a variable name.
Solution: Give an error when an expression is evaluated without an effect. (closes #8538)
Files: src/ex_eval.c, src/errors.h, src/vim9compile.c, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.3138 (after 8.2.3137)

Problem: Debugger test fails.
Solution: Adjust eval command.
Files: src/testdir/test_debugger.vim

Patch 8.2.3139

Problem: Functions for string manipulation are spread out.
Solution: Move string related functions to a new source file. (Yegappan Lakshmanan, closes #8470)
Files: Filelist, src/Make_ami.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/eval.c, src/evalfunc.c, src/mbyte.c, src/misc1.c, src/misc2.c, src/proto.h, src/proto/eval.pro, src/proto/evalfunc.pro,

src/proto/mbyte.pro, src/proto/misc1.pro, src/proto/misc2.pro,
src/proto/strings.pro, src/strings.c

Patch 8.2.3140 (after 8.2.3131)

Problem: MS-Windows: ipv6 channel test is very flaky also without the GUI.
Solution: Skip the test also without the GUI.
Files: src/testdir/test_channel.vim

Patch 8.2.3141

Problem: No error when using :complete for :command without -nargs.
Solution: Give an error. (Martin Tournoij, closes #8544, closes #8541)
Files: src/usercmd.c, src/errors.h, src/testdir/test_usercommands.vim

Patch 8.2.3142

Problem: Vim9: type check for has_key() argument is too strict.
Solution: Also allow for a number key argument. (closes #8542)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3143

Problem: Vim9: A lambda may be compiled with the wrong context if it is
called from a profiled function.
Solution: Compile the lambda with and without profiling. (closes #8543)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3144

Problem: Vim9: no error when using an invalid value for a line number.
Solution: Give an error if the string value is not recognized.
(closes #8536)
Files: src/errors.h, src/eval.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3145

Problem: Vim9: profile test fails without profile feature.
Solution: Check the profile feature is present.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3146

Problem: Vim9: line number wrong for :execute argument.
Solution: Use the line number of the :execute command itself. (closes #8537)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3147

Problem: Vim9: profiling does not work with a nested function.
Solution: Also compile a nested function without profiling. (closes #8543)
Handle that compiling may cause the table of compiled functions to
change.
Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_script.vim

Patch 8.2.3148

Problem: Vim9: function arg type check does not handle base offset.
Solution: Take the base offset into account when checking builtin function
argument types.
Files: src/evalfunc.c, src/vim9compile.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.3149 (after 8.2.3141)

Problem: Some plugins have a problem with the error check for using
:command with -complete but without -nargs.

Solution: In legacy script only give a warning message.

Files: src/usercmd.c, src/message.c, src/proto/message.pro,
src/testdir/test_usercommands.vim

Patch 8.2.3150

Problem: Vim9: argument types are not checked at compile time.

Solution: Add more type checks. (Yegappan Lakshmanan, closes #8545)

Files: src/evalfunc.c, src/testing.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3151

Problem: Vim9: profiling fails if nested function is also profiled.

Solution: Use the compile type from the outer function. (closes #8543)

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3152

Problem: Vim9: accessing "s:" results in an error.

Solution: Do not try to lookup a script variable for "s:". (closes #8549)

Files: src/evalvars.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3153

Problem: URLs with a dash in the scheme are not recognized.

Solution: Allow for a scheme with a dash, but not at the start or end.
(Tsuyoshi CHO, closes #8299)

Files: src/misc1.c, src/testdir/test_buffer.vim

Patch 8.2.3154

Problem: Vim9: some type checks for builtin functions fail.

Solution: Correct the type checks. (Yegappan Lakshmanan, closes #8551,
closes #8550)

Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3155

Problem: Some option related code not covered by tests.

Solution: Add a few test cases. (Dominique Pellé, closes #8552)

Files: src/testdir/test_options.vim, src/testdir/test_set.vim

Patch 8.2.3156

Problem: Vim9: term_getansicolors() test fails without +termguicolors.

Solution: Add a check for the feature. (Dominique Pellé, closes #8555)

Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3157

Problem: Crypt test may fail on MS-Windows.

Solution: Ignore "[unix]" in the file message. (Christian Brabandt,
closes #8561)

Files: src/testdir/test_crypt.vim

Patch 8.2.3158

Problem: Strange error message when using islocked() with a number.
(Yegappan Lakshmanan)

Solution: Check that the name is empty.
Files: src/evalfunc.c, src/testdir/test_functions.vim

Patch 8.2.3159

Problem: Cursor displayed in wrong position after deleting line.
Solution: When deleting lines do not approximate botline. (fixes #8559)
Files: src/change.c

Patch 8.2.3160

Problem: **'breakindent'** does not work well for bulleted and numbered lists.
Solution: Add the "list" entry to **'breakindentopt'**. (Christian Brabandt, closes #8564, closes #1661)
Files: runtime/doc/options.txt, src/indent.c, src/structs.h, src/testdir/test_breakindent.vim

Patch 8.2.3161

Problem: Vim9: no error when reltime() has invalid arguments.
Solution: Add an error. (closes #8562)
Files: src/time.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3162

Problem: Vim9: argument types are not checked at compile time.
Solution: Add more type checks. (Yegappan Lakshmanan, closes #8560)
Files: runtime/doc/channel.txt, src/clientserver.c, src/cmdhist.c, src/errors.h, src/evalfunc.c, src/evalwindow.c, src/filepath.c, src/globals.h, src/popupwin.c, src/proto/typval.pro, src/sign.c, src/strings.c, src/terminal.c, src/testdir/test_normal.vim, src/testdir/test_reftime.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim, src/testing.c, src/textprop.c, src/time.c, src/typval.c

Patch 8.2.3163

Problem: Location list window may open a wrong file.
Solution: Also update the qf_ptr field. (Wei-Chung Wen, closes #8565, closes #8566)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.3164

Problem: MS-Windows: reported version lacks patchlevel, causing some update tools to update too often. (Klaus Frank)
Solution: Add the patchlevel to the version. (Christian Brabandt)
Files: src/dosinst.c

Patch 8.2.3165

Problem: Vim9: in a || expression the error line number may be wrong.
Solution: Save and restore the line number when checking the type. (closes #8569)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3166

Problem: Vim9: nested autoload call error overruled by "Unknown error".
Solution: Check need_rethrow before giving an "Unknown error". (closes #8568)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3167

Problem: Get E12 in a job callback when searching for tags. (Andy Stewart)
Solution: Use the sandbox only for executing a command, not for searching.
(closes #8511)
Files: src/tag.c

Patch 8.2.3168

Problem: Vim9: type error for constant of type any.
Solution: Do add a runtime type check if a constant has type any.
(closes #8570)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3169

Problem: Vim9: cannot handle nested inline function.
Solution: Check for nested inline function. (closes #8575)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.3170

Problem: Illegal memory access in test.
Solution: Check pointer is not before the start of the line.
Files: src/userfunc.c

Patch 8.2.3171

Problem: Another illegal memory access in test.
Solution: Check pointer is after the start of the line.
Files: src/userfunc.c

Patch 8.2.3172

Problem: MzScheme test fails. (Christian Brabandt)
Solution: Correct function name.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3173

Problem: Vim9: argument types are not checked at compile time.
Solution: Add more type checks. (Yegappan Lakshmanan, closes #8581)
Files: src/diff.c, src/errors.h, src/evalfunc.c, src/globals.h,
src/job.c, src/proto/typval.pro, src/strings.c, src/terminal.c,
src/testdir/test_vim9_builtin.vim, src/typval.c

Patch 8.2.3174

Problem: Vim9: "legacy undo" finds "undo" variable.
Solution: Do not pass lookup function to find_ex_command(). (closes #8563)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3175

Problem: Vim9: using illegal pointer with inline function inside a lambda.
Solution: Clear eval_tofree_cmdline when advancing to the next line.
(closes #8578)
Files: src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.3176

Problem: Vim9: no type error for comparing number with string.

Solution: Add a runtime type check. (closes #8571)
Files: src/typval.c, src/errors.h, src/testdir/test_vim9_expr.vim

Patch 8.2.3177

Problem: Vim9: can not use "for _ in expr" at script level.
Solution: Skip assignment if the loop variable is "_".
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3178

Problem: Vim9: the file name of an :import cannot be an expression.
Solution: Accept an expression that results in a string. Do not support :import in a function.
Files: runtime/doc/vim9.txt, src/vim9script.c, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3179

Problem: Vim9: cannot assign to an imported variable at script level.
Solution: Lookup imported items when assigning.
Files: src/evalvars.c, src/errors.h, src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3180

Problem: Vim9: memory leak when concatenating to an imported string.
Solution: Clear the destination.
Files: src/evalvars.c

Patch 8.2.3181

Problem: Vim9: builtin function test fails without channel feature.
Solution: Add feature checks. (Dominique Pellé, closes #8586) Make feature checks more consistent.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3182

Problem: Vim9: crash when using removing items from a constant list. (Yegappan Lakshmanan)
Solution: When a list was allocated with items copy them.
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3183

Problem: Duplicate error numbers.
Solution: Adjust the error numbers.
Files: src/errors.h, src/testdir/test_vim9_builtin.vim

Patch 8.2.3184

Problem: Cannot add a digraph with a leading space. It is not easy to list existing digraphs.
Solution: Add setdigraph(), setdigraphlist(), getdigraph() and getdigraphlist(). (closes #8580)
Files: runtime/doc/digraph.txt, runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/digraph.c, src/evalfunc.c, src/ex_docmd.c, src/globals.h, src/errors.h, src/proto/digraph.pro, src/testdir/test_digraph.vim

Patch 8.2.3185

Problem: Vim9: start of inline function found in comment line.
Solution: Do not check for inline function in comment line. (closes #8589)
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3186

Problem: Vim9: not all failures for import tested
Solution: Test more import failures
Files: src/errors.h, src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.3187

Problem: Vim9: popup timer callback is not compiled.
Solution: Compile the callback when creating the timer.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/popupwin.c

Patch 8.2.3188

Problem: Vim9: argument types are not checked at compile time.
Solution: Add several more type checks, also at runtime. (Yegappan Lakshmanan, closes #8587)
Files: src/blob.c, src/channel.c, src/clientserver.c, src/cmdexpand.c, src/cmdhist.c, src/dict.c, src/diff.c, src/errors.h, src/eval.c, src/evalbuffer.c, src/evalfunc.c, src/evalvars.c, src/evalwindow.c, src/filepath.c, src/globals.h, src/insexpand.c, src/job.c, src/list.c, src/map.c, src/match.c, src/proto/typval.pro, src/quickfix.c, src/search.c, src/sign.c, src/strings.c, src/terminal.c, src/testdir/test_blob.vim, src/testdir/test_gui.vim, src/testdir/test_vim9_builtin.vim, src/testing.c, src/textprop.c, src/time.c, src/typval.c

Patch 8.2.3189

Problem: Vim9: error when using "try|".
Solution: Allow for "|" right after a command.
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.3190

Problem: Error messages are spread out.
Solution: Move error messages to errors.h and give them a clear name.
Files: src/globals.h, src/errors.h, src/buffer.c, src/debugger.c, src/digraph.c, src/edit.c, src/ex_cmds.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c, src/gui.c, src/list.c, src/main.c, src/map.c, src/match.c, src/quickfix.c, src/regexp.c, src/regexp_bt.c, src/regexp_nfa.c, src/register.c, src/search.c, src/session.c, src/spell.c, src/syntax.c, src/time.c, src/userfunc.c, src/vim9execute.c, src/window.c

Patch 8.2.3191

Problem: Vim9: not enough code is tested.
Solution: Use CheckLegacyAndVim9Success() in more places. Fix uncovered problems.
Files: src/vim9compile.c, src/vim9execute.c, src/testdir/test_listdict.vim

Patch 8.2.3192 (after 8.2.3190)

Problem: Build failure with small version (Tony Mechelynck).
Solution: Remove stray #ifdef.

Files: src/errors.h

Patch 8.2.3193

Problem: screenpos() is wrong when the last line is partially visible and **'display'** is "lastline".

Solution: Also compute the position for a partially visible line.
(closes #8599)

Files: src/move.c, src/testdir/test_cursor_func.vim

Patch 8.2.3194

Problem: Vim9: argument types are not checked at compile time.

Solution: Add several more type checks, simplify some. (Yegappan Lakshmanan, closes #8598)

Files: src/diff.c, src/evalbuffer.c, src/evalfunc.c, src/job.c,
src/proto/typval.pro, src/sign.c, src/terminal.c,
src/testdir/test_vim9_builtin.vim, src/typval.c

Patch 8.2.3195

Problem: Vim9: unclear error when passing too many arguments to lambda.

Solution: Pass the expression itself instead of "[expression]".
(closes #8604)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.3196

Problem: Vim9: bool expression with numbers only fails at runtime.

Solution: Check constant to be bool at compile time. (closes #8603)

Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3197

Problem: Error messages are spread out.

Solution: Move a few more error messages to errors.h.

Files: src/globals.h, src/errors.h, src/edit.c, src/ex_cmds.c,
src/ex_docmd.c, src/evalvars.c, src/option.c, src/quickfix.c,
src/regexp_bt.c, src/regexp_nfa.c, src/regexp.c, src/undo.c,
src/vim9compile.c, src/vim9script.c

Patch 8.2.3198

Problem: Cannot use **'formatlistpat'** for breakindent.

Solution: Use a negative list indent. (Maxim Kim, closes #8594)

Files: runtime/doc/options.txt, src/indent.c,
src/testdir/test_breakindent.vim

Patch 8.2.3199

Problem: Vim9: execution speed can be improved.

Solution: Make the break counter static.

Files: src/vim9execute.c

Patch 8.2.3200

Problem: Vim9: hard to guess where a type error is given.

Solution: Add the function name where possible. (closes #8608)

Files: src/dict.c, src/proto/dict.pro, src/eval.c, src/list.c,
src/vim9compile.c, src/vim9execute.c, src/structs.h,
src/vim9type.c, src/proto/vim9type.pro, src/if_py_both.h,
src/errors.h, src/testdir/test_vim9_builtin.vim

Patch 8.2.3201 (after 8.2.3200)

Problem: Crash in test.
Solution: Initialize "where".
Files: src/eval.c, src/evalvars.c

Patch 8.2.3202

Problem: Vim9: tests are only executed for legacy script.
Solution: Run more tests also for Vim9 script. Fix uncovered problems.
Files: src/vim9execute.c, src/ex_docmd.c, src/testdir/test_listdict.vim

Patch 8.2.3203

Problem: Vim9: compiled string expression causes type error. (Yegappan Lakshmanan)
Solution: Remove the string type from the stack.
Files: src/vim9compile.c, src/evalfunc.c

Patch 8.2.3204

Problem: Display garbled when '**cursorline**' is set and lines wrap. (Gabriel Dupras)
Solution: Avoid inserting lines twice. (closes #7255)
Files: src/drawscreen.c, src/testdir/test_cursorline.vim,
src/testdir/dumps/Test_cursorline_redraw_1.dump,
src/testdir/dumps/Test_cursorline_redraw_2.dump

Patch 8.2.3205

Problem: Coverity reports a null pointer dereference.
Solution: Change the logic to avoid Coverity gets confused.
Files: src/vim9compile.c

Patch 8.2.3206

Problem: Vim9: argument types are not checked at compile time.
Solution: Add several more type checks. (Yegappan Lakshmanan, closes #8611)
Files: runtime/doc/eval.txt, src/blob.c, src/cmdhist.c, src/dict.c,
src/errors.h, src/evalfunc.c, src/filepath.c, src/globals.h,
src/job.c, src/list.c, src/match.c, src/misc1.c, src/popupwin.c,
src/proto/typval.pro, src/sign.c, src/terminal.c,
src/testdir/test_blob.vim, src/testdir/test_vim9_builtin.vim,
src/typval.c

Patch 8.2.3207

Problem: Vim9: crash when compiling string fails. (Yegappan Lakshmanan)
Solution: Adjust the type stack length.
Files: src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3208

Problem: Dynamic library load error does not mention why it failed.
Solution: Add the error message. (Martin Tournoij, closes #8621)
Files: src/globals.h, src/if_cscope.c, src/if_lua.c, src/if_mzsch.c,
src/if_perl.xs, src/if_python.c, src/if_python3.c, src/if_ruby.c,
src/if_tcl.c, src/mbyte.c, src/os_win32.c, src/proto/os_win32.pro,
src/terminal.c

Patch 8.2.3209

Problem: Vim9: lambda doesn't find block-local variable.
Solution: Adjust how a script-local variable is found. (closes #8614)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.3210

Problem: Vim9: searchpair() sixth argument is compiled. (Yegappan Lakshmanan)
Solution: Only compile the fifth argument.
Files: src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3211

Problem: Vim9: argument types are not checked at compile time.
Solution: Add several more type checks. Fix type check for matchaddpos(). (Yegappan Lakshmanan, closes #8619)
Files: src/channel.c, src/evalfunc.c, src/evalvars.c, src/if_cscope.c, src/job.c, src/proto/typval.pro, src/testdir/test_vim9_builtin.vim, src/time.c, src/typval.c

Patch 8.2.3212

Problem: Vim9: execution speed can be improved.
Solution: Use __builtin_expect() to have the compiler produce better code. (Dominique Pellé, closes #8613)
Files: src/vim9execute.c

Patch 8.2.3213

Problem: NOCOMPOUNDSUGS entry in spell file not tested.
Solution: Add a test. (Dominique Pellé, closes #8624)
Files: src/testdir/test_spellfile.vim

Patch 8.2.3214

Problem: MS-Windows: passing /D does not set the install location.
Solution: Adjust how the installer uses \$VIM. Update the documentation. (Christian Brabandt, Ken Takata, closes #8605)
Files: nsis/gvim.nsi, runtime/doc/os_win32.txt

Patch 8.2.3215

Problem: Vim9: argument types are not checked at compile time.
Solution: Add several more type checks. Sort the argument lists. (Yegappan Lakshmanan, closes #8626)
Files: src/change.c, src/evalfunc.c, src/filepath.c, src/sound.c, src/testdir/test_gui.vim, src/testdir/test_vim9_builtin.vim, src/testing.c

Patch 8.2.3216

Problem: Vim9: crash when using variable in a loop at script level.
Solution: Do not clear the variable if a function was defined. Do not create a new entry in sn_var_vals every time. (closes #8628)
Files: src/eval.c, src/ex_eval.c, src/vim9script.c, src/userfunc.c, src/evalvars.c, src/structs.h

Patch 8.2.3217 (after 8.2.3216)

Problem: Build failure.
Solution: Add missing changes.

Files: src/globals.h

Patch 8.2.3218

Problem: When using xchacha20 crypt undo file is not removed.

Solution: Reset **'undofile'** and delete the file. (Christian Brabandt, closes #8630, closes #8467)

Files: src/bufwrite.c, src/crypt.c, src/proto/undo.pro, src/testdir/test_crypt.vim, src/undo.c

Patch 8.2.3219

Problem: :find searches non-existing directories.

Solution: Check the path is not "..". Update help. (Christian Brabandt, closes #8612, closes #8533)

Files: runtime/doc/editing.txt, src/findfile.c, src/testdir/test_findfile.vim

Patch 8.2.3220

Problem: Test_term_setansicolors() fails in some configurations.

Solution: Check available features. (Dominique Pellé, closes #8636)

Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3221

Problem: Vim9: argument types are not checked at compile time.

Solution: Add several more type checks. (Yegappan Lakshmanan, closes #8632)

Files: src/evalfunc.c, src/popupwin.c, src/proto/typval.pro, src/testdir/test_assert.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_script.vim, src/testing.c, src/typval.c

Patch 8.2.3222

Problem: Vim9: cannot use loop variable later as lambda argument.

Solution: When not in function context check the current block ID. (closes #8637)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.3223

Problem: Vim: using {} block in autoloading omnifunc fails.

Solution: Allow using {} block when text is locked. (closes #8631)

Files: src/ex_cmds.h, src/testdir/test_ins_complete.vim

Patch 8.2.3224

Problem: Cannot call script-local function after :vim9cmd. (Christian J. Robinson)

Solution: Skip over "<SNR>123".

Files: src/vim9compile.c, src/eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3225

Problem: Incsearch highlighting is attempted halfway a mapping.

Solution: Only do incsearch highlighting if keys were typed or there is no more typeahead.

Files: src/ex_getln.c

Patch 8.2.3226

Problem: New digraph functions use old naming scheme.

Solution: Use the digraph_ prefix. (Hirohito Higashi, closes #8642)

Files: runtime/doc/digraph.txt, runtime/doc/eval.txt,
runtime/doc/usr_41.txt, src/digraph.c, src/edit.c, src/errors.h,
src/evalfunc.c, src/proto/digraph.pro,
src/testdir/test_digraph.vim

Patch 8.2.3227

Problem: **'virtualedit'** can only be set globally.
Solution: Make **'virtualedit'** global-local. (Gary Johnson, closes #8638)
Files: runtime/doc/options.txt, src/buffer.c, src/change.c,
src/drawscreen.c, src/edit.c, src/misc2.c, src/normal.c,
src/ops.c, src/option.c, src/option.h, src/optiondefs.h,
src/optionstr.c, src/proto/option.pro, src/register.c,
src/structs.h, src/testdir/test_virtualedit.vim

Patch 8.2.3228

Problem: Cannot use a simple block for the :command argument. (Maarten Tournouij)
Solution: Recognize a simple {} block. (issue #8623)
Files: runtime/doc/map.txt, src/misc2.c, src/proto/misc2.pro,
src/usercmd.c, src/testdir/test_usercommands.vim

Patch 8.2.3229

Problem: Vim9: runtime and compile time type checks are not the same.
Solution: Add more runtime type checks for builtin functions. (Yegappan Lakshmanan, closes #8646)
Files: src/arglist.c, src/change.c, src/channel.c, src/cindent.c,
src/clientserver.c, src/cmdhist.c, src/dict.c, src/diff.c,
src/digraph.c, src/errors.h, src/eval.c, src/evalbuffer.c,
src/evalfunc.c, src/evalwindow.c, src/ex_docmd.c, src/ex_getln.c,
src/filepath.c, src/findfile.c, src/float.c, src/fold.c,
src/getchar.c, src/indent.c, src/insexpand.c, src/job.c,
src/json.c, src/list.c, src/mark.c, src/match.c, src/mbyte.c,
src/menu.c, src/misc1.c, src/move.c, src/popupwin.c,
src/proto/typval.pro, src/quickfix.c, src/search.c, src/sign.c,
src/sound.c, src/strings.c, src/terminal.c,
src/testdir/test_assert.vim, src/testdir/test_blob.vim,
src/testdir/test_execute_func.vim,
src/testdir/test_float_func.vim, src/testdir/test_functions.vim,
src/testdir/test_glob2regpat.vim, src/testdir/test_listdict.vim,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_script.vim, src/testing.c, src/textprop.c,
src/time.c, src/typval.c, src/undo.c

Patch 8.2.3230

Problem: Vim9: type error when function return type is not known yet.
Solution: When return type is unknown, use "any". (closes #8644)
Files: src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3231

Problem: Build failure with small features.
Solution: Adjust #ifdef.
Files: src/errors.h

Patch 8.2.3232 (after 8.2.3229)

Problem: system() does not work without a second argument.
Solution: Do not require a second argument. (Yegappan Lakshmanan, closes #8651, closes #8650)
Files: src/misc1.c, src/proto/typval.pro, src/testdir/test_vim9_builtin.vim, src/typval.c

Patch 8.2.3233

Problem: prop_list() and prop_find() do not indicate the buffer for the used type.
Solution: Add "type_bufnr" to the results. (closes #8647)
Files: runtime/doc/textprop.txt, src/testdir/test_textprop.vim, src/textprop.c

Patch 8.2.3234

Problem: Crash when printing long string with Lua.
Solution: Remove lua_pop(). (Martin Tournoij, closes #8648)
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.3235

Problem: Cannot use lambda in {} block in user command. (Martin Tournoij)
Solution: Do not go over the end of the lambda.
Files: src/userfunc.c, src/testdir/test_usercommands.vim

Patch 8.2.3236

Problem: mode() does not indicate using **CTRL-O** in Select mode.
Solution: Use "vs" and similar. (closes #8640)
Files: runtime/doc/eval.txt, src/globals.h, src/misc1.c, src/normal.c, src/testdir/test_functions.vim

Patch 8.2.3237

Problem: When a builtin function gives an error processing continues.
Solution: In Vim9 script return FAIL in get_func_tv().
Files: src/userfunc.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3238

Problem: Vim9: error message does not indicate the location.
Solution: Add the relevant text. (issue #8634)
Files: src/errors.h, src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3239

Problem: Vim9: no error using heredoc for a number variable.
Solution: Add a type check. (closes #8627)
Files: src/vim9compile.c, src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3240

Problem: Lua print() does not work properly.
Solution: Put back lua_pop().
Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.3241

Problem: Vim9: memory leak when function reports an error.
Solution: Clear the return value.
Files: src/userfunc.c

Patch 8.2.3242

Problem: Vim9: valgrind reports leaks in builtin function test.
Solution: Do not start a job.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3243

Problem: MS-Windows: the "edit with multiple Vim" choice is not that useful.
Solution: Change it to "Edit with multiple tabs". (Michael Soyka, closes #8645)
Files: src/GvimExt/gvimext.cpp, src/GvimExt/gvimext.h

Patch 8.2.3244

Problem: Lua 5.3 print() with a long string crashes.
Solution: Use a growarray instead of a Lua buffer. (Yegappan Lakshmanan, closes #8655)
Files: src/if_lua.c, src/misc2.c, src/proto/misc2.pro

Patch 8.2.3245

Problem: The crypt key may appear in a swap partition.
Solution: When using xchacha20 use sodium_mlock(). (Christian Brabandt, closes #8657)
Files: src/buffer.c, src/crypt.c, src/errors.h, src/fileio.c, src/memline.c, src/vim.h

Patch 8.2.3246

Problem: Memory use after free.
Solution: When clearing a string option set the pointer to "empty_option".
Files: src/option.c

Patch 8.2.3247

Problem: Using uninitialized memory when checking for crypt method.
Solution: Check the header length before using the salt and seed.
Files: src/fileio.c

Patch 8.2.3248

Problem: Vim9: error message for wrong input uses wrong line number.
Solution: Use the line number of the start of the command. (issue #8653)
Files: src/vim9script.c, src/testdir/test_vim9_script.vim

Patch 8.2.3249

Problem: Vim9: error for re-imported function with default argument.
Solution: Do not check argument type if it is still unknown. (closes #8653)
Files: src/vim9type.c, src/proto/vim9type.pro, src/vim9script.c, src/vim.h, src/eval.c, src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3250

Problem: MS-Windows: cannot build with libsodium.
Solution: Change FEAT_SODIUM into HAVE_SODIUM. (Christian Brabandt, closes #8668, closes #8663)
Files: src/Make_mvc.mak

Patch 8.2.3251

Problem: Listing builtin_gui as an available terminal is confusing.
Solution: Do not list builtin_gui. (Christian Brabandt, closes #8669, closes #8661)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.3252

Problem: Duplicated code for adding buffer lines.
Solution: Move code to a common function. Also move map functions to map.c. (Yegappan Lakshmanan, closes #8665)
Files: src/evalbuffer.c, src/evalfunc.c, src/map.c, src/proto/map.pro

Patch 8.2.3253

Problem: Channel test fails randomly.
Solution: Add a sleep after sending the "echoerr" command. (Michael Soyka)
Files: src/testdir/test_channel.vim, src/testdir/test_channel.py

Patch 8.2.3254

Problem: win_gettype() does not recognize a quickfix window.
Solution: Add "quickfix" and "loclist". (Yegappan Lakshmanan, closes #8676)
Files: runtime/doc/eval.txt, src/evalwindow.c, src/misc2.c, src/testdir/test_quickfix.vim

Patch 8.2.3255

Problem: ci" finds following string but ci< and others don't.
Solution: When not inside an object find the start. (Connor Lane Smit, closes #8670)
Files: src/search.c, src/testdir/test_textobjects.vim, src/textobject.c

Patch 8.2.3256

Problem: Executable test may fail on new Ubuntu system.
Solution: Consider /usr/bin/cat and /bin/cat the same.
Files: src/testdir/test_functions.vim

Patch 8.2.3257

Problem: Calling prop_find() with -1 for ID gives erroneous error. (Naohiro Ono)
Solution: When passing -1 use -2. (closes #8674)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.3258

Problem: Error messages have the wrong text.
Solution: Adjust the error message.
Files: src/errors.h, src/typval.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3259

Problem: When **'indentexpr'** causes an error the did_throw flag may remain set.
Solution: Reset did_throw and show the error. (closes #8677)
Files: src/indent.c, src/ex_docmd.c, src/proto/ex_docmd.pro

Patch 8.2.3260

Problem: Build failure with small features.
Solution: Add #ifdef.

Files: src/ex_docmd.c

Patch 8.2.3261

Problem: Vim9: when compiling repeat(123, N) return type is number.

Solution: Make return type a string. (closes #8664)

Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3262

Problem: Build failure when ABORT_ON_INTERNAL_ERROR is defined.

Solution: Adjust how estack_len_before is used.

Files: src/ex_docmd.c

Patch 8.2.3263

Problem: Vim9: "..=" does not accept same types as the ".." operator.

Solution: Convert value to string like ".." does. (issue #8664)

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3264 (after 8.2.3263)

Problem: Vim9: assign test fails.

Solution: Add missing change.

Files: src/eval.c

Patch 8.2.3265

Problem: Smartcase does not work correctly in very magic pattern.

Solution: Take the magicness into account when skipping over regexp items.
(Christian Brabandt, closes #8682, closes #7845)

Files: src/search.c, src/testdir/test_search.vim

Patch 8.2.3266

Problem: Vim9: assignment with two indexes may check next line.

Solution: Limit the number of lines to avoid checking the next line when
assigning to a LHS subscript. (closes #8660)

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3267

Problem: Vim9: crash when disassembling a function that uses a deleted
script variable.

Solution: Check the variable still exists. (closes #8683)

Files: src/vim9execute.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3268

Problem: Cannot use a block with :autocmd like with :command.

Solution: Add support for a {} block after :autocmd. (closes #8620)

Files: runtime/doc/autocmd.txt, runtime/doc/map.txt, src/autocmd.c,
src/proto/autocmd.pro, src/usercmd.c, src/proto/usercmd.pro,
src/ex_docmd.c, src/vim.h, src/testdir/test_autocmd.vim

Patch 8.2.3269

Problem: Vim9: wrong argument check for partial. (Naohiro Ono)

Solution: Handle getting return type without arguments. Correct the minimal
number of arguments for what is included in the partial.
(closes #8667)

Files: src/evalfunc.c, src/vim9type.c, src/testdir/test_vim9_func.vim

Patch 8.2.3270

Problem: prop_find() finds property with ID -2.
Solution: Use a separate flag to indicate an ID was specified. (issue #8674)
Files: src/textprop.c

Patch 8.2.3271

Problem: Vim9: cannot use :command or :au with a block in a :def function.
Solution: Recognize the start of the block.
Files: src/userfunc.c, src/usercmd.c, src/ex_docmd.c,
src/proto/ex_docmd.pro, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.3272

Problem: Cannot use id zero with prop_find(). (Naohiro Ono)
Solution: Also accept id zero.
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.3273

Problem: Autocmd test fails.
Solution: Require white space before the "{" that starts a block.
Files: src/userfunc.c

Patch 8.2.3274

Problem: Macro for printf format check can be simplified.
Solution: Add ATTRIBUTE_FORMAT_PRINTF(). (Dominique Pellé, issue #8635)
Files: src/channel.c, src/gui_xim.c, src/if_mzsch.c, src/nbdebug.c,
src/nbdebug.h, src/netbeans.c, src/proto.h, src/term.c, src/vim.h,
src/vim9execute.c

Patch 8.2.3275

Problem: Optimizer can use hints about ga_grow() normally succeeding.
Solution: Use GA_GROW_FAILS() and GA_GROW_OK() in several places. (Dominique Pellé, issue #8635)
Files: src/arglist.c, src/macros.h, src/vim9execute.c, src/vim9compile.c

Patch 8.2.3276

Problem: Vim9: exists() can only be evaluated at runtime.
Solution: Evaluate at compile time for option name literals. (closes #8437)
Files: src/vim9compile.c, src/evalfunc.c, src/proto/evalfunc.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.3277 (after 8.2.3276)

Problem: Vim9: compiled has() does not work properly.
Solution: Fix check for has() vs exists().
Files: src/vim9compile.c

Patch 8.2.3278

Problem: Vim9: error when adding 1 to float.
Solution: Accept t_number_bool. (closes #8687)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3279

Problem: Vim9: cannot use block in cmdline window.

Solution: Add EX_CMDWIN to the CMD_block flags. (closes #8689)
Files: src/ex_cmds.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.3280

Problem: **'virtualedit'** local to buffer is not the best solution.
Solution: Make it window-local. (Gary Johnson, closes #8685)
Files: runtime/doc/options.txt, src/buffer.c, src/drawscreen.c,
src/ops.c, src/option.c, src/option.h, src/optionstr.c,
src/structs.h, src/testdir/test_virtualedit.vim

Patch 8.2.3281

Problem: Vim9: TODO items in tests can be taken care of.
Solution: Update test for now working functionality. (closes #8694)
Files: src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.3282

Problem: Vim9: error about using -complete without -nargs is confusing.
Solution: Change the wording.
Files: src/usercmd.c, src/errors.h

Patch 8.2.3283

Problem: Julia filetype is not recognized
Solution: Add filetype detection. (Christian Clason, closes #8700)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3284

Problem: No error for insert() or remove() changing a locked blob.
Solution: Check a blob is not locked before changing it. (Sean Dewar,
closes #8696)
Files: src/blob.c, src/errors.h, src/eval.c, src/list.c,
src/proto/blob.pro, src/testdir/test_blob.vim,
src/testdir/test_eval_stuff.vim

Patch 8.2.3285

Problem: Scdoc filetype is not recognized.
Solution: Add filetype detection. (Gregory Anders, closes #8701)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3286

Problem: win_enter_ext() has too many boolean arguments.
Solution: use one flags argument with defined values.
Files: src/window.c

Patch 8.2.3287

Problem: Channel events not handled in BufEnter autocommand.
Solution: Decrement dont_parse_messages earlier. (Tim Pope, closes #8697)
Files: src/window.c, src/testdir/test_channel.vim

Patch 8.2.3288

Problem: Cannot easily access namespace dictionaries from Lua.
Solution: Add vim.g, vim.b, etc. (Yegappan Lakshmanan, closes #8693,
from NeoVim)
Files: runtime/doc/if_lua.txt, src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.3289 (after 8.2.3287)

Problem: Compiler warning for unused variable with small features.

Solution: Rearrange #ifdefs.

Files: src/window.c

Patch 8.2.3290

Problem: Vim9: compiling dict may use pointer after free and leak memory on failure.

Solution: Pass a pointer to generate_PUSHES(). (Zdenek Dohnal, closes #8699)

Files: src/vim9compile.c

Patch 8.2.3291

Problem: Coverity warns for not checking return value.

Solution: If dict_add() fails give an error message.

Files: src/if_lua.c, src/testdir/test_lua.vim

Patch 8.2.3292

Problem: Underscore in very magic pattern causes a hang. Pattern with \V are case sensitive. (Yutao Yuan)

Solution: Adjust condition for magicness and advance pointer. (Christian Brabandt, closes #8707, closes #8704, closes #8705)

Files: src/search.c, src/testdir/test_search.vim

Patch 8.2.3293

Problem: Finding completions may cause an endless loop.

Solution: Use a better way to check coming back where the search started. (Andy Gozas, closes #8672, closes #8671)

Files: src/insexpand.c, src/testdir/Make_all.mak, src/testdir/test_ins_complete_no_halt.vim

Patch 8.2.3294

Problem: Lua: memory leak when adding dict item fails.

Solution: Free the typval and the dict item.

Files: src/if_lua.c

Patch 8.2.3295

Problem: **'cursorline'** should not apply to **'breakindent'**.

Solution: Make **'cursorline'** apply to **'breakindent'** and **'showbreak'** consistently. (closes #8684)

Files: src/drawline.c, src/testdir/dumps/Test_Xcursorline_19.dump, src/testdir/dumps/Test_Xcursorline_20.dump, src/testdir/dumps/Test_Xcursorline_21.dump, src/testdir/dumps/Test_Xcursorline_22.dump, src/testdir/dumps/Test_Xcursorline_23.dump, src/testdir/dumps/Test_Xcursorline_24.dump, src/testdir/dumps/Test_diff_with_cul_bri_01.dump, src/testdir/dumps/Test_diff_with_cul_bri_02.dump, src/testdir/dumps/Test_diff_with_cul_bri_03.dump, src/testdir/dumps/Test_diff_with_cul_bri_04.dump, src/testdir/test_cursorline.vim, src/testdir/test_diffmode.vim

Patch 8.2.3296

Problem: Vim9: cannot add a number to a float.

Solution: Accept a number if the destination is a float. (closes #8703)
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3297

Problem: Cannot use all commands inside a {} block after :command and :autocmd.

Solution: Do consider \n to separate commands. (closes #8620)

Files: runtime/doc/map.txt, src/ex_docmd.c, src/proto/ex_docmd.pro, src/ex_eval.c, src/proto/ex_eval.pro, src/eval.c, src/evalvars.c, src/ex_cmds.c, src/syntax.c, src/userfunc.c, src/vim9compile.c, src/vim9script.c, src/errors.h, src/testdir/test_autocmd.vim, src/testdir/test_usercommands.vim

Patch 8.2.3298

Problem: Build failure with small features.

Solution: Add #ifdef.

Files: src/ex_docmd.c

Patch 8.2.3299

Problem: Vim9: exists() does not handle much at compile time.

Solution: Handle variable names. (closes #8688)

Files: src/vim9compile.c, src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3300

Problem: Lua: can only execute one Vim command at a time. Not easy to get the Vim version.

Solution: Make vim.command() accept multiple lines. Add vim.version(). (Yegappan Lakshmanan, closes #8716)

Files: runtime/doc/if_lua.txt, src/evalfunc.c, src/if_lua.c, src/proto/evalfunc.pro, src/testdir/test_lua.vim, src/testdir/test_shell.vim

Patch 8.2.3301

Problem: Memory allocation functions don't have their own place.

Solution: Move memory allocation functions to alloc.c. (Yegappan Lakshmanan, closes #8717)

Files: Filelist, src/Make_ami.mak, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/README.md, src/alloc.c, src/misc2.c, src/proto.h, src/proto/alloc.pro, src/proto/misc2.pro

Patch 8.2.3302

Problem: Coverity is not run from github.

Solution: Add a coverity script. (James McCoy, closes #8714)

Files: .github/workflows/coverity.yml, Filelist

Patch 8.2.3303

Problem: Some structures could be smaller.

Solution: Rearrange members to reduce size. (Dominique Pellé, closes #8725)

Files: src/structs.h, src/vim9.h, src/vim9execute.c

Patch 8.2.3304

Problem: Popup window title with wide characters is truncated.

Solution: Use vim_strsize() instead of MB_CHARLEN(). (Naruhiko Nishino, closes #8721)

Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_multibytetitle.dump

Patch 8.2.3305

Problem: Vim9: :finally in skipped block not handled correctly.

Solution: Check whether :finally is in a skipped block. (Naruhiko Nishino, closes #8724)

Files: src/ex_eval.c, src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3306

Problem: Unexpected "No matching autocommands".

Solution: Do not give the message when aborting. Mention the arguments in the message. (closes #8690)

Files: src/autocmd.c,

Patch 8.2.3307

Problem: Vim9: :echoconsole cannot access local variables.

Solution: Handle like other :echo commands. (closes #8708)

Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3308

Problem: Vim9: no runtime check for argument type if a function only has varargs.

Solution: Also check argument types if uf_va_type is set. (closes #8715)

Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.3309

Problem: Vim9: divide by zero causes a crash.

Solution: Give an error message. (closes #8727)

Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3310

Problem: Vim9: unpack assignment does not mention source of type error.

Solution: Mention the argument number. (closes #8719)

Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3311

Problem: Vim9: check for DO_NOT_FREE_CNT is very slow.

Solution: Move to a separate function so it can be skipped by setting \$TEST_SKIP_PAT.

Files: src/testdir/test_vim9_expr.vim, src/testdir/runtest.vim

Patch 8.2.3312

Problem: Vim9: after "if false" line breaks in expression not skipped.

Solution: Do parse the expression. (closes #8723)

Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3313

Problem: Unused code in win_exchange() and frame_remove().

Solution: Remove the code. (closes #8728)
Files: src/window.c

Patch 8.2.3314

Problem: Behavior of exists() in a :def function is unpredictable.
Solution: Add exists_compiled().
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/errors.h, src/vim9compile.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3315

Problem: Cannot use single quote in a float number for readability.
Solution: Support single quotes like in numbers. (closes #8713)
Files: src/typval.c, src/float.c, src/proto/float.pro, src/json.c,
src/viminfo.c, src/testdir/test_float_func.vim

Patch 8.2.3316 (after 8.2.3315)

Problem: Float test fails.
Solution: Add missing change.
Files: src/evalfunc.c

Patch 8.2.3317

Problem: Vim9: No error for missing white space before return type.
Solution: Check for white space. (closes #8733)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3318

Problem: Vim9: cannot ignore quotes in number at the command line.
Solution: Use in_vim9script() so that after ":vim9" quotes are ignored.
Files: src/typval.c, src/testdir/test_float_func.vim

Patch 8.2.3319

Problem: Coverity action on github does not work.
Solution: Remove undefined \$SRCDIR. (James McCoy, closes #8739)
Files: .github/workflows/coverity.yml

Patch 8.2.3320

Problem: Some local functions are not static.
Solution: Add "static". Move snprintf() related code to strings.c.
(Yegappan Lakshmanan, closes #8734)
Files: src/alloc.c, src/channel.c, src/dict.c, src/digraph.c, src/edit.c,
src/ex_docmd.c, src/getchar.c, src/job.c, src/list.c,
src/message.c, src/profiler.c, src/proto/channel.pro,
src/proto/dict.pro, src/proto/digraph.pro, src/proto/edit.pro,
src/proto/ex_docmd.pro, src/proto/getchar.pro, src/proto/job.pro,
src/proto/list.pro, src/proto/profiler.pro, src/proto/spell.pro,
src/proto/vim9compile.pro, src/proto/vim9script.pro,
src/proto/vim9type.pro, src/spell.c, src/strings.c,
src/vim9compile.c, src/vim9script.c, src/vim9type.c, src/window.c

Patch 8.2.3321

Problem: Some code is not tested.
Solution: Add some more tests. (Dominique Pellé, closes #8735)
Files: src/testdir/test_excmd.vim, src/testdir/test_writefile.vim

Patch 8.2.3322

Problem: Vim9: checking type of dict does not check member type.
Solution: When getting the type of a typval use dv_type and lv_type.
(closes #8732)
Files: src/vim9type.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3323

Problem: Help tag for exists_compiled() is wrong. (Maxim Kim)
Solution: Adjust the help tag.
Files: runtime/doc/eval.txt

Patch 8.2.3324

Problem: Vim9: Cannot use :silent with :endwhile.
Solution: Allow for using the :silent modifier. (closes #8737)
Files: src/ex_eval.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3325

Problem: Digraph test fails when LC_ALL is set to "C".
Solution: When restoring '**encoding**' set it to "utf-8". (closes #8742)
Files: src/testdir/test_digraph.vim

Patch 8.2.3326

Problem: Vim9: no error passing an empty list of the wrong type.
Solution: Use ISN_SETTYPE also for "list<any>". (closes #8732)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3327

Problem: No check for sysconf() failing.
Solution: If sysconf() fails use SIGSTKSZ for the signal stack size.
(Zdenek Dohnal, closes #8743)
Files: src/os_unix.c

Patch 8.2.3328

Problem: Coverity error for not checking return value.
Solution: Check value is not negative.
Files: src/spellfile.c

Patch 8.2.3329

Problem: v_lock not set when getting value of environment variable.
Solution: Set v_lock to zero.
Files: src/typval.c

Patch 8.2.3330

Problem: Coverity reports using uninitialized field.
Solution: Initialize the field early.
Files: src/tag.c

Patch 8.2.3331

Problem: Coverity warns for using value without boundary check.
Solution: Add a boundary check.
Files: src/viminfo.c

Patch 8.2.3332

Problem: Vim9: cannot assign to range in list.
Solution: Implement overwriting a list range.
Files: src/vim9compile.c, src/vim9execute.c, src/list.c,
src/proto/list.pro, src/eval.c, src/proto/eval.pro,
src/testdir/test_listdict.vim, src/testdir/test_vim9_assign.vim

Patch 8.2.3333

Problem: Vim9: not enough tests run with Vim9.
Solution: Run a few more tests in Vim9 script and :def function.
Files: src/testdir/test_listdict.vim, src/testdir/vim9.vim

Patch 8.2.3334

Problem: Vim9: not enough tests run with Vim9.
Solution: Run a few more tests in Vim9 script and :def function. Fix
islocked(). Fix error for locking local variable.
Files: src/evalfunc.c, src/vim9compile.c, src/testdir/test_listdict.vim

Patch 8.2.3335

Problem: Vim9: not enough tests run with Vim9.
Solution: Run a few more tests in Vim9 script and :def function. Fix that
items(), keys() and values() return zero for a NULL dict.
Make join() return an empty string for a NULL list. Make sort()
return an empty list for a NULL list.
Files: src/dict.c, src/list.c, src/testdir/test_listdict.vim,
src/testdir/vim9.vim

Patch 8.2.3336

Problem: Behavior of negative index in list change changed. (Naruhiko
Nishino)
Solution: Only change it for Vim9 script. (closes #8749)
Files: src/list.c, src/testdir/test_listdict.vim

Patch 8.2.3337

Problem: Completing "call g:" returns entries with just "g:". (Naohiro Ono)
Solution: Skip empty strings returned by get_user_func_name(). (closes #8753)
Files: src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 8.2.3338

Problem: Vim9: no type check when assigning a list range. (Naohiro Ono)
Solution: Check the member type. (closes #8750)
Files: src/list.c, src/testdir/test_listdict.vim

Patch 8.2.3339

Problem: Vim9: cannot lock a member in a local dict.
Solution: Get the local dict from the stack and pass it to get_lval().
Files: src/eval.c, src/vim9execute.c, src/vim9compile.c, src/vim9.h,
src/globals.h, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3340 (after 8.2.3339)

Problem: Accessing uninitialized pointer.
Solution: Set pointer to NULL.
Files: src/eval.c

Patch 8.2.3341

Problem: Vim9: function call aborted despite try/catch. (Naohiro Ono)
Solution: Ignore error caught by try/catch. (closes #8755)
Files: src/evalvars.c, src/vim9execute.c, src/message.c, src/time.c,
src/globals.h, src/testdir/vim9.vim, src/testdir/test_vim9_func.vim

Patch 8.2.3342 (after 8.2.3341)

Problem: Test for :let errors fails.
Solution: Adjust the test and how to avoid a second error message.
Files: src/evalvars.c, src/testdir/test_let.vim

Patch 8.2.3343 (after 8.2.3342)

Problem: Vim9: autoload test fails.
Solution: Adjust the way the second message is avoided
Files: src/evalvars.c

Patch 8.2.3344 (after 8.2.3343)

Problem: Vimscript test fails.
Solution: Have test verify first error instead of second
Files: src/testdir/test_vimscript.vim

Patch 8.2.3345

Problem: Some code not covered by tests.
Solution: Add a few more tests. (Dominique Pellé, closes #8757)
Files: src/testdir/test_arglist.vim, src/testdir/test_cmdline.vim,
src/testdir/test_spellfile.vim, src/testdir/test_substitute.vim

Patch 8.2.3346

Problem: Vim9: no error for using "." for concatenation after ":vim9cmd".
(Naohiro Ono)
Solution: Check for Vim9 script syntax. (closes #8756)
Files: src/eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3347

Problem: Check for legacy script is incomplete. (Naohiro Ono)
Solution: Also check the :legacy modifier. Use for string concatenation
with "." and others (issue #8756)
Files: src/vim9script.c, src/proto/vim9script.pro, src/eval.c,
src/typval.c, src/evalvars.c, src/errors.h, src/ex_docmd.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.3348

Problem: line2byte() returns wrong value after adding textprop. (Yuto
Kimura)
Solution: Reduce the length by the size of the text property. (closes #8759)
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.2.3349 (after 8.2.3347)

Problem: Eval test for scriptversion fails.
Solution: Fix off-by-one error.
Files: src/vim9script.c

Patch 8.2.3350 (after 8.2.3348)

Problem: Text properties test fails on MS-Windows.
Solution: Set fileformat to unix.
Files: src/testdir/test_textprop.vim

Patch 8.2.3351

Problem: Vim9: using a function by name may delete it. (Naohiro Ono)
Solution: Increment the reference count when using a function by name.
(closes #8760)
Files: src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.3352

Problem: Vim9: error for nested :enddef has wrong line number.
Solution: Compute the line number.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3353

Problem: Vim9: type of argument for negate not checked at compile time.
Solution: Add a compile time check.
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3354

Problem: Build failure with +byte_offset but without +textprop. (John
Marriott)
Solution: Adjust the #ifdef.
Files: src/memline.c

Patch 8.2.3355

Problem: MS-Windows: compiler warning for 64-32 bit conversion.
Solution: Add type casts.
Files: src/memline.c

Patch 8.2.3356

Problem: Adding many text properties requires a lot of function calls.
Solution: Add the prop_add_list() function. (Yegappan Lakshmanan,
closes #8751)
Files: runtime/doc/eval.txt, runtime/doc/textprop.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/textprop.pro,
src/testdir/test_textprop.vim, src/testdir/test_vim9_builtin.vim,
src/textprop.c

Patch 8.2.3357

Problem: Crash when **'virtualedit'** is set and window is narrow.
Solution: Check that width is not zero. (closes #8767)
Files: src/misc2.c, src/testdir/test_number.vim

Patch 8.2.3358

Problem: Structurizr files are not recognized.
Solution: Recognize the file by contents. (Bastian Venthur, closes #8764)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3359

Problem: Vim9: error for type when variable is not set.

Solution: Give a specific error for a NULL function. (closes #8773)
Files: src/vim9type.c, src/errors.h, src/testdir/test_vim9_func.vim

Patch 8.2.3360

Problem: User function completion fails with dict function.
Solution: Do not stop sequencing through the list if user functions when encountering an empty name. (Naohiro Ono, closes #8765, closes #8774)
Files: src/evalfunc.c, src/testdir/test_cmdline.vim

Patch 8.2.3361

Problem: Vim9: crash with nested :while.
Solution: Handle skipping better. (Naruhiko Nishino, closes #8778)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3362

Problem: Buffer overflow when completing long tag name.
Solution: Allocate the buffer dynamically. (Gregory Anders, closes #8769)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.2.3363

Problem: When :edit reuses the current buffer the alternate file is set to the same buffer.
Solution: Only set the alternate file when not reusing the buffer. (closes #8783)
Files: src/ex_cmds.c, src/testdir/test_undo.vim, src/testdir/test_cmdline.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_script.vim

Patch 8.2.3364

Problem: Vim9: crash when :for is skipped.
Solution: Skip more code generation. (Naruhiko Nishino, closes #8777)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3365

Problem: Vim9: cannot use option for all operations.
Solution: Recognize more operations. (closes #8779)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3366

Problem: Vim9: debugging elseif does not stop before condition.
Solution: Move debug statement to after the jump. (closes #8781)
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3367

Problem: Vim9: :@r executing a register is inconsistent.
Solution: Use "@r" as the start of an expression. (issue #8779)
Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3368

Problem: Not all Racket files are recognized.
Solution: Also recognize .rctl and .rkt files. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3369

Problem: Auto formatting after "cw" leaves cursor in wrong spot.
Solution: Do not auto-format after the delete. (closes #8789)
Files: src/ops.c, src/testdir/test_textformat.vim

Patch 8.2.3370

Problem: Vim9: no check for white space before type in declaration.
(Naohiro Ono)
Solution: Check for white space like in a compiled function. (closes #8785)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3371

Problem: Vim9: :\$ENV cannot be followed by ->func() in next line.
Solution: Use "\$ENV" as the start of an expression. (closes #8790)
Files: src/ex_docmd.c, src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3372

Problem: line2byte() value wrong when adding a text property. (Yuto Kimura)
Solution: Adjust length for text property. (closes #8772) Also fix it for deleting a line.
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.2.3373 (after 8.2.3372)

Problem: text property test fails on MS-Windows.
Solution: Set fileformat to "unix"
Files: src/testdir/test_textprop.vim

Patch 8.2.3374

Problem: Pyret files are not recognized.
Solution: Recognize .arr files as Pyret. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3375

Problem: Using uninitialized memory.
Solution: Initialize textprop_save_len.
Files: src/memline.c

Patch 8.2.3376

Problem: Vim9: no warning that "@r" does not do anything.
Solution: Give a "no effect" error. (closes #8779)
Files: src/ex_eval.c, src/proto/ex_eval.pro, src/vim9compile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.3377

Problem: Vim9: :disass completion does not understand "s:".
Solution: Expand "s:" to a pattern. (closes #8780)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3378

Problem: MS-Windows: completing environment variables with % is wrong.
Solution: Only complete environment variables with \$. (Albert Liu,
closes #8791)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3379

Problem: Crash when using NULL job.
Solution: Copy static string into buffer. (issue #8260)
Files: src/job.c, src/testdir/test_channel.vim

Patch 8.2.3380

Problem: Crash when using NULL string for funcref().
Solution: Check for NULL argument. (issue #8260)
Files: src/evalfunc.c, src/testdir/test_expr.vim

Patch 8.2.3381

Problem: Crash when using NULL list with sign functions.
Solution: Handle a NULL list like an empty list. (issue #8260)
Files: src/globals.h, src/testdir/test_signs.vim

Patch 8.2.3382

Problem: Crash when getting the type of a NULL partial.
Solution: Check for NULL. (closes #8260)
Files: src/vim9type.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3383

Problem: Vim9: completion for :disassemble adds parenthesis.
Solution: Don't add parenthesis. (Naohiro Ono, closes #8802)
Files: src/userfunc.c, src/testdir/test_cmdline.vim

Patch 8.2.3384

Problem: Cannot disable modeline for an individual file.
Solution: Recognize "nomodeline" in a modeline. (Hu Jialun, closes #8798)
Files: runtime/doc/options.txt, src/buffer.c,
src/testdir/test_modeline.vim

Patch 8.2.3385

Problem: Escaping for fish shell does not work properly.
Solution: Insert a backslash before a backslash. (Jason Cox, closes #8810)
Files: runtime/doc/eval.txt, src/strings.c, src/testdir/test_shell.vim

Patch 8.2.3386

Problem: Using uninitialized memory.
Solution: Initialize the rm_ic field. (Dominique Pellé, closes #8800)
Files: src/indent.c

Patch 8.2.3387

Problem: Compiler warning for non-static function.
Solution: Make the function static. (Dominique Pellé, closes #8816)
Files: src/strings.c

Patch 8.2.3388

Problem: fnamemodify('path/..', ':p') differs from using 'path/./'. (David
Briscoe)
Solution: Include the "/.." in the directory name. (closes #8808)
Files: src/os_unix.c, src/testdir/test_fnamemodify.vim

Patch 8.2.3389

Problem: Cannot stop insert mode completion without side effects.
Solution: Add **CTRL-X CTRL-Z**. (closes #8821)
Files: runtime/doc/index.txt, runtime/doc/insert.txt, src/insexpand.c,
src/testdir/test_ins_complete.vim

Patch 8.2.3390

Problem: Included xdiff code is outdated.
Solution: Sync with xdiff in git 2.33. (Christian Brabandt, closes #8431)
Files: src/diff.c, src/xdiff/README.txt, src/xdiff/xdiff.h,
src/xdiff/xdiffi.c, src/xdiff/xdiffi.h, src/xdiff/xemit.c,
src/xdiff/xemit.h, src/xdiff/xhistogram.c, src/xdiff/xinclude.h,
src/xdiff/xmacros.h, src/xdiff/xpatience.c, src/xdiff/xprepare.h,
src/xdiff/xtypes.h, src/xdiff/xutils.c, src/xdiff/xutils.h

Patch 8.2.3391

Problem: Crash with combination of '**linebreak**' and other options.
Solution: Avoid n_extra to become negative. (Christian Brabandt,
closes #8817)
Files: src/drawline.c

Patch 8.2.3392

Problem: augroup completion escapes regexp pattern characters.
Solution: Do not escape the augroup name. (closes #8826)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3393

Problem: Escaping for fish shell is skipping some characters.
Solution: Escape character after backslash if needed. (Jason Cox,
closes #8827)
Files: src/strings.c, src/testdir/test_shell.vim

Patch 8.2.3394

Problem: Filler lines are wrong when changing text in diff mode.
Solution: Don't change the filler lines on every change. Check
scrollbinding when updating the filler lines. (closes #8809)
Files: src/move.c, src/diff.c, src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_scroll_change_01.dump,
src/testdir/dumps/Test_diff_scroll_change_02.dump

Patch 8.2.3395

Problem: Vim9: expression breakpoint not checked in :def function.
Solution: Always compile a function for debugging if there is an expression
breakpoint. (closes #8803)
Files: src/vim9execute.c, src/proto/vim9execute.pro, src/debugger.c,
src/proto/debugger.pro, src/vim.h, src/vim9.h,
src/testdir/test_debugger.vim

Patch 8.2.3396

Problem: When libcall() fails invalid pointer may be used.
Solution: Initialize the string to NULL. (Yasuhiro Matsumoto, closes #8829)
Files: src/evalfunc.c

Patch 8.2.3397

Problem: No test for what 8.2.3391 fixes.

Solution: Add a test. (Yegappan Lakshmanan, closes #8828)
Files: src/testdir/test_breakindent.vim

Patch 8.2.3398

Problem: Html text objects are not fully tested.
Solution: Add tests for dbcs encoding and different number of backslashes.
(Dominique Pellé, closes #8831)
Files: src/testdir/test_textobjects.vim

Patch 8.2.3399

Problem: Octave files are not recognized.
Solution: Detect Octave files. (Doug Kearns)
Files: runtime/autoload/dist/ft.vim, runtime/doc/filetype.txt,
runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3400

Problem: ":z!" is not supported.
Solution: Make ":z!" work and add tests. (Dominique Pellé, closes #8836)
Use display height instead of current window height.
Files: runtime/doc/various.txt, src/ex_cmds.h, src/ex_cmds.c,
src/testdir/test_ex_z.vim

Patch 8.2.3401

Problem: Vim9: cannot use a negative count with finddir() and findfile().
Solution: Adjust the return type. (closes #8776)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3402

Problem: Invalid memory access when using :retab with large value.
Solution: Check the number is positive.
Files: src/indent.c, src/option.c, src/optionstr.c,
src/testdir/test_retab.vim

Patch 8.2.3403 (after 8.2.3402)

Problem: Memory leak for :retab with invalid argument.
Solution: Free the memory. Make error messages consistent.
Files: src/indent.c

Patch 8.2.3404

Problem: Vim9: no error for white space before "(".
Solution: Give an error, like in a compiled function.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3405

Problem: Cannot have a comment line in a {} block of a user command.
Solution: Continue after the line break. (closes #8837)
Files: src/ex_docmd.c, src/testdir/test_usercommands.vim

Patch 8.2.3406

Problem: On some systems tests fail without _REENTRANT. (Elimar Riesebieter)
Solution: Add -D_REENTRANT in configure. (closes #7402)
Files: src/configure.ac, src/auto/configure

Patch 8.2.3407

Problem: Using uninitialized memory with "let g:['bar'] = 2".
Solution: Initialize v_type of a new dict item.
Files: src/dict.c

Patch 8.2.3408

Problem: Can delete a numbered function. (Naohiro Ono)
Solution: Disallow deleting a numbered function. (closes #8760)
Files: src/userfunc.c, src/testdir/test_user_func.vim

Patch 8.2.3409

Problem: Reading beyond end of line with invalid utf-8 character.
Solution: Check for NUL when advancing.
Files: src/regexp_nfa.c, src/testdir/test_regexp_utf8.vim

Patch 8.2.3410

Problem: Crash with linebreak, listchars and large tabstop.
Solution: Account for different size listchars for a tab. (closes #8841)
Files: src/drawline.c, src/testdir/test_listlbr_utf8.vim

Patch 8.2.3411

Problem: Vim9: crash when using base name of import. (Naohiro Ono)
Solution: Check the import flags. (closes #8843)
Files: src/evalvars.c, src/errors.h, src/testdir/test_vim9_script.vim

Patch 8.2.3412 (after 8.2.3411)

Problem: Vim9: importing the wrong file.
Solution: Correct the file name. Delete the file afterwards.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3413

Problem: Vim9: too many characters are allowed in import name.
Solution: Disallow ':' and '#', check for white space. (closes #8845)
Files: src/vim9script.c, src/errors.h, src/testdir/test_vim9_script.vim

Patch 8.2.3414

Problem: fullcommand() gives the wrong name if there is a buffer-local user command. (Naohiro Ono)
Solution: Use a separate function to get the user command name. (closes #8840)
Files: src/usercmd.c, src/proto/usercmd.pro, src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.3415

Problem: Vim9: Not all function argument types are properly checked.
Solution: Add and improve argument type checks. (Yegappan Lakshmanan, closes #8839)
Files: src/channel.c, src/digraph.c, src/evalfunc.c, src/terminal.c, src/testdir/test_digraph.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3416

Problem: Second error is reported while exception is being thrown.
Solution: Do not check for trailing characters when already aborting. (closes #8842)

Files: src/userfunc.c, src/testdir/test_trycatch.vim

Patch 8.2.3417

Problem: Vim9: a failing debug expression aborts script sourcing.
Solution: Do not let expression failure abort script sourcing. (closes #8848)
Files: src/debugger.c, src/testdir/test_debugger.vim

Patch 8.2.3418

Problem: Garbage collection while evaluating may cause trouble.
Solution: Disable garbage collection while evaluating an expression.
(Christian Brabandt, issue #8848)
Files: src/eval.c

Patch 8.2.3419

Problem: A failing debug expression may make Vim unusable.
Solution: Suppress error messages. (closes #8848)
Files: src/debugger.c, src/testdir/test_debugger.vim

Patch 8.2.3420

Problem: _REENTRANT defined more than once.
Solution: Fix configure script. (Christian Brabandt, closes #8852)
Files: src/configure.ac, src/auto/configure

Patch 8.2.3421

Problem: A bit of code is not covered by tests.
Solution: Add a few more test cases. (Dominique Pellé, closes #8857)
Files: src/testdir/test_functions.vim, src/testdir/test_history.vim,
src/testdir/test_startup.vim

Patch 8.2.3422

Problem: Vim9: no failure if return type differs from returned variable.
Solution: Copy type when copying a list. (closes #8847)
Files: src/list.c, src/testdir/test_vim9_func.vim

Patch 8.2.3423

Problem: Vim9: list += list creates a new list in :def function.
Solution: Append to the existing list.
Files: src/structs.h, src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.3424

Problem: A sequence of spaces is hard to see in list mode.
Solution: Add the "multispace" option to '**listchars**'. (closes #8834)
Files: runtime/doc/options.txt, src/drawline.c, src/message.c,
src/screen.c, src/structs.h, src/testdir/test_listchars.vim

Patch 8.2.3425

Problem: Warning for using uninitialized variable.
Solution: Initialize it. (John Marriott)
Files: src/screen.c

Patch 8.2.3426

Problem: Crash when deleting a listener in a listener callback. (Naohiro Ono)

Solution: Mark the listener and delete it later. (closes #8863)
Files: src/change.c, src/testdir/test_listener.vim

Patch 8.2.3427

Problem: Double free when list is copied.
Solution: Allocate the type when making a copy. (closes #8862)
Clear the type for flattennew(). Avoid a memory leak when
flattennew() fails.
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3428

Problem: Using freed memory when replacing. (Dhiraj Mishra)
Solution: Get the line pointer after calling ins_copychar().
Files: src/normal.c, src/testdir/test_edit.vim

Patch 8.2.3429

Problem: Leaking memory when assigning to list or dict.
Solution: Free the list or dict type before overwriting it.
Files: src/vim9type.c, src/evalvars.c

Patch 8.2.3430

Problem: No generic way to trigger an autocommand on mode change.
Solution: Add the ModeChanged autocommand event. (Magnus Gross, closes #8856)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/edit.c,
src/ex_docmd.c, src/ex_getln.c, src/globals.h, src/misc1.c,
src/normal.c, src/proto/autocmd.pro, src/proto/misc1.pro,
src/testdir/test_edit.vim, src/vim.h

Patch 8.2.3431

Problem: Completion for :disas sorts local functions first.
Solution: Sort local functions last, like with :delfunc. (Naohiro Ono,
closes #8860)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3432

Problem: Octave/Matlab filetype detection does not work properly.
Solution: Update the patterns used for matching. (Doug Kearns)
Files: runtime/autoload/dist/ft.vim, src/testdir/test_filetype.vim

Patch 8.2.3433

Problem: :delcommand does not take a -buffer option.
Solution: Add the -buffer option.
Files: runtime/doc/map.txt, src/usercmd.c, src/errors.h,
src/testdir/test_usercommands.vim

Patch 8.2.3434 (after 8.2.3430)

Problem: Function prototype for trigger_modechanged() is incomplete.
Solution: Add "void".
Files: src/proto/misc1.pro

Patch 8.2.3435

Problem: Vim9: dict is not passed to dict function.
Solution: Keep the dict used until a function call.
Files: src/vim9compile.c, src/vim9execute.c, src/vim9.h,

src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3436

Problem: Check for optional bool type has confusing return type.
Solution: Explicitly return OK.
Files: src/typval.c

Patch 8.2.3437

Problem: Compiler warnings for 32/64 bit usage.
Solution: Add type casts. (Mike Williams, closes #8870)
Files: src/screen.c, src/xdiff/xemit.c, src/xdiff/xutils.c

Patch 8.2.3438

Problem: Cannot manipulate blobs.
Solution: Add blob2list() and list2blob(). (Yegappan Lakshmanan, closes #8868)
Files: runtime/doc/eval.txt, runtime/doc/usr_41.txt, src/blob.c, src/errors.h, src/evalfunc.c, src/proto/blob.pro, src/proto/typval.pro, src/testdir/test_blob.vim, src/testdir/test_vim9_builtin.vim, src/typval.c

Patch 8.2.3439

Problem: Deleted lines go to wrong yank register.
Solution: Reset y_append when not calling get_yank_register(). (Christian Brabandt, closes #8872)
Files: src/ops.c, src/proto/register.pro, src/register.c, src/testdir/test_registers.vim

Patch 8.2.3440

Problem: Recover test fails if there is an old swap file.
Solution: Delete old swap files.
Files: src/testdir/test_recover.vim

Patch 8.2.3441

Problem: MS-Windows: vimtutor can't handle path with spaces.
Solution: Add double quotes. (Christian Brabandt, closes #8871)
Files: vimtutor.bat

Patch 8.2.3442

Problem: Vim9: || and && are not handled at compile time when possible.
Solution: When using constants generate fewer instructions.
Files: src/vim9.h, src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3443

Problem: Vim9: memory leak when and/or fails.
Solution: Also clear the growarray when the length is zero.
Files: src/vim9compile.c

Patch 8.2.3444

Problem: concealed text not revealed when leaving insert mode. (Michael Soyka)
Solution: Check if concealing changed when leaving insert mode.

(closes #8880)
Files: src/edit.c, src/testdir/test_conceal.vim,
src/testdir/dumps/Test_conceal_two_windows_07in.dump

Patch 8.2.3445

Problem: On Solaris longVersion may be declared twice. (Vladimir Marek)
Solution: Always declare longVersion in version.c
Files: src/globals.h, src/version.c

Patch 8.2.3446

Problem: Not enough tests for empty string arguments.
Solution: Add tests, fix type check. (Yegappan Lakshmanan, closes #8881)
Files: runtime/doc/sign.txt, runtime/doc/textprop.txt, src/sign.c,
src/testdir/test_blob.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3447

Problem: A couple of declarations are not ANSI C.
Solution: Put argument type inside (). (Yegappan Lakshmanan, closes #8890)
Files: src/os_unix.h

Patch 8.2.3448

Problem: :endtry after function call that throws not found.
Solution: Do check for following :endtry if an exception is being thrown.
(closes #8889)
Files: src/userfunc.c, src/testdir/test_trycatch.vim

Patch 8.2.3449

Problem: Sort fails if the sort compare function returns 999.
Solution: Adjust value to -1 / 0 / 1. (Yasuhiro Matsumoto, closes #8884)
Files: src/list.c, src/testdir/test_sort.vim

Patch 8.2.3450

Problem: Coveralls action fails.
Solution: Disable it for now.
Files: .github/workflows/ci.yml

Patch 8.2.3451

Problem: Not all apache files are recognized.
Solution: Adjust the filetype pattern. (Zdenek Dohnal, closes #8882)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3452

Problem: MPD files are not recognized.
Solution: Recognize MPD files as XML. (Steven Penny, closes #8893)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3453

Problem: Autocmd not executed when editing a directory ending in a path
separator inside try block.
Solution: Return NOTDONE instead of FAIL. (closes #8885)
Files: src/fileio.c, src/testdir/test_autocmd.vim

Patch 8.2.3454

Problem: Using a count with "gp" leaves cursor in wrong position. (Naohiro

Ono)
Solution: Count the inserted lines. (closes #8899)
Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.3455 (after 8.2.3454)

Problem: Using a count with "gp" leaves ']' in wrong position. (Naohiro Ono)
Solution: Correct the mark position. (closes #8899)
Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.3456

Problem: Vim9: Not all functions are tested with an empty string argument.
Solution: Add tests with empty strings. (Yegappan Lakshmanan, closes #8915)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3457

Problem: MS-Windows Vim9: test executed and fails.
Solution: Add extra check for not being on MS-Windows.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3458

Problem: Not all dictdconf files are recognized.
Solution: Adjust the pattern. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3459

Problem: Vim9: need more tests for empty string arguments.
Solution: Add more tests. Also use empty argument with menu_info() to get the top-level menu names. (Yegappan Lakshmanan, closes #8925)
Files: runtime/doc/eval.txt, src/menu.c, src/testdir/test_menu.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3460

Problem: Some type casts are not needed.
Solution: Remove unnecessary type casts. (closes #8934)
Files: src/autocmd.c, src/buffer.c, src/debugger.c, src/getchar.c, src/hardcopy.c, src/if_cscope.c, src/move.c, src/tag.c, src/version.c

Patch 8.2.3461

Problem: Cannot distinguish Normal and Terminal-Normal mode.
Solution: Make mode() return "nt" for Terminal-Normal mode. (issue #8856)
Files: runtime/doc/eval.txt, src/misc1.c, src/testdir/test_functions.vim

Patch 8.2.3462

Problem: The ModeChanged event only uses one character for the new_mode and old_mode values.
Solution: Pass one as first argument to mode(). (issue #8856)
Files: src/misc1.c, src/testdir/test_edit.vim

Patch 8.2.3463

Problem: Pattern matching with ModeChanged not tested.
Solution: Add a few more test lines. (issue #8856)
Files: src/testdir/test_edit.vim

Patch 8.2.3464

Problem: nginx files are not recognized.
Solution: Add several file patterns. (Chris Aumann, closes #8922)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3465

Problem: Cannot detect insert scroll mode.
Solution: Add "scroll" to complete_info(). (closes #8943)
Files: runtime/doc/eval.txt, src/insexpand.c, src/testdir/test_popup.vim

Patch 8.2.3466

Problem: Completion submode not indicated for virtual replace.
Solution: Add submode to "Rv". (closes #8945)
Files: runtime/doc/eval.txt, src/misc1.c, src/testdir/test_functions.vim

Patch 8.2.3467

Problem: CursorHoldI event interferes with "**CTRL-G** U". (Naohiro Ono)
Solution: Restore the flag for "**CTRL-G** U" after triggering CursorHoldI.
(closes #8937)
Files: src/edit.c, src/testdir/test_autocmd.vim

Patch 8.2.3468

Problem: Problem with :cd when editing file in non-existent directory. (Yee Cheng Chin)
Solution: Prepend the current directory to get the full path. (closes #8903)
Files: src/os_unix.c, src/testdir/test_cd.vim

Patch 8.2.3469

Problem: Some files with json syntax are not recognized.
Solution: Add a few file patterns. (Emiliano Ruiz Carletti, closes #8947)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3470

Problem: Crash with error in :catch and also in :finally.
Solution: Only discard an exception if there is one. (closes #8954)
Files: src/ex_eval.c, src/testdir/test_trycatch.vim

Patch 8.2.3471

Problem: Crash when using **CTRL-T** after an empty search pattern.
Solution: Bail out when there is no previous search pattern. (closes #8953)
Files: src/ex_getln.c, src/testdir/test_search.vim

Patch 8.2.3472

Problem: Other crashes with empty search pattern not tested.
Solution: Add a few more test lines. (Dominique Pellé)
Files: src/testdir/test_search.vim

Patch 8.2.3473

Problem: Some files with tcl syntax are not recognized.
Solution: Add a few file patterns. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3474

Problem: Some places use "Vimscript" instead of "Vim script".

Solution: Consistently use "Vim script". (Hirohito Higashi, closes #8910)
Files: runtime/doc/if_lua.txt, src/getchar.c, src/if_lua.c

Patch 8.2.3475

Problem: Expression register set by not executed put command.
Solution: Do not set the register if the command is skipped. (closes #8909)
Files: src/ex_docmd.c, src/testdir/test_excmd.vim

Patch 8.2.3476

Problem: Renaming a buffer on startup may cause using freed memory.
Solution: Check if the buffer is used in a window. (closes #8955)
Files: src/buffer.c, src/testdir/test_startup.vim

Patch 8.2.3477 (after 8.2.3476)

Problem: Startup test fails on MS-Windows.
Solution: Skip the test if not on Unix.
Files: src/testdir/test_startup.vim

Patch 8.2.3478 (after 8.2.3470)

Problem: Still crash with error in :catch and also in :finally.
Solution: Only call finish_exception() once. (closes #8954)
Files: src/ex_eval.c, src/structs.h

Patch 8.2.3479

Problem: Crash when calling job_start with an invalid argument. (Virginia Senioria)
Solution: Clear the first item in argv. (closes #8957)
Files: src/misc2.c, src/testdir/test_channel.vim

Patch 8.2.3480 (after 8.2.3478)

Problem: Test does not fail without the fix for a crash.
Solution: Write the bad code in a file and source it. (Dominique Pellé, closes #8961)
Files: src/testdir/test_trycatch.vim

Patch 8.2.3481

Problem: Failures when char is unsigned.
Solution: Use int8_T. Make a CI run with unsigned char. (James McCoy, closes #8936)
Files: src/structs.h, .github/workflows/ci.yml

Patch 8.2.3482

Problem: Reading beyond end of line ending in quote and backslash.
Solution: Check for non-NUL after backslash. (closes #8964)
Files: src/cindent.c, src/testdir/test_cindent.vim

Patch 8.2.3483

Problem: #ifdef for using sysinfo() is incomplete.
Solution: Also check for HAVE_SYSINFO. Make autoconf check use TRY_LINK. (closes #8952)
Files: src/memline.c, src/configure.ac, src/auto/configure

Patch 8.2.3484

Problem: Crash when going through spell suggestions.

Solution: Limit the text length for finding suggestions to the original length. Do not update buffers when exiting. (closes #8965)
Files: src/spellsuggest.c, src/clipboard.c, src/testdir/test_spell_utf8.vim

Patch 8.2.3485

Problem: Python 3 test fails with Python 3.10.
Solution: Adjust expected error message. (zdohnal Dohnal, closes #8969)
Files: src/testdir/test_python3.vim

Patch 8.2.3486

Problem: Illegal memory access with invalid sequence of commands.
Solution: Do not call leave_block() when not in a try block. (closes #8966)
Reset did_emsg so that exception is shown as an error.
Files: src/ex_eval.c, src/testdir/test_trycatch.vim

Patch 8.2.3487

Problem: Illegal memory access if buffer name is very long.
Solution: Make sure not to go over the end of the buffer.
Files: src/drawscreen.c, src/testdir/test_statusline.vim

Patch 8.2.3488

Problem: Issue template is not easy to use.
Solution: Use a yaml template. (closes #8928)
Files: .github/ISSUE_TEMPLATE/bug_report.md, .github/ISSUE_TEMPLATE/bug_report.yml

Patch 8.2.3489

Problem: ml_get error after search with range.
Solution: Limit the line number to the buffer line count.
Files: src/ex_docmd.c, src/testdir/test_search.vim

Patch 8.2.3490

Problem: Superfluous return statements.
Solution: Remove superfluous return statements from void functions. (closes #8977)
Files: src/buffer.c, src/getchar.c, src/memline.c, src/move.c, src/option.c

Patch 8.2.3491

Problem: xpm2 filetype detection is not so good.
Solution: Adjust the check for xpm2. (closes #8914)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3492

Problem: Crash when pasting too many times.
Solution: Limit the size to what fits in an int. (closes #8962)
Files: src/register.c, src/errors.h, src/testdir/test_put.vim

Patch 8.2.3493 (after 8.2.3492)

Problem: Large count test fails on MS-Windows.
Solution: Skip the test on MS-Windows.
Files: src/testdir/test_put.vim

Patch 8.2.3494

Problem: Illegal memory access in utf_head_off.
Solution: Check cursor position when reselecting the Visual area.
(closes #8963)
Files: src/normal.c, src/testdir/test_visual.vim

Patch 8.2.3495

Problem: GUI geometry startup test fails on some systems. (Drew Vogel)
Solution: Add tolerance to the size check. (closes #8815)
Files: src/testdir/test_startup.vim

Patch 8.2.3496

Problem: Crypt test fails on MS-Windows if xxd was not installed yet.
Solution: Use the just built xxd executable if it exists. (James McCoy,
closes #8929)
Files: src/testdir/test_crypt.vim

Patch 8.2.3497

Problem: Put test fails when run by itself.
Solution: Source check.vim. (Dominique Pellé, closes #8990)
Files: src/testdir/test_put.vim

Patch 8.2.3498

Problem: Recover test may fail on some systems.
Solution: Adjust the little endian and 64 bit detection. (James McCoy,
closes #8941)
Files: src/testdir/test_recover.vim

Patch 8.2.3499

Problem: GUI geometry startup test fails.
Solution: Check string values instead of numbers
Files: src/testdir/test_startup.vim

Patch 8.2.3500

Problem: Github CI fails to install clang.
Solution: Install llvm-11 explicitly. (Christian Brabandt, closes #8993)
Files: .github/workflows/ci.yml

Patch 8.2.3501

Problem: tmux filetype detection is incomplete
Solution: Also use tmux for files having text after .conf. (Eric Pruitt,
closes #8971)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3502 (after 8.2.2919)

Problem: Cannot enter password in shell command.
Solution: Revert patch 8.2.2919.
Files: src/os_unix.c

Patch 8.2.3503

Problem: Vim9: using g:pat:cmd is confusing.
Solution: Do not recognize g: as the :global command. Also for s:pat:repl.
(closes #8982)
Files: runtime/doc/vim9.txt, src/ex_docmd.c, src/ex_cmds.c, src/errors.h,

src/vim9compile.c, src/proto/vim9compile.pro,
src/testdir/test_vim9_cmd.vim

Patch 8.2.3504 (after 8.2.3503)

Problem: Vim9: warning for signed vs unsigned.

Solution: Add type cast.

Files: src/vim9compile.c

Patch 8.2.3505 (after 8.2.3503)

Problem: Vim9: build failure without the +eval feature.

Solution: Add #ifdef.

Files: src/ex_cmds.c

Patch 8.2.3506 (after 8.2.3503)

Problem: Vim9: special cases for "g" and "s" insufficiently tested.

Solution: Add a few more test cases.

Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.3507

Problem: Generating proto files may fail.

Solution: Define __attribute().

Files: src/Makefile

Patch 8.2.3508 (after 8.2.3503)

Problem: Vim9: bad separators for "g" and "s" insufficiently tested.

Solution: Add a few more test cases.

Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.3509

Problem: Undo file is not synced. (Sami Farin)

Solution: Sync the undo file if '**fsync**' is set. (Christian Brabandt,
closes #8879, closes #8920)

Files: runtime/doc/options.txt, src/undo.c

Patch 8.2.3510

Problem: Changes are only detected with one second accuracy.

Solution: Use the nanosecond time if possible. (Leah Neukirchen,
closes #8873, closes #8875)

Files: runtime/doc/eval.txt, src/auto/configure, src/bufwrite.c,
src/config.h.in, src/configure.ac, src/fileio.c,
src/proto/fileio.pro, src/memline.c, src/netbeans.c,
src/structs.h, src/evalfunc.c, src/testdir/test_stat.vim

Patch 8.2.3511

Problem: Vim9: entry for loop variable is created every round.

Solution: Only create the entry once. (closes #8996)

Files: src/evalvars.c, src/vim9script.c

Patch 8.2.3512

Problem: Timestamp test fails on some systems.

Solution: Sleep for a short while.

Files: src/testdir/test_stat.vim

Patch 8.2.3513

Problem: Using freed memory when using a timer and searching. (Dominique Pellé)
Solution: Allocated mr_pattern.
Files: src/search.c

Patch 8.2.3514

Problem: Autoread test with nanosecond time sometimes fails.
Solution: Mark the test as being flaky.
Files: src/testdir/test_stat.vim

Patch 8.2.3515

Problem: Nano time test fails on Mac and FreeBSD.
Solution: Also check nano time when not on Linux. (Ozaki Kiichi, closes #9000)
Files: src/fileio.c

Patch 8.2.3516

Problem: Terminal window does not have transparent background when **'termguicolors'** is used.
Solution: Fix the background color. (closes #2361, closes #9002)
Files: runtime/doc/terminal.txt, src/highlight.c, src/proto/terminal.pro, src/terminal.c

Patch 8.2.3517

Problem: TextChanged does not trigger after TextChangedI.
Solution: Store the tick separately for TextChangedI. (Christian Brabandt, closes #8968, closes #8932)
Files: src/buffer.c, src/bufwrite.c, src/edit.c, src/structs.h, src/testdir/test_autocmd.vim

Patch 8.2.3518

Problem: Test_xrestore sometimes fails.
Solution: Mark the test as flaky. Move marking test as flaky to the test instead of listing them in runtest.
Files: src/testdir/test_paste.vim, src/testdir/runtest.vim, src/testdir/test_autocmd.vim, src/testdir/test_channel.vim, src/testdir/test_clientserver.vim, src/testdir/test_diffmode.vim, src/testdir/test_functions.vim, src/testdir/test_gui.vim, src/testdir/test_mapping.vim, src/testdir/test_popup.vim, src/testdir/test_quotestar.vim, src/testdir/test_reftime.vim, src/testdir/test_terminal.vim, src/testdir/test_terminal2.vim, src/testdir/test_timers.vim

Patch 8.2.3519

Problem: TOML files are not recognized.
Solution: Add filetype patterns for TOML. (Aman Verma, closes #8984)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3520

Problem: Cannot define a function for thesaurus completion.
Solution: Add **'thesaurusfunc'**. (Yegappan Lakshmanan, closes #8987, closes 8950)
Files: runtime/doc/insert.txt, runtime/doc/options.txt, runtime/doc/quickref.txt, src/buffer.c, src/insexpand.c,

src/option.c, src/option.h, src/optiondefs.h, src/optionstr.c,
src/structs.h, src/testdir/test_edit.vim

Patch 8.2.3521 (after 8.2.3520)

Problem: Options completion test fails.
Solution: Add **'thesaurusfunc'** to the results.
Files: src/testdir/test_options.vim

Patch 8.2.3522

Problem: Cannot use \x and \u when setting **'listchars'**.
Solution: Support hex and unicode in hex form. (closes #9006)
Files: runtime/doc/options.txt, src/screen.c, src/charset.c,
src/testdir/test_listchars.vim

Patch 8.2.3523

Problem: Duplicated code in xxd.
Solution: Remove duplicated lines. (closes #8972)
Files: src/xxd/xxd.c

Patch 8.2.3524

Problem: GUI: ligatures are not used.
Solution: Add the **'guiligatures'** option. (Dusan Popovic, closes #8933)
Files: runtime/doc/options.txt, src/gui.c, src/gui.h, src/gui_gtk_x11.c,
src/option.h, src/optiondefs.h, src/optionstr.c, src/errors.h,
src/proto/gui.pro, src/proto/gui_gtk_x11.pro,
src/testdir/test_gui.vim

Patch 8.2.3525

Problem: Option variable name does not match option name. (Christ van Willegen)
Solution: Rename the variable.
Files: src/buffer.c, src/insexpand.c, src/option.c, src/optionstr.c,
src/structs.h

Patch 8.2.3526

Problem: Tests have clumsy check for X11 based GUI.
Solution: Add CheckX11BasedGui.
Files: src/testdir/check.vim, src/testdir/test_gui.vim,
src/testdir/test_gui_init.vim, src/testdir/setup_gui.vim

Patch 8.2.3527

Problem: Gcc complains about uninitialized variable. (Tony Mechelynck)
Solution: Initialize it.
Files: src/gui_gtk_x11.c

Patch 8.2.3528

Problem: **'thesaurus'** and **'thesaurusfunc'** do not have the same scope.
Solution: Make **'thesaurusfunc'** global-local.
Files: runtime/doc/options.txt, runtime/doc/insert.txt,
src/optiondefs.h, src/option.h, src/option.c, src/structs.h,
src/insexpand.c, src/testdir/test_edit.vim

Patch 8.2.3529

Problem: Xxd usage output is incomplete.

Solution: Add "bytes" to "-g" flag. (Atsushi Sugawara, closes #8944)
Files: src/xxd/xxd.c

Patch 8.2.3530

Problem: ":buf \{a}" fails while ":edit \{a}" works.
Solution: Unescape "\{". (closes #8917)
Files: src/vim.h, src/cmdexpand.c, src/evalfunc.c, src/ex_getln.c,
src/proto/ex_getln.pro, src/normal.c, src/session.c,
src/terminal.c, src/vim9execute.c, src/testdir/test_cmdline.vim

Patch 8.2.3531 (after 8.2.3530)

Problem: Command line completion test fails on MS-Windows.
Solution: Do not test with "\{" on MS-Windows.
Files: src/testdir/test_cmdline.vim

Patch 8.2.3532

Problem: The previous ' mark is restored after moving the cursor to the original jump position. (Tony Chen)
Solution: Forget the previous position after checking. (closes #8985)
Files: src/mark.c, src/testdir/test_marks.vim

Patch 8.2.3533

Problem: Inefficient code in xxd.
Solution: Don't use "p" when "hextype" is non-zero. (closes #9013)
Files: src/xxd/xxd.c

Patch 8.2.3534

Problem: Autoread test is a bit flaky.
Solution: Wait a brief moment before overwriting the file.
Files: src/testdir/test_stat.vim

Patch 8.2.3535

Problem: If-else indenting is confusing.
Solution: Add curly brackets and indent. (Dominique Pellé, closes #9010)
Files: src/drawscreen.c

Patch 8.2.3536

Problem: The do_highlight() function is way too long.
Solution: Split it into several functions. (Yegappan Lakshmanan, closes #9011)
Files: src/highlight.c

Patch 8.2.3537

Problem: mode() does not return the right value in **'operatorfunc'**.
Solution: Reset finish_op while calling **'operatorfunc'**.
Files: src/ops.c, src/testdir/test_functions.vim

Patch 8.2.3538

Problem: Else-if indenting is confusing.
Solution: Add curly brackets. (Yegappan Lakshmanan, closes #9017)
Files: src/highlight.c

Patch 8.2.3539

Problem: GTK3: with **'rightleft'** set scrollbar may move unintentionally.

Solution: Ignore events while moving the scrollbar thumb. (closes #8958)
Files: src/gui_gtk.c

Patch 8.2.3540

Problem: The mark ']' is wrong after put with a count. (Naohiro Ono)
Solution: Use the right line number. (closes #8956)
Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.3541

Problem: Compiler warning for unused variable in tiny version.
Solution: Add #ifdef. (John Marriott)
Files: src/highlight.c

Patch 8.2.3542

Problem: Too many comments are old style.
Solution: Change comments to // style. (closes #9021)
Files: src/buffer.c

Patch 8.2.3543

Problem: Swapname has double slash when '**directory**' ends in double slash.
(Shane Smith)
Solution: Remove the superfluous slash. (closes #8876)
Files: src/memline.c, src/testdir/test_swap.vim

Patch 8.2.3544

Problem: Unix: may leak file descriptor when using a non-existing
directory.
Solution: Always close the file. (closes #9023)
Files: src/os_unix.c

Patch 8.2.3545

Problem: setcellwidths() may make '**listchars**' or '**fillchars**' invalid.
Solution: Check the value and give an error. (closes #9024)
Files: runtime/doc/eval.txt, src/optionstr.c, src/errors.h, src/mbyte.c,
src/testdir/test_utf8.vim

Patch 8.2.3546 (after 8.2.3545)

Problem: Build failure without the +eval feature.
Solution: Add #ifdef. (closes #9025)
Files: src/errors.h

Patch 8.2.3547

Problem: Opening the quickfix window triggers BufWinEnter twice. (Yorick
Peterse)
Solution: Only trigger BufWinEnter with "quickfix". (closes #9022)
Files: src/ex_cmds.c, src/vim.h, src/quickfix.c, src/buffer.c,
src/testdir/test_quickfix.vim

Patch 8.2.3548

Problem: GTK GUI crashes when reading from stdin.
Solution: Do not overwrite the NUL after the string. (closes #9028)
Files: src/gui_gtk_x11.c, src/testdir/test_gui.vim

Patch 8.2.3549

Problem: Mistakes in test comments.
Solution: Fix the comments. (closes #9029)
Files: src/testdir/test_autocmd.vim

Patch 8.2.3550

Problem: completion() does not work properly.
Solution: Set xp_line and add WILD_HOME_REPLACE. (Shougo Matsushita, closes #9016)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3551

Problem: Checking first character of url twice.
Solution: Only check once. (closes #9026)
Files: src/misc1.c

Patch 8.2.3552

Problem: Xxd revert does not handle end of line correctly.
Solution: Check for newline first. (closes #9034)
Files: src/xxd/xxd.c, src/testdir/test_xxd.vim

Patch 8.2.3553 (after 8.2.3552)

Problem: Xxd test fails on MS-Windows.
Solution: Split shell command in two.
Files: src/testdir/test_xxd.vim

Patch 8.2.3554

Problem: Xxd has various way to exit.
Solution: Add function to print error and exit. (closes #9035)
Files: src/xxd/xxd.c

Patch 8.2.3555

Problem: ModeChanged is not triggered on every mode change.
Solution: Also trigger on minor mode changes. (Maguns Gross, closes #8999)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/insexpand.c, src/misc1.c, src/normal.c, src/terminal.c, src/testdir/test_edit.vim

Patch 8.2.3556

Problem: Filler lines are incorrect for other window in diff mode after making a change.
Solution: Copy filler lines from the current window. (closes #8809)
Files: src/diff.c, src/testdir/test_diffmode.vim, src/testdir/dumps/Test_diff_scroll_change_03.dump

Patch 8.2.3557

Problem: Vim9: cannot call imported funcref at script level.
Solution: Check for an imported function. (closes #9007)
Files: src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.3558 (after 8.2.3557)

Problem: Vim9: asserting the wrong variable.
Solution: Don't use Foo, use Goo.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3559

Problem: Loop variable recreated every time.
Solution: Keep the loop variable when looping.
Files: src/ex_eval.c

Patch 8.2.3560

Problem: Using freed memory with lambda.
Solution: Do not free lines early, keep them until the expression is finished. (closes #9020)
Files: src/eval.c, src/proto/eval.pro, src/userfunc.c, src/vim9compile.c, src/structs.h, src/globals.h, src/testdir/test_vim9_func.vim

Patch 8.2.3561

Problem: Cscope has a complicated way of giving an error message.
Solution: Use semsg(). (James McCoy, closes #9038)
Files: src/if_cscope.c

Patch 8.2.3562

Problem: Cannot add color names.
Solution: Add the v:colornames dictionary. (Drew Vogel, closes #8761)
Files: Filelist, READMEdir/README_extra.txt, nsis/gvim.nsi, runtime/colors/README.txt, runtime/colors/lists/csscolors.vim, runtime/colors/lists/default.vim, runtime/doc/eval.txt, runtime/doc/gui_w32.txt, runtime/doc/message.txt, runtime/doc/os_haiku.txt, runtime/doc/syntax.txt, runtime/doc/usr_06.txt, src/Makefile, src/evalvars.c, src/gui_haiku.cc, src/highlight.c, src/gui.c, src/job.c, src/proto/highlight.pro, src/proto/term.pro, src/term.c, src/vim.h, src/globals.h, src/errors.h, src/testdir/test_highlight.vim

Patch 8.2.3563 (after 8.2.3562)

Problem: Build failure with +eval but without GUI or +termguicolors
Solution: Adjust #ifdef. (John Marriott)
Files: src/highlight.c

Patch 8.2.3564

Problem: Invalid memory access when scrolling without a valid screen.
Solution: Do not set VALID_BOTLINE in w_valid.
Files: src/move.c, src/testdir/test_normal.vim

Patch 8.2.3565

Problem: Makefile dependencies are outdated. (Gary Johnson)
Solution: Run "make depend" and add missing dependencies.
Files: src/Makefile

Patch 8.2.3566

Problem: Build failure on old systems when using nano timestamp.
Solution: Define _BSD_SOURCE, _SVID_SOURCE and _DEFAULT_SOURCE. (Gary Johnson, closes #9054)
Files: src/vim.h

Patch 8.2.3567

Problem: **CTRL-I** in Insert mode is not tested

Solution: Add a test case. (Dominique Pellé, closes #8866)
Files: src/testdir/test_edit.vim

Patch 8.2.3568

Problem: Ctrl-hat test fails with Athena and Motif. (Elimar Riesebieter)
Solution: Run the test only with GTK. (Dominique Pellé, closes #9069)
Files: src/testdir/test_edit.vim

Patch 8.2.3569

Problem: Error for :let when vimrc is Vim 9 script.
Solution: Prepend :legacy in the code for converting arguments. (Christian Brabandt, closes #9068, closes #9077)
Files: src/os_win32.c

Patch 8.2.3570

Problem: Test_very_large_count fails on 32bit systems.
Solution: Bail out when using 32 bit numbers. (closes #9072)
Files: src/testdir/test_put.vim

Patch 8.2.3571

Problem: Some unicode control characters are considered printable.
Solution: Make 0x2060 - 0x2069 not printable.
Files: src/mbyte.c

Patch 8.2.3572

Problem: Memory leak when closing window and using "multispace" in **'listchars'**.
Solution: Free the memory. (closes #9071)
Files: src/window.c, src/testdir/test_listchars.vim

Patch 8.2.3573

Problem: Cannot decide whether to skip test that fails with 64 bit ints. (closes #9072)
Solution: Add v:sizeofint, v:sizeoflong and v:sizeofpointer. Improve the check for multiply overflow.
Files: runtime/doc/eval.txt, src/vim.h, src/evalvars.c, src/register.c, src/testdir/test_put.vim

Patch 8.2.3574 (after 8.2.3573)

Problem: Divide by zero.
Solution: Don't check for overflow if multiplicand is zero.
Files: src/register.c

Patch 8.2.3575 (after 8.2.3574)

Problem: Overflow check still fails when sizeof(int) == sizeof(long).
Solution: Use a float to check the result.
Files: src/register.c

Patch 8.2.3576

Problem: Some functions are not documented for use with a method.
Solution: Add examples. Fix that sign_unplacelist() only takes one argument. (Sean Dewar, closes #9081)
Files: src/evalfunc.c, runtime/doc/eval.txt

Patch 8.2.3577 (after 8.2.3574)

Problem: Overflow check fails with 32 bit ints.

Solution: Only test with 64 bit ints.

Files: src/testdir/test_put.vim

Patch 8.2.3578

Problem: Manipulating highlighting is complicated.

Solution: Add the hlget() and hlset() functions. (Yegappan Lakshmanan, closes #9039)

Files: runtime/doc/eval.txt, runtime/doc/syntax.txt,
runtime/doc/usr_41.txt, runtime/doc/windows.txt, src/evalfunc.c,
src/highlight.c, src/proto/highlight.pro,
src/testdir/test_highlight.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3579

Problem: CI sometimes fails for MinGW.

Solution: Use backslashes in HandleSwapExists(). (Christian Brabandt, closes #9078)

Files: src/testdir/runtest.vim

Patch 8.2.3580

Problem: gj does not move properly with a wide character.

Solution: Move one to the right. (Christian Brabandt, closes #8702)

Files: src/normal.c, src/testdir/test_normal.vim

Patch 8.2.3581

Problem: Reading character past end of line.

Solution: Correct the cursor column.

Files: src/ex_docmd.c, src/testdir/test_put.vim

Patch 8.2.3582

Problem: Reading uninitialized memory when giving spell suggestions.

Solution: Check that preword is not empty.

Files: src/spellsuggest.c, src/testdir/test_spell.vim

Patch 8.2.3583

Problem: The "gd" and "gD" commands do not update search stats. (Gary Johnson)

Solution: Clear search stats.

Files: src/normal.c, src/testdir/test_search_stat.vim,
src/testdir/dumps/Test_searchstatgd_1.dump,
src/testdir/dumps/Test_searchstatgd_2.dump

Patch 8.2.3584

Problem: "verbose set efm" reports the location of the :compiler command. (Gary Johnson)

Solution: Add the "--keepscrip" argument to :command and use it when defining CompilerSet.

Files: runtime/doc/map.txt, src/ex_cmds2.c, src/usercmd.c, src/ex_cmds.h,
src/testdir/test_compiler.vim

Patch 8.2.3585

Problem: Crash when passing float to "term_rows" in the options argument of term_start(). (Virginia Senioria)

Solution: Bail out if the argument is not a number. (closes #9116)
Files: src/job.c, src/terminal.c, src/testdir/test_terminal.vim

Patch 8.2.3586 (after 8.2.3584)

Problem: Command completion test fails.
Solution: Add new argument to expected output
Files: src/testdir/test_usercommands.vim

Patch 8.2.3587 (after 8.2.3584)

Problem: Compiler test fails with backslash file separator.
Solution: Accept slash and backslash.
Files: src/testdir/test_compiler.vim

Patch 8.2.3588

Problem: Break statement is never reached.
Solution: Rely on return value of set_chars_option() not changing.
(closes #9103)
Files: src/optionstr.c

Patch 8.2.3589

Problem: Failure when the "term_rows" argument of term_start() is an unusual value.
Solution: Limit to range of zero to 1000. (closes #9116)
Files: runtime/doc/terminal.txt, src/job.c, src/testdir/test_terminal.vim

Patch 8.2.3590

Problem: Test for v:colornames sometimes fails. (Dominique Pellé)
Solution: Check features. Clear v:colornames between tests. (Drew Vogel, closes #9105, closes #9073)
Files: runtime/doc/eval.txt, src/highlight.c, src/proto/highlight.pro, src/testdir/test_highlight.vim

Patch 8.2.3591

Problem: No event is triggered when closing a window.
Solution: Add the WinClosed event. (Naohiro Ono, closes #9110)
Files: runtime/doc/autocmd.txt, src/autocmd.c, src/testdir/test_autocmd.vim, src/vim.h, src/window.c

Patch 8.2.3592

Problem: Test_hlset fails when terminal has many columns.
Solution: Set the number of columns to 80. (Dominique Pellé, closes #9101, closes #9100)
Files: src/testdir/test_highlight.vim

Patch 8.2.3593

Problem: Directory is wrong after executing "lcd" with win_execute().
Solution: Correct the directory when going back to the original window.
(closes #9132)
Files: src/evalwindow.c, src/window.c, src/proto/window.pro, src/testdir/test_execute_func.vim

Patch 8.2.3594

Problem: Xxd code is a bit difficult to understand.
Solution: Move some lines to a separate function. (closes #9037)

Files: src/xxd/xxd.c

Patch 8.2.3595

Problem: Check for signed overflow might not work everywhere.

Solution: Limit to 32 bit int. (closes #9043, closes #9067)

Files: src/getchar.c

Patch 8.2.3596

Problem: Crash when using :pedit in Vim9 script.

Solution: Move check for arguments to after checking there are arguments.
(Yegappan Lakshmanan, closes #9134, closes #9135)

Files: src/popupwin.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3597

Problem: Vim seems to hang when writing a very long text to a terminal window.

Solution: Limit the amount of text based on **'termwincscroll'**. (issue #9080)

Files: runtime/doc/options.txt, src/terminal.c

Patch 8.2.3598

Problem: RouterOS filetype is not recognized.

Solution: Add file and script patterns. (closes #9097)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3599

Problem: Not all gdbinit files are recognized.

Solution: Add "gdbinit". (Doug Kearns)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3600 (after 8.2.3598)

Problem: Filetype test fails.

Solution: Add missing change.

Files: runtime/scripts.vim

Patch 8.2.3601

Problem: Check for overflow in put count does not work well.

Solution: Improve the overflow check. (Ozaki Kiichi, closes #9102)

Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.3602

Problem: Python3 test fails with Python 3.10 on MS-Windows.

Solution: Adjust the expected error. (Ken Takata, closes #9118)

Files: src/testdir/test_python3.vim

Patch 8.2.3603

Problem: Fish filetype not recognized.

Solution: Add a file pattern and match script line. (Doug Kearns)

Files: runtime/filetype.vim, runtime/scripts.vim,
src/testdir/test_filetype.vim

Patch 8.2.3604

Problem: Not all sudoers files are recognized.

Solution: Add a file pattern. (Doug Kearns, closes #1192)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3605

Problem: Cannot clear and unlink a highlight group with hlset() in a single call.
Solution: Add the "force" option. (Yegappan Lakshmanan, closes #9117)
Files: runtime/doc/eval.txt, src/highlight.c, src/testdir/test_highlight.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.3606

Problem: File missing from list of distributed files.
Solution: Add the file.
Files: Filelist

Patch 8.2.3607

Problem: GTK3 screen updating is slow.
Solution: Remove some of the GTK3-specific code. (closes #9052)
Files: src/gui.h, src/gui_gtk_x11.c

Patch 8.2.3608

Problem: Users who type "q:" instead of ":q" are confused.
Solution: Add an autocmd to give a message that explains this is the command-line window. (Egor Zvorykin, closes #9146)
Files: runtime/defaults.vim, src/testdir/test_autocmd.vim, src/testdir/test_cmdline.vim

Patch 8.2.3609

Problem: Internal error when ModeChanged is triggered when v:event is already in use.
Solution: Save and restore v:event if needed.
Files: src/misc1.c, src/proto/misc1.pro, src/testdir/test_edit.vim, src/insexpand.c, src/structs.h, src/register.c

Patch 8.2.3610

Problem: Crash when ModeChanged triggered too early.
Solution: Trigger ModeChanged after setting VIsual.
Files: src/normal.c, src/testdir/test_edit.vim

Patch 8.2.3611

Problem: Crash when using **CTRL-W** f without finding a file name.
Solution: Bail out when the file name length is zero.
Files: src/findfile.c, src/normal.c, src/testdir/test_visual.vim

Patch 8.2.3612

Problem: Using freed memory with regexp using a mark.
Solution: Get the line again after getting the mark position.
Files: src/regexp.c, src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.2.3613

Problem: :find test fails.
Solution: Put length check inside if block.
Files: src/findfile.c

Patch 8.2.3614

Problem: zindex of popup windows not used when redrawing popup menu.
Solution: Check the zindex when redrawing the popup menu. (closes #9129, closes #9089)

Files: src/popupmenu.c, src/popupwin.c, src/proto/popupmenu.pro, src/screen.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_popupmenu_masking_1.dump, src/testdir/dumps/Test_popupwin_popupmenu_masking_2.dump

Patch 8.2.3615

Problem: When re-formatting with an indent expression the first line of a paragraph may get the wrong indent. (Martin F. Krafft)

Solution: Apply the correct indenting function for the first line. (Christian Brabandt, closes #9150, closes #9056)

Files: src/textformat.c, src/testdir/test_indent.vim

Patch 8.2.3616

Problem: Arglist test does not clear the argument list consistently.

Solution: Call Reset_arglist(). (Shougo Matsushita, closes #9154)

Files: src/testdir/test_arglist.vim

Patch 8.2.3617

Problem: ":verbose pwd" does not mention 'autochdir' was applied.

Solution: Remember the last chdir was done by 'autochdir'. (issue #9142)

Files: src/globals.h, src/buffer.c, src/ex_docmd.c, src/window.c, src/main.c, src/netbeans.c, src/os_win32.c, src/testdir/test_autochdir.vim

Patch 8.2.3618

Problem: getcwd() is unclear about how 'autochdir' is used.

Solution: Update the help for getcwd(). Without any arguments always return the actual current directory. (closes #9142)

Files: runtime/doc/eval.txt, src/filepath.c, src/testdir/test_cd.vim

Patch 8.2.3619

Problem: Cannot use a lambda for 'operatorfunc'.

Solution: Support using a lambda or partial. (Yegappan Lakshmanan, closes #8775)

Files: runtime/doc/map.txt, runtime/doc/options.txt, src/ops.c, src/option.c, src/optionstr.c, src/proto/ops.pro, src/proto/option.pro, src/quickfix.c, src/testdir/test_normal.vim

Patch 8.2.3620

Problem: Memory leak reported in libtlib.

Solution: Call del_curterm() when cleaning up memory. Rename term.h to termdefs.h to avoid a name clash.

Files: src/term.c, src/proto/term.pro, src/alloc.c, src/configure.ac, src/auto/configure, src/config.h.in, src/Makefile, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/term.h, src/termdefs.h

Patch 8.2.3621 (after 8.2.3620)

Problem: Build failure.

Solution: Add missing change.

Files: src/vim.h

Patch 8.2.3622

Problem: "verbose pwd" shows confusing info when :lcd does not change directory.
Solution: Clear last_chdir_reason also when the directory does not change. (closes #9160)
Files: src/ex_docmd.c, src/testdir/test_autochdir.vim

Patch 8.2.3623

Problem: "\$*" is expanded to "nonomatch".
Solution: Only add "set nonomatch" when using a csh-like shell. (Christian Brabandt, closes #9159, closes #9153)
Files: src/os_unix.c, src/testdir/test_expand.vim

Patch 8.2.3624

Problem: When renaming a terminal buffer the status text is not updated.
Solution: Clear the cached status text when renaming a terminal buffer. (closes #9162)
Files: src/buffer.c, src/terminal.c, src/proto/terminal.pro, src/testdir/test_terminal.vim

Patch 8.2.3625

Problem: Illegal memory access when C-indenting.
Solution: Also set the cursor column.
Files: src/cindent.c, src/testdir/test_cindent.vim

Patch 8.2.3626

Problem: "au!" and "au! event" cannot be followed by another command as documented.
Solution: When a bar is found set nextcmd.
Files: src/autocmd.c, src/testdir/test_autocmd.vim

Patch 8.2.3627

Problem: difficult to know where the text starts in a window. (Sergey Vlasov)
Solution: Add the "textoff" entry in the result of getwininfo(). (closes #9163)
Files: runtime/doc/eval.txt, src/evalwindow.c, src/testdir/test_bufwintabinfo.vim

Patch 8.2.3628

Problem: Looking up terminal colors is a bit slow.
Solution: Cache the terminal colors. (closes #9130, closes #9058)
Files: src/highlight.c, src/libvterm/include/vterm.h, src/option.c, src/optionstr.c, src/popupwin.c, src/proto/terminal.pro, src/structs.h, src/terminal.c, src/window.c, src/testdir/test_terminal3.vim, src/testdir/dumps/Test_terminal_color_MyTermCol.dump, src/testdir/dumps/Test_terminal_color_MyTermCol_over_Terminal.dump, src/testdir/dumps/Test_terminal_color_MyWinCol.dump, src/testdir/dumps/Test_terminal_color_MyWinCol_over_group.dump, src/testdir/dumps/Test_terminal_color_Terminal.dump, src/testdir/dumps/Test_terminal_color_gui_MyTermCol.dump, src/testdir/dumps/Test_terminal_color_gui_MyWinCol.dump,

src/testdir/dumps/Test_terminal_color_gui_Terminal.dump,
src/testdir/dumps/Test_terminal_color_gui_transp_MyTermCol.dump,
src/testdir/dumps/Test_terminal_color_gui_transp_MyWinCol.dump,
src/testdir/dumps/Test_terminal_color_gui_transp_Terminal.dump,
src/testdir/dumps/Test_terminal_color_transp_MyTermCol.dump,
src/testdir/dumps/Test_terminal_color_transp_MyWinCol.dump,
src/testdir/dumps/Test_terminal_color_transp_Terminal.dump,
src/testdir/dumps/Test_terminal_popup_MyPopupHlCol.dump,
src/testdir/dumps/Test_terminal_popup_MyTermCol_over_Terminal.dump,
src/testdir/dumps/Test_terminal_popup_MyWinCol.dump,
src/testdir/dumps/Test_terminal_popup_MyWinCol_over_group.dump,
src/testdir/dumps/Test_terminal_popup_gui_MyPopupHlCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_MyTermCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_MyWinCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_Terminal.dump,
src/testdir/dumps/Test_terminal_popup_gui_transp_MyPopupHlCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_transp_MyTermCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_transp_MyWinCol.dump,
src/testdir/dumps/Test_terminal_popup_gui_transp_Terminal.dump,
src/testdir/dumps/Test_terminal_popup_transp_MyPopupHlCol.dump,
src/testdir/dumps/Test_terminal_popup_transp_MyTermCol.dump,
src/testdir/dumps/Test_terminal_popup_transp_MyWinCol.dump,
src/testdir/dumps/Test_terminal_popup_transp_Terminal.dump,
src/testdir/dumps/Test_terminal_wincolor_split_MyWinCol.dump,
src/testdir/dumps/Test_terminal_wincolor_split_MyWinCol2.dump

Patch 8.2.3629

Problem: Command completion in cmdline window uses global user commands, not local commands for the window where it was opened from.
Solution: Use local commands. (closes #9168)
Files: src/ex_getln.c, src/proto/ex_getln.pro, src/evalvars.c, src/usercmd.c, src/testdir/test_ins_complete.vim

Patch 8.2.3630

Problem: Printf() with %S does not handle multi-byte correctly.
Solution: Count cells instead of bytes. (closes #9169, closes #7486)
Files: src/strings.c, src/testdir/test_expr.vim

Patch 8.2.3631

Problem: "syntax enable" does not work properly in Vim9 context.
Solution: Also handle Vim9 context. (closes #9161)
Files: src/syntax.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3632

Problem: GTK3: undercurl does not get removed properly.
Solution: Set the cairo cursor first. (closes #9170)
Files: src/gui_gtk_x11.c

Patch 8.2.3633

Problem: Vim9: line number of lambda is off by one.
Solution: Add one to the line number. (closes #9083)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3634

Problem: Error for already defined function uses wrong line number.
Solution: Set SOURCING_LNUM before giving the error message. (closes #9085)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3635

Problem: GTK: composing underline does not show.
Solution: Include composing character in pango call. A few more optimizations for ligatures. (Dusan Popovic, closes #9171, closes #9147)
Files: src/gui_gtk_x11.c

Patch 8.2.3636

Problem: Coverity warns for unreachable code.
Solution: Remove unreachable else block.
Files: src/gui_gtk_x11.c

Patch 8.2.3637

Problem: Typos in test files.
Solution: Correct the typos. (Dominique Pellé, closes #9175)
Files: src/testdir/runtest.vim, src/testdir/test_debugger.vim, src/testdir/test_diffmode.vim, src/testdir/test_edit.vim, src/testdir/test_excmod.vim, src/testdir/test_flatten.vim, src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim, src/testdir/test_options.vim, src/testdir/test_python2.vim, src/testdir/test_python3.vim, src/testdir/test_quickfix.vim, src/testdir/test_recover.vim, src/testdir/test_spellfile.vim, src/testdir/test_syntax.vim, src/testdir/test_termcodes.vim, src/testdir/test_textobjects.vim, src/testdir/test_trycatch.vim, src/testdir/test_vim9_script.vim, src/testdir/test_viminfo.vim

Patch 8.2.3638

Problem: getcompletion() always passes zero as position to custom completion function.
Solution: Pass the pattern length. (closes #9173)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.3639 (after 8.2.2922)

Problem: Line commented out accidentally.
Solution: Uncomment. (Volodymyr Kot, closes #9172)
Files: src/main.c

Patch 8.2.3640

Problem: Freeze when calling term_wait() in a close callback.
Solution: Set a "closing" flag to tell term_wait() to return. (closes #9152)
Files: src/channel.c, src/terminal.c, src/proto/terminal.pro, src/testdir/test_terminal.vim

Patch 8.2.3641

Problem: Xxd code has duplicate expressions.
Solution: Refactor to avoid duplication. (closes #9185)
Files: src/xxd/xxd.c

Patch 8.2.3642

Problem: List of distributed files is outdated.

Solution: Rename term.h to termdefs.h.
Files: Filelist

Patch 8.2.3643

Problem: Header for source file is outdated.
Solution: Make the header more accurate. (closes #9186)
Files: src/map.c, src/getchar.c

Patch 8.2.3644

Problem: Count for **'operatorfunc'** in Visual mode is not redone.
Solution: Add the count to the redo buffer. (closes #9174)
Files: src/normal.c, src/proto/normal.pro, src/ops.c,
src/testdir/test_normal.vim

Patch 8.2.3645

Problem: Vim9: The "no effect" error is not given for all registers.
Solution: Include any character following '@'. (closes #8779)
Files: src/ex_eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3646

Problem: Using `<sfile>` in a function gives an unexpected result.
Solution: Give an error in a Vim9 function. (issue #9189)
Files: src/scriptfile.c, src/errors.h, src/testdir/test_vim9_builtin.vim

Patch 8.2.3647

Problem: GTK: when using ligatures the cursor is drawn wrong.
Solution: Clear more characters when ligatures are used. (Dusan Popovic,
closes #9190)
Files: src/gui.c

Patch 8.2.3648

Problem: "verbose pwd" is incorrect after dropping files on Vim.
Solution: Set the chdir reason to "drop".
Files: src/gui.c

Patch 8.2.3649

Problem: Vim9: error for variable declared in while loop.
Solution: Do not keep the first variable. (closes #9191)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3650

Problem: Vim9: for loop variable can be a list member.
Solution: Check for valid variable name. (closes #9179)
Files: src/vim9compile.c, src/dict.c, src/eval.c, src/evalvars.c,
src/proto/evalvars.pro, src/testdir/test_vim9_script.vim

Patch 8.2.3651

Problem: Vim9: no error for :lock or :unlock with unknown variable.
Solution: Give an error. (closes #9188)
Files: src/evalvars.c, src/errors.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.3652

Problem: Can only get text properties one line at a time.
Solution: Add options to prop_list() to use a range of lines and filter by

Files: types. (Yegappan Lakshmanan, closes #9138)
runtime/doc/textprop.txt, src/textprop.c,
src/testdir/test_textprop.vim

Patch 8.2.3653

Problem: Terminal ANSI colors may be wrong.
Solution: Initialize the color type. (closes #9198, closes #9197)
Files: src/terminal.c

Patch 8.2.3654

Problem: GTK: a touch-drag does not update the selection.
Solution: Add GDK_BUTTON1_MASK to the state. (Chris Dalton, close #9196,
closes #9194)
Files: src/gui_gtk_x11.c

Patch 8.2.3655

Problem: Compiler warning for using size_t for int.
Solution: Add a type cast. (Mike Williams, closes #9199)
Files: src/vim9compile.c

Patch 8.2.3656

Problem: Vim9: no error for an environment variable by itself.
Solution: Give a "without effect" error. (closes #9166)
Files: src/ex_eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3657

Problem: Vim9: debug text misses one line of return statement.
Solution: Add a line when not at a debug instruction. (closes #9137)
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.3658

Problem: Duplicate code in xxd.
Solution: Merge duplicated code. Add more tests. (closes #9192)
Files: src/xxd/xxd.c, src/testdir/test_xxd.vim

Patch 8.2.3659

Problem: Integer overflow with large line number.
Solution: Check for overflow. (closes #9202)
Files: src/errors.h, src/ex_docmd.c, src/testdir/test_excmd.vim
src/normal.c, src/testdir/test_normal.vim

Patch 8.2.3660 (after 8.2.3659)

Problem: Overflow check uses wrong number.
Solution: Divide by ten.
Files: src/normal.c

Patch 8.2.3661 (after 8.2.3659)

Problem: Test for put with large count fails.
Solution: Adjust the counts in the test.
Files: src/testdir/test_put.vim

Patch 8.2.3662

Problem: Illegal memory access if malloc() fails.
Solution: Check **'foldmethod'** is not empty. (closes #9207)

Files: src/fold.c

Patch 8.2.3663

Problem: Using %S in printf() does not work correctly.

Solution: Fix the problem and add more tests. (closes #9208)

Files: src/strings.c, src/testdir/test_expr.vim

Patch 8.2.3664

Problem: Cannot adjust sign highlighting for '**cursorline**'.

Solution: Add CursorLineSign and CursorLineFold highlight groups.
(Gregory Anders, closes #9201)

Files: runtime/doc/sign.txt, runtime/doc/syntax.txt, src/drawline.c,
src/highlight.c, src/optiondefs.h, src/popupwin.c,
src/proto/sign.pro, src/sign.c, src/structs.h, src/vim.h,
src/testdir/test_signs.vim

Patch 8.2.3665

Problem: Cannot use a lambda for '**tagfunc**'.

Solution: Use '**tagfunc**' like '**opfunc**'. (Yegappan Lakshmanan, closes #9204)

Files: runtime/doc/options.txt, src/buffer.c, src/option.c,
src/optionstr.c, src/proto/tag.pro, src/structs.h, src/tag.c,
src/testdir/test_tagfunc.vim

Patch 8.2.3666

Problem: Libvterm is outdated.

Solution: Include patches from revision 769 to revision 789.

Files: Filelist, src/libvterm/Makefile, src/libvterm/doc/seqs.txt,
src/libvterm/include/vterm.h, src/libvterm/src/mouse.c,
src/libvterm/src/parser.c, src/libvterm/src/state.c,
src/libvterm/src/vterm.c, src/libvterm/src/vterm_internal.h,
src/libvterm/t/02parser.test, src/libvterm/t/17state_mouse.test,
src/libvterm/t/29state_fallback.test,
src/libvterm/t/40state_selection.test, src/libvterm/t/harness.c,
src/libvterm/t/run-test.pl, src/libvterm/vterm.pc.in,
src/terminal.c

Patch 8.2.3667

Problem: Building libvterm fails with MSVC.

Solution: Don't use C99 construct.

Files: src/libvterm/src/state.c

Patch 8.2.3668

Problem: Messages may be corrupted.

Solution: Use another buffer instead of IObuff. (Yegappan Lakshmanan,
closes #9195)

Files: src/highlight.c, src/testdir/test_highlight.vim

Patch 8.2.3669

Problem: Buffer overflow with long help argument.

Solution: Use snprintf().

Files: src/help.c, src/testdir/test_help.vim

Patch 8.2.3670

Problem: Error checks repeated several times.

Solution: Move the checks to functions. (closes #9213)
Files: src/xd/xd.c

Patch 8.2.3671

Problem: Restarting Insert mode in prompt buffer too often when a callback switches windows and comes back. (Sean Dewar)

Solution: Do not set "restart_edit" when already in Insert mode.
(closes #9212)

Files: src/window.c, src/testdir/test_prompt_buffer.vim

Patch 8.2.3672 (after 8.2.3670)

Problem: Build failure with unsigned char.

Solution: Use int instead of char.

Files: src/xd/xd.c

Patch 8.2.3673

Problem: Crash when allocating signal stack fails.

Solution: Only using sourcing info when available. (closes #9215)

Files: src/globals.h, src/message.c

Patch 8.2.3674

Problem: When ml_get_buf() fails it messes up IObuff.

Solution: Return a local pointer. (closes #9214)

Files: src/memline.c

Patch 8.2.3675

Problem: Using freed memory when vim_strsave() fails.

Solution: Clear "last_sourcing_name". Check for msg_source() called recursively. (closes #8217)

Files: src/message.c

Patch 8.2.3676

Problem: Unused runtime file.

Solution: Remove rgb.txt.

Files: runtime/rgb.txt

Patch 8.2.3677

Problem: After a put the ']' mark is on the last byte of a multi-byte character.

Solution: Move it to the first byte. (closes #9047)

Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.3678 (after 8.2.3677)

Problem: Illegal memory access.

Solution: Ignore changed indent when computing byte offset.

Files: src/register.c

Patch 8.2.3679

Problem: objc file detected as Octave. (Antony Lee)

Solution: Detect objc by preprocessor lines. (Doug Kearns, closes #9223, closes #9220)

Files: runtime/autoload/dist/ft.vim, src/testdir/test_filetype.vim

Patch 8.2.3680

Problem: Repeated code in xxd.
Solution: Change exit_on_ferror() to getc_or_die(). (closes #9226)
Files: src/xxd/xxd.c

Patch 8.2.3681

Problem: Cannot drag popup window after click on a status line. (Sergey Vlasov)
Solution: Reset on_status_line. (closes #9221)
Files: src/mouse.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_drag_04.dump

Patch 8.2.3682

Problem: Vim9: assigning to a script variable drops the required type.
Solution: Lookup the type of the variable and use it. (closes #9219)
Files: src/evalvars.c, src/vim9script.c, src/proto/vim9script.pro, src/testdir/test_vim9_assign.vim

Patch 8.2.3683

Problem: Vim9: cannot use `=expr` in :...do commands.
Solution: Add EX_EXPAND to the commands. (closes #9232)
Files: src/ex_cmds.h, src/testdir/test_vim9_cmd.vim

Patch 8.2.3684

Problem: Blockwise insert does not handle autoindent properly.
Solution: Adjust text column for indent. (closes #9229)
Files: src/ops.c, src/testdir/test_blockedit.vim

Patch 8.2.3685

Problem: Visual Studio project files are not recognized.
Solution: Use the xml file type. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3686

Problem: Filetype detection often mixes up Forth and F#.
Solution: Add a function to inspect the file contents. (Doug Kearns)
Files: runtime/autoload/dist/ft.vim, runtime/doc/filetype.txt, runtime/doc/syntax.txt, runtime/filetype.vim, runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.3687

Problem: Blockwise insert does not handle autoindent properly when tab is inserted.
Solution: Adjust text column for indent before computing column. (closes #9229)
Files: src/ops.c, src/testdir/test_blockedit.vim

Patch 8.2.3688

Problem: The window title is not updated when dragging the scrollbar.
Solution: Call maketitle(). (Christian Brabandt, closes #9238, closes #5383)
Files: src/gui.c

Patch 8.2.3689

Problem: ex_let_one() is too long.
Solution: Split into multiple functions.

Files: src/evalvars.c

Patch 8.2.3690

Problem: Vim9: "filter #pat# cmd" does not work.

Solution: Do not see #pat# as a comment.

Files: src/ex_docmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3691

Problem: Build failure with small features.

Solution: Add #ifdef. (Dominique Pellé)

Files: src/gui.c

Patch 8.2.3692

Problem: Vim9: cannot use :func inside a :def function.

Solution: Make it work.

Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/errors.h,
src/structs.h, src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3693

Problem: Coverity warns for possibly using a NULL pointer.

Solution: Check for NULL and give an error.

Files: src/vim9execute.c, src/errors.h

Patch 8.2.3694

Problem: Cannot use quotes in the count of an Ex command.

Solution: Add getdigits_quoted(). Give an error when misplacing a quote in
a range. (closes #9240)

Files: src/ex_docmd.c, src/charset.c, src/proto/charset.pro,
src/testdir/test_usercommands.vim

Patch 8.2.3695

Problem: Confusing error for missing key.

Solution: Use the actual key for the error. (closes #9241)

Files: src/eval.c, src/testdir/test_listdict.vim

Patch 8.2.3696

Problem: Vim9: error for invalid assignment when skipping.

Solution: Do not check white space when skipping. (closes #9243)

Files: src/evalvars.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3697

Problem: Cannot drag a popup without a border.

Solution: Add the "dragall" option. (closes #9218)

Files: runtime/doc/popup.txt, src/mouse.c, src/popupwin.c, src/vim.h,
src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_drag_05.dump,
src/testdir/dumps/Test_popupwin_drag_06.dump

Patch 8.2.3698

Problem: Match highlighting continues over breakindent.

Solution: Stop before the end column. (closes #9242)

Files: src/match.c, src/proto/match.pro, src/drawline.c,
src/testdir/test_match.vim,
src/testdir/dumps/Test_match_linebreak.dump

Patch 8.2.3699

Problem: The +title feature adds a lot of #ifdef but little code.
Solution: Graduate the +title feature.
Files: src/feature.h, src/alloc.c, src/arglist.c, src/autocmd.c,
src/buffer.c, src/bufwrite.c, src/change.c, src/drawscreen.c,
src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/gui.c,
src/gui_gtk_x11.c, src/if_xcmdsrv.c, src/locale.c, src/main.c,
src/misc2.c, src/netbeans.c, src/option.c, src/optionstr.c,
src/os_amiga.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c,
src/regexp.c, src/term.c, src/ui.c, src/version.c, src/window.c,
src/globals.h, src/option.h, src/optiondefs.h,
runtime/doc/options.txt, runtime/doc/various.txt

Patch 8.2.3700

Problem: Text property highlighting continues over breakindent.
Solution: Stop before the end column. (closes #9242)
Files: src/drawline.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_prop_linebreak.dump

Patch 8.2.3701

Problem: Vim9: invalid LHS is not possible.
Solution: Remove unreachable error message.
Files: src/vim9compile.c

Patch 8.2.3702

Problem: First key in dict is seen as curly expression and fails.
Solution: Ignore failure of curly expression. (closes #9247)
Files: src/typval.c, src/dict.c, src/testdir/test_listdict.vim

Patch 8.2.3703 (after 8.2.3686)

Problem: Most people call F# "fsharp" and not "fs".
Solution: Rename filetype "fs" to "fsharp".
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim,
src/testdir/test_filetype.vim

Patch 8.2.3704

Problem: Vim9: cannot use a list declaration in a :def function.
Solution: Make it work.
Files: runtime/doc/vim9.txt, src/vim9compile.c, src/errors.h,
src/testdir/test_vim9_assign.vim

Patch 8.2.3705

Problem: Cannot pass a lambda name to function() or funcref(). (Yegappan Lakshmanan)
Solution: Handle a lambda name differently.
Files: src/userfunc.c, src/proto/userfunc.pro, src/evalfunc.c,
src/testdir/test_expr.vim

Patch 8.2.3706 (after 8.2.3700)

Problem: Text property highlighting is used on Tab.
Solution: Only set in_linebreak when not on a Tab. (closes #9242)
Files: src/drawline.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_prop_after_tab.dump

Patch 8.2.3707

Problem: Vim9: constant expression of elseif not recognized.
Solution: Set instruction count before generating the expression.
Files: src/vim9compile.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3708 (after 8.2.3707)

Problem: Vim9: test fails with different error.
Solution: Correct the error number.
Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.3709

Problem: Vim9: backtick expression expanded when not desired.
Solution: Only expand a backtick expression for commands that expand their argument. Remove a few outdated TODO comments.
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3710

Problem: Vim9: backtick expression expanded for :global.
Solution: Check the following command.
Files: runtime/doc/vim9.txt, src/vim9compile.c,
src/testdir/test_vim9_cmd.vim

Patch 8.2.3711

Problem: Vim9: memory leak when compiling :elseif fails.
Solution: Cleanup ppconst.
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3712

Problem: Cannot use Vim9 lambda for '**tagfunc**'.
Solution: Make it work, add more tests. (Yegappan Lakshmanan, closes #9250)
Files: runtime/doc/options.txt, src/insexpand.c, src/option.c,
src/testdir/test_tagfunc.vim

Patch 8.2.3713

Problem: MS-Windows: No error message if vimgrep pattern is not matching.
Solution: Give an error message. (Christian Brabandt, closes #9245,
closes #8762)
Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.3714

Problem: Some unused assignments and ugly code in xxd.
Solution: Leave out assignments. Use marco for fprintf(). (closes #9246)
Files: src/xxd/xxd.c

Patch 8.2.3715

Problem: Vim9: valgrind reports spurious problems for a test.
Solution: Move the test to the set that is known to fail.
Files: src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_fails.vim

Patch 8.2.3716

Problem: Vim9: range without a command is not compiled.
Solution: Add the ISN_EXECRANGE byte code.
Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/vim9compile.c,

src/vim9execute.c, src/vim9.h,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3717

Problem: Vim9: error for constant list size is only given at runtime.
Solution: Give the error at compile time if possible.
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3718

Problem: Compiler warns for unused variable without the +textprop feature.
(John Marriott)
Solution: Adjust #ifdefs.
Files: src/drawline.c

Patch 8.2.3719

Problem: MS-Windows: test sometimes runs into existing swap file.
Solution: Use a different file name.
Files: src/testdir/test_buffer.vim

Patch 8.2.3720

Problem: Vim9: Internal error when invoking closure in legacy context.
Solution: Give a more appropriate error message. (closes #9251)
Files: src/errors.h, src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.3721

Problem: Using memory freed by losing the clipboard selection. (Dominique Pellé)
Solution: Check y_array is still valid after calling changed_lines().
(closes #9253)
Files: src/errors.h, src/register.c

Patch 8.2.3722

Problem: Amiga: superfluous messages for freeing lots of yanked text.
Solution: Assume that the machine isn't that slow these days.
Files: src/register.c

Patch 8.2.3723

Problem: When using '**linebreak**' a text property starts too early.
Solution: Decrement "bcol" when looking for property start. (closes #9242)
Files: src/drawline.c, src/testdir/test_textprop.vim,
src/testdir/dumps/Test_prop_after_linebreak.dump

Patch 8.2.3724

Problem: Build error for missing error message in small build.
Solution: Correct #ifdef.
Files: src/errors.h

Patch 8.2.3725

Problem: Cannot use a lambda for '**complefunc**' and '**omnifunc**'.
Solution: Implement lambda support. (Yegappan Lakshmanan, closes #9257)
Files: runtime/doc/options.txt, src/buffer.c, src/insexpand.c,
src/option.c, src/optionstr.c, src/proto/insexpand.pro,

src/proto/tag.pro, src/proto/userfunc.pro, src/structs.h,
src/tag.c, src/userfunc.c, src/testdir/test_ins_complete.vim,
src/testdir/test_tagfunc.vim

Patch 8.2.3726

Problem: README file in a config directory gets wrong filetype.
Solution: Match README before patterns that match everything in a directory.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3727

Problem: In a gnome terminal keys are recognized as mouse events.
Solution: Only recognize DEC mouse events when four numbers are following.
(closes #9256)
Files: src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.3728

Problem: Internal error when passing range() to list2blob().
Solution: Materialize the list first. (closes #9262)
Files: src/blob.c, src/testdir/test_blob.vim

Patch 8.2.3729

Problem: No support for squirrels.
Solution: Recognize nuts. (closes #9259)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3730

Problem: "/etc/Muttrc.d/README" gets filetype muttrc.
Solution: Move the Muttrc.d pattern down, add exception for *.rc files.
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3731

Problem: "set! termcap" shows codes in one column, but not keys.
Solution: Also use one column for keys. (closes #9258)
Files: src/option.c, src/term.c, src/proto/term.pro,
src/testdir/test_set.vim

Patch 8.2.3732 (after 8.2.3731)

Problem: "set! termcap" test fails.
Solution: Account for keys without a t_xx entry.
Files: src/testdir/test_set.vim

Patch 8.2.3733

Problem: Vim9: using "legacy" before range does not work.
Solution: Skip over range before parsing command. (closes #9270)
Files: src/vim9compile.c, src/usercmd.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.3734

Problem: Vim9: crash when no pattern match found.
Solution: Check for error.
Files: src/vim9execute.c

Patch 8.2.3735

Problem: Cannot use a lambda for 'imactivatefunc'.
Solution: Add lambda support for 'imactivatefunc' and 'imstatusfunc'.

(Yegappan Lakshmanan, closes #9275)
Files: runtime/doc/options.txt, src/alloc.c, src/gui_xim.c,
src/optionstr.c, src/proto/gui_xim.pro,
src/testdir/test_iminsert.vim, src/testdir/test_ins_complete.vim

Patch 8.2.3736

Problem: Test fails without the channel feature. (Dominique Pellé)
Solution: Source the check.vim script. (closes #9277)
Files: src/testdir/test_vim9_fails.vim

Patch 8.2.3737

Problem: Test fails without the **'autochdir'** option.
Solution: Check that the option is available. (Dominique Pellé, closes #9272)
Files: src/testdir/test_cd.vim

Patch 8.2.3738

Problem: Screen is cleared when a FocusLost autocommand triggers.
Solution: Do not redraw when at the hit-enter or more prompt. (closes #9274)
Files: src/misc1.c

Patch 8.2.3739

Problem: In wrong directory when using win_execute() with **'acd'** set.
Solution: Restore the directory when returning to the window. (closes #9276)
Files: src/window.c, src/testdir/test_autochdir.vim

Patch 8.2.3740

Problem: Memory left allocated on exit when using Tcl.
Solution: Call Tcl_Finalize().
Files: src/if_tcl.c, src/proto/if_tcl.pro, src/alloc.c

Patch 8.2.3741

Problem: Using freed memory in open command.
Solution: Make a copy of the current line.
Files: src/ex_docmd.c, src/testdir/test_ex_mode.vim

Patch 8.2.3742

Problem: Dec mouse test fails without gnome terminfo entry.
Solution: Check if there is a gnome entry. Also fix **'acd'** test on
MS-Windows. (Ozaki Kiichi, closes #9282)
Files: src/testdir/test_termcodes.vim, src/testdir/test_autochdir.vim

Patch 8.2.3743

Problem: **":sign"** can add a highlight group without a name.
Solution: Give an error if the group name is missing. (closes #9280)
Files: src/sign.c, src/errors.h, src/testdir/test_signs.vim

Patch 8.2.3744

Problem: E854 is not tested; some spelling suggestions are not tested.
Solution: Add a couple of tests. (Dominique Pellé, closes #9279)
Files: src/testdir/test_options.vim, src/testdir/test_spell.vim

Patch 8.2.3745

Problem: Autochdir test fails without the +channel feature.
Solution: Remove the ch_logfile() call. (Dominique Pellé, closes #9281)

Files: src/testdir/test_autochdir.vim

Patch 8.2.3746

Problem: Cannot disassemble function starting with "debug" or "profile".

Solution: Check for white space following. (closes #9273)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3747 (after 8.2.3743)

Problem: Cannot remove highlight from an existing sign. (James McCoy)

Solution: Only reject empty argument for a new sign.

Files: src/sign.c, src/testdir/test_signs.vim

Patch 8.2.3748 (after 8.2.3747)

Problem: Giving an error for an empty sign argument breaks a plugin.

Solution: Do not give an error.

Files: src/sign.c, src/errors.h, src/testdir/test_signs.vim

Patch 8.2.3749

Problem: Error messages are everywhere.

Solution: Move more error messages to errors.h and adjust the names.

Files: src/errors.h, src/regexp_bt.c, src/regexp.c, src/regexp_nfa.c,
src/globals.h, src/memfile.c, src/tag.c, src/getchar.c,
src/bufwrite.c, src/cmdexpand.c

Patch 8.2.3750

Problem: Error messages are everywhere.

Solution: Move more error messages to errors.h and adjust the names.

Files: src/globals.h, src/errors.h, src/blob.c, src/buffer.c,
src/channel.c, src/ex_docmd.c, src/job.c, src/list.c, src/mark.c,
src/misc1.c, src/os_unix.c, src/popupwin.c, src/register.c,
src/session.c, src/spellfile.c, src/term.c, src/userfunc.c

Patch 8.2.3751

Problem: Cannot assign a lambda to an option that takes a function.

Solution: Automatically convert the lambda to a string. (Yegappan Lakshmanan, closes #9286)

Files: runtime/doc/options.txt, src/eval.c, src/proto/eval.pro,
src/evalvars.c, src/if_mzsch.c, src/if_ruby.c, src/if_tcl.c,
src/option.c, src/option.h, src/optiondefs.h,
src/proto/option.pro, src/spell.c, src/typval.c,
src/vim9compile.c, src/testdir/test_iminsert.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_tagfunc.vim

Patch 8.2.3752

Problem: Build error when using Photon GUI.

Solution: Adjust #ifdef. (closes #9288)

Files: src/beval.c

Patch 8.2.3753

Problem: Vim9: function unreferenced while called is never deleted.

Solution: Delete a function when no longer referenced.

Files: src/vim9execute.c, src/userfunc.c, src/proto/userfunc.pro

Patch 8.2.3754 (after 8.2.3615)

Problem: Undesired changing of the indent of the first formatted line.
Solution: Do not indent the first formatted line.
Files: src/textformat.c, src/testdir/test_indent.vim

Patch 8.2.3755

Problem: Coverity warns for using a buffer in another scope.
Solution: Declare the buffer in a common scope.
Files: src/evalvars.c

Patch 8.2.3756

Problem: might crash when callback is not valid.
Solution: Check for valid callback. (Yegappan Lakshmanan, closes #9293)
Files: src/insexpand.c, src/option.c, src/tag.c, src/job.c,
src/userfunc.c, src/testdir/test_iminsert.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_tagfunc.vim

Patch 8.2.3757

Problem: An overlong highlight group name is silently truncated.
Solution: Give an error if the name is too long. (closes #9289)
Files: src/errors.h, src/highlight.c, src/testdir/test_highlight.vim

Patch 8.2.3758

Problem: Options that take a function insufficiently tested.
Solution: Add additional tests and enhance existing tests. (Yegappan Lakshmanan, closes #9298)
Files: src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim,
src/testdir/test_tagfunc.vim

Patch 8.2.3759

Problem: Quickfix buffer becomes hidden while still in a window.
Solution: Check if the closed window is the last window showing the quickfix buffer. (Yegappan Lakshmanan, closes #9303, closes #9300)
Files: src/quickfix.c, src/testdir/test_quickfix.vim, src/window.c

Patch 8.2.3760

Problem: Not automatically handling gnome terminal mouse like xterm.
Solution: Default **'ttymouse'** to "xterm" and recognize Focus events. (issue #9296)
Files: src/os_unix.c

Patch 8.2.3761

Problem: Focus change is not passed on to a terminal window.
Solution: If the current window is a terminal and focus events are enabled send a focus event escape sequence to the terminal.
Files: src/ui.c, src/terminal.c, src/proto/terminal.pro,
src/testdir/test_terminal.vim,
src/testdir/dumps/Test_terminal_focus_1.dump,
src/testdir/dumps/Test_terminal_focus_2.dump

Patch 8.2.3762

Problem: If the quickfix buffer is wiped out getqflist() still returns its number.
Solution: Use zero if the buffer is no longer present. (Yegappan Lakshmanan, closes #9306)

Files: src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.3763

Problem: When editing the command line a FocusLost callback may cause the screen to scroll up.

Solution: Do not redraw at the last line but at the same place where the command line was before. (closes #9295)

Files: src/ex_getln.c, src/ui.c, src/beval.c, src/channel.c, src/drawscreen.c, src/proto/drawscreen.pro, src/job.c, src/popupwin.c, src/sound.c, src/terminal.c, src/time.c, src/testdir/test_terminal.vim, src/testdir/dumps/Test_terminal_focus_1.dump, src/testdir/dumps/Test_terminal_focus_2.dump, src/testdir/dumps/Test_terminal_focus_3.dump

Patch 8.2.3764

Problem: Cannot see any text when window was made zero lines or zero columns.

Solution: Ensure there is at least one line and column. (fixes #9307)

Files: src/window.c, src/proto/window.pro, src/normal.c, src/edit.c, src/testdir/test_window_cmd.vim

Patch 8.2.3765

Problem: Vim9: cannot use a lambda for '**opfunc**' and others.

Solution: Convert the lambda to a string.

Files: src/vim9compile.c, src/vim9.h, src/vim9execute.c, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3766

Problem: Converting a funcref to a string leaves out "g:", causing the meaning of the name depending on the context.

Solution: Prepend "g:" for a global function.

Files: src/eval.c, src/testdir/test_functions.vim

Patch 8.2.3767 (after 8.2.3766)

Problem: Crash when using NULL partial.

Solution: Check for NULL.

Files: src/eval.c

Patch 8.2.3768

Problem: timer_info() has the wrong repeat value in a timer callback. (Sergey Vlasov)

Solution: Do not add one to the repeat value when in the callback. (closes #9294)

Files: src/time.c, src/testdir/test_timers.vim

Patch 8.2.3769

Problem: Zig files are not recognized.

Solution: Add *.zig. (Gregory Anders, closes #9313)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3770

Problem: New compiler warnings from clang-12 and clang-13.

Solution: Adjust CI and suppress some warnings. (Ozaki Kiichi, closes #9314)
Files: .github/workflows/ci.yml, ci/config.mk.clang-12.sed,
src/os_unix.c, src/spellfile.c

Patch 8.2.3771

Problem: Vim9: accessing freed memory when checking type.
Solution: Make a copy of a function type.
Files: src/structs.h, src/evalvars.c, src/vim9script.c,
src/testdir/test_vim9_func.vim

Patch 8.2.3772

Problem: Timer info test fails on slow machine.
Solution: Use WaitForAssert().
Files: src/testdir/test_timers.vim

Patch 8.2.3773

Problem: Wrong window size when a modeline changes 'columns' and there is more than one tabpage. (Michael Soyka)
Solution: Adjust the frames of all tabpages. (closes #9315)
Files: src/window.c

Patch 8.2.3774 (after 8.2.3773)

Problem: Test for command line height fails.
Solution: Use another way to handle window size change.
Files: src/structs.h, src/window.c

Patch 8.2.3775

Problem: Vim9: lambda compiled without outer context when debugging.
Solution: When compiling a lambda for debugging also compile it without. (closes #9302)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.3776

Problem: When a tags file line is long a tag may not be found.
Solution: When increasing the buffer size read the same line again.
Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.2.3777

Problem: Spell file write error not checked.
Solution: Check writing the prefix conditions. (Bjorn Linse, closes #9323)
Files: src/spellfile.c

Patch 8.2.3778

Problem: Lambda debug test fails in some configurations.
Solution: Check feature in a legacy function.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.3779

Problem: Using freed memory when defining a user command from a user command.
Solution: Do not use the command pointer after executing the command. (closes #9318)
Files: src/usercmd.c, src/testdir/test_usercommands.vim

Patch 8.2.3780

Problem: `":cd"` works differently on MS-Windows.
Solution: Add the `'cdhome'` option. (closes #9324)
Files: runtime/doc/editing.txt, runtime/doc/options.txt,
runtime/doc/quickref.txt, runtime/optwin.vim, src/ex_docmd.c,
src/option.h, src/optiondefs.h, src/testdir/runtest.vim,
src/testdir/test_options.vim

Patch 8.2.3781

Problem: The option `window script` is outdated.
Solution: Add several changes.
Files: runtime/optwin.vim

Patch 8.2.3782

Problem: Vim9: no error if a function shadows a script variable.
Solution: Check the function doesn't shadow a variable. (closes #9310)
Files: src/userfunc.c, src/evalvars.c, src/vim.h,
src/testdir/test_vim9_script.vim

Patch 8.2.3783

Problem: Confusing error for using a variable as a function.
Solution: If a function is not found but there is a variable, give a more
useful error. (issue #9310)
Files: src/eval.c, src/userfunc.c, src/proto/userfunc.pro,
src/structs.h, src/vim9execute.c, src/testdir/test_functions.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_vim9_func.vim

Patch 8.2.3784

Problem: The help for options is outdated.
Solution: Include all the recent changes.
Files: runtime/doc/options.txt

Patch 8.2.3785

Problem: Running CI on macOS with gcc is not useful.
Solution: Only use clang. (Ozaki Kiichi, closes #9326) Also build with
normal features.
Files: .github/workflows/ci.yml

Patch 8.2.3786

Problem: Test fails because of using Vim9 syntax in legacy function.
Solution: Add `"call"`.
Files: src/testdir/test_functions.vim

Patch 8.2.3787

Problem: No proper formatting of a C line comment after a statement.
Solution: Find the start of the line comment, insert the comment leader and
indent the comment properly.
Files: src/change.c, src/proto/change.pro, src/search.c,
src/proto/search.pro, src/cindent.c, src/edit.c, src/normal.c,
src/textformat.c, src/testdir/test_textformat.vim,
src/testdir/test_cindent.vim

Patch 8.2.3788

Problem: Lambda for option that is a function may be garbage collected.

Solution: Set a reference in the funcref. (Yegappan Lakshmanan, closes #9330)
Files: src/eval.c, src/evalbuffer.c, src/evalvars.c, src/gui_xim.c, src/insexpand.c, src/ops.c, src/proto/eval.pro, src/proto/gui_xim.pro, src/proto/insexpand.pro, src/proto/ops.pro, src/proto/tag.pro, src/quickfix.c, src/tag.c, src/testdir/test_iminsert.vim, src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim, src/testdir/test_quickfix.vim, src/testdir/test_tagfunc.vim

Patch 8.2.3789

Problem: Test_window_minimal_size can fail on a slow machine.
Solution: Do not rely on timers firing at the expected time. (Ozaki Kiichi, closes #9335)
Files: src/testdir/test_window_cmd.vim

Patch 8.2.3790

Problem: Test for term_gettitle() fails in some environments.
Solution: Make the digits after "VIM" optional. (Kenta Sato, closes #9334)
Files: src/testdir/test_terminal2.vim

Patch 8.2.3791

Problem: Build error with +cindent but without +smartindent.
Solution: Move declaration of "do_cindent". (John Marriott)
Files: src/change.c

Patch 8.2.3792

Problem: Setting *func options insufficiently tested.
Solution: Improve tests. (Yegappan Lakshmanan, closes #9337)
Files: src/testdir/test_iminsert.vim, src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim, src/testdir/test_quickfix.vim, src/testdir/test_tagfunc.vim

Patch 8.2.3793

Problem: Using "g:Func" as a funcref does not work in script context because "g:" is dropped.
Solution: Keep "g:" in the name. Also add parenthesis to avoid confusing operator precedence. (closes #9336)
Files: src/evalvars.c, src/testdir/test_vim9_func.vim

Patch 8.2.3794

Problem: Vim9: cannot find script-local func using "s:". (Yegappan Lakshmanan)
Solution: Skip the "s:".
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3795

Problem: Too many #ifdefs.
Solution: Graduate the jumplist feature.
Files: runtime/doc/various.txt, runtime/doc/motion.txt, src/feature.h, src/buffer.c, src/change.c, src/evalfunc.c, src/ex_docmd.c, src/mark.c, src/normal.c, src/undo.c, src/version.c, src/viminfo.c, src/window.c, src/structs.h, src/testdir/test_changelist.vim, src/testdir/test_jumplist.vim,

src/testdir/test_normal.vim

Patch 8.2.3796

Problem: The funcexe_T struct members are not named consistently.

Solution: Prefix "fe_" to all the members.

Files: src/structs.h, src/eval.c, src/list.c, src/regexp.c,
src/terminal.c, src/userfunc.c, src/vim9execute.c

Patch 8.2.3797

Problem: No good reason to limit the message history in the tiny version.

Solution: Always use 200.

Files: runtime/doc/message.txt, src/feature.h

Patch 8.2.3798

Problem: A :def callback function postpones an error message.

Solution: Display the error after calling the function. (closes #9340)

Files: src/userfunc.c, src/testdir/test_vim9_func.vim,
src/testdir/dumps/Test_opfunc_error.dump

Patch 8.2.3799 (after 8.2.3798)

Problem: Edit test hangs or fails.

Solution: Do not rethrow an exception when inside try/catch.

Files: src/userfunc.c

Patch 8.2.3800

Problem: When cross compiling the output of "uname" cannot be set. (Ben Reeves)

Solution: Use cache variables. (closes #9338)

Files: src/configure.ac, src/auto/configure

Patch 8.2.3801

Problem: If a terminal shows in two windows, only one is redrawn.

Solution: Reset the dirty row range only after redrawing all windows.
(closes #9341)

Files: src/terminal.c, src/proto/terminal.pro, src/drawscreen.c,
src/testdir/test_terminal.vim

Patch 8.2.3802

Problem: Terminal in two windows test fails on some systems.

Solution: Wait a bit between commands.

Files: src/testdir/test_terminal.vim

Patch 8.2.3803

Problem: Crash when **'writedelay'** is set and using a terminal window to execute a shell command.

Solution: Check that "tl_vterm" isn't NULL. (closes #9346)

Files: src/terminal.c

Patch 8.2.3804

Problem: Script context not set when copying **'swf'** and **'ts'**.

Solution: Use COPY_OPT_SCTX with the right argument. (closes #9347)

Files: src/option.c

Patch 8.2.3805

Problem: i3config files are not recognized.
Solution: Add patterns to match i3config files. (Quentin Hixon, closes #7969)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3806

Problem: Terminal focus test fails sometimes.
Solution: Run the test function before others.
Files: src/testdir/test_terminal.vim

Patch 8.2.3807

Problem: Vim9: can call import with star directly.
Solution: Check that the import used star.
Files: src/userfunc.c, src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.3808

Problem: Vim9: obsolete TODO items
Solution: Remove the comments.
Files: src/vim9execute.c

Patch 8.2.3809

Problem: Vim9: crash when garbage collecting a nested partial. (Virginia Senioria)
Solution: Set references in all the funcstacks. (closes #9348)
Files: src/vim9execute.c, src/proto/vim9execute.pro, src/structs.h, src/eval.c, src/testdir/test_vim9_func.vim

Patch 8.2.3810

Problem: Vim9: expr4 test fails on MS-Windows.
Solution: Do not give an error for a missing function name when skipping.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3811

Problem: The opfunc error test fails on a slow machine.
Solution: Use WaitForAssert().
Files: src/testdir/test_vim9_func.vim

Patch 8.2.3812

Problem: Vim9: leaking memory in numbered function test.
Solution: Skip "g:" when checking for numbered function. Clean up after errors properly.
Files: src/userfunc.c

Patch 8.2.3813

Problem: confusing error when using :cc without error list. (Gary Johnson)
Solution: Give the "no errors" error.
Files: src/ex_docmd.c, src/testdir/test_quickfix.vim

Patch 8.2.3814

Problem: .csx files and .sln files are not recognized.
Solution: Add filetype patterns. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3815

Problem: Vim9: cannot have a multi-line dict inside a block.
Solution: Do not split the command at a line break, handle NL characters as white space.
Files: src/ex_docmd.c, src/charset.c, src/proto/charset.pro, src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3816

Problem: Compiler warning for possible loss of data on MS-Windows.
Solution: Add type cast. (Mike Williams, closes #9349)
Files: src/userfunc.c

Patch 8.2.3817 (after 8.2.3815)

Problem: Vim9: Not using NL as command end does not work for :autocmd.
Solution: Only ignore NL for commands with an expression argument.
Files: src/ex_cmds.h, src/ex_docmd.c, src/testdir/test_usercommands.vim

Patch 8.2.3818

Problem: Cannot filter or map characters in a string.
Solution: Make filter() and map() work on a string. (Naruhiko Nishino, closes #9327)
Files: runtime/doc/eval.txt, src/errors.h, src/list.c, src/testdir/test_filter_map.vim

Patch 8.2.3819 (after 8.2.3818)

Problem: Test fails because error message changed.
Solution: Update screendumps.
Files: src/testdir/dumps/Test_popupwin_three_errors_1.dump, src/testdir/dumps/Test_popupwin_three_errors_2.dump

Patch 8.2.3820

Problem: "vrc" does not replace composing characters, while "rc" does.
Solution: Check the byte length including composing characters. (closes #9351)
Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.2.3821

Problem: ASAN test run fails.
Solution: Use asan_symbolize-13 instead of asan_symbolize-11.
Files: .github/workflows/ci.yml

Patch 8.2.3822

Problem: Leaking memory in map() and filter(), cannot use a string argument in Vim9 script.
Solution: Fix the leak, adjust the argument check, also run the tests as Vim9 script. (Yegappan Lakshmanan, closes #9354)
Files: src/errors.h, src/evalfunc.c, src/list.c, src/proto/typval.pro, src/testdir/test_filter_map.vim, src/testdir/test_vim9_builtin.vim, src/typval.c

Patch 8.2.3823

Problem: Test for visual replace is in wrong function.
Solution: Move it to another function.
Files: src/testdir/test_visual.vim

Patch 8.2.3824

Problem: No ASAN support for MSVC.

Solution: Add ASAN support and fix a couple of uncovered problems. (Yegappan Lakshmanan, closes #9357)

Files: src/Make_mvc.mak, src/findfile.c, src/os_mswin.c,
src/testdir/test_fnamemodify.vim

Patch 8.2.3825

Problem: Various comments could be improved.

Solution: Improve the comments.

Files: src/getchar.c, src/mbyte.c, src/regexp_nfa.c,
src/testdir/test_edit.vim, src/gui_motif.c

Patch 8.2.3826

Problem: Vim9: using "g:Func" as a funcref does not work in a :def function.

Solution: Include "g:" in the function name. (closes #9336)

Files: src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3827

Problem: Huntr badge does not really fit in the list.

Solution: Move the link to Huntr to the issue template.

Files: Filelist, .github/ISSUE_TEMPLATE/bug_report.yml, README.md

Patch 8.2.3828

Problem: when opening a terminal from a timer the first typed character is lost. (Virginia Senioria)

Solution: When opening a terminal while waiting for a character put K_IGNORE in the input buffer.

Files: src/terminal.c, src/edit.c, src/testdir/test_terminal.vim

Patch 8.2.3829

Problem: No error when setting a func option to a script-local function.

Solution: Give an error if the name starts with "s:". (closes #9358)

Files: src/option.c, src/testdir/test_tagfunc.vim,
src/testdir/dumps/Test_set_tagfunc_on_cmdline.dump

Patch 8.2.3830

Problem: Error messages are spread out.

Solution: Move more error messages to errors.h.

Files: src/globals.h, src/errors.h, src/buffer.c, src/dict.c, src/diff.c,
src/digraph.c, src/eval.c, src/evalfunc.c, src/evalvars.c,
src/misc2.c, src/quickfix.c, src/typval.c, src/ui.c,
src/userfunc.c, src/vim9compile.c, src/vim9execute.c,
src/vim9type.c, src/window.c

Patch 8.2.3831

Problem: Opfunc test fails when missing feature changes function name. (Dominique Pellé)

Solution: Check the relevant screen line instead of using a screendump. (closes #9360)

Files: src/testdir/test_vim9_func.vim,
src/testdir/dumps/Test_opfunc_error.dump

Patch 8.2.3832 (after 8.2.3830)

Problem: Test fails because of changed error message.

Solution: Adjust the expected error message.

Files: src/testdir/test_vimscript.vim

Patch 8.2.3833

Problem: Error from term_start() not caught by try/catch.

Solution: save and restore did_emsg when applying autocommands. (Ozaki Kiichi, closes #9361)

Files: src/autocmd.c, src/testdir/test_terminal3.vim

Patch 8.2.3834

Problem: Test_out_cb often fails on Mac.

Solution: Increase the timeout with every retry.

Files: src/testdir/test_channel.vim

Patch 8.2.3835

Problem: The inline-function example does not work.

Solution: Drop ":let". Add EX_EXPR_ARG to CMD_var. (issue #9352)

Files: runtime/doc/vim9.txt, src/ex_cmds.h,
src/testdir/test_vim9_expr.vim

Patch 8.2.3836

Problem: Vim9: comment after expression not skipped to find NL.

Solution: After evaluating an expression look for a newline after a # comment.

Files: src/eval.c

Patch 8.2.3837

Problem: QNX: crash when compiled with GUI but using terminal.

Solution: Check gui.in_use is set. (Hirohito Higashi, closes #9363)

Files: src/autocmd.c

Patch 8.2.3838

Problem: Cannot use script-local function for setting *func options.

Solution: Use the script context. (Yegappan Lakshmanan, closes #9362)

Files: src/option.c, src/testdir/dumps/Test_set_tagfunc_on_cmdline.dump,
src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim,
src/testdir/test_quickfix.vim, src/testdir/test_tagfunc.vim

Patch 8.2.3839

Problem: Using \z() with \z1 not tested for syntax highlighting.

Solution: Add a test. (Dominique Pellé, closes #9365)

Files: src/testdir/test_syntax.vim

Patch 8.2.3840

Problem: Useless test for negative index in check functions.

Solution: Remove the test for negative index. (Naruhiko Nishino, closes #9364)

Files: src/typval.c

Patch 8.2.3841

Problem: Vim9: outdated TODO items, disabled tests that work.

Solution: Remove TODO items, run tests that work now. Check that a dict item isn't locked.
Files: src/vim9execute.c, src/evalvars.c, src/errors.h, src/globals.h, src/testdir/test_listdict.vim, src/testdir/test_vim9_assign.vim

Patch 8.2.3842

Problem: Vim9: can change locked list and list items.
Solution: Check that a list and list item isn't locked.
Files: src/vim9execute.c, src/testdir/test_listdict.vim

Patch 8.2.3843

Problem: Dep3patch files are not recognized.
Solution: Recognize dep3patch files by their location and content. (James McCoy, closes #9367)
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3844

Problem: Vim9: no type error if assigning a value with type func(number) to a variable of type func(string).
Solution: Use check_type_maybe(): return MAYBE if a runtime type check is useful. (issue #8492)
Files: src/vim9type.c, src/proto/vim9type.pro, src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3845

Problem: Vim9: test fails when the channel feature is missing.
Solution: Check for the channel feature. (Dominique Pellé, closes #9368)
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.3846

Problem: No error when using control character for **'lcs'** or **'fcs'**.
Solution: Use char2cells() to check the width. (closes #9369)
Files: src/screen.c, src/testdir/test_display.vim, src/testdir/test_listchars.vim

Patch 8.2.3847

Problem: Illegal memory access when using a lambda with an error.
Solution: Avoid skipping over the NUL after a string.
Files: src/eval.c, src/testdir/test_lambda.vim

Patch 8.2.3848

Problem: Cannot use reduce() for a string.
Solution: Make reduce() work with a string. (Naruhiko Nishino, closes #9366)
Files: runtime/doc/eval.txt, src/errors.h, src/evalfunc.c, src/list.c, src/typval.c, src/proto/typval.pro, src/testdir/test_listdict.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3849

Problem: Functions implementing reduce and map are too long.
Solution: Use a function for each type of value. Add a few more test cases and add to the help. (Yegappan Lakshmanan, closes #9370)
Files: runtime/doc/eval.txt, src/list.c, src/testdir/test_listdict.vim

Patch 8.2.3850

Problem: Illegal memory access when displaying a partial.
Solution: Terminate the string with a NUL. (closes #9371)
Files: src/eval.c, src/testdir/test_messages.vim

Patch 8.2.3851

Problem: Vim9: overhead when comparing string, dict or function.
Solution: Call the intended compare function directly. Refactor to avoid duplicated code.
Files: src/vim9execute.c, src/typval.c, src/proto/typval.pro

Patch 8.2.3852

Problem: Vim9: not enough tests.
Solution: Also run existing tests for Vim9 script. Make errors more consistent.
Files: src/testdir/test_listdict.vim, src/eval.c, src/vim9compile.c, src/errors.h

Patch 8.2.3853

Problem: Vim9: not enough tests.
Solution: Run more existing tests for Vim9 script.
Files: src/testdir/test_listdict.vim

Patch 8.2.3854

Problem: Vim9: inconsistent arguments for test functions.
Solution: When :def function and script have different arguments use a list with two items instead of a separate function.
Files: src/testdir/vim9.vim, src/testdir/test_execute_func.vim, src/testdir/test_float_func.vim, src/testdir/test_functions.vim, src/testdir/test_glob2regpat.vim, src/testdir/test_listdict.vim, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.3855

Problem: Illegal memory access when displaying a blob.
Solution: Append a NUL at the end. (Yegappan Lakshmanan, closes #9372)
Files: src/blob.c, src/regexp_nfa.c, src/testdir/test_blob.vim, src/testdir/test_messages.vim

Patch 8.2.3856

Problem: Vim9: not enough tests.
Solution: Run more expression tests also with Vim9. Fix an uncovered problem.
Files: src/vim9compile.c, src/testdir/test_expr.vim, src/testdir/vim9.vim

Patch 8.2.3857

Problem: Vim9: inconsistent error for using function().
Solution: Use a runtime type check for the result of function(). (closes #8492)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3858

Problem: Vim9: not enough tests.
Solution: Add tests for :try/:catch and :redir. Add missing type check.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_cmd.vim

Patch 8.2.3859

Problem: Vim9: some code lines not tested.
Solution: Add a few specific tests.
Files: src/vim9compile.c, src/testdir/test_vim9_cmd.vim, src/errors.h,
src/testdir/test_vim9_script.vim

Patch 8.2.3860

Problem: Vim9: codecov struggles with the file size.
Solution: Split vim9compile.c into four files.
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9instr.c,
src/proto/vim9instr.pro, src/vim9expr.c, src/proto/vim9expr.pro,
src/vim9cmds.c, src/proto/vim9cmds.pro, src/vim9.h,
src/vim9execute.c, src/vim9script.c, src/proto.h,
src/Makefile, src/Make_ami.mak, src/Make_cyg_ming.mak,
src/Make_mvc.mak, src/Make_vms.mms

Patch 8.2.3861

Problem: List of distributed files is outdated.
Solution: Add new files.
Files: Filelist

Patch 8.2.3862

Problem: Crash on exit with EXITFREE and using win_execute().
Solution: Also save and restore tp_topframe. (issue #9374)
Files: src/evalwindow.c, src/testdir/test_execute_func.vim

Patch 8.2.3863 (after 8.2.3860)

Problem: Various build flags accidentally enabled.
Solution: Revert several lines in Makefile.
Files: src/Makefile

Patch 8.2.3864

Problem: Cannot disable requesting key codes from xterm.
Solution: Add the '**xtermcodes**' option, default on.
Files: runtime/doc/options.txt, runtime/doc/term.txt, src/option.h,
src/optiondefs.h, src/term.c, runtime/optwin.vim

Patch 8.2.3865

Problem: Vim9: compiler complains about using "try" as a struct member.
Solution: Rename "try" to "tryref".
Files: src/vim9.h, src/vim9cmds.c, src/vim9execute.c, src/vim9instr.c

Patch 8.2.3866

Problem: Vim9: type checking global variables is inconsistent.
Solution: Use the "unknown" type in more places.
Files: src/globals.h, src/vim9expr.c, src/vim9instr.c, src/vim9cmds.c,
src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3867

Problem: Implementation of some list functions too complicated.
Solution: Refactor do_sort_uniq(), f_count() and extend() (Yegappan Lakshmanan, closes #9378)
Files: src/list.c

Patch 8.2.3868 (after 8.2.3866)

Problem: Vim9: function test fails.
Solution: Add missing changes. Add test for earlier patch.
Files: src/vim9type.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3869

Problem: Vim9: type checking for "any" is inconsistent.
Solution: Always use a runtime type check for using "any" for a more specific type.
Files: src/vim9type.c, src/vim9compile.c, src/vim9expr.c, src/testdir/test_vim9_func.vim

Patch 8.2.3870

Problem: MS-Windows: wrong working directory when opening two files with right-click context menu. (Gabriel Dupras)
Solution: Use the working directory and pass it on to the process creation. (Nir Lichtman, closes #9382, closes #8874)
Files: src/GvimExt/gvimext.cpp, src/GvimExt/gvimext.h

Patch 8.2.3871

Problem: List.c contains code for dict and blob.
Solution: Refactor to put code where it belongs. (Yegappan Lakshmanan, closes #9386)
Files: src/blob.c, src/dict.c, src/list.c, src/proto/blob.pro, src/proto/dict.pro, src/proto/list.pro, src/proto/strings.pro, src/strings.c, src/structs.h, src/testdir/test_filter_map.vim, src/testdir/test_listdict.vim, src/testdir/test_sort.vim

Patch 8.2.3872

Problem: Vim9: finddir() and uniq() return types can be more specific.
Solution: Adjust the return type.
Files: src/evalfunc.c, src/testdir/vim9.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3873

Problem: go.mod files are not recognized.
Solution: Check for the file name. (closes #9380)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3874

Problem: Cannot highlight the number column for a sign.
Solution: Add the "numhl" argument. (James McCoy, closes #9381)
Files: runtime/doc/options.txt, runtime/doc/sign.txt, src/drawline.c, src/popupwin.c, src/proto/sign.pro, src/sign.c, src/structs.h, src/testdir/test_signs.vim

Patch 8.2.3875

Problem: gcc complains about buffer overrun.
Solution: Use mch_memmove() instead of STRCPY(). (John Marriott)

Files: src/dict.c

Patch 8.2.3876

Problem: 'cindent' does not recognize inline namespace.

Solution: Skip over "inline" to find "namespace". (closes #9383)

Files: src/cindent.c, src/testdir/test_cindent.vim

Patch 8.2.3877

Problem: Function does not abort after a type error in compare

Solution: Check getting number fails. (closes #9384)

Files: src/typval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.3878

Problem: Vim9: debugger tries to read more lines than there are.

Solution: Check the number of lines. (closes #9394)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.3879

Problem: getreg() and getregtype() contain dead code.

Solution: Remove the needless check. (closes #9392) Also refactor to put common code in a shared function.

Files: src/evalfunc.c

Patch 8.2.3880

Problem: Solution filter files are not recognized.

Solution: Add pattern *.slnf and use json. (Doug Kearns)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3881

Problem: QNX: crash when compiled with GUI but using terminal.

Solution: Check the gui.in_use flag. (Hirohito Higashi, closes #9391)

Files: src/main.c

Patch 8.2.3882 (after 8.2.3879)

Problem: More duplicated code in f_getreginfo().

Solution: Also use getreg_get_regname(). (closes #9398)

Files: src/evalfunc.c

Patch 8.2.3883

Problem: Crash when switching to other regexp engine fails.

Solution: Check for regprog being NULL.

Files: src/ex_cmds.c

Patch 8.2.3884

Problem: Crash when clearing the argument list while using it.

Solution: Lock the argument list for ":all".

Files: src/arglist.c, src/testdir/test_arglist.vim

Patch 8.2.3885

Problem: Arglist test fails.

Solution: Adjust for locking the arglist for ":all".

Files: src/testdir/test_arglist.vim

Patch 8.2.3886

Problem: Can define autocmd for every event by using "au!".
Solution: Check if a command is present also for "au!".
Files: src/autocmd.c, src/testdir/test_autocmd.vim,
src/testdir/test_arglist.vim

Patch 8.2.3887

Problem: E1135 is used for two different errors.
Solution: Renumber one error.
Files: src/errors.h, src/testdir/test_mapping.vim

Patch 8.2.3888

Problem: The argument list may contain duplicates.
Solution: Add the :argdedupe command. (Nir Lichtman, closes #6235)
Files: runtime/doc/editing.txt, runtime/doc/index.txt, src/arglist.c,
src/ex_cmdidxs.h, src/ex_cmds.h, src/proto/arglist.pro,
src/testdir/test_arglist.vim

Patch 8.2.3889

Problem: Duplicate code for translating script-local function name.
Solution: Move the code to get_scriptlocal_funcname(). (Yegappan Lakshmanan,
closes #9393)
Files: src/evalfunc.c, src/evalvars.c, src/option.c, src/userfunc.c,
src/proto/userfunc.pro, src/testdir/test_expr.vim,
src/testdir/test_normal.vim

Patch 8.2.3890

Problem: Vim9: type check for using v: variables is basic.
Solution: Specify a more precise type.
Files: src/evalvars.c, src/proto/evalvars.pro, src/vim9instr.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.3891

Problem: Github CI: workflows may overlap.
Solution: Cancel previous workflows when starting a new one. (Yegappan
Lakshmanan, closes #9400)
Files: .github/workflows/ci.yml, .github/workflows/codeql-analysis.yml

Patch 8.2.3892

Problem: When modifyOtherKeys is used **CTRL-C** is not recognized.
Solution: Check for uppercase C as well, fix minimum length.
Files: src/ui.c

Patch 8.2.3893

Problem: Vim9: many local variables are initialized with an instruction.
Solution: Initialize local variables to zero to avoid the instructions.
Files: src/vim9execute.c, src/vim9compile.c, src/vim9instr.c,
src/proto/vim9instr.pro, src/vim9cmds.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.3894

Problem: Vim9: no proper type check for first argument of call().
Solution: Add specific type check.
Files: src/evalfunc.c, src/typval.c, src/proto/typval.pro,
src/errors.h, src/testdir/test_vim9_builtin.vim

Patch 8.2.3895

Problem: Vim9: confusing error when using function() with a number.
Solution: Check for a function or string argument.
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3896

Problem: Vim9: no test for nested function not available later.
Solution: Add a test.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.3897

Problem: Vim9: the second argument of map() and filter() is not checked at compile time.
Solution: Add more specific type check for the second argument.
Files: src/evalfunc.c, src/globals.h, src/list.c,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.3898

Problem: Vim9: not sufficient testing for variable initialization.
Solution: Add another test case.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.3899 (after 8.2.3897)

Problem: Vim9: test for map() on string fails.
Solution: Expect string return type.
Files: src/evalfunc.c

Patch 8.2.3900

Problem: It is not easy to use a script-local function for an option.
Solution: recognize s: and <SID> at the start of the expression. (Yegappan Lakshmanan, closes #9401)
Files: runtime/doc/diff.txt, runtime/doc/fold.txt,
runtime/doc/options.txt, runtime/doc/print.txt, src/optionstr.c,
src/testdir/test_diffmode.vim, src/testdir/test_edit.vim,
src/testdir/test_fold.vim, src/testdir/test_gf.vim,
src/testdir/test_gui.vim, src/testdir/test_hardcopy.vim,
src/testdir/test_normal.vim

Patch 8.2.3901

Problem: Vim9: Cannot set 'cpo' in main .vimrc if using Vim9 script.
Solution: Do not restore 'cpo' at the end of the main .vimrc.
Files: runtime/doc/vim9.txt, runtime/doc/options.txt, src/scriptfile.c,
src/structs.h, src/testdir/test_vim9_script.vim

Patch 8.2.3902

Problem: Vim9: double free with nested :def function.
Solution: Pass "line_to_free" from compile_def_function() and make sure cmdlinep is valid.
Files: src/vim9compile.c, src/userfunc.c, src/proto/userfunc.pro,
src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.3903

Problem: "gM" does not count tabs as expected.
Solution: Use `linetabsize()` instead of `mb_string2cells()`. (closes #9409)
Files: `src/normal.c`, `src/testdir/test_normal.vim`

Patch 8.2.3904

Problem: Vim9: skip expression type is not checked at compile time.
Solution: Add argument type checks.
Files: `src/evalfunc.c`, `src/testdir/test_vim9_builtin.vim`

Patch 8.2.3905

Problem: Dockerfile using prefix name not recognized.
Solution: Recognize `Dockerfile.*`. (closes #9410)
Files: `runtime/filetype.vim`, `src/testdir/test_filetype.vim`

Patch 8.2.3906

Problem: Vim9 help still contains "under development" warnings.
Solution: Remove the explicit warning.
Files: `runtime/doc/vim9.txt`

Patch 8.2.3907

Problem: Error messages are spread out.
Solution: Move error messages to `errors.h`. Avoid duplicates.
Files: `src/userfunc.c`, `src/ex_cmds.c`, `src/viminfo.c`, `src/errors.h`,
`src/testdir/test_user_func.vim`

Patch 8.2.3908

Problem: Cannot use a script-local function for **'foldtext'**.
Solution: Expand `"s:"` and `"<SID>"`. (Yegappan Lakshmanan, closes #9411)
Files: `runtime/doc/fold.txt`, `src/optionstr.c`, `src/strings.c`,
`src/testdir/test_blob.vim`, `src/testdir/test_expr.vim`,
`src/testdir/test_filter_map.vim`, `src/testdir/test_fold.vim`,
`src/testdir/test_listdict.vim`

Patch 8.2.3909

Problem: Containerfile using prefix name not recognized.
Solution: Recognize `Containerfile.*`.
Files: `runtime/filetype.vim`, `src/testdir/test_filetype.vim`

Patch 8.2.3910

Problem: When the compare function of `sort()` produces an error then `sort()` does not abort.
Solution: Check if `did_emsg` was incremented.
Files: `src/list.c`, `src/testdir/test_vim9_builtin.vim`

Patch 8.2.3911

Problem: Vim9: type check for `filter()` does not accept unknown.
Solution: Also accept unknown for the return type. (closes #9413)
Files: `src/evalfunc.c`, `src/testdir/test_vim9_builtin.vim`

Patch 8.2.3912

Problem: The `ins_complete()` function is much too long.
Solution: Split it up into multiple functions. (Yegappan Lakshmanan, closes #9414)
Files: `src/insexpand.c`

Patch 8.2.3913

Problem: Help for expressions does not mention Vim9 syntax.
Solution: Add the rules for Vim9 to the expression help. Rename functions to match the help.
Files: runtime/doc/vim9.txt, runtime/doc/eval.txt, src/vim9expr.c

Patch 8.2.3914

Problem: Various spelling mistakes in comments.
Solution: Fix the mistakes. (Dominique Pellé, closes #9416)
Files: src/alloc.c, src/blowfish.c, src/buffer.c, src/cindent.c, src/clipboard.c, src/diff.c, src/drawline.c, src/edit.c, src/ex_cmds.c, src/ex_docmd.c, src/findfile.c, src/fold.c, src/getchar.c, src/gui.c, src/gui_athena.c, src/gui_gtk.c, src/gui_motif.c, src/gui_photon.c, src/gui_w32.c, src/gui_xmew.c, src/if_python.c, src/if_python3.c, src/if_xcmdsrv.c, src/main.c, src/memline.c, src/menu.c, src/message.c, src/misc1.c, src/move.c, src/option.c, src/os_amiga.c, src/os_mac.h, src/os_mac_conv.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/os_win32.h, src/quickfix.c, src/regexp_nfa.c, src/screen.c, src/scriptfile.c, src/spell.c, src/spellfile.c, src/spellsuggest.c, src/strings.c, src/term.c, src/terminal.c, src/testdir/test_debugger.vim, src/testdir/test_source.vim, src/textformat.c, src/userfunc.c, src/vim.h, src/vim9.h, src/vim9cmds.c, src/vim9execute.c, src/winclip.c, src/window.c

Patch 8.2.3915

Problem: illegal memory access when completing with invalid bytes.
Solution: Avoid going over the end of the completion text.
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.3916

Problem: No error for passing an invalid line number to append().
Solution: In Vim9 script check for a non-negative number. (closes #9417)
Files: src/evalbuffer.c, src/textprop.c, src/errors.h, src/indent.c, src/eval.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3917

Problem: The eval.txt helpfile is way too big.
Solution: Move the builtin function details to a separate file.
Files: runtime/doc/eval.txt, runtime/doc/builtin.txt, runtime/doc/Makefile, runtime/doc/help.txt, runtime/doc/remote.txt

Patch 8.2.3918 (after 8.2.3916)

Problem: Function list test fails.
Solution: Adjust the test for the new location of the function list.
Files: src/testdir/test_function_lists.vim

Patch 8.2.3919

Problem: Vim9: wrong argument for append() results in two errors.
Solution: Check did_emsg. Also for setline(). Adjust the help for appendbuflines().
Files: runtime/doc/builtin.txt, src/evalbuffer.c, src/typval.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3920

Problem: Restoring directory after using another window is inefficient.
Solution: Only restore the directory for win_execute(). Apply **'autochdir'** only when needed.
Files: src/evalwindow.c, src/testdir/test_autochdir.vim

Patch 8.2.3921

Problem: The way xdiff is used is inefficient.
Solution: Use hunk_func instead of the out_line callback. (Lewis Russell, closes #9344)
Files: src/diff.c

Patch 8.2.3922

Problem: Cannot build with dynamic Ruby 3.1.
Solution: Add "_EXTRA" variables for CI. Add missing functions. (Ozaki Kiichi, closes #9420)
Files: ci/config.mk.clang-12.sed, ci/config.mk.clang.sed, ci/config.mk.sed, src/Makefile, src/auto/configure, src/config.mk.in, src/configure.ac, src/if_ruby.c, src/vim.h

Patch 8.2.3923

Problem: Vim9: double free if a nested function has a line break in the argument list.
Solution: Set cmdlinep when freeing the previous line.
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.3924

Problem: Vim9: no error if something follows :endef in a nested function.
Solution: Give an error. Move common code to a function.
Files: src/userfunc.c, src/vim9compile.c, src/errors.h, src/testdir/test_vim9_func.vim

Patch 8.2.3925

Problem: Diff mode confused by NUL bytes.
Solution: Handle NUL bytes differently. (Christian Brabandt, closes #9421, closes #9418)
Files: src/diff.c, src/testdir/test_diffmode.vim, src/testdir/dumps/Test_diff_bin_01.dump, src/testdir/dumps/Test_diff_bin_02.dump, src/testdir/dumps/Test_diff_bin_03.dump, src/testdir/dumps/Test_diff_bin_04.dump

Patch 8.2.3926 (after 8.2.3920)

Problem: Build failure without the **'autochdir'** option. (John Marriott)
Solution: Add #ifdefs.
Files: src/evalwindow.c

Patch 8.2.3927

Problem: Vim9: double free when using lambda.
Solution: Don't free both cmdline and line_to_free.
Files: src/userfunc.c

Patch 8.2.3928

Problem: Heredoc test fails.
Solution: Correct order of function arguments.
Files: src/userfunc.c

Patch 8.2.3929

Problem: Using uninitialized variable.
Solution: Set the option flags to zero for a terminal option.
Files: src/option.c

Patch 8.2.3930

Problem: getcmdline() argument has a misleading type.
Solution: Use the correct type, even though the value is not used.
Files: src/ex_getln.c, src/proto/ex_getln.pro, src/ex_docmd.c,
src/normal.c, src/register.c, src/userfunc.c

Patch 8.2.3931

Problem: Coverity reports a memory leak.
Solution: Free memory in case of failure.
Files: src/diff.c

Patch 8.2.3932

Problem: C line comment not formatted properly.
Solution: If a line comment follows after "#if" the next line is not the end
of a paragraph.
Files: src/textformat.c, src/testdir/test_textformat.vim

Patch 8.2.3933

Problem: After ":cd" fails ":cd -" is incorrect.
Solution: Set the previous directory only after successfully changing
directory. (Richard Doty, closes #9419, closes #8983)
Files: src/ex_docmd.c, src/testdir/test_cd.vim

Patch 8.2.3934

Problem: Repeating line comment is undesired for "0" command.
Solution: Do not copy line comment leader for "0". (closes #9426)
Files: src/change.c, src/testdir/test_textformat.vim

Patch 8.2.3935

Problem: **CTRL-U** in Insert mode does not fix the indent.
Solution: Fix the indent when '**cindent**' is set.
Files: src/edit.c, src/testdir/test_textformat.vim

Patch 8.2.3936

Problem: No proper test for maintaining change mark in diff mode.
Solution: Run the test with internal and external diff. (Sean Dewar,
closes #9424)
Files: src/testdir/test_diffmode.vim

Patch 8.2.3937

Problem: Insert mode completion function is too long.
Solution: Refactor into multiple functions. (Yegappan Lakshmanan,
closes #9423)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.3938

Problem: Line comment start is also found in a string.
Solution: Skip line comments in a string.
Files: src/cindent.c, src/proto/cindent.pro, src/search.c,
src/testdir/test_textformat.vim

Patch 8.2.3939

Problem: MS-Windows: fnamemodify('', ':p') does not work.
Solution: Do not consider an empty string a full path. (Yegappan Lakshmanan,
closes #9428, closes #9427)
Files: src/os_mswin.c, src/testdir/test_fnamemodify.vim

Patch 8.2.3940

Problem: Match highlight disappears when doing incsearch for ":s/pat".
Solution: Only use line limit for incsearch highlighting. (closes #9425)
Files: src/match.c, src/testdir/test_match.vim,
src/testdir/dumps/Test_match_with_incsearch_1.dump,
src/testdir/dumps/Test_match_with_incsearch_2.dump

Patch 8.2.3941

Problem: SIGTSTP is not handled.
Solution: Handle SIGTSTP like pressing **CTRL-Z**. (closes #9422)
Files: runtime/doc/autocmd.txt, src/ex_docmd.c, src/os_unix.c,
src/proto/ex_docmd.pro, src/testdir/test_signals.vim

Patch 8.2.3942

Problem: Coverity reports a possible memory leak.
Solution: Free the array if allocation fails.
Files: src/insexpand.c

Patch 8.2.3943

Problem: Compiler warning from gcc for uninitialized variable.
Solution: Initialize variable. (closes #9429)
Files: src/diff.c

Patch 8.2.3944

Problem: Insert mode completion functions are too long.
Solution: Split up into multiple functions. (Yegappan Lakshmanan,
closes #9431)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.3945

Problem: Vim9: partial variable argument types are wrong, leading to a
crash.
Solution: When adjusting the argument count also adjust the argument types.
(closes #9433)
Files: src/vim9type.c, src/userfunc.c, src/testdir/test_vim9_assign.vim

Patch 8.2.3946

Problem: When an internal error makes Vim exit the error is not seen.
Solution: Add the error to the test output.
Files: src/message.c, src/testdir/runtest.vim

Patch 8.2.3947

Problem: Unnecessary check for NULL pointer.
Solution: Remove the check. (closes #9434)
Files: src/ex_docmd.c

Patch 8.2.3948

Problem: Vim9: failure with partial with unknown argument count.
Solution: Do not copy argument types if there aren't any.
Files: src/vim9type.c

Patch 8.2.3949

Problem: Using freed memory with /\%V.
Solution: Get the line again after getvvcol().
Files: src/regexp.c, src/testdir/test_regexp_latin.vim

Patch 8.2.3950

Problem: Going beyond the end of the line with /\%V.
Solution: Check for valid column in getvvcol().
Files: src/charset.c, src/testdir/test_regexp_latin.vim

Patch 8.2.3951

Problem: Vim9: memory leak when text after a nested function.
Solution: Free the function if text is found after "enddef".
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.3952

Problem: First line not redrawn when adding lines to an empty buffer.
Solution: Adjust the argument to appended_lines(). (closes #9439,
closes #9438)
Files: src/ex_cmds.c, src/testdir/test_excmd.vim

Patch 8.2.3953

Problem: Insert completion code is too complicated.
Solution: More refactoring. Move function arguments into a struct.
(Yegappan Lakshmanan, closes #9437)
Files: src/insexpand.c

Patch 8.2.3954

Problem: Vim9: no error for shadowing if script var is declared later.
Solution: Check argument names when compiling a function.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.3955

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/globals.h, src/debugger.c, src/ex_cmds.c,
src/help.c, src/sign.c, src/spellfile.c

Patch 8.2.3956

Problem: Duplicate assignment.
Solution: Remove the second assignment. (closes #9442)
Files: src/evalfunc.c

Patch 8.2.3957

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/globals.h, src/arglist.c, src/bufwrite.c,
src/evalvars.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/help.c, src/scriptfile.c, src/usercmd.c, src/userfunc.c,
src/vim9cmds.c, src/vim9compile.c

Patch 8.2.3958

Problem: Build failure compiling xxd with "-std=c2x".
Solution: define _XOPEN_SOURCE. (Yegappan Lakshmanan, closes #9444)
Files: src/xxd/xxd.c

Patch 8.2.3959

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/autocmd.c, src/bufwrite.c, src/evalvars.c,
src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/fileio.c,
src/getchar.c, src/gui.c, src/locale.c, src/map.c

Patch 8.2.3960

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/alloc.c, src/arglist.c, src/autocmd.c,
src/blob.c, src/blowfish.c, src/buffer.c, src/bufwrite.c

Patch 8.2.3961

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/globals.h, src/arglist.c, src/autocmd.c,
src/blob.c, src/bufwrite.c, src/channel.c, src/clipboard.c,
src/cmdexpand.c, src/debugger.c, src/dict.c, src/eval.c,
src/evalfunc.c, src/evalvars.c, src/evalwindow.c, src/ex_cmds.c,
src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/fileio.c,
src/filepath.c, src/gui_gtk_x11.c, src/gui_haiku.c,
src/gui_photon.c, src/gui_w32.c, src/gui_x11.c, src/highlight.c,
src/indent.c, src/insexpand.c, src/job.c, src/json.c, src/list.c,
src/map.c, src/mark.c, src/match.c, src/mbyte.c, src/menu.c,
src/message.c, src/misc2.c, src/ops.c, src/option.c,
src/optionstr.c, src/popupwin.c, src/quickfix.c, src/screen.c,
src/scriptfile.c, src/search.c, src/sign.c, src/spell.c,
src/spellfile.c, src/strings.c, src/syntax.c, src/terminal.c,
src/testing.c, src/textprop.c, src/time.c, src/userfunc.c,
src/vim9cmds.c, src/vim9execute.c, src/vim9script.c, src/window.c

Patch 8.2.3962 (after 8.2.3961)

Problem: Build fails for missing error message.
Solution: Add changes in missed file.
Files: src/regexp_bt.c

Patch 8.2.3963

Problem: Build failure with tiny and small features. (Tony Mechelynck)
Solution: Adjust #ifdefs.
Files: src/errors.h, src/message.c

Patch 8.2.3964

Problem: Some common lisp and scheme files not recognized.
Solution: Recognize *.asd as lisp and *.sld as scheme. (Alex Vear, closes #9447)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3965

Problem: Vim9: no easy way to check if Vim9 script is supported.
Solution: Add has('vim9script').
Files: runtime/doc/vim9.txt, src/evalfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.3966

Problem: When using feedkeys() abbreviations may be blocked.
Solution: Reset tb_no_abbr_cnt when running out of characters. (closes #9448)
Files: src/getchar.c, src/testdir/test_feedkeys.vim

Patch 8.2.3967

Problem: Error messages are spread out.
Solution: Move more errors to errors.h.
Files: src/errors.h, src/globals.h, src/feature.h, src/arglist.c, src/autocmd.c, src/blob.c, src/bufwrite.c, src/channel.c, src/cmdexpand.c, src/dict.c, src/diff.c, src/eval.c, src/evalfunc.c, src/evalvars.c, src/ex_cmds.c, src/ex_docmd.c, src/fileio.c, src/filepath.c, src/getchar.c, src/gui_gtk_x11.c, src/gui_x11.c, src/hardcopy.c, src/help.c, src/highlight.c, src/if_cscope.c, src/if_lua.c, src/if_mzsch.c, src/if_perl.xs, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/if_xcmdsrv.c, src/indent.c, src/insexpand.c, src/job.c, src/list.c, src/main.c, src/map.c, src/match.c, src/mbyte.c, src/message.c, src/misc1.c, src/option.c, src/optionstr.c, src/os_mswin.c, src/os_unix.c, src/os_win32.c, src/popupwin.c, src/profiler.c, src/quickfix.c, src/scriptfile.c, src/search.c, src/session.c, src/sign.c, src/spell.c, src/spellfile.c, src/spellsuggest.c, src/syntax.c, src/tag.c, src/terminal.c, src/testing.c, src/textprop.c, src/typval.c, src/userfunc.c, src/vim9execute.c, src/vim9expr.c, src/vim9instr.c, src/vim9script.c

Patch 8.2.3968

Problem: Build failure.
Solution: Add missing changes.
Files: src/strings.c, src/vim9compile.c

Patch 8.2.3969

Problem: Value of MAXCOL not available in Vim script.
Solution: Add v:maxcol. (Naohiro Ono, closes #9451)
Files: runtime/doc/builtin.txt, runtime/doc/eval.txt, src/evalvars.c, src/testdir/test_cursor_func.vim, src/testdir/test_normal.vim, src/testdir/test_put.vim, src/vim.h

Patch 8.2.3970

Problem: Error messages are spread out.

Solution: Move more errors to errors.h.
Files: src/errors.h, src/globals.h, src/buffer.c, src/bufwrite.c, src/clientserver.c, src/cmdhist.c, src/dict.c, src/edit.c, src/eval.c, src/evalfunc.c, src/evalvars.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_eval.c, src/ex_getln.c, src/gui_w32.c, src/gui_x11.c, src/if_xcmdsrv.c, src/insexpand.c, src/json.c, src/match.c, src/menu.c, src/option.c, src/optionstr.c, src/os_mswin.c, src/quickfix.c, src/regexp_bt.c, src/regexp_nfa.c, src/scriptfile.c, src/sign.c, src/spellfile.c, src/undo.c, src/userfunc.c, src/vim9cmds.c, src/vim9compile.c, src/vim9execute.c, src/vim9expr.c, src/window.c

Patch 8.2.3971

Problem: Build fails.
Solution: Use the right error message name.
Files: src/typval.c

Patch 8.2.3972

Problem: Error messages are spread out.
Solution: Move the last errors from globals.h to errors.h.
Files: src/errors.h, src/globals.h, src/eval.c, src/evalfunc.c, src/evalvars.c, src/evalwindow.c, src/ex_eval.c, src/list.c, src/match.c, src/menu.c, src/popupmenu.c, src/search.c, src/vim9cmds.c, src/vim9expr.c

Patch 8.2.3973

Problem: Tiny build fails.
Solution: Adjust #ifdefs
Files: src/errors.h

Patch 8.2.3974

Problem: Vim9: LISTAPPEND instruction does not check for a locked list.
Solution: Check whether the list is locked. (closes #9452)
Files: src/vim9execute.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3975

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/clientserver.c, src/fileio.c, src/gui.c, src/gui_beval.c, src/gui_w32.c, src/gui_x11.c, src/if_cscope.c, src/if_xcmdsrv.c, src/os_mswin.c, src/sign.c, src/viminfo.c, src/window.c

Patch 8.2.3976

Problem: FEARG_LAST is never used. (Dominique Pellé)
Solution: Remove FEARG_LAST and the related code.
Files: src/evalfunc.c

Patch 8.2.3977

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/change.c, src/clientserver.c, src/eval.c, src/gui_xim.c, src/if_cscope.c, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/if_ruby.c, src/if_tcl.c, src/main.c,

src/mark.c, src/match.c, src/memfile.c, src/memline.c,
src/terminal.c, src/textprop.c, src/userfunc.c

Patch 8.2.3978

Problem: Build error when using dynamically loaded Python 3.
Solution: Adjust #ifdef.
Files: src/errors.h

Patch 8.2.3979

Problem: Vim9: the feature is not mentioned in the right places.
Solution: Add +vim9script to the help and :version output.
Files: runtime/doc/builtin.txt, runtime/doc/various.txt, src/version.c

Patch 8.2.3980

Problem: If '**operatorfunc**' invokes an operator the remembered Visual mode may be changed. (Naohiro Ono)
Solution: Save and restore the information for redoing the Visual area. (closes #9455)
Files: src/ops.c, src/testdir/test_normal.vim

Patch 8.2.3981

Problem: Vim9: debugging a for loop doesn't stop before it starts.
Solution: Keep the DEBUG instruction before the expression is evaluated. (closes #9456)
Files: src/vim9cmds.c, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3982

Problem: Some lines of code not covered by tests.
Solution: Add a few more test cases. (Dominique Pellé, closes #9453)
Files: src/testdir/test_filter_map.vim, src/testdir/test_highlight.vim, src/testdir/test_regexp_latin.vim, src/testdir/test_search.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3983

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/ex_docmd.c, src/fileio.c, src/filepath.c, src/findfile.c, src/hardcopy.c, src/memfile.c, src/memline.c, src/menu.c, src/normal.c, src/regexp_bt.c

Patch 8.2.3984 (after 8.2.3981)

Problem: Debugger test fails.
Solution: Adjust the test for modified debugging of a for loop.
Files: src/testdir/test_debugger.vim

Patch 8.2.3985

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/findfile.c, src/fold.c, src/hardcopy.c, src/highlight.c, src/map.c, src/message.c, src/normal.c, src/option.c, src/os_amiga.c, src/os_unix.c, src/os_win32.c, src/quickfix.c, src/regexp.c, src/register.c, src/search.c, src/syntax.c, src/tag.c, src/term.c, src/typval.c, src/undo.c, src/window.c

Patch 8.2.3986

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/evalvars.c, src/ex_cmds.c, src/ex_docmd.c,
src/fileio.c, src/fold.c, src/gui_x11.c, src/hardcopy.c,
src/help.c, src/highlight.c, src/if_cscope.c, src/json.c,
src/map.c, src/netbeans.c, src/popupwin.c, src/usercmd.c,
src/userfunc.c

Patch 8.2.3987

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/digraph.c, src/ex_eval.c, src/gui.c,
src/hardcopy.c, src/if_cscope.c, src/if_tcl.c, src/if_xcmdsrv.c,
src/mbyte.c, src/misc2.c, src/netbeans.c, src/option.c,
src/optionstr.c, src/quickfix.c, src/regexp.c, src/tag.c,
src/term.c, src/viminfo.c

Patch 8.2.3988 (after 8.2.3987)

Problem: Tiny build fails.
Solution: Fix misplaced #ifdef.
Files: src/errors.h

Patch 8.2.3989

Problem: Some insert completion code is not tested.
Solution: Add a few tests. Refactor thesaurus completion. (Yegappan
Lakshmanan, closes #9460)
Files: src/insexpand.c, src/testdir/test_edit.vim,
src/testdir/test_ins_complete.vim

Patch 8.2.3990

Problem: Testing wrong operator.
Solution: Test "g@" instead of "r_". (Naohiro Ono, closes #9463)
Files: src/testdir/test_normal.vim

Patch 8.2.3991

Problem: Vim9: error when extending dict<any> with another type that it was
initialized with.
Solution: Also set the type for dict<any> if the initializer has a more
specific type. (closes #9461)
Files: src/vim9compile.c, src/vim9type.c, src/vim9.h, src/eval.c,
src/list.c, src/vim9script.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_func.vim

Patch 8.2.3992

Problem: Wrong local-additions in the help with language mix.
Solution: Adjust how the local additions list is generated. (Hirohito
Higashi, closes #9464)
Files: src/help.c, src/testdir/test_help.vim

Patch 8.2.3993

Problem: When recording a change in Select mode the first typed character
appears twice.

Solution: When putting the character back into typeahead remove it from recorded characters. (closes #9462)
Files: src/getchar.c, src/proto/getchar.pro, src/normal.c, src/testdir/test_registers.vim

Patch 8.2.3994

Problem: Vim9: extend() complains about the type even when it was not declared.
Solution: Only check the list or dict type when it was declared.
Files: src/list.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.3995

Problem: Not all sshconfig files are detected as such.
Solution: Adjust the patterns used for sshconfig detection. (David Auer, closes #9322)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.3996

Problem: Vim9: type checking for list and dict lacks information about declared type.
Solution: Add dv_decl_type and lv_decl_type. Refactor the type stack to store two types in each entry.
Files: src/structs.h, src/dict.c, src/list.c, src/vim9type.c, src/proto/vim9type.pro, src/vim9instr.c, src/proto/vim9instr.pro, src/vim9compile.c, src/evalfunc.c, src/proto/evalfunc.pro, src/evalbuffer.c, src/proto/evalbuffer.pro, src/vim9expr.c, src/vim9cmds.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.3997

Problem: Vim9: not enough testing for extend() and map().
Solution: Add more test cases. Fix uncovered problems. Remove unused type fields.
Files: src/structs.h, src/dict.c, src/list.c, src/vim9compile.c, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.3998

Problem: Asan error for adding zero to NULL.
Solution: Do not compute pointer if there are no entries.
Files: src/vim9type.c

Patch 8.2.3999

Problem: Redundant check for NUL byte.
Solution: Remove the check for a NUL byte. (closes #9471)
Files: src/ex_docmd.c

Patch 8.2.4000

Problem: Coverity warns for checking for NULL pointer after using it.
Solution: Remove check for NULL.
Files: src/help.c

Patch 8.2.4001

Problem: Insert complete code uses global variables.

Solution: Make variables local to the file and use accessor functions.
(Yegappan Lakshmanan, closes #9470)
Files: src/edit.c, src/getchar.c, src/globals.h, src/insexpand.c,
src/proto/insexpand.pro, src/search.c

Patch 8.2.4002

Problem: First char typed in Select mode can be wrong.
Solution: Escape special bytes in the input buffer. (closes #9469)
Files: src/getchar.c, src/testdir/test_utf8.vim

Patch 8.2.4003

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/channel.c, src/ex_docmd.c, src/ex_eval.c,
src/gui_at_fs.c, src/hardcopy.c, src/if_cscope.c, src/menu.c,
src/netbeans.c, src/optionstr.c, src/os_mswin.c, src/sign.c,
src/typval.c

Patch 8.2.4004

Problem: Old compiler complains about struct init with variable.
Solution: Set the struct member later. (John Marriott)
Files: src/evalfunc.c

Patch 8.2.4005

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/dict.c, src/eval.c, src/evalfunc.c,
src/evalvars.c, src/ex_cmds2.c, src/ex_docmd.c, src/ex_eval.c,
src/filepath.c, src/gui.c, src/gui_w32.c, src/hardcopy.c,
src/help.c, src/highlight.c, src/if_python.c, src/list.c,
src/misc1.c, src/normal.c, src/quickfix.c, src/regexp.c,
src/regexp_bt.c, src/regexp_nfa.c, src/typval.c, src/userfunc.c

Patch 8.2.4006

Problem: Vim9: crash when declaring variable on the command line.
Solution: Use a temporary type list. (closes #9474)
Files: src/eval.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4007

Problem: Session does not restore help buffer properly when "options" is
missing from **'sessionoptions'**.
Solution: Use a ":help" command to create the help window. (closes #9475,
closes #9458, closes #9472)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4008

Problem: Error messages are spread out.
Solution: Move more error messages to errors.h.
Files: src/errors.h, src/diff.c, src/digraph.c, src/evalfunc.c,
src/evalvars.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c,
src/insexpand.c, src/match.c, src/memline.c, src/menu.c,
src/ops.c, src/profiler.c, src/quickfix.c, src/regexp.c,
src/regexp_bt.c, src/regexp_nfa.c, src/register.c, src/spell.c,
src/spell.h, src/spellfile.c, src/strings.c, src/syntax.c,

src/typval.c, src/undo.c, src/userfunc.c

Patch 8.2.4009

Problem: Reading one byte beyond the end of the line.

Solution: Check for NUL byte first.

Files: src/vim9compile.c, src/ex_docmd.c, src/testdir/test_vim9_func.vim

Patch 8.2.4010

Problem: Error messages are spread out.

Solution: Move more error messages to errors.h.

Files: src/errors.h, src/crypt.c, src/diff.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/findfile.c, src/float.c, src/gui.c, src/highlight.c, src/if_mzsch.c, src/if_py_both.h, src/if_python.c, src/if_python3.c, src/insexpand.c, src/match.c, src/memline.c, src/option.c, src/popupwin.c, src/regexp.c, src/regexp_nfa.c, src/spellfile.c, src/strings.c, src/syntax.c, src/textprop.c, src/typval.c, src/undo.c, src/usercmd.c, src/userfunc.c, src/window.c

Patch 8.2.4011

Problem: Test fails because of changed error number.

Solution: Restore old duplicate error message.

Files: src/errors.h, src/match.c

Patch 8.2.4012

Problem: Error messages are spread out.

Solution: Move the last error messages to errors.h.

Files: src/errors.h, src/channel.c, src/clientserver.c, src/diff.c, src/evalfunc.c, src/evalvars.c, src/ex_cmds2.c, src/ex_docmd.c, src/gui_w32.c, src/help.c, src/if_mzsch.c, src/if_py_both.h, src/if_python.c, src/job.c, src/json.c, src/list.c, src/option.c, src/optionstr.c, src/quickfix.c, src/regexp.c, src/regexp_nfa.c, src/register.c, src/scriptfile.c, src/sign.c, src/syntax.c, src/tag.c, src/terminal.c, src/textprop.c, src/typval.c, src/undo.c, src/userfunc.c, src/vim9compile.c, src/viminfo.c

Patch 8.2.4013

Problem: Build failure without the spell feature.

Solution: Adjust #ifdefs.

Files: src/errors.h

Patch 8.2.4014

Problem: Git and gitcommit file types not properly recognized.

Solution: Adjust filetype detection. (Tim Pope, closes #9477)

Files: runtime/filetype.vim, runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.4015

Problem: Build failure with tiny features. (Tony Mechelynck)

Solution: Adjust #ifdefs.

Files: src/errors.h

Patch 8.2.4016

Problem: Vim9: incorrect error for argument that is shadowing var.

Solution: Ignore variable that is not in block where the function was defined.
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.4017

Problem: Gcc warns for misleading indent in Athena menu code.
Solution: Add curlyes around the two statements. (Dominique Pellé, closes #9480)
Files: src/gui_athena.c

Patch 8.2.4018

Problem: ml_get error when win_execute redraws with Visual selection.
Solution: Disable Visual area temporarily. (closes #9479)
Files: src/evalwindow.c, src/proto/evalwindow.pro, src/structs.h, src/evalbuffer.c, src/proto/evalbuffer.pro, src/evalvars.c, src/if_py_both.h, src/evalfunc.c, src/testdir/test_execute_func.vim

Patch 8.2.4019

Problem: Vim9: import mechanism is too complicated.
Solution: Do not use the Javascript mechanism but a much simpler one.
Files: runtime/doc/vim9.txt, src/vim9script.c, src/proto/vim9script.pro, src/errors.h, src/structs.h, src/eval.c, src/proto/eval.pro, src/evalvars.c, src/proto/evalvars.pro, src/userfunc.c, src/vim9expr.c, src/vim9compile.c, src/vim9execute.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_script.vim

Patch 8.2.4020 (after 8.2.4019)

Problem: Debugger test fails.
Solution: Fix import statement.
Files: src/testdir/test_debugger.vim

Patch 8.2.4021 (after 8.2.4019)

Problem: Missing part of the :import changes.
Solution: Add changes in vim9cmds.c.
Files: src/vim9cmds.c

Patch 8.2.4022

Problem: Two error messages in the wrong file.
Solution: Use the error message from errors.h.
Files: src/popupwin.c, src/usercmd.c

Patch 8.2.4023

Problem: Using uninitialized variable.
Solution: Initialize "ufunc" also when an item is not exported.
Files: src/vim9script.c

Patch 8.2.4024

Problem: Confusing error message if imported name is used directly.
Solution: Give a better error message.
Files: src/eval.c, src/proto/eval.pro, src/evalvars.c, src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.4025

Problem: Error for import not ending in .vim does not work for .vimrc.
Solution: Check that .vim is the end. (closes #9484)
Files: src/vim9script.c, src/errors.h, src/testdir/test_vim9_script.vim

Patch 8.2.4026

Problem: ml_get error with specific win_execute() command. (Sean Dewar)
Solution: Check cursor and Visual area are OK.
Files: src/evalwindow.c, src/testdir/test_execute_func.vim

Patch 8.2.4027

Problem: Import test fails on MS-Windows.
Solution: Use a different directory name.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.4028

Problem: ml_get error with :doautoall and Visual area. (Sean Dewar)
Solution: Disable Visual mode while executing autocommands.
Files: src/structs.h, src/autocmd.c, src/testdir/test_autocmd.vim

Patch 8.2.4029

Problem: Debugging NFA regexp my crash, cached indent may be wrong.
Solution: Fix some debug warnings in the NFA regexp code. Make sure log_fd is set when used. Fix breakindent and indent caching. (Christian Brabandt, closes #9482)
Files: src/indent.c, src/optionstr.c, src/regexp_nfa.c

Patch 8.2.4030

Problem: A script local funcref is not found from a mapping.
Solution: When looking for a function, also find a script-local funcref. (closes #9485)
Files: src/evalvars.c, src/proto/evalvars.pro, src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.4031

Problem: Crash in xterm with only two lines. (Dominique Pellé)
Solution: Only perform xterm compatibility test if possible. (closes #9488)
Files: src/term.c, src/testdir/test_startup.vim

Patch 8.2.4032

Problem: ATTRIBUTE_NORETURN is not needed.
Solution: Use NORETURN(). (Ozaki Kiichi, closes #9487)
Files: src/if_ruby.c, src/vim.h

Patch 8.2.4033

Problem: Running filetype tests leaves directory behind.
Solution: Delete the top directory. (closes #9483)
Files: src/testdir/test_filetype.vim

Patch 8.2.4034

Problem: Coverity warns for possibly using a NULL pointer.
Solution: Check v_partial is not NULL.
Files: src/vim9type.c

Patch 8.2.4035

Problem: Timer triggered at the debug prompt may cause trouble.
Solution: Do not trigger any timer at the debug prompt. (closes #9481)
Files: src/time.c

Patch 8.2.4036

Problem: Vim9: script test file is getting too long.
Solution: Split the import/export functionality to a separate file.
Files: src/testdir/test_vim9_script.vim, src/testdir/test_vim9_import.vim,
src/testdir/Make_all.mak

Patch 8.2.4037

Problem: Insert mode completion is insufficiently tested.
Solution: Add more tests. Fix uncovered memory leak. (Yegappan Lakshmanan,
closes #9489)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.4038

Problem: Various code not used when features are disabled.
Solution: Add #ifdefs. (Dominique Pellé, closes #9491)
Files: src/alloc.c, src/buffer.c, src/charset.c, src/clipboard.c,
src/cmdhist.c, src/crypt.c, src/edit.c, src/eval.c,
src/evalbuffer.c, src/evalfunc.c, src/ex_docmd.c, src/globals.h,
src/gui_xim.c, src/hashtab.c, src/highlight.c, src/insexpand.c,
src/main.c, src/mark.c, src/message.c, src/misc1.c, src/misc2.c,
src/ops.c, src/option.c, src/option.h, src/optionstr.c,
src/register.c, src/scriptfile.c, src/tag.c, src/term.c,
src/typval.c, src/usercmd.c, src/userfunc.c, src/vim9script.c,
src/vim9type.c

Patch 8.2.4039

Problem: The xdiff library is linked in even when not used.
Solution: Use configure to decide whether xdiff object files are included.
Files: src/Makefile, src/config.mk.in, src/configure.ac,
src/auto/configure, src/feature.h

Patch 8.2.4040

Problem: Keeping track of allocated lines in user functions is too complicated.
Solution: Instead of freeing individual lines keep them all until the end.
Files: src/alloc.c, src/proto/alloc.pro, src/vim9compile.c,
src/userfunc.c, src/proto/userfunc.pro, src/message.c,
src/usercmd.c, src/viminfo.c, src/testdir/test_vim9_func.vim

Patch 8.2.4041

Problem: Using uninitialized pointer.
Solution: Store "ht" when variable is in another script.
Files: src/evalvars.c

Patch 8.2.4042

Problem: Vim9: build error.
Solution: Use grow array instead of character pointer.
Files: src/vim9execute.c

Patch 8.2.4043

Problem: Using int for second argument of ga_init2().
Solution: Remove unnecessary type cast (int) when using sizeof().
Files: src/arglist.c, src/channel.c, src/cmdexpand.c, src/dict.c,
src/digraph.c, src/eval.c, src/evalfunc.c, src/evalvars.c,
src/evalwindow.c, src/ex_docmd.c, src/fileio.c, src/filepath.c,
src/findfile.c, src/fold.c, src/hardcopy.c, src/help.c,
src/job.c, src/list.c, src/menu.c, src/os_win32.c, src/register.c,
src/scriptfile.c, src/spellfile.c, src/spellsuggest.c,
src/strings.c, src/syntax.c, src/tag.c, src/terminal.c,
src/undo.c, src/usercmd.c, src/userfunc.c, src/vim9execute.c,
src/viminfo.c, src/window.c, src/if_py_both.h

Patch 8.2.4044

Problem: Vim9: no error when importing the same script twice.
Solution: Give an error, unless it is a reload.
Files: src/vim9script.c, src/errors.h, src/testdir/test_vim9_import.vim

Patch 8.2.4045

Problem: Some global functions are only used in one file.
Solution: Make the functions static. (Yegappan Lakshmanan, closes #9492)
Files: src/ex_getln.c, src/highlight.c, src/proto/ex_getln.pro,
src/proto/highlight.pro, src/proto/vim9compile.pro,
src/proto/vim9instr.pro, src/proto/window.pro, src/vim9compile.c,
src/vim9instr.c, src/window.c

Patch 8.2.4046

Problem: Some error messages not in the right place.
Solution: Adjust the errors file. Fix typo.
Files: src/errors.h, src/regexp_bt.c, src/typval.c,

Patch 8.2.4047

Problem: Depending on the build features error messages are unused.
Solution: Add #ifdefs. (Dominique Pellé, closes #9493)
Files: src/errors.h

Patch 8.2.4048

Problem: gcc complains about use of "%p" in printf.
Solution: Add (void *) typecast. (Dominique Pellé, closes #9494)
Files: src/if_py_both.h

Patch 8.2.4049

Problem: Vim9: reading before the start of the line with "\$" by itself.
Solution: Do not subtract one when reporting the error.
Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4050

Problem: Vim9: need to prefix every item in an autoload script.
Solution: First step in supporting "vim9script autoload" and "import
autoload".
Files: runtime/doc/repeat.txt, runtime/doc/vim9.txt, src/structs.h,
src/errors.h, src/vim9script.c, src/scriptfile.c,
src/proto/scriptfile.pro, src/userfunc.c, src/eval.c,
src/evalvars.c, src/vim9compile.c, src/proto/vim9compile.pro,
src/vim9expr.c, src/testdir/test_vim9_script.vim

Patch 8.2.4051

Problem: Compiler complains about possibly uninitialized variable.
Solution: Add code to avoid a compiler warning. (John Marriott)
Files: src/scriptfile.c

Patch 8.2.4052

Problem: Not easy to resize a window from a plugin.
Solution: Add win_move_separator() and win_move_statusline() functions.
(Daniel Steinberg, closes #9486)
Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/evalwindow.c, src/proto/evalwindow.pro,
src/testdir/test_window_cmd.vim

Patch 8.2.4053

Problem: Vim9: autoload mechanism doesn't fully work yet.
Solution: Define functions and variables with their autoload name, add the
prefix when calling a function, find the variable in the table of
script variables.
Files: src/structs.h, src/scriptfile.c, src/proto/scriptfile.pro,
src/vim9script.c, src/proto/vim9script.pro, src/userfunc.c,
src/evalvars.c, src/testdir/test_vim9_script.vim

Patch 8.2.4054 (after 8.2.4053)

Problem: Vim9 script test fails.
Solution: Add missing change.
Files: src/vim9compile.c

Patch 8.2.4055

Problem: Vim9: line break in expression causes v:errmsg to be filled.
(Yegappan Lakshmanan)
Solution: Do not give an error when skipping over an expression.
Files: src/userfunc.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4056

Problem: Vim9: memory leak when exporting function in autoload script.
Solution: Free the name if replacing it.
Files: src/scriptfile.c

Patch 8.2.4057

Problem: Vim9: not fully implementing the autoload mechanism.
Solution: Allow for exporting a legacy function. Improve test coverage.
Files: src/vim9script.c, src/testdir/test_vim9_import.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.4058

Problem: Vim9: import test failure in wrong line.
Solution: Adjust line number.
Files: src/testdir/test_vim9_import.vim

Patch 8.2.4059

Problem: Vim9: an expression of a map cannot access script-local items.
(Maxim Kim)
Solution: Use the script ID of where the map was defined.

Files: src/getchar.c, src/map.c, src/proto/map.pro,
src/testdir/test_vim9_import.vim

Patch 8.2.4060

Problem: win_execute() is slow on systems where getcwd() or chdir() is slow. (Rick Howe)

Solution: Avoid using getcwd() and chdir() if no local directory is used and 'acd' is not set. (closes #9504)

Files: src/evalwindow.c

Patch 8.2.4061

Problem: Codecov bash script is deprecated.

Solution: Use the codecov action. (Ozaki Kiichi, closes #9505)

Files: .github/workflows/ci.yml

Patch 8.2.4062

Problem: Match highlighting of tab too short.

Solution: Do not stop match highlighting if on a Tab. (Christian Brabandt, closes #9507, closes #9500)

Files: src/drawline.c, src/testdir/test_match.vim,
src/testdir/dumps/Test_match_tab_linebreak.dump

Patch 8.2.4063

Problem: Vim9: exported function in autoload script not found. (Yegappan Lakshmanan)

Solution: Use the autoload prefix to search for the function.

Files: src/userfunc.c, src/testdir/test_vim9_import.vim

Patch 8.2.4064

Problem: Foam files are not detected.

Solution: Detect the foam filetype by the path and file contents. (Mohammed Elwardi Fadeli, closes #9501)

Files: runtime/filetype.vim, runtime/autoload/dist/ft.vim,
src/testdir/test_filetype.vim

Patch 8.2.4065

Problem: Computation overflow with large count for :yank.

Solution: Avoid an overflow.

Files: src/ex_docmd.c, src/testdir/test_excmd.vim

Patch 8.2.4066

Problem: Vim9: imported autoload script loaded again.

Solution: Do not create a new imported_T every time.

Files: src/vim9script.c, src/vim9compile.c,
src/testdir/test_vim9_import.vim

Patch 8.2.4067

Problem: Vim9: cannot call imported function with :call. (Drew Vogel)

Solution: Translate the function name. (closes #9510)

Files: src/userfunc.c, src/testdir/test_vim9_import.vim

Patch 8.2.4068 (after 8.2.4066)

Problem: Vim9: import test fails.

Solution: Add missing change.

Files: src/scriptfile.c

Patch 8.2.4069

Problem: Vim9: import test fails on MS-Windows.

Solution: Ignore case. Adjust test to avoid name that only differs in case.

Files: src/eval.c, src/scriptfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4070

Problem: Using uninitialized memory when reading empty file.

Solution: Check for empty file before checking for NL. (Dominique Pellé, closes #9511)

Files: src/filepath.c, src/testdir/test_eval_stuff.vim

Patch 8.2.4071

Problem: Vim9: no detection of return in try/endtry. (Dominique Pellé)

Solution: Check if any of the blocks inside try/endtry did not end in return.

Files: src/vim9.h, src/vim9compile.c, src/vim9cmds.c, src/testdir/test_vim9_script.vim

Patch 8.2.4072

Problem: Vim9: compiling function fails when autoload script is not loaded yet.

Solution: Depend on runtime loading.

Files: src/vim9expr.c, src/vim9script.c, src/vim9instr.c, src/vim9execute.c, src/testdir/test_vim9_import.vim

Patch 8.2.4073

Problem: Coverity warns for using NULL pointer.

Solution: Bail out when running out of memory. Check for running over end of a string.

Files: src/userfunc.c,

Patch 8.2.4074

Problem: Going over the end of NameBuff.

Solution: Check length when appending a space.

Files: src/drawscreen.c, src/testdir/test_edit.vim

Patch 8.2.4075 (after 8.2.4073)

Problem: Test failures.

Solution: Change check for NULL pointer.

Files: src/userfunc.c

Patch 8.2.4076

Problem: Memory leak in autoload import.

Solution: Do not overwrite the autoload prefix.

Files: src/vim9script.c

Patch 8.2.4077

Problem: Not all Libsensors files are recognized.

Solution: Add "sensors.d/*" pattern. (Doug Kearns)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4078

Problem: Terminal test for current directory not used on FreeBSD.
Solution: Make it work on FreeBSD. (Ozaki Kiichi, closes #9516) Add TermWait() inside Run_shell_in_terminal() as a generic solution.
Files: src/testdir/test_terminal3.vim, src/testdir/term_util.vim, src/testdir/test_terminal.vim, src/testdir/test_terminal2.vim, src/testdir/test_mapping.vim

Patch 8.2.4079

Problem: MS-Windows: "gvim --version" didn't work when build with VIMDLL.
Solution: Adjust #ifdef. (Ken Takata, closes #9517)
Files: src/main.c

Patch 8.2.4080

Problem: Not sufficient test coverage for xxd.
Solution: Add a few more test cases. (Erki Auerswald, closes #9515)
Files: src/testdir/test_xxd.vim

Patch 8.2.4081

Problem: CodeQL reports problem in if_cscope causing it to fail.
Solution: Use execvp() instead of execl(). Merge the header file into the source file. (Ozaki Kiichi, closes #9519)
Files: Filelist, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/if_cscope.c, src/if_cscope.h, src/testdir/test_cscope.vim

Patch 8.2.4082

Problem: Check for autoload file name and prefix fails. (Christian J. Robinson)
Solution: Only lower case the prefix on systems where the file name is not case sensitive.
Files: src/scriptfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4083

Problem: Vim9: no test for "vim9script autoload" and using script variable in the same script.
Solution: Add a simple test. Fix uncovered problem.
Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4084

Problem: Memory leak when looking for autoload prefixed variable.
Solution: Free the concatenated string.
Files: src/evalvars.c

Patch 8.2.4085

Problem: Vim9: no test for using import in legacy script.
Solution: Add a test.
Files: src/testdir/test_vim9_import.vim

Patch 8.2.4086

Problem: "cctx" argument of find_func_even_dead() is unused.
Solution: Remove the argument.
Files: src/userfunc.c, src/proto/userfunc.pro, src/vim9compile.c, src/vim9instr.c, src/evalfunc.c, src/evalvars.c, src/testing.c, src/vim9execute.c, src/vim9expr.c, src/vim9script.c,

src/vim9type.c

Patch 8.2.4087

Problem: Cannot test items from an autoload script easily.
Solution: Add the "autoload" value for test_override().
Files: runtime/doc/testing.txt, src/testing.c, src/globals.h,
src/vim9script.c, src/testdir/test_vim9_import.vim

Patch 8.2.4088

Problem: Xxd cannot output everything in one line.
Solution: Make zero columns mean infinite columns. (Erik Auerswald,
closes #9524)
Files: runtime/doc/xxd.1, runtime/doc/xxd.man, src/testdir/test_xxd.vim,
src/xxd/xxd.c

Patch 8.2.4089 (after 8.2.4078)

Problem: Terminal test for current directory fails on FreeBSD.
Solution: Skip the test.
Files: src/testdir/test_terminal3.vim

Patch 8.2.4090

Problem: After restoring a session buffer order can be quite different.
Solution: Create buffers first. (Evgeni Chasnovski, closes #9520)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4091

Problem: Virtcol is recomputed for statusline unnecessarily.
Solution: Just use "w_virtcol". (closes #9523)
Files: src/buffer.c, src/testdir/test_statusline.vim

Patch 8.2.4092

Problem: macOS CI: unnecessarily doing "Install packages".
Solution: Only do "Install packages" for huge build. (Ozaki Kiichi,
closes #9521)
Files: .github/workflows/ci.yml

Patch 8.2.4093

Problem: Cached breakindent values not initialized properly.
Solution: Initialize and cache formatlistpat. (Christian Brabandt,
closes #9526, closes #9512)
Files: runtime/doc/options.txt, src/indent.c, src/option.c,
src/proto/option.pro, src/testdir/test_breakindent.vim

Patch 8.2.4094

Problem: **'virtualedit'** is window-local but using buffer-local enum.
Solution: Use window-local enum. (closes #9529)
Files: src/option.h, src/optiondefs.h

Patch 8.2.4095

Problem: Sed script not recognized by the first line.
Solution: Recognize a sed script starting with "#n". (Doug Kearns)
Files: runtime/scripts.vim, src/testdir/test_filetype.vim

Patch 8.2.4096

Problem: Linux CI: unnecessarily installing packages
Solution: Only install packages for huge build. (Ozaki Kiichi, closes #9530)
Files: .github/workflows/ci.yml

Patch 8.2.4097

Problem: Wrong number in error message on 32 bit system. (John Paul Adrian Glaubitz)
Solution: Add type cast. (closes #9527)
Files: src/vim9compile.c

Patch 8.2.4098

Problem: Typing "interrupt" at debug prompt may keep exception around, causing function calls to fail.
Solution: Discard any exception at the toplevel. (closes #9532)
Files: src/main.c

Patch 8.2.4099

Problem: Vim9: cannot use Vim9 syntax in mapping.
Solution: Add `<ScriptCmd>` to use the script context for a command.
Files: runtime/doc/map.txt, src/normal.c, src/getchar.c, src/proto/getchar.pro, src/ex_getln.c, src/edit.c, src/terminal.c, src/keymap.h, src/insexpand.c, src/misc2.c, src/ops.c, src/testdir/test_vim9_import.vim

Patch 8.2.4100

Problem: Early return when getting the '`formatlistpat`' value.
Solution: Remove the first line. (Christian Brabandt)
Files: src/option.c, src/testdir/test_breakindent.vim

Patch 8.2.4101

Problem: Warning for unused argument in tiny version.
Solution: Add "UNUSED".
Files: src/getchar.c

Patch 8.2.4102

Problem: Vim9: import cannot be used after method.
Solution: Recognize an imported function name. (closes #9496)
Files: src/eval.c, src/testdir/test_vim9_import.vim

Patch 8.2.4103

Problem: Vim9: variable declared in for loop not initialized.
Solution: Always initialize the variable. (closes #9535)
Files: src/vim9instr.c, src/proto/vim9instr.pro, src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4104

Problem: Vim9: lower casing the autoload prefix causes problems.
Solution: Always store the prefix with case preserved.
Files: src/scriptfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4105

Problem: Translation related comment in the wrong place.
Solution: Move it back with the text. (Ken Takata, closes #9537)

Files: src/errors.h, src/ex_docmd.c

Patch 8.2.4106

Problem: Going over the end of the w_lines array.

Solution: Break out of the loop when "idx" is too big. (issue #9540)

Files: src/drawscreen.c

Patch 8.2.4107

Problem: Script context not restored after using <ScriptCmd>.

Solution: Also restore context when not in a script. (closes #9536)

Add the 'c' flag to feedkeys() to be able to test this.

Files: runtime/doc/builtin.txt, src/getchar.c, src/evalfunc.c,
src/testdir/test_mapping.vim

Patch 8.2.4108

Problem: Going over the end of the w_lines array.

Solution: Check not going over the end and limit to Rows. (issue #9540)

Files: src/drawscreen.c

Patch 8.2.4109

Problem: MS-Windows: high dpi support is outdated.

Solution: Improve High DPI support by using PerMonitorV2. (Ken Takata
closes #9525, closes #3102)

Files: src/gui.c, src/gui.h, src/gui_w32.c, src/vim.manifest

Patch 8.2.4110

Problem: Coverity warns for using NULL pointer.

Solution: Check "evalarg" is not NULL. Skip errors when "verbose" is false.

Files: src/eval.c

Patch 8.2.4111

Problem: Potential problem when map is deleted while executing.

Solution: Reset last used map pointer when deleting a mapping.

Files: src/map.c

Patch 8.2.4112

Problem: Function not deleted at end of test.

Solution: Delete the function.

Files: src/testdir/test_diffmode.vim

Patch 8.2.4113

Problem: Typo on DOCMD_RANGEOK results in not recognizing command.

Solution: Correct the typo. (closes #9539)

Files: src/vim.h, src/testdir/test_mapping.vim

Patch 8.2.4114

Problem: Vim9: type checking for a funcref does not work for when it is
used in a method.

Solution: Pass the base to where the type is checked.

Files: src/vim9type.c, src/proto/vim9type.pro, src/userfunc.c,
src/testdir/test_vim9_expr.vim

Patch 8.2.4115

Problem: Cannot use a method with a complex expression.

Solution: Evaluate the expression after "->" and use the result.
Files: src/eval.c, src/errors.h, src/testdir/test_vim9_expr.vim

Patch 8.2.4116

Problem: Vim9: cannot use a method with a complex expression in a :def function.

Solution: Implement compiling the expression.

Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4117

Problem: Vim9: wrong white space error after using imported item.

Solution: Don't skip over white space. (closes #9544)

Files: src/eval.c, src/testdir/test_vim9_import.vim

Patch 8.2.4118

Problem: Using UNUSED for argument that is used.

Solution: Remove UNUSED.

Files: src/usercmd.c

Patch 8.2.4119

Problem: Build failure when disabling the channel feature.

Solution: Adjust #ifdef. (Dominique Pellé, closes #9545)

Files: src/misc2.c

Patch 8.2.4120

Problem: Block insert goes over the end of the line.

Solution: Handle invalid byte better. Fix inserting the wrong text.

Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.2.4121

Problem: Visual test fails on MS-Windows.

Solution: Set '**isprint**' so that the character used is not printable.

Files: src/testdir/test_visual.vim

Patch 8.2.4122

Problem: ":command Cmd" does not show custom completion argument.

Solution: Show the completion argument when using ":verbose".

Files: src/usercmd.c, src/testdir/test_usercommands.vim

Patch 8.2.4123

Problem: Complete function cannot be import.Name.

Solution: Dereference the function name if needed. Also: do not see "import.Name" as a builtin function. (closes #9541)

Files: src/userfunc.c, src/eval.c, src/testdir/test_vim9_import.vim

Patch 8.2.4124

Problem: Vim9: method in compiled function may not see script item.

Solution: Make sure not to skip to the next line. (closes #9496)

Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4125

Problem: Completion tests fail.

Solution: Disable error messages while dereferencing the function name.

Files: src/eval.c

Patch 8.2.4126

Problem: Crash on exit when built with dynamic Tcl and EXITFREE is defined.
(Dominique Pellé)
Solution: Only call Tcl_Finalize() when initialized. (closes #9546)
Files: src/if_tcl.c

Patch 8.2.4127

Problem: Build failure without the +eval feature.
Solution: Add #ifdef.
Files: src/usercmd.c

Patch 8.2.4128

Problem: Crash when method cannot be found. (Christian J. Robinson)
Solution: Don't mix up pointer names.
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4129

Problem: Building with +sound but without +eval fails. (Dominique Pellé)
Solution: Disable canberra in tiny and small build. (closes #9548)
Files: src/configure.ac, src/auto/configure

Patch 8.2.4130

Problem: MS-Windows: MSVC build may have libraries duplicated.
Solution: Improve the MSVC Makefile. (Ken Takata, closes #9547)
Files: src/Make_mvc.mak

Patch 8.2.4131

Problem: Vim9: calling function in autoload import does not work in a :def
function.
Solution: When a variable is not found and a PCALL follows use a funcref.
(closes #9550)
Files: src/vim9execute.c, src/testdir/test_vim9_import.vim

Patch 8.2.4132

Problem: Vim9: wrong error message when autoload script can't be found.
Solution: Correct check for using autoload with wrong name.
Files: src/vim9script.c, src/testdir/test_vim9_import.vim

Patch 8.2.4133

Problem: output of ":scriptnames" goes into the message history, while this
does not happen for other commands, such as ":ls".
Solution: Use msg_outtrans() instead of msg(). (closes #9551)
Files: src/scriptfile.c, src/testdir/test_scriptnames.vim

Patch 8.2.4134

Problem: MS-Windows: test for import with absolute path fails.
Solution: Handle path starting with slash as an absolute path.
Files: src/vim9script.c

Patch 8.2.4135

Problem: Vim9: ":scriptnames" shows unloaded imported autoload script.
Solution: Mark the unloaded script with "A". (closes #9552)
Files: runtime/doc/repeat.txt, src/scriptfile.c,

src/testdir/test_vim9_import.vim

Patch 8.2.4136

Problem: Vim9: the "autoload" argument of ":vim9script" is not useful.
Solution: Remove the argument. (closes #9555)
Files: runtime/doc/vim9.txt, runtime/doc/repeat.txt, src/vim9script.c,
src/errors.h, src/testdir/test_vim9_import.vim

Patch 8.2.4137

Problem: Vim9: calling import with and without method is inconsistent.
Solution: Set a flag that a parenthesis follows to compile_load_scriptvar().
Add some more tests. Improve error message.
Files: src/vim9expr.c, src/vim9execute.c, src/vim9script.c,
src/testdir/test_vim9_import.vim

Patch 8.2.4138

Problem: Vim9: no error for return with argument when the function does not
return anything.
Solution: Give an error for the invalid argument. (issue #9497)
Files: src/vim9cmds.c, src/testdir/test_vim9_func.vim

Patch 8.2.4139

Problem: Using freed memory if an expression abbreviation deletes the
abbreviation.
Solution: Do not access the pointer after evaluating the expression.
Files: src/map.c, src/testdir/test_mapping.vim

Patch 8.2.4140

Problem: maparg() does not indicate the type of script where it was defined.
Solution: Add "scriptversion".
Files: runtime/doc/builtin.txt, src/map.c, src/testdir/test_maparg.vim

Patch 8.2.4141 (after 8.2.4140)

Problem: Vim9 builtin functions test fails.
Solution: Add "scriptversion" item to maparg() result.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.4142

Problem: Build failure with normal features without persistent undo.
Solution: Adjust #ifdef. (closes #9557)
Files: src/fileio.c

Patch 8.2.4143

Problem: MS-Windows: IME support for Win9x is obsolete.
Solution: Remove the Win9x code. (Ken Takata, closes #9559)
Files: src/gui_w32.c

Patch 8.2.4144

Problem: Cannot load libsodium dynamically.
Solution: Support dynamic loading on MS-Windows. (Ken Takata, closes #9554)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak, src/buffer.c,
src/crypt.c, src/memline.c, src/proto/crypt.pro

Patch 8.2.4145

Problem: Confusing error when using name of import for a function.
Solution: Pass a flag to trans_function_name().
Files: src/vim.h, src/userfunc.c, src/proto/userfunc.pro, src/eval.c,
src/testdir/test_vim9_import.vim

Patch 8.2.4146

Problem: Vim9: shadowed function can be used in compiled function but not at script level.
Solution: Also give an error in a compiled function. (closes #9563)
Files: src/vim9expr.c

Patch 8.2.4147

Problem: E464 does not always include the offending command.
Solution: Add another error message with "%s". (closes #9564)
Files: src/errors.h, src/vim9compile.c, src/ex_docmd.c,
src/testdir/test_vim9_script.vim

Patch 8.2.4148

Problem: Deleting any mapping may cause `<ScriptCmd>` to not set the script context.
Solution: Only reset last_used_map if it is the deleted mapping. (closes #9568)
Files: src/map.c, src/getchar.c, src/proto/getchar.pro,
src/testdir/test_mapping.vim

Patch 8.2.4149

Problem: Test override not restored, autocommand left behind.
Solution: Correct restoring test override. Delete autocommand afterwards.
Files: src/testdir/test_autocmd.vim, src/testdir/test_mapping.vim

Patch 8.2.4150

Problem: Coverity warns for using pointer after free.
Solution: Swap statements, even though using the pointer is no problem.
Files: src/map.c

Patch 8.2.4151

Problem: Reading beyond the end of a line.
Solution: For block insert only use the offset for correcting the length.
Files: src/ops.c, src/testdir/test_visual.vim

Patch 8.2.4152

Problem: Block insert with double wide character fails.
Solution: Adjust the expected output.
Files: src/testdir/test_utf8.vim

Patch 8.2.4153

Problem: MS-Windows: Global IME is no longer supported.
Solution: Remove the Global IME implementation. (Ken Takata, closes #9562)
Files: Filelist, runtime/doc/mbyte.txt, src/Make_mvc.mak, src/dimm.idl,
src/glbl_ime.cpp, src/glbl_ime.h, src/gui_w32.c, src/vim.h

Patch 8.2.4154

Problem: ml_get error when exchanging windows in Visual mode.
Solution: Correct end of Visual area when entering another buffer.

Files: src/window.c, src/testdir/test_visual.vim

Patch 8.2.4155

Problem: Translating strftime() argument results in check error.

Solution: Add gettext comment.

Files: src/time.c

Patch 8.2.4156

Problem: Fileinfo message overwrites echo'ed message.

Solution: Reset need_fileinfo when displaying a message. (Rob Pilling, closes #9569)

Files: src/message.c, src/testdir/test_messages.vim, src/testdir/dumps/Test_fileinfo_after_echo.dump

Patch 8.2.4157

Problem: Terminal test fails because Windows sets the title.

Solution: Add the "vterm_title" testing override and use it in the test. (Ozaki Kiichi, closes #9556)

Files: runtime/doc/testing.txt, src/globals.h, src/terminal.c, src/testing.c, src/testdir/test_terminal.vim

Patch 8.2.4158

Problem: MS-Windows: memory leak in :browse.

Solution: Free stuff before returning. (Ken Takata, closes #9574)

Files: src/gui_w32.c

Patch 8.2.4159

Problem: MS-Windows: _WndProc() is very long.

Solution: Move code to separate functions. (Ken Takata, closes #9573)

Files: src/gui_w32.c

Patch 8.2.4160

Problem: Cannot change the register used for Select mode delete.

Solution: Make **CTRL-R** set the register to be used when deleting text for Select mode. (Shougo Matsushita, closes #9531)

Files: runtime/doc/visual.txt, src/globals.h, src/normal.c, src/ops.c, src/testdir/test_selectmode.vim

Patch 8.2.4161

Problem: Vim9: warning for missing white space after imported variable.

Solution: Do not skip white space. (closes #9567)

Files: src/vim9expr.c, src/testdir/test_vim9_import.vim

Patch 8.2.4162

Problem: Vim9: no error for redefining function with export.

Solution: Check for existing function with/without prefix. (closes #9577)

Files: src/userfunc.c, src/scriptfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4163

Problem: No error for omitting function name after autoload prefix.

Solution: Check for missing function name. (issue #9577)

Files: src/userfunc.c, src/testdir/test_vim9_import.vim

Patch 8.2.4164 (after 8.2.4162)

Problem: Error in legacy code for function shadowing variable.
Solution: Only give the error in Vim9 script.
Files: src/userfunc.c

Patch 8.2.4165

Problem: The nv_g_cmd() function is too long.
Solution: Move code to separate functions. (Yegappan Lakshmanan, closes #9576)
Files: src/normal.c

Patch 8.2.4166

Problem: Undo synced when switching buffer in another window.
Solution: Do not sync undo when not needed. (closes #9575)
Files: src/buffer.c, src/testdir/test_timers.vim

Patch 8.2.4167

Problem: Vim9: error message for old style import.
Solution: Use another error message. Add a test.
Files: src/evalvars.c, src/errors.h, src/testdir/test_vim9_import.vim

Patch 8.2.4168 (after 8.2.4163)

Problem: Disallowing empty function name breaks existing plugins.
Solution: Allow empty function name in legacy script.
Files: src/userfunc.c, src/testdir/test_autoload.vim, src/testdir/sautest/autoload/foo.vim

Patch 8.2.4169

Problem: MS-Windows: unnecessary casts and other minor things.
Solution: Clean up the MS-Windows code. (Ken Takata, closes #9583)
Files: src/gui_w32.c

Patch 8.2.4170

Problem: MS-Windows: still using old message API calls.
Solution: Call the "W" functions directly. (Ken Takata, closes #9582)
Files: src/gui_w32.c, src/os_mswin.c, src/os_win32.c, src/os_win32.h

Patch 8.2.4171

Problem: Cannot invoke option function using autoload import.
Solution: Expand the import to an autoload function name. (closes #9578)
Files: src/userfunc.c, src/evalvars.c, src/proto/evalvars.pro, src/option.c, src/testdir/test_vim9_import.vim

Patch 8.2.4172

Problem: Filetype detection for BASIC is not optimal.
Solution: Improve BASIC filetype detection. (Doug Kearns)
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4173

Problem: Cannot use an import in '**foldexpr**'.
Solution: Set the script context to where '**foldexpr**' was set. (closes #9584)
Fix that the script context was not set for all buffers.
Files: src/eval.c, src/proto/eval.pro, src/fold.c, src/structs.h, src/option.c, src/testdir/test_vim9_import.vim

Patch 8.2.4174

Problem: Vim9: can use an autoload name in normal script.
Solution: Disallow using an autoload name.
Files: src/userfunc.c, src/errors.h, src/testdir/test_vim9_import.vim

Patch 8.2.4175

Problem: MS-Windows: runtime check for multi-line balloon is obsolete.
Solution: Remove the obsolete code. (Ken Takata, closes #9592)
Files: src/evalfunc.c, src/gui_w32.c, src/proto/gui_w32.pro

Patch 8.2.4176

Problem: Vim9: cannot use imported function with call().
Solution: Translate the function name. (closes #9590)
Files: src/evalfunc.c, src/testdir/test_vim9_import.vim

Patch 8.2.4177

Problem: Vim9: autoload script not loaded after "vim9script noclear".
Solution: Check IMP_FLAGS_AUTOLOAD properly. (closes #9593)
Files: src/vim9compile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4178

Problem: Vim9: invalid error for return type of lambda when debugging.
Solution: Do not check the return type of a lambda. (closes #9589)
Files: src/vim9cmds.c

Patch 8.2.4179

Problem: **'foldtext'** is evaluated in the current script context.
Solution: Use the script context where the option was set.
Files: src/fold.c, src/buffer.c, src/eval.c, src/proto/eval.pro,
src/findfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4180

Problem: **'balloonexpr'** is evaluated in the current script context.
Solution: Use the script context where the option was set.
Files: src/beval.c, src/option.c, src/proto/option.pro,
src/testdir/test_balloon.vim,
src/testdir/dumps/Test_balloon_eval_term_01.dump,
src/testdir/dumps/Test_balloon_eval_term_01a.dump,
src/testdir/dumps/Test_balloon_eval_term_02.dump

Patch 8.2.4181

Problem: Vim9: cannot use an import in **'diffexpr'**.
Solution: Set the script context when evaluating **'diffexpr'**. Do not require **'diffexpr'** to return a bool, it was ignored anyway.
Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4182 (after 8.2.4182)

Problem: Memory leak when evaluating **'diffexpr'**.
Solution: Use free_tv() instead of clear_tv().
Files: src/evalvars.c

Patch 8.2.4183

Problem: Cannot use an import in **'formatexpr'**.

Solution: Set the script context when evaluating **'formatexpr'**.
Files: src/textformat.c, src/testdir/test_vim9_import.vim

Patch 8.2.4184

Problem: Cannot use an import in **'includeexpr'**.
Solution: Set the script context when evaluating **'includeexpr'**.
Files: src/findfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4185

Problem: Cannot use an import in **'indentexpr'**.
Solution: Set the script context when evaluating **'indentexpr'**.
Files: src/indent.c, src/testdir/test_vim9_import.vim

Patch 8.2.4186

Problem: Cannot use an import in **'patchexpr'**.
Solution: Set the script context when evaluating **'patchexpr'**. Do not require **'patchexpr'** to return a bool, it was ignored anyway.
Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4187

Problem: Gnuplot file not recognized.
Solution: Recognize ".gnuplot". (closes #9588)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4188

Problem: Not all gitconfig files are recognized.
Solution: Add a few more patterns. (Tim Pope, closes #9597)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4189

Problem: MS-Windows: code for "old look" is obsolete.
Solution: Delete obsolete code. Use "MS Shell Dlg" font. (Ken Takata, closes #9596)
Files: src/gui_w32.c

Patch 8.2.4190

Problem: All conceal tests are skipped without the screendumps feature.
Solution: Only skip the tests that use screendumps. (closes #9599)
Files: src/testdir/test_conceal.vim

Patch 8.2.4191

Problem: json5 files are not recognized.
Solution: Add a pattern for json5 files. (closes #9601)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4192

Problem: Cannot use an import in **'printexpr'**.
Solution: Set the script context when evaluating **'printexpr'**.
Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4193

Problem: Cannot use an import in **'charconvert'**.
Solution: Set the script context when evaluating **'charconvert'**. Also expand script-local functions in **'charconvert'**.

Files: src/evalvars.c, src/optionstr.c, src/testdir/test_vim9_import.vim

Patch 8.2.4194

Problem: MS-Windows: code for calculating font size is duplicated.

Solution: Move the code to a function. (Ken Takata, closes #9603)

Files: src/gui_w32.c

Patch 8.2.4195

Problem: Resizing terminal may cause to behave like **CTRL-Z**.

Solution: Set "got_tstp" only when in_mch_suspend is set. (Dorian Bivolaru, closes #9602, closes #9586)

Files: src/os_unix.c

Patch 8.2.4196

Problem: Various file types not recognized.

Solution: Add patterns to recognize more file types (closes #9607)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4197

Problem: Cannot use an import in the "expr" part of '**spellsuggest**'.

Solution: Set the script context when evaluating "expr" of '**spellsuggest**'.

Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4198

Problem: Vim9: the switch for executing instructions is too long.

Solution: Move some code to separate functions.

Files: src/vim9execute.c

Patch 8.2.4199

Problem: MS-Windows: Support for MSVC before 2003 is not useful.

Solution: Remove the exceptions for MSVC 6.0. (Ken Takata, closes #9616)

Files: src/GvimExt/gvimext.h, src/ex_docmd.c, src/feature.h, src/gui_w32.c, src/if_cscope.c, src/if_ole.cpp, src/if_ruby.c, src/macros.h, src/os_mswin.c, src/os_win32.c, src/os_win32.h, src/proto/os_win32.pro, src/time.c, src/vim.h

Patch 8.2.4200

Problem: Some tests do not clean up properly.

Solution: Delete created files. (Yegappan Lakshmanan, closes #9611)

Files: src/testdir/test_filetype.vim, src/testdir/test_messages.vim, src/testdir/test_vim9_import.vim

Patch 8.2.4201

Problem: When using the GUI **CTRL-Z** does not stop gvim.

Solution: When using the GUI set SIGTSTP to SIG_DFL. (Andrew Maltsev, closes #9570)

Files: src/os_unix.c

Patch 8.2.4202

Problem: Vim9: cannot export function that exists globally.

Solution: When checking if a function already exists only check for script-local functions. (closes #9615)

Files: src/userfunc.c, src/proto/userfunc.pro, src/vim.h, src/vim9compile.c, src/vim9instr.c,

src/testdir/test_vim9_import.vim

Patch 8.2.4203

Problem: Entering a character with **CTRL-V** may include modifiers.
Solution: Reset "mod_mask" when entering a character with digits after **CTRL-V**. (closes #9610)
Files: src/edit.c, src/testdir/test_edit.vim

Patch 8.2.4204

Problem: screenpos() has non-zero row for invisible text.
Solution: Only add the window row when the text is visible. (closes #9618)
Files: src/move.c, src/testdir/test_cursor_func.vim

Patch 8.2.4205

Problem: The normal_cmd() function is too long.
Solution: Move parts to separate functions. (Yegappan Lakshmanan, closes #9608)
Files: src/normal.c

Patch 8.2.4206

Problem: Condition with many "(" causes a crash.
Solution: Limit recursion to 1000.
Files: src/errors.h, src/eval.c, src/testdir/test_eval_stuff.vim

Patch 8.2.4207 (after 8.2.4206)

Problem: Recursion test fails with MSVC.
Solution: Use a smaller limit for MSVC.
Files: src/eval.c

Patch 8.2.4208

Problem: Using setbufvar() may change the window title.
Solution: Do not redraw when creating the autocommand window. (closes #9613)
Files: src/autocmd.c, src/testdir/test_functions.vim

Patch 8.2.4209

Problem: partial in **'opfunc'** cannot use an imported function.
Solution: Also expand the function name in a partial. (closes #9614)
Files: src/evalvars.c, src/testdir/test_vim9_import.vim

Patch 8.2.4210 (after 8.2.4208)

Problem: Window title test fails in some configurations.
Solution: Only run the test if the title can be obtained.
Files: src/testdir/test_functions.vim

Patch 8.2.4211 (after 8.2.4208)

Problem: Window title test still fails in some configurations.
Solution: Use WaitForAssert().
Files: src/testdir/test_functions.vim

Patch 8.2.4212 (after 8.2.4208)

Problem: Window title test still fails in some configurations.
Solution: Explicitly set the **'title'** option.
Files: src/testdir/test_functions.vim

Patch 8.2.4213

Problem: Too much code for supporting old MSVC versions.
Solution: Remove MSVC 2003 support. (Ken Takata, closes #9623)
Files: Filelist, src/INSTALLpc.txt, src/Make_mvc.mak, src/gui_w32.c, src/msvcsetup.bat, src/os_win32.c

Patch 8.2.4214

Problem: Illegal memory access with large 'tabstop' in Ex mode.
Solution: Allocate enough memory.
Files: src/ex_getln.c, src/testdir/test_ex_mode.vim

Patch 8.2.4215

Problem: Illegal memory access when copying lines in Visual mode.
Solution: Adjust the Visual position after copying lines.
Files: src/ex_cmds.c, src/testdir/test_visual.vim

Patch 8.2.4216

Problem: Vim9: cannot use a function from an autoload import directly.
Solution: Add the AUToload instruction to figure out at runtime. (closes #9620)
Files: src/vim9expr.c, src/vim9.h, src/vim9execute.c, src/vim9instr.c, src/proto/vim9instr.pro, src/testdir/test_vim9_import.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.4217

Problem: Illegal memory access when undo makes Visual area invalid.
Solution: Correct the Visual area after undo.
Files: src/undo.c, src/testdir/test_visual.vim

Patch 8.2.4218

Problem: Illegal memory access with bracketed paste in Ex mode.
Solution: Reserve space for the trailing NUL.
Files: src/edit.c, src/testdir/test_paste.vim

Patch 8.2.4219

Problem: Reading before the start of the line.
Solution: Check boundary before trying to read the character.
Files: src/register.c, src/testdir/test_visual.vim

Patch 8.2.4220

Problem: MS-Windows: some old compiler support remains.
Solution: Remove obsolete compiler support. (Ken Takata, closes #9627)
Files: src/Make_mvc.mak, src/vim.h

Patch 8.2.4221

Problem: Some functions in normal.c are very long.
Solution: Move code to separate functions. (Yegappan Lakshmanan, closes #9628)
Files: src/normal.c

Patch 8.2.4222

Problem: MS-Windows: clumsy way to suppress progress on CI.
Solution: Check for "\$CI" in the Makefile itself. (Ken Takata, closes #9631)
Files: .github/workflows/ci.yml, ci/appveyor.bat, src/Make_mvc.mak

Patch 8.2.4223

Problem: Long/int compiler warnings; function arguments swapped.
Solution: Add type casts. Swap arguments. (Ken Takata, closes #9632)
Files: src/alloc.c, src/eval.c, src/vim9script.c

Patch 8.2.4224

Problem: Vim9: no error when using a number for map() second argument
Solution: Disallow number to string conversion. (closes #9630)
Files: src/eval.c, src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4225

Problem: Vim9: depth argument of :lockvar not parsed in :def function.
Solution: Parse the optional depth argument. (closes #9629)
Fix that locking doesn't work for a non-materialize list.
Files: src/vim9cmds.c, src/evalvars.c, src/structs.h, src/evalfunc.c,
src/errors.h, src/vim9execute.c, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4226 (after 8.2.4224)

Problem: Filter-map test fails.
Solution: Only reject number argument in Vim9 script.
Files: src/eval.c

Patch 8.2.4227

Problem: Vim9: using "lockvar!" in :def function does not work.
Solution: Add "!" instead of "-1". (closes #9634)
Files: src/vim9cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4228

Problem: No tests for clicking in the GUI tabline.
Solution: Add test functions to generate the events. Add tests using the
functions. (Yegappan Lakshmanan, closes #9638)
Files: runtime/doc/builtin.txt, runtime/doc/testing.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/normal.c,
src/proto/testing.pro, src/testdir/test_diffmode.vim,
src/testdir/test_gui.vim, src/testdir/test_normal.vim,
src/testing.c

Patch 8.2.4229

Problem: Possible crash when invoking timer callback fails.
Solution: Initialize the typval. Give an error for an empty callback.
(closes #9636)
Files: src/time.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4230

Problem: MS-Windows: set_guifontwide() is included but won't work.
Solution: Include set_guifontwide() only for X11. (Ken Takata, closes #9640)
Files: src/gui.c

Patch 8.2.4231

Problem: Vim9: map() gives type error when type was not declared.
Solution: Only check the type when it was declared, like extend() does.
(closes #9635)

Files: src/list.c, src/evalfunc.c, src/vim9instr.c,
src/testdir/test_vim9_builton.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.4232 (after 8.2.4231)

Problem: Some compilers don't like a goto label without statement.

Solution: Return instead of using a goto.

Files: src/list.c

Patch 8.2.4233

Problem: Crash when recording and using Select mode.

Solution: When deleting the last recorded character check there is something to delete.

Files: src/getchar.c, src/testdir/test_registers.vim

Patch 8.2.4234

Problem: test_garbagecollect_now() does not check v:testing as documented.

Solution: Give an error if v:testing is not set.

Files: src/testing.c, src/errors.h, src/testdir/test_functions.vim

Patch 8.2.4235

Problem: Invalid check for NULL pointer.

Solution: Remove the check.

Files: src/getchar.c

Patch 8.2.4236

Problem: Accessing freed memory.

Solution: Set the bh_curr pointer to NULL.

Files: src/getchar.c

Patch 8.2.4237

Problem: Record buffer wrong if character in Select mode was not typed.

Solution: Only delete the tail from the record buffer if the character was typed. (closes #9650)

Files: src/normal.c, src/testdir/test_registers.vim

Patch 8.2.4238

Problem: *.tf file could be filetype "tf" or "terraform".

Solution: Detect the type from the file contents. (closes #9642)

Files: runtime/filetype.vim, runtime/autoload/dist/ft.vim,
src/testdir/test_filetype.vim

Patch 8.2.4239

Problem: Build fails with unusual configuration.

Solution: Adjust #ifdef. (closes #9651)

Files: src/testing.c

Patch 8.2.4240

Problem: Error for using flatten() in Vim9 script is unclear.

Solution: Add a remark to use flattennew().

Files: src/errors.h

Patch 8.2.4241

Problem: Some type casts are redundant.

Solution: Remove the type casts. (closes #9643)
Files: src/blob.c, src/buffer.c, src/channel.c, src/clientserver.c, src/clipboard.c, src/drawline.c, src/drawscreen.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_eval.c, src/fold.c, src/if_cscope.c, src/json.c, src/match.c, src/memline.c, src/message.c, src/misc1.c, src/normal.c, src/ops.c, src/option.c, src/optionstr.c, src/os_unix.c, src/register.c, src/sign.c, src/spellfile.c, src/tag.c, src/ui.c, src/undo.c, src/window.c

Patch 8.2.4242

Problem: Put in Visual mode cannot be repeated.
Solution: Use "P" to put without yanking the deleted text into the unnamed register. (Shougo Matsushita, closes #9591)
Files: runtime/doc/visual.txt, src/normal.c, src/register.c, src/testdir/test_visual.vim

Patch 8.2.4243

Problem: Lua tests fail with Lua 5.4.4.
Solution: Check messages like before Lua 5.4.3. (Jakub Kulík, closes #9652)
Files: src/testdir/test_lua.vim

Patch 8.2.4244

Problem: MS-Windows: warning from MSVC on debug build.
Solution: Adjust "/opt" options. Remove unused variables. Make variables uppercase for consistency. (Ken Takata, closes #9647)
Files: src/Make_mvc.mak

Patch 8.2.4245

Problem: ":retab 0" may cause illegal memory access.
Solution: Limit the value of 'tabstop' to 10000.
Files: src/option.c, src/vim.h, src/indent.c, src/testdir/test_options.vim

Patch 8.2.4246

Problem: One error message not in errors.h. (Antonio Colombo)
Solution: Move the message and rename.
Files: src/errors.h, src/if_perl.xs

Patch 8.2.4247

Problem: Stack corruption when looking for spell suggestions.
Solution: Prevent the depth increased too much. Add a five second time limit to finding suggestions.
Files: src/spellsuggest.c, src/testdir/test_spell.vim

Patch 8.2.4248

Problem: No proper test for moving the window separator.
Solution: Add a test. Add comment in code. (closes #9656)
Files: src/window.c, src/testdir/test_window_cmd.vim

Patch 8.2.4249

Problem: The timeout limit for spell suggestions is always 5000 milli seconds.
Solution: Add the "timeout" entry to 'spellsuggest'.

Files: runtime/doc/options.txt, src/spellsuggest.c,
src/testdir/test_spell.vim

Patch 8.2.4250

Problem: Channel out callback test is flaky on Mac.

Solution: Assign high priority to the test process. (Ozaki Kiichi,
closes #9653)

Files: src/testdir/test_channel_pipe.py, src/testdir/thread_util.py

Patch 8.2.4251

Problem: Vala files are not recognized.

Solution: Add the *.vala pattern. (closes #9654)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4252

Problem: Generating the normal command table at runtime is inefficient.

Solution: Generate the table with a Vim script and put it in a header file.
(Yegappan Lakshmanan, closes #9648)

Files: Filelist, runtime/doc/builtin.txt, runtime/doc/usr_41.txt,
src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms,
src/Makefile, src/create_nvcmdidxs.vim, src/evalfunc.c,
src/main.c, src/normal.c, src/nv_cmdidxs.h, src/proto/normal.pro

Patch 8.2.4253

Problem: Using freed memory when substitute uses a recursive function call.

Solution: Make a copy of the substitute text.

Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.2.4254

Problem: Using short instead of int.

Solution: Use int. (closes #9658)

Files: src/if_cscope.c

Patch 8.2.4255

Problem: Theoretical computation overflow.

Solution: Perform multiplication in a wider type. (closes #9657)

Files: src/alloc.c, src/drawline.c, src/eval.c, src/evalfunc.c,
src/ex_docmd.c, src/hardcopy.c, src/list.c, src/memfile.c,
src/memline.c, src/popupwin.c

Patch 8.2.4256

Problem: MS-Windows: compiler warnings when compiled with /W4.

Solution: Small adjustments to the code. (Ken Takata, closes #9659)

Files: src/gui_w32.c, src/os_win32.c

Patch 8.2.4257

Problem: Vim9: finding global function without g: prefix but not finding
global variable is inconsistent.

Solution: Require using g: for a global function. Change the vim9.vim
script into a Vim9 script with exports. Fix that import in legacy
script does not work.

Files: src/vim9expr.c, src/evalfunc.c, src/eval.c, src/userfunc.c,
src/testdir/vim9.vim, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim,

src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_import.vim,
src/testdir/test_vim9_script.vim, src/testdir/test_blob.vim,
src/testdir/test_execute_func.vim, src/testdir/test_debugger.vim,
src/testdir/test_expr.vim, src/testdir/test_filter_map.vim,
src/testdir/test_float_func.vim, src/testdir/test_functions.vim,
src/testdir/test_glob2regpat.vim, src/testdir/test_highlight.vim,
src/testdir/test_imsinsert.vim, src/testdir/test_ins_complete.vim,
src/testdir/test_listdict.vim, src/testdir/test_mapping.vim,
src/testdir/test_normal.vim, src/testdir/test_popupwin.vim,
src/testdir/test_profile.vim, src/testdir/test_quickfix.vim,
src/testdir/test_tagfunc.vim, src/testdir/test_textprop.vim,
src/testdir/test_usercommands.vim

Patch 8.2.4258

Problem: Coverity warns for array overrun.
Solution: Restrict depth to MAXWLEN - 1.
Files: src/spellsuggest.c

Patch 8.2.4259

Problem: Number of test functions for GUI events is growing.
Solution: Use one function with a dictionary. (Yegappan Lakshmanan,
closes #9660)
Files: runtime/doc/builtin.txt, runtime/doc/testing.txt,
runtime/doc/usr_41.txt, src/evalfunc.c, src/proto/testing.pro,
src/testdir/test_gui.vim, src/testdir/test_vim9_builtin.vim,
src/testing.c

Patch 8.2.4260

Problem: Vim9: can still use a global function without g: at the script
level.
Solution: Also check for g: at the script level. (issue #9637)
Files: src/userfunc.c, src/proto/userfunc.pro, src/evalvars.c,
src/vim9expr.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_import.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_scroll_11.dump,
src/testdir/dumps/Test_popupwin_scroll_12.dump

Patch 8.2.4261

Problem: Accessing invalid memory when a regular expression checks the
Visual area while matching in a string.
Solution: Do not try matching the Visual area in a string.
Files: src/regexp.c, src/testdir/test_help.vim

Patch 8.2.4262 (after 8.2.4261)

Problem: Some search tests fail.
Solution: Use a better way to reject searching for the Visual area.
Files: src/regexp.c

Patch 8.2.4263

Problem: No test for the GUI find/replace dialog.
Solution: Add a test function and a test. (Yegappan Lakshmanan, closes #9662)
Files: runtime/doc/testing.txt, src/testdir/test_gui.vim, src/testing.c

Patch 8.2.4264

Problem: Vim9: can use old style autoloader function name.
Solution: Give an error for old style autoloader function name.
Files: src/errors.h, src/userfunc.c, src/testdir/test_vim9_import.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim

Patch 8.2.4265 (after 8.2.4264)

Problem: Autoloader tests fails.
Solution: Use export instead of name with #.
Files: src/testdir/sautest/autoloader/auto9.vim, src/testdir/test_autoloader.vim, src/testdir/test_ins_complete.vim

Patch 8.2.4266

Problem: Compiler warning for uninitialized variable.
Solution: Initialize saved_did_emsg.
Files: src/userfunc.c

Patch 8.2.4267

Problem: Unused entry in keymap enum.
Solution: Remove the entry.
Files: src/keymap.h

Patch 8.2.4268

Problem: CI log output is long.
Solution: Group output in sections. (Ozaki Kiichi, closes #9670)
Files: .github/workflows/ci.yml

Patch 8.2.4269

Problem: Coverity warns for using a NULL pointer.
Solution: Check for "name" to not be NULL.
Files: src/userfunc.c

Patch 8.2.4270

Problem: Generating nv_cmdidxs.h requires building Vim twice.
Solution: Move the table into a separate file and use a separate executable to extract the command characters. (Ozaki Kiichi, closes #9669)
Files: src/normal.c, src/nv_cmds.h, Filelist, runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/Make_vms.mms, src/Makefile, src/create_nvcmdidxs.c, src/create_nvcmdidxs.vim, src/evalfunc.c, src/proto/normal.pro

Patch 8.2.4271

Problem: MS-Windows: cannot build with Ruby 3.1.0.
Solution: Adjust the DLL name and include directory. (Ken Takata, closes #9666)
Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.2.4272

Problem: Vim9 expr test fails without the channel feature. (Dominique Pellé)
Solution: Remove "g:" before "CheckFeature". (closes #9671)
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.4273

Problem: The EBCDIC support is outdated.
Solution: Remove the EBCDIC support.
Files: src/ascii.h, src/charset.c, src/cindent.c, src/digraph.c, src/edit.c, src/eval.c, src/evalfunc.c, src/ex_cmds.c, src/feature.h, src/filepath.c, src/findfile.c, src/getchar.c, src/gui.c, src/gui_motif.c, src/hardcopy.c, src/help.c, src/macros.h, src/map.c, src/mark.c, src/misc2.c, src/normal.c, src/ops.c, src/option.c, src/option.h, src/optiondefs.h, src/os_unix.c, src/proto/evalfunc.pro, src/regexp.c, src/regexp_bt.c, src/regexp_nfa.c, src/register.c, src/screen.c, src/spell.c, src/strings.c, src/structs.h, src/term.c, src/version.c, src/viminfo.c, src/testdir/test_edit.vim, src/testdir/test_exec_while_if.vim, src/testdir/test_expr.vim, src/testdir/test_gf.vim, src/testdir/test_regexp_utf8.vim

Patch 8.2.4274

Problem: Basic and form filetype detection is incomplete.
Solution: Add a separate function for .frm files. (Doug Kearns, closes #9675)
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4275

Problem: Cannot use an autoload function from a package under start.
Solution: Also look in the "start" package directory. (Bjorn Linse, closes #7193)
Files: src/scriptfile.c, src/testdir/test_packadd.vim

Patch 8.2.4276

Problem: Separate test function for the GUI scrollbar.
Solution: Use test_gui_event(). (Yegappan Lakshmanan, closes #9674)
Files: runtime/doc/builtin.txt, runtime/doc/testing.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/testing.c, src/proto/testing.pro, src/testdir/test_gui.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.4277

Problem: Vim9: an import does not shadow a command modifier.
Solution: Do not accept a command modifier followed by a dot.
Files: src/ex_docmd.c, src/testdir/test_vim9_import.vim

Patch 8.2.4278

Problem: Build with Athena GUI fails. (Elimar Riesebieter)
Solution: Add #ifdef.
Files: src/testing.c

Patch 8.2.4279

Problem: Vim9: cannot change item type with map() after range().
Solution: Split the return type in current type and declared type.

(closes #9665)
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9instr.c,
src/vim9type.c, src/proto/vim9type.pro,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4280 (after 8.2.4279)
Problem: list-dict test crashes.
Solution: Check declared type for add().
Files: src/vim9expr.vim

Patch 8.2.4281
Problem: Using freed memory with :lopen and :bwipe.
Solution: Do not use a wiped out buffer.
Files: src/buffer.c, src/testdir/test_quickfix.vim

Patch 8.2.4282
Problem: Restricted mode requires the -Z command line option.
Solution: Use restricted mode when \$SHELL ends in "nologin" or "false".
(closes #9681)
Files: runtime/doc/starting.txt, src/option.c,
src/testdir/test_restricted.vim

Patch 8.2.4283
Problem: Using a variable for the return value is not needed.
Solution: Return the value directly. (closes #9687)
Files: src/ex_docmd.c, src/misc2.c

Patch 8.2.4284
Problem: Old mac resources files are no longer used.
Solution: Delete the unused files. (Ozaki Kiichi, closes #9688)
Files: Filelist, src/Makefile, src/dehqx.py, src/infplist.xml,
src/os_mac.rsr.hqx, src/os_mac_rsrc/app.icns,
src/os_mac_rsrc/doc-txt.icns, src/os_mac_rsrc/doc.icns

Patch 8.2.4285
Problem: Vim9: type of item in for loop not checked properly.
Solution: Adjust the type checking. (closes #9683)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9cmds.c,
src/testdir/test_vim9_script.vim

Patch 8.2.4286
Problem: Vim9: strict type checking after copy() and deepcopy().
Solution: Allow type to change after making a copy. (closes #9644)
Files: src/eval.c, src/proto/eval.pro, src/dict.c, src/proto/dict.pro,
src/list.c, src/proto/list.pro, src/evalfunc.c, src/vim9execute.c,
src/vim9type.c, src/proto/vim9type.pro, src/evalvars.c,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_assign.vim

Patch 8.2.4287
Problem: Cannot assign empty list with any list type to variable with
specific list type.
Solution: Use unknown list type for empty list if the specified type is any.
Files: src/vim9type.c, src/testdir/test_vim9_assign.vim,

src/testdir/test_vim9_func.vim

Patch 8.2.4288

Problem: Preprocessor indents are inconsistent.
Solution: Fix preprocessor indents. (Ken Takata, closes #9691)
Files: src/arglist.c, src/change.c, src/ex_cmds.c, src/gui.c,
src/hashtab.c, src/indent.c, src/ops.c, src/os_win32.c

Patch 8.2.4289

Problem: Warnings reported by MSVC.
Solution: Rename variables and other fixes. (Ken Takata, closes #9689)
Files: src/cmdexpand.c, src/drawscreen.c, src/filepath.c, src/getchar.c,
src/menu.c, src/os_win32.c, src/version.c

Patch 8.2.4290

Problem: MS-Windows: using type casts for timer IDs.
Solution: Remove type casts and use the right type. (Ken Takata,
closes #9690) Remove old debug comments. Rename variables and
functions.
Files: src/gui_w32.c

Patch 8.2.4291

Problem: Error number used twice.
Solution: Renumber the errors.
Files: src/errors.h

Patch 8.2.4292 (after 8.2.4291)

Problem: Test fails.
Solution: Adjust the expected error number.
Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.4293

Problem: Vim9: when copying a list it gets type list<any> even when the
original list did not have a type.
Solution: Only set the type when the original list has a type. (closes #9692)
Files: src/list.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4294

Problem: MS-Windows: #ifdefs for Cygwin are too complicated.
Solution: Simplify the conditions. (Ken Takata, closes #9693)
Files: src/evalfunc.c, src/main.c, src/os_unix.c, src/os_win32.c,
src/os_win32.h

Patch 8.2.4295

Problem: Vim9: concatenating two lists may result in wrong type.
Solution: Remove the type instead of using list<any>. (closes #9692)
Files: src/list.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4296

Problem: Vim9: not all code covered by tests.
Solution: Add a few more tests for corner cases. Fix hang when single quote
is missing.
Files: src/vim9expr.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.4297

Problem: Vim9: not all code covered by tests.

Solution: Add a couple more tests.

Files: src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4298

Problem: Divide by zero with huge tabstop value.

Solution: Reject tabstop value that overflows to zero.

Files: src/indent.c, src/testdir/test_vartabs.vim

Patch 8.2.4299

Problem: SafeState autocommand interferes with debugging.

Solution: Do not trigger SafeState while debugging. (closes #9697)

Files: src/main.c

Patch 8.2.4300 (after 8.2.4299)

Problem: Cannot build tiny version. (Tony Mechelynck)

Solution: Add #ifdef.

Files: src/main.c

Patch 8.2.4301

Problem: Vim9: type error for copy of dict.

Solution: Do not use dict<any> but no type. (closes #9696)

Files: src/dict.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4302

Problem: Vim9: return type of getline() is too strict.

Solution: Make the declared type list<any>. Also do this for other functions returning a list of a specific type.

Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4303

Problem: A few messages should not be translated.

Solution: Remove _(). (Dominique Pellé, closes #9702)

Files: src/syntax.c

Patch 8.2.4304

Problem: Vim9: slice() makes a copy but doesn't change the type.

Solution: Change the declared type like copy(). (closes #9696)

Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4305

Problem: Tex filetype detection fails.

Solution: Check value to be positive. (closes #9704)

Files: runtime/autoload/dist/ft.vim, src/testdir/test_filetype.vim

Patch 8.2.4306

Problem: No test for fixed perl filetype check.

Solution: Add a test. Sort test functions.

Files: src/testdir/test_filetype.vim

Patch 8.2.4307

Problem: A few more messages should not be translated.
Solution: Remove _().
Files: src/syntax.c

Patch 8.2.4308

Problem: Vim9: cannot list autoload function.
Solution: Don't give an error for using # when listing a function.
(closes #9703)
Files: src/userfunc.c, src/testdir/test_vim9_import.vim

Patch 8.2.4309

Problem: Vim9: crash when using a partial in the wrong context.
Solution: Don't use an NULL outer pointer. (closes #9706)
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim

Patch 8.2.4310

Problem: Vim9: constant list and dict get a declaration type other than "any".
Solution: A constant list and dict have a declared member type "any".
(closes #9701)
Files: src/vim9instr.c, src/vim9type.c, src/proto/vim9type.pro, src/testdir/test_vim9_builtin.vim

Patch 8.2.4311

Problem: Vim9: changing script variable type not caught at compile time.
Solution: Set the declared type.
Files: src/vim9instr.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4312

Problem: No error for using :vim9script in a :def function.
Solution: Give an error when compiling.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.4313

Problem: Vim9: cannot change type of list after making a slice.
Solution: Adjust the declared member type. (closes #9696)
Files: src/vim9expr.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4314 (after 8.2.4312)

Problem: Test fails where lines are skipped.
Solution: Only give an error when not skipping commands.
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.4315

Problem: Put in Visual mode not fully tested.
Solution: Add a few more test cases. (closes #9708)
Files: src/testdir/test_visual.vim

Patch 8.2.4316

Problem: __CYGWIN32__ is not defined on 64 bit systems.
Solution: Update #ifdefs. (Ken Takata, closes #9709)
Files: src/main.c, src/os_unix.c, src/pty.c, src/vim.h

Patch 8.2.4317

Problem: MS-Windows: Vim exits when Python 3 initialisation fails.
Solution: Hook into the exit() function to recover from the failure.
(Ken Takata, closes #9710)
Files: runtime/doc/if_pyth.txt, src/if_python3.c, src/os_win32.c,
src/errors.h, src/proto/os_win32.pro

Patch 8.2.4318

Problem: Various comment and indent mistakes, returning wrong zero.
Solution: Fix the mistakes. Return NULL instead of FAIL.
Files: src/clientserver.c, src/eval.c, src/evalvars.c, src/vim9cmds.c,
src/window.c

Patch 8.2.4319

Problem: :put does not work properly in compiled function. (John Beckett)
Solution: Adjust the direction when using line zero.
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4320

Problem: Athena and Motif: when maximized scrollbar position is wrong.
Solution: Implement the scrollbar padding functions. (closes #9712)
Files: src/gui_athena.c, src/gui_motif.c

Patch 8.2.4321

Problem: Vim9: crash when using a funcref to a closure.
Solution: Copy pt_outer to the new partial. (closes #9714)
Files: src/evalfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.4322

Problem: Vim9: crash when using funcref with closure.
Solution: Keep a reference to the funcref that has the outer context.
(closes #9716)
Files: src/evalfunc.c, src/structs.h, src/eval.c, src/vim9execute.c,
src/testdir/test_vim9_func.vim

Patch 8.2.4323

Problem: Vim9: nested function name can start with "_".
Solution: Use same rule for function name for nested functions.
(closes #9713)
Files: src/vim9compile.c, src/testdir/test_vim9_func.vim

Patch 8.2.4324

Problem: Vim9: script-local function name can start with "_".
Solution: Check for leading capital after "s:". Correct error message.
Files: src/userfunc.c, src/errors.h, src/vim9compile.c,
src/testdir/test_vim9_func.vim

Patch 8.2.4325

Problem: **'wildmenu'** only shows few matches.
Solution: Add the "pum" option: use a popup menu to show the matches.
(Yegappan Lakshmanan et al., closes #9707)
Files: runtime/doc/options.txt, src/vim.h, src/cmdexpand.c,
src/drawscreen.c, src/evalfunc.c, src/ex_getln.c, src/option.h,
src/optionstr.c, src/popupmenu.c, src/proto/cmdexpand.pro,
src/testdir/test_cmdline.vim,

src/testdir/dumps/Test_wildmenu_pum_01.dump,
src/testdir/dumps/Test_wildmenu_pum_02.dump,
src/testdir/dumps/Test_wildmenu_pum_03.dump,
src/testdir/dumps/Test_wildmenu_pum_04.dump,
src/testdir/dumps/Test_wildmenu_pum_05.dump,
src/testdir/dumps/Test_wildmenu_pum_06.dump,
src/testdir/dumps/Test_wildmenu_pum_07.dump,
src/testdir/dumps/Test_wildmenu_pum_08.dump,
src/testdir/dumps/Test_wildmenu_pum_09.dump,
src/testdir/dumps/Test_wildmenu_pum_10.dump,
src/testdir/dumps/Test_wildmenu_pum_11.dump,
src/testdir/dumps/Test_wildmenu_pum_12.dump,
src/testdir/dumps/Test_wildmenu_pum_13.dump,
src/testdir/dumps/Test_wildmenu_pum_14.dump,
src/testdir/dumps/Test_wildmenu_pum_15.dump,
src/testdir/dumps/Test_wildmenu_pum_16.dump,
src/testdir/dumps/Test_wildmenu_pum_17.dump,
src/testdir/dumps/Test_wildmenu_pum_18.dump,
src/testdir/dumps/Test_wildmenu_pum_19.dump,
src/testdir/dumps/Test_wildmenu_pum_20.dump,
src/testdir/dumps/Test_wildmenu_pum_21.dump,
src/testdir/dumps/Test_wildmenu_pum_22.dump,
src/testdir/dumps/Test_wildmenu_pum_23.dump,
src/testdir/dumps/Test_wildmenu_pum_24.dump,
src/testdir/dumps/Test_wildmenu_pum_25.dump,
src/testdir/dumps/Test_wildmenu_pum_26.dump,
src/testdir/dumps/Test_wildmenu_pum_27.dump,
src/testdir/dumps/Test_wildmenu_pum_28.dump,
src/testdir/dumps/Test_wildmenu_pum_29.dump

Patch 8.2.4326

Problem: "o" and "O" copying comment not sufficiently tested.
Solution: Add a test case. (closes #9718)
Files: src/testdir/test_textformat.vim

Patch 8.2.4327

Problem: May end up with no current buffer.
Solution: When deleting the current buffer to not pick a quickfix buffer as the new current buffer.
Files: src/buffer.c, src/testdir/test_quickfix.vim

Patch 8.2.4328

Problem: Command line complete matches cleared when typing character.
(Dominique Pellé)
Solution: Only remove a popup menu if there is one.
Files: src/ex_getln.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_pum_30.dump,
src/testdir/dumps/Test_wildmenu_pum_31.dump

Patch 8.2.4329

Problem: No support for end line number and column in **'errorformat'**.
Solution: Add %e and %k. (closes #9624)
Files: runtime/doc/quickfix.txt, src/quickfix.c,
src/testdir/test_quickfix.vim

Patch 8.2.4330

Problem: Vim9: no error if script imports itself.
Solution: Give an error when a script imports itself.
Files: src/vim9script.c, src/errors.h, src/testdir/test_vim9_import.vim

Patch 8.2.4331

Problem: Vim9: no test for existing script variable in block.
Solution: Add a test.
Files: src/testdir/test_vim9_func.vim

Patch 8.2.4332

Problem: Vim9: incomplete test for existing script variable in block.
Solution: Add a couple more tests. Fix uncovered problem.
Files: src/userfunc.c, src/vim9compile.c, src/proto/vim9compile.pro,
src/vim9script.c, src/vim9expr.c, src/testdir/test_vim9_func.vim

Patch 8.2.4333

Problem: cstack not always passed to where it is needed.
Solution: Pass cstack through functions.
Files: src/eval.c, src/vim9expr.c, src/vim9script.c,
src/proto/vim9script.pro, src/vim9compile.c,
src/proto/vim9compile.pro

Patch 8.2.4334

Problem: Command line popup menu not positioned correctly.
Solution: Also use vim_strsize() on the existing text. (Naruhiko Nishino,
closes #9727)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_pum_32.dump

Patch 8.2.4335

Problem: No autocommand event triggered before changing directory. (Ronnie
Magatti)
Solution: Add DirChangedPre. (closes #9721)
Files: runtime/doc/autocmd.txt, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/vim.h, src/autocmd.c, src/misc2.c,
src/testdir/test_autocmd.vim

Patch 8.2.4336

Problem: Using :filter for :scriptnames does not work. (Ben Jackson)
Solution: Call message_filtered(). (closes #9720)
Files: src/scriptfile.c, src/testdir/test_filter_cmd.vim

Patch 8.2.4337

Problem: Part of condition is always true.
Solution: Remove that part of the condition. (closes #9729)
Files: src/filepath.c

Patch 8.2.4338

Problem: An error from an expression mapping messes up the display.
Solution: When the expression results in an empty string return K_IGNORE.
In cmdline mode redraw the command line. (closes #9726)
Files: src/getchar.c, src/testdir/test_mapping.vim,

```
src/testdir/dumps/Test_map_expr_2.dump,  
src/testdir/dumps/Test_map_expr_3.dump,  
src/testdir/dumps/Test_map_expr_4.dump
```

Patch 8.2.4339

Problem: **CTRL-A** does not work properly with the cmdline popup menu.

Solution: Fix issues with **CTRL-A**. Add more tests for the cmdline popup menu. Remove TermWait() before VeriryScreenDump(). Refactor the cmdline popup code. (Yegappan Lakshmanan, closes #9735)

Files: src/cmdexpand.c, src/ex_getln.c, src/popupmenu.c,
src/testdir/screendump.vim, src/testdir/test_buflines.vim,
src/testdir/test_cmdline.vim, src/testdir/test_conceal.vim,
src/testdir/test_cursorline.vim, src/testdir/test_diffmode.vim,
src/testdir/test_display.vim, src/testdir/test_highlight.vim,
src/testdir/test_match.vim, src/testdir/test_popup.vim,
src/testdir/test_search_stat.vim, src/testdir/test_terminal.vim,
src/testdir/test_textprop.vim,
src/testdir/dumps/Test_wildmenu_pum_33.dump,
src/testdir/dumps/Test_wildmenu_pum_34.dump,
src/testdir/dumps/Test_wildmenu_pum_35.dump,
src/testdir/dumps/Test_wildmenu_pum_36.dump,
src/testdir/dumps/Test_wildmenu_pum_37.dump

Patch 8.2.4340

Problem: Amiga: mch_can_exe() is not implemented.

Solution: Implement mch_can_exe() for Amiga OS 4. (Ola Söder, closes #9731)

Files: src/os_amiga.c

Patch 8.2.4341

Problem: Command line not redrawn when finishing popup menu and the screen has scrolled up.

Solution: Redraw the command line after updating the screen. (closes #9722)

Files: src/cmdexpand.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_pum_38.dump

Patch 8.2.4342

Problem: CI will soon switch to other windows version.

Solution: Use "windows-2019" instead of "windows-latest". (Ozaki Kiichi, closes #9740)

Files: .github/workflows/ci.yml

Patch 8.2.4343

Problem: When reloading not all properties are detected.

Solution: Add the "edit" value to v:fcs_choice. (Rob Pilling, closes #9579)

Files: runtime/doc/editing.txt, runtime/doc/eval.txt, src/fileio.c,
src/proto/fileio.pro, src/message.c, src/spellfile.c,
src/testdir/test_filechanged.vim

Patch 8.2.4344

Problem: Amiga: header file included twice.

Solution: Remove #include. (Ola Söder, closes #9733)

Files: src/memfile.c

Patch 8.2.4345

Problem: `<amatch>` is expanded like a file name for `DirChangedPre`.
Solution: Do not expand `<amatch>`. (closes #9742) Also for the `User` event.
Files: `src/autocmd.c`, `src/testdir/test_autocmd.vim`

Patch 8.2.4346

Problem: A custom statusline may cause `Esc` to work like `Enter` on the command line when the popup menu is displayed.
Solution: Save and restore `KeyTyped`. (closes #9749)
Files: `src/drawscreen.c`, `src/testdir/test_cmdline.vim`,
`src/testdir/dumps/Test_wildmenu_pum_39.dump`

Patch 8.2.4347

Problem: In some build setups `UNUSED` is not defined.
Solution: Change the logic of how `UNUSED` is defined. (Ola Söder, closes #9734)
Files: `src/vim.h`

Patch 8.2.4348

Problem: `"legacy exe cmd"` does not do what one would expect.
Solution: Apply the `"legacy"` and `"vim9script"` command modifiers to the argument of `":execute"`.
Files: `runtime/doc/vim9.txt`, `src/globals.h`, `src/eval.c`, `src/ex_docmd.c`,
`src/testdir/test_vim9_cmd.vim`

Patch 8.2.4349

Problem: `FileChangedShell` test fails on MS-Windows.
Solution: Skip the test on MS-Windows.
Files: `src/testdir/test_filechanged.vim`

Patch 8.2.4350

Problem: `FEAT_GUI_ENABLED` defined but never used.
Solution: Remove the `#define`. (Ola Söder, closes #9732)
Files: `src/vim.h`

Patch 8.2.4351

Problem: No coverage is measured on MS-Windows CI.
Solution: Enable coverage on MS-Windows. (Ozaki Kiichi, closes #9750)
Files: `.github/workflows/ci.yml`

Patch 8.2.4352

Problem: `ReScript` files are not recognized.
Solution: Add the `*.res` and `*.resi` patterns. (Ananda Umamil, closes #9752)
Files: `runtime/filetype.vim`, `src/testdir/test_filetype.vim`

Patch 8.2.4353

Problem: CI does not use the latest Lua and Python.
Solution: Use Lua 5.4.2 and Python 3.10. (closes #9744)
Files: `.github/workflows/ci.yml`

Patch 8.2.4354

Problem: Dynamic loading of `libsodium` not handled properly.
Solution: Fix `has()` and `:version`. Show an error message when loading fails. Fix memory leaks. (Ken Takata, closes #9754)
Files: `src/crypt.c`, `src/evalfunc.c`, `src/gui_dwrite.cpp`, `src/if_cscope.c`,

src/os_win32.c, src/proto/crypt.pro, src/proto/os_win32.pro,
src/version.c

Patch 8.2.4355

Problem: Unnecessary call to check_colorcolumn().
Solution: Remove the call. (Sean Dewar, closes #9748)
Files: src/option.c, src/window.c

Patch 8.2.4356

Problem: Command line completion functions are very long.
Solution: Refactor into multiple functions. (Yegappan Lakshmanan,
closes #9753)
Files: src/cmdexpand.c

Patch 8.2.4357 (after 8.2.4348)

Problem: sticky command modifiers are too sticky.
Solution: Do not apply command modifiers to a sourced script. (closes #9751)
Files: src/scriptfile.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4358

Problem: Vim9: line number of exception is not set.
Solution: Set the line number before throwing an exception. (closes #9755)
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.4359

Problem: crash when repeatedly using :retab.
Solution: Bail out when the line is getting too long.
Files: src/indent.c, src/testdir/test_retab.vim

Patch 8.2.4360

Problem: Vim9: allowing use of "s:" leads to inconsistencies.
Solution: Disallow using "s:" in Vim9 script at the script level.
Files: src/userfunc.c, src/proto/userfunc.pro, src/errors.h,
src/vim9compile.c, src/eval.c, src/testdir/vim9.vim,
src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_import.vim, src/testdir/test_vim9_script.vim

Patch 8.2.4361 (after 8.2.4360)

Problem: Vim9: some tests fail.
Solution: Fix the tests, mostly by removing "s:".
Files: src/testdir/test_expr.vim, src/testdir/test_functions.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_normal.vim,
src/testdir/test_tagfunc.vim

Patch 8.2.4362

Problem: :retab may allocate too much memory.
Solution: Bail out when allocating more than MAXCOL bytes.
Files: src/indent.c

Patch 8.2.4363

Problem: MS-Windows: running out of memory for a very long line.

Solution: Use a 32 bit value for MAXCOL also when ints are 64 bits.
Files: src/vim.h

Patch 8.2.4364

Problem: MS-Windows: still running out of memory for a very long line.
Solution: Check for negative length.
Files: src/indent.c

Patch 8.2.4365 (after 8.2.4348)

Problem: sticky command modifiers are too sticky.
Solution: Do not apply command modifiers to a called function. (closes #9751)
Files: src/userfunc.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4366

Problem: Not enough tests for command line completion.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #9760)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim,
src/testdir/test_usercommands.vim

Patch 8.2.4367

Problem: Calling in_vim9script() multiple times.
Solution: Call it once and keep the result.
Files: src/userfunc.c, src/eval.c

Patch 8.2.4368

Problem: Amiga: a few compiler warnings.
Solution: Adjust #ifdefs. Add "UNUSED". (Ola Söder, closes #9756,
closes #9757)
Files: src/term.c, src/os_amiga.c

Patch 8.2.4369

Problem: Redundant #ifdef argument.
Solution: Remove unused MSWIN. (Ola Söder, closes #9758)
Files: src/feature.h

Patch 8.2.4370

Problem: MS-Windows: libsodium.dll not included with the installer.
Solution: Add the file to the installer if it exists. (Christian Brabandt,
closes #9762)
Files: nsis/gvim.nsi

Patch 8.2.4371

Problem: Vim9: can create a script variable from a legacy function.
Solution: Disallow creating a script variable from a function.
Files: src/evalvars.c, src/errors.h, src/testdir/test_vim9_script.vim

Patch 8.2.4372

Problem: Filetype detection from file contents is in legacy script.
Solution: Use a compiled function for filetype detection.
Files: runtime/scripts.vim, runtime/autoload/dist/script.vim

Patch 8.2.4373

Problem: Expression test fails.
Solution: Make the test work with latest Vim9 syntax.

Files: src/testdir/test_expr.vim

Patch 8.2.4374

Problem: Unreachable code.
Solution: Remove outdated code lines.
Files: src/vim9compile.c

Patch 8.2.4375

Problem: ctx_imports is not used.
Solution: Delete ctx_imports. Add missing dependency.
Files: src/vim9.h, src/vim9compile.c, src/proto/vim9compile.pro,
src/eval.c, src/evalfunc.c, src/evalvars.c, src/userfunc.c,
src/vim9expr.c, src/vim9script.c, src/Makefile

Patch 8.2.4376

Problem: Not enough tests for command line completion.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #9771)
Files: src/testdir/test_cmdline.vim, src/testdir/test_usercommands.vim

Patch 8.2.4377

Problem: CI steps for Windows are a bit unorganized.
Solution: Organize CI test steps on Windows. (Ozaki Kiichi, closes #9764)
Files: .github/workflows/ci.yml

Patch 8.2.4378

Problem: Incsearch highlight broken when calling searchcount() in 'tabLine'
function. (Mirko Palmer)
Solution: Save and restore the incsearch state. (Christian Brabandt,
closes #9763, closes #9633)
Files: src/search.c, src/testdir/test_search_stat.vim,
src/testdir/dumps/Test_searchstat_inc_1.dump,
src/testdir/dumps/Test_searchstat_inc_2.dump,
src/testdir/dumps/Test_searchstat_inc_3.dump

Patch 8.2.4379

Problem: An empty change is reported to a listener.
Solution: Do not report an empty change. (closes #9768) Remove unused
return value.
Files: src/undo.c, src/change.c, src/testdir/test_listener.vim

Patch 8.2.4380

Problem: Small differences between Chinese translation files.
Solution: Add rule for converting UTF-8 to gb2312. (closes #9773)
Files: src/po/Makefile, src/po/Make_all.mak

Patch 8.2.4381 (after 8.2.4380)

Problem: Translation file listed twice.
Solution: Remove one entry.
Files: src/po/Make_all.mak

Patch 8.2.4382 (after 8.2.4346)

Problem: A custom '**tabline**' may cause Esc to work like Enter on the
command line when the popup menu is displayed.
Solution: Save and restore KeyTyped. (closes #9776)

Files: src/drawscreen.c, src/screen.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_wildmenu_pum_40.dump

Patch 8.2.4383

Problem: Vim9: unused code lines.
Solution: Rely on either "cctx" or "cstack" to not be NULL.
Files: src/vim9compile.c

Patch 8.2.4384

Problem: Vim9: error message not tested, some code not tested.
Solution: Add a couple of test cases. Give an error for a command modifier
without a command.
Files: src/errors.h, src/vim9compile.c, src/ex_docmd.c,
src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.4385 (after 8.2.4384)

Problem: Cannot build tiny version.
Solution: Adjust #ifdefs.
Files: src/errors.h

Patch 8.2.4386 (after 8.2.4384)

Problem: Still cannot build tiny version.
Solution: Adjust #ifdefs.
Files: src/ex_docmd.c

Patch 8.2.4387

Problem: Command line completion doesn't always work properly.
Solution: Adjust triggering after a "|". Add more tests. (Yegappan
Lakshmanan, closes #9779)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.4388

Problem: Dead code in op_insert().
Solution: Remove condition and else block. (closes #9782)
Files: src/ops.c

Patch 8.2.4389

Problem: screenpos() does not handle a position in a closed fold.
Solution: Check if the position is inside a closed fold. (closes #9778)
Files: src/move.c, src/testdir/test_cursor_func.vim

Patch 8.2.4390

Problem: Vim9: list from declaration with inferred type does not set the
type on the value.
Solution: When inferring the type in a variable declaration also set the
type of the list or dictionary. (closes #9705) Do not set the
type when the member is "any".
Files: src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4391

Problem: Command line executed when typing Esc in the GUI.
Solution: Move saving/restoring KeyTyped to build_stl_str_hl().

(closes #9783)
Files: src/buffer.c, src/screen.c

Patch 8.2.4392 (after 8.2.4002)
Problem: MS-Windows with VIMDLL: Escaping CSI is wrong.
Solution: Put back #ifdef. (Ken Takata, closes #9769)
Files: src/getchar.c

Patch 8.2.4393
Problem: Possible number overflow with nested folds.
Solution: Avoid a negative line number.
Files: src/fold.c

Patch 8.2.4394 (after 8.2.4392)
Problem: UTF8 select mode test fails on MS-Windows.
Solution: Revert the #ifdef change.
Files: src/getchar.c

Patch 8.2.4395
Problem: Some code lines not covered by tests.
Solution: Add a few more test cases. Fix getting more than one error for invalid assignment.
Files: src/evalvars.c, src/errors.h, src/vim9compile.c,
src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.4396 (after 8.2.4395)
Problem: Python3 test fails.
Solution: Remove "let".
Files: src/testdir/test_vim9_func.vim

Patch 8.2.4397
Problem: Crash when using many composing characters in error message.
Solution: Use mb_cptr2char_adv() instead of mb_ptr2char_adv().
Files: src/testing.c, src/testdir/test_assert.vim

Patch 8.2.4398
Problem: Some command completion functions are too long.
Solution: Refactor code into separate functions. Add a few more tests.
(Yegappan Lakshmanan, closes #9785)
Files: src/cmdexpand.c, src/ex_getln.c, src/usercmd.c,
src/proto/usercmd.pro, src/testdir/test_cmdline.vim

Patch 8.2.4399
Problem: Crash after ml_get error.
Solution: When returning "???" flush the line and set ml_line_lnum.
Files: src/memline.c

Patch 8.2.4400 (after 8.2.4394)
Problem: MS-Windows: cannot use the mouse in the console with VIMDLL.
Solution: use add_char2buf() instead of fix_input_buffer(). (closes #9784,
closes #9769)
Files: src/getchar.c

Patch 8.2.4401

Problem: Map listing does not clear the rest of the command line.
Solution: Call msg_clear_eos(). (closes #5623, closes #5962)
Files: src/map.c, src/testdir/test_mapping.vim,
src/testdir/dumps/Test_map_list_1.dump

Patch 8.2.4402

Problem: Missing parenthesis may cause unexpected problems.
Solution: Add more parenthesis in macros. (closes #9788)
Files: src/autocmd.c, src/charset.c, src/drawline.c, src/drawscreen.c,
src/evalfunc.c, src/fileio.c, src/fold.c, src/getchar.c,
src/highlight.c, src/memline.c, src/normal.c, src/quickfix.c,
src/regexp.c, src/search.c, src/sha256.c, src/spell.c,
src/spellfile.c, src/spellsuggest.c, src/syntax.c, src/window.c

Patch 8.2.4403

Problem: ml_get error with nested folds and deleting lines.
Solution: Correct the last line number before calling hasFoldingWin().
Files: src/change.c

Patch 8.2.4404

Problem: Vim9: some code not covered by tests.
Solution: Add a few specific test cases.
Files: src/vim9execute.c, src/testdir/test_vim9_func.vim,
src/testdir/test_vim9_import.vim

Patch 8.2.4405

Problem: Compiler warning for unused variable without the +folding feature.
(Tony Mechelynck)
Solution: Add #ifdef.
Files: src/change.c

Patch 8.2.4406

Problem: Expand functions use confusing argument names.
Solution: Rename "file" to "match". Refactor some completion code. Add a
few more tests. (Yegappan Lakshmanan, closes #9790)
Files: src/cmdexpand.c, src/testdir/test_usercommands.vim

Patch 8.2.4407

Problem: Vim9: some code not covered by tests.
Solution: Add more tests. Avoid giving two errors. Remove dead code.
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_func.vim

Patch 8.2.4408

Problem: Vim9: some code not covered by tests.
Solution: Add a few more tests. Correct error message. Allow unlet on dict
with a number key.
Files: src/vim9execute.c, src/errors.h, src/testdir/test_vim9_assign.vim

Patch 8.2.4409

Problem: Vim9: some code not covered by tests.
Solution: Add a few more tests. Fix reported line number.
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4410

Problem: Vim9: some code not covered by tests.
Solution: Add a few more tests. Remove dead code.
Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_script.vim

Patch 8.2.4411

Problem: Bicep files are not recognized.
Solution: Match *.bicep files. (Dundar Goc, closes #9791)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4412

Problem: Translation cleanup script does not remove empty lines at end.
Solution: Remove empty lines at the end. (Ken Takata, closes #9794)
Files: src/po/cleanup.vim

Patch 8.2.4413

Problem: Vim9: Coverity warns for using NULL pointer.
Solution: Give an internal error when funcref function can't be found.
Files: src/vim9execute.c

Patch 8.2.4414

Problem: Solidity files are not recognized.
Solution: Add the *.sol pattern. (Dundar Goc, closes #9792)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4415

Problem: Function argument name conflicts with C++ keyword.
Solution: Rename the argument.
Files: src/usercmd.c, src/proto/usercmd.pro

Patch 8.2.4416

Problem: Vim9: using a script-local function requires using "s:" when
setting **'completestfunc'**.
Solution: Do not require "s:" in Vim9 script. (closes #9796)
Files: runtime/doc/options.txt, src/userfunc.c,
src/testdir/test_ins_complete.vim

Patch 8.2.4417 (after 8.2.4416)

Problem: Using NULL pointer.
Solution: Set offset after checking for NULL pointer.
Files: src/userfunc.c

Patch 8.2.4418

Problem: Crash when using special multi-byte character.
Solution: Don't use isalpha() for an arbitrary character.
Files: src/charset.c, src/proto/charset.pro, src/filepath.c,
src/testdir/test_autochdir.vim

Patch 8.2.4419

Problem: Illegal memory access when using exactly 20 highlights.
Solution: Add one more item in the array. (Brandon Richardson,

closes #9800)
Files: src/buffer.c, src/testdir/test_tabline.vim

Patch 8.2.4420

Problem: Menu translations are inconsistent.

Solution: Add a Makefile to convert between encodings. (Ada (Haowen) Yu, closes #9801)

Files: runtime/lang/Makefile, runtime/lang/menu_af_af.latin1.vim,
runtime/lang/menu_ca_es.latin1.vim,
runtime/lang/menu_chinese_gb.936.vim,
runtime/lang/menu_chinese_taiwan.950.vim,
runtime/lang/menu_cs_cz.iso_8859-2.vim,
runtime/lang/menu_cs_cz.utf-8.vim,
runtime/lang/menu_czech_czech_republic.1250.vim,
runtime/lang/menu_czech_czech_republic.ascii.vim,
runtime/lang/menu_da.utf-8.vim,
runtime/lang/menu_de_de.latin1.vim,
runtime/lang/menu_eo.utf-8.vim,
runtime/lang/menu_es_es.latin1.vim,
runtime/lang/menu_fi_fi.latin1.vim,
runtime/lang/menu_fr_fr.latin1.vim,
runtime/lang/menu_hu_hu.iso_8859-2.vim,
runtime/lang/menu_hu_hu.utf-8.vim,
runtime/lang/menu_is_is.latin1.vim,
runtime/lang/menu_it_it.latin1.vim,
runtime/lang/menu_ja_jp.euc-jp.vim,
runtime/lang/menu_ja_jp.utf-8.vim,
runtime/lang/menu_japanese_japan.932.vim,
runtime/lang/menu_ko_kr.euckr.vim,
runtime/lang/menu_ko_kr.utf-8.vim,
runtime/lang/menu_nl_nl.latin1.vim,
runtime/lang/menu_no_no.latin1.vim,
runtime/lang/menu_pl_pl.iso_8859-2.vim,
runtime/lang/menu_pl_pl.utf-8.vim,
runtime/lang/menu_polish_poland.1250.vim,
runtime/lang/menu_pt_br.vim, runtime/lang/menu_pt_pt.vim,
runtime/lang/menu_ru.utf-8.vim,
runtime/lang/menu_ru_ru.koi8-r.vim,
runtime/lang/menu_ru_ru.utf-8.vim, runtime/lang/menu_ru_ru.vim,
runtime/lang/menu_sk_sk.iso_8859-2.vim,
runtime/lang/menu_sl_si.cp1250.vim,
runtime/lang/menu_sl_si.latin2.vim,
runtime/lang/menu_sl_si.utf-8.vim,
runtime/lang/menu_slovak_slovak_republic.1250.vim,
runtime/lang/menu_sr_rs.ascii.vim,
runtime/lang/menu_sr_rs.iso_8859-2.vim,
runtime/lang/menu_sr_rs.iso_8859-5.vim,
runtime/lang/menu_sr_rs.utf-8.vim,
runtime/lang/menu_sv_se.latin1.vim,
runtime/lang/menu_tr_tr.cp1254.vim,
runtime/lang/menu_tr_tr.iso_8859-9.vim,
runtime/lang/menu_tr_tr.utf-8.vim,
runtime/lang/menu_uk_ua.cp1251.vim,
runtime/lang/menu_uk_ua.koi8-u.vim,

runtime/lang/menu_uk_ua.utf-8.vim, runtime/lang/menu_vi_vn.vim,
runtime/lang/menu_zh_cn.utf-8.vim,
runtime/lang/menu_zh_tw.utf-8.vim

Patch 8.2.4421

Problem: Some installed files and directories have wrong permissions.
Solution: Adjust the Makefile and shell to set permissions. (closes #9793)
Files: src/Makefile, src/installman.sh

Patch 8.2.4422

Problem: Autochdir test fails on MS-Windows.
Solution: Expect another error on MS-Windows.
Files: src/testdir/test_autochdir.vim

Patch 8.2.4423

Problem: "make nvcmdidxs" fails.
Solution: Use "-S" instead of "-u" to source the script.
Files: src/Makefile

Patch 8.2.4424

Problem: ".gts" and ".gjs" files are not recognized.
Solution: Recognize Glimmer flavored typescript and javascript.
(closes #9799)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4425

Problem: map() function does not check function arguments at compile time.
Solution: Give an error if the arguments of a map() function are wrong.
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.4426

Problem: map() function on string and blob does not check argument types at compile time.
Solution: Check string and blob argument types. Support "0z1234->func()".
Files: src/vim9compile.c, src/evalfunc.c, src/ex_docmd.c,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4427

Problem: getchar() may return modifiers if no character is available.
Solution: Do not process modifiers when there is no character. (closes #9806)
Files: src/getchar.c, src/testdir/test_functions.vim

Patch 8.2.4428

Problem: Crash when switching tabpage while in the cmdline window.
Solution: Disallow switching tabpage when in the cmdline window.
Files: src/window.c, src/proto/window.pro, src/evalvars.c,
src/evalvars.c, src/usercmd.c

Patch 8.2.4429

Problem: Using script-local function from the wrong script when using a partial. (Yegappan Lakshmanan)
Solution: Include the script ID in the partial name.
Files: src/userfunc.c, src/proto/userfunc.pro, src/evalfunc.c,

src/vim9type.c, src/testdir/test_vim9_import.vim

Patch 8.2.4430

Problem: GTK: crash when using **'guiligatures'** and reading from stdin.
Solution: Make a copy of the message. (Amon Sha, closes #9719, closes #9814)
Files: src/fileio.c

Patch 8.2.4431

Problem: Unnecessary condition when assigning to a variable.
Solution: Remove the condition.
Files: src/evalvars.c

Patch 8.2.4432 (after 8.2.4428)

Problem: Cannot use settabvar() while the cmdline window is open.
Solution: Only give an error when actually switching tabpage.
(closes #9813)
Files: src/window.c

Patch 8.2.4433

Problem: CI: cannot see interface versions for MS-Windows.
Solution: List the interface versions. (Ken Takata, closes #9811)
Files: .github/workflows/ci.yml

Patch 8.2.4434

Problem: Duplicate check for cmdline window.
Solution: Remove the second check. (Sean Dewar, closes #9816)
Files: src/window.c

Patch 8.2.4435

Problem: Dead code in checking map() arguments. (Dominique Pellé)
Solution: Remove the first return statement. (closes #9815)
Files: src/evalfunc.c

Patch 8.2.4436

Problem: Crash with weird **'vartabstop'** value.
Solution: Check for running into the end of the line.
Files: src/indent.c, src/testdir/test_vartabs.vim

Patch 8.2.4437

Problem: Vartabs test fails on MS-Windows.
Solution: Use iso8859-1 **'encoding'**. (Ken Takata, closes #9818)
Files: src/testdir/test_vartabs.vim

Patch 8.2.4438

Problem: Crash on exit when using cmdline window.
Solution: Reset "cmdwin_type" before exiting. (closes #9817)
Files: src/ui.c, src/testdir/test_exit.vim

Patch 8.2.4439

Problem: Accepting "iso8859" **'encoding'** as "iso-8859-".
Solution: use "iso8859" as "iso-8859-1".
Files: src/mbyte.c, src/testdir/test_options.vim

Patch 8.2.4440

Problem: Crash with specific regexp pattern and string.
Solution: Stop at the start of the string.
Files: src/regexp_bt.c, src/testdir/test_regexp_utf8.vim

Patch 8.2.4441

Problem: Vim9: function argument of filter() not checked like map().
Solution: Also check the function argument of filter().
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4442 (after 8.2.4438)

Problem: Test for error reading input fails on MS-Windows.
Solution: Don't run the test on MS-Windows.
Files: src/testdir/test_exit.vim

Patch 8.2.4443 (after 8.2.4440)

Problem: Regexp pattern test fails on Mac.
Solution: Do not use a swapfile for the buffer.
Files: src/testdir/test_regexp_utf8.vim

Patch 8.2.4444

Problem: Beep caused by test. ASAN reports leaks.
Solution: Do not put a NL at the end of the script. Make the text work on MS-Windows. Do not run the test with ASAN.
Files: src/testdir/test_exit.vim

Patch 8.2.4445

Problem: Exit test fails on MS-Windows anyway.
Solution: Skip the test on MS-Windows.
Files: src/testdir/test_exit.vim

Patch 8.2.4446

Problem: Vim9: cannot refer to a global function like a local one.
Solution: When g:name is not a variable but a function, use a function reference. (closes #9826)
Files: src/vim9execute.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4447

Problem: Vim9: can still use s:var in a compiled function.
Solution: Disallow using s:var for Vim9 script. (closes #9824)
Files: runtime/doc/vim9.txt, src/vim9expr.c, src/vim9compile.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4448 (after 8.2.4447)

Problem: Filetype detection is failing.
Solution: Do not use "s:" where it is no longer allowed.
Files: runtime/autoload/dist/ft.vim,

Patch 8.2.4449

Problem: vim9: function argument of sort() not checked at compile time.
Solution: Add a compile time check.
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4450 (after 8.2.4449)

Problem: List sort test fails.

Solution: Pass a valid "how" argument.
Files: src/testdir/test_listdict.vim

Patch 8.2.4451 (after 8.2.4450)
Problem: sort() fails when ignoring case.
Solution: Accept a number one argument in sort().
Files: src/evalfunc.c, src/testdir/test_listdict.vim

Patch 8.2.4452
Problem: Test for what 8.2.4436 fixes does not check for regression.
Solution: Set several options. (Ken Takata, closes #9830)
Files: src/testdir/test_vartabs.vim

Patch 8.2.4453
Problem: :helpgrep may free an option that was not allocated. (Yegappan Lakshmanan)
Solution: Check if the value was allocated.
Files: src/option.c, src/proto/option.pro, src/quickfix.c, src/testdir/test_quickfix.vim

Patch 8.2.4454
Problem: Resetting cmdwin_type only for one situation.
Solution: Reset cmdwin_type before closing windows. (closes #9822)
Files: src/ui.c, src/window.c, src/testdir/test_exit.vim

Patch 8.2.4455
Problem: Accepting one and zero for the second sort() argument is strange.
Solution: Disallow using one and zero in Vim9 script.
Files: runtime/doc/builtin.txt, src/evalfunc.c, src/list.c, src/testdir/test_listdict.vim

Patch 8.2.4456
Problem: Terminal test may fail on some machines.
Solution: Increase wait time. (Zdenek Dohnal, closes #9834)
Files: src/testdir/test_terminal.vim

Patch 8.2.4457
Problem: The GPM library can only be linked statically.
Solution: Make it possible to load the GPM library dynamically. (Damien)
Files: runtime/doc/various.txt, src/config.h.in, src/configure.ac, src/Makefile, src/evalfunc.c, src/feature.h, src/os_unix.c, src/proto/os_unix.pro, src/version.c

Patch 8.2.4458
Problem: Vim9: compiling filter() call fails with funcref that has unknown arguments.
Solution: Do not check the arguments if they are unknown at compile time. (closes #9835)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4459
Problem: Vim9: compiling sort() call fails with a funcref that has unknown arguments.
Solution: Do not check the arguments if they are unknown at compile time.

(closes #9835)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4460

Problem: Vim9: wrong error for defining dict function.
Solution: Explicitly check for trying to define a dict function.
(closes #9827)
Files: src/errors.h, src/userfunc.c, src/vim9compile.c,
src/testdir/test_vim9_func.vim

Patch 8.2.4461

Problem: MS-Windows: garbage characters on stdout with VIMDLL.
Solution: Don't call gui_focus_change() when about to quit. (Ken Takata,
closes #9840)
Files: src/gui_w32.c

Patch 8.2.4462

Problem: Not enough testing for quickfix code.
Solution: Add more tests. Fix uncovered problem. (Yegappan Lakshmanan,
closes #9839)
Files: src/quickfix.c, src/window.c, src/testdir/test_makeencoding.vim,
src/testdir/test_quickfix.vim

Patch 8.2.4463

Problem: Completion only uses strict matching.
Solution: Add the "fuzzy" item for '**wildoptions**'. (Yegappan Lakshmanan,
closes #9803)
Files: runtime/doc/options.txt, src/buffer.c, src/cmdexpand.c,
src/option.c, src/option.h, src/optionstr.c,
src/proto/cmdexpand.pro, src/proto/option.pro,
src/proto/search.pro, src/search.c, src/structs.h,
src/testdir/gen_opt_test.vim, src/testdir/test_cmdline.vim

Patch 8.2.4464

Problem: Dtrace files are recognized as filetype D.
Solution: Add a pattern for Dtrace files. (Teubel György, closes #9841)
Add some more testing.
Files: runtime/autoload/dist/ft.vim, runtime/filetype.vim,
src/testdir/test_filetype.vim

Patch 8.2.4465

Problem: Fuzzy completion does not order matches properly.
Solution: Do not use regular expression match. (Yegappan Lakshmanan,
closes #9843)
Files: src/cmdexpand.c, src/search.c, src/testdir/test_cmdline.vim

Patch 8.2.4466

Problem: MS-Windows: illegal memory access in installer when using
"create-directories" as the final argument.
Solution: Check the argument count. (Cam Sinclair, closes #9844)
Files: src/dosinst.c

Patch 8.2.4467

Problem: Running filetype test leaves file behind.

Solution: Delete the file.
Files: src/testdir/test_filetype.vim

Patch 8.2.4468

Problem: Coverity warns for uninitialized struct member.
Solution: Set color.index to zero.
Files: src/terminal.c

Patch 8.2.4469

Problem: Coverity warns for uninitialized variable.
Solution: Set the value to zero.
Files: src/ex_getln.c

Patch 8.2.4470

Problem: Coverity warns for uninitialized variable.
Solution: Set can_spell to zero.
Files: src/drawline.c

Patch 8.2.4471

Problem: Coverity warns for uninitialized variable.
Solution: Set flags to zero.
Files: src/vim9cmds.c

Patch 8.2.4472

Problem: Coverity warns for use of a freed function name.
Solution: Only check an autoload name when is prefixed.
Files: src/userfunc.c

Patch 8.2.4473

Problem: Coverity warns for not checking return value of ftell().
Solution: Bail out if ftell() returns a negative value.
Files: src/spellfile.c

Patch 8.2.4474

Problem: Memory allocation failures not tested in quickfix code.
Solution: Add alloc IDs and tests. (Yegappan Lakshmanan, closes #9848)
Files: src/alloc.h, src/quickfix.c, src/vim.h,
src/testdir/test_quickfix.vim

Patch 8.2.4475

Problem: Fuzzy cmdline completion does not work for lower case.
Solution: Also use fuzzy completion for lower case input. (Yegappan Lakshmanan, closes #9849)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.4476

Problem: Operator name spelled wrong.
Solution: Change trinary to ternary. (Goc Dunder, closes #9850)
Files: src/testdir/test_expr.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vimscript.vim

Patch 8.2.4477

Problem: Crash when using fuzzy completion.
Solution: Temporary fix: put back regexp. (closes #9851)

Files: src/cmdexpand.c

Patch 8.2.4478

Problem: Crash when using fuzzy completion.

Solution: Temporary fix: put back regexp. (closes #9852, closes #9851)

Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.4479

Problem: No fuzzy completion for maps and abbreviations.

Solution: Fuzzy complete maps and abbreviations. (Yegappan Lakshmanan, closes #9856)

Files: src/cmdexpand.c, src/map.c, src/proto/map.pro, src/search.c, src/testdir/test_cmdline.vim

Patch 8.2.4480

Problem: Suspending with **CTRL-Z** does not work on Android.

Solution: Do not handle SIGTSTP. (closes #9854)

Files: src/os_unix.c

Patch 8.2.4481

Problem: Cmdline popup menu not removed when '**lazyredraw**' is set.

Solution: Temporarily reset '**lazyredraw**' when removing the popup menu. (closes #9857)

Files: src/cmdexpand.c, src/testdir/test_cmdline.vim, src/testdir/dumps/Test_wildmenu_pum_41.dump

Patch 8.2.4482

Problem: No fuzzy cmdline completion for user defined completion.

Solution: Add fuzzy completion for user defined completion. (Yegappan Lakshmanan, closes #9858)

Files: src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.4483

Problem: Command completion makes two rounds to collect matches.

Solution: Use a growarray to collect matches. (Yegappan Lakshmanan, closes #9860)

Files: src/buffer.c, src/cmdexpand.c, src/map.c, src/testdir/test_cmdline.vim

Patch 8.2.4484

Problem: Vim9: some error messages are not tested.

Solution: Add a few more test cases. Delete dead code.

Files: src/vim9execute.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim

Patch 8.2.4485

Problem: Compiler warning for uninitialized variable.

Solution: Initialize the variable. (John Marriott)

Files: src/cmdexpand.c

Patch 8.2.4486

Problem: MS-Windows GUI: slow scrolling with maximized window.

Solution: Use a better way to check the window is on screen. (Ken Takata, closes #9865)

Files: src/gui_w32.c

Patch 8.2.4487

Problem: Vim9: cannot compare with v:null.

Solution: Allow comparing anything with v:null. (closes #9866)

Files: src/vim9instr.c, src/typval.c, src/proto/typval.pro,
src/vim9.h, src/vim9execute.c, src/evalvars.c,
src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4488 (after 8.2.4487)

Problem: Build error with +eval but without +channel or +job.

Solution: Add #ifdef. (John Marriott)

Files: src/typval.c

Patch 8.2.4489 (after 8.2.4487)

Problem: Failing test for comparing v:null with number.

Solution: Allow comparing v:null with number in legacy script.
(Ken Takata, closes #9873) Also do this for float.

Files: src/typval.c, src/testdir/test_vimscript.vim

Patch 8.2.4490

Problem: Terminal focus reporting only works for xterm-like terminals.
(Jonathan Rascher)

Solution: Remove the "focus_mode" flag. (closes #9859)

Files: src/term.c

Patch 8.2.4491

Problem: MS-Windows makefile dependencies are outdated.

Solution: Update dependencies. (Ken Takata, closes #9876)

Files: src/Make_cyg_ming.mak, src/Make_mvc.mak

Patch 8.2.4492

Problem: No error if an option is given an invalid value with
":let &opt = val".

Solution: Give the error. (closes #9864)

Files: src/evalvars.c, src/testdir/test_options.vim

Patch 8.2.4493 (after 8.2.4492)

Problem: Options test fails in the GUI.

Solution: Do not save and restore 'term'.

Files: src/testdir/gen_opt_test.vim

Patch 8.2.4494

Problem: The find_tags() function is much too long.

Solution: Refactor the function. (Yegappan Lakshmanan, closes #9869)

Files: src/quickfix.c, src/tag.c, src/testdir/test_tagjump.vim

Patch 8.2.4495

Problem: Help test fails in 24 line terminal.

Solution: Use up to 23 lines for text.

Files: src/testdir/test_help.vim

Patch 8.2.4496 (after 8.2.4494)

Problem: Coverity gives warnings after tags code refactoring.
Solution: Avoid the warnings. (Yegappan Lakshmanan, closes #9882)
Files: src/tag.c

Patch 8.2.4497

Problem: Wrong color for half of wide character next to pum scrollbar.
Solution: Redraw the screen cell with the right color. (closes #9874)
Files: src/screen.c, src/testdir/test_ins_complete.vim,
src/testdir/dumps/Test_scrollbar_on_wide_char.dump

Patch 8.2.4498

Problem: Using <Plug> with "noremap" does not work.
Solution: Always remap <Plug>. (closes #9879, closes #9789)
Files: runtime/doc/map.txt, src/getchar.c, src/testdir/test_mapping.vim

Patch 8.2.4499

Problem: Vim9: at the script level declarations leak from try block to catch and finally block.
Solution: End the block and start a new one. (closes #9883)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.4500

Problem: Vim9: can declare a global variable on the command line.
Solution: Disallow declaring a variable on the command line. (closes #9881)
Files: src/errors.h, src/evalvars.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_script.vim,
src/testdir/dumps/Test_vim9_reject_declaration.dump

Patch 8.2.4501

Problem: With 'showbreak' set and after the end of the line the cursor may be displayed in the wrong position.
Solution: Do not apply 'showbreak' after the end of the line. (closes #9884)
Files: src/charset.c, src/testdir/test_breakindent.vim,
src/testdir/dumps/Test_cursor_position_with_showbreak.dump

Patch 8.2.4502

Problem: In the GUI a modifier is not recognized for the key typed after **CTRL-X**, which may result in a mapping to be used. (Daniel Steinberg)
Solution: Recognize a modifier starting with CSI. (closes #9889)
Files: src/getchar.c, src/testdir/test_ins_complete.vim

Patch 8.2.4503

Problem: Vim9: there is no point in supporting :Print and :mode.
Solution: Do not recognize :Print and :mode as commands. (closes #9870)
Files: src/ex_docmd.c, src/testdir/test_vim9_script.vim

Patch 8.2.4504

Problem: When there is a partially matching map and modifyOtherKeys is active a full map may not work.
Solution: Only simplify modifiers when there is no matching mapping. (closes #8792)
Files: src/getchar.c, src/testdir/test_termcodes.vim

Patch 8.2.4505

Problem: Vim9: outdated "autocmd nested" still works.

Solution: Do not accept the :autocmd argument "nested" without "++" in Vim9 script.

Files: src/autocmd.c, src/errors.h, src/testdir/test_autocmd.vim

Patch 8.2.4506

Problem: "pattern not found" for :global is not an error message.

Solution: In Vim9 script make this an actual error, so that try/catch can be used as expected.

Files: src/ex_cmds.c, src/errors.h, src/testdir/test_global.vim

Patch 8.2.4507 (after 8.2.4506)

Problem: Test fails because of new error message.

Solution: Avoid the test fails.

Files: src/testdir/test_vim9_cmd.vim

Patch 8.2.4508

Problem: Vim9: cannot assign to a global variable on the command line.

Solution: Allow using :vim9cmd for assignment on the command line.

Files: src/evalvars.c, src/testdir/test_vim9_script.vim,
src/testdir/dumps/Test_vim9_reject_declaration.dump,
src/testdir/dumps/Test_vim9_reject_declaration_1.dump,
src/testdir/dumps/Test_vim9_reject_declaration_2.dump

Patch 8.2.4509

Problem: Vim9: can declare a variable with ":va".

Solution: Disallow using ":va", require using ":var".

Files: src/evalvars.c, src/errors.h, src/vim9compile.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.4510

Problem: Vim9: shortening commands leads to confusing script.

Solution: In Vim9 script require at least ":cont" for ":continue", "const" instead of "cons", "break" instead of "brea", "catch" instead of "cat", "else" instead of "el" "elseif" instead of "elsei" "endfor" instead of "endfo" "endif" instead of "en" "endtry" instead of "endt", "finally" instead of "fina", "throw" instead of "th", "while" instead of "wh".

Files: src/ex_cmds.h, src/ex_docmd.c, src/errors.h, src/evalvars.c,
src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.4511

Problem: Filetype test fails.

Solution: Change "endw" to "endwhile".

Files: runtime/autoload/dist/ft.vim

Patch 8.2.4512

Problem: The find_tags_in_file() function is much too long.

Solution: Refactor into multiple smaller functions. (Yegappan Lakshmanan, closes #9892)

Files: Filelist, src/Makefile, src/quickfix.c, src/tag.c,
src/testdir/test83-tags2, src/testdir/test83-tags3,
src/testdir/test_tagjump.vim

Patch 8.2.4513

Problem: Window-local directory is not applied if 'acd' fails.
Solution: Don't call do_autochdir(). (closes #9891)
Files: src/window.c, src/testdir/test_autochdir.vim

Patch 8.2.4514

Problem: Vim9: some flow commands can be shortened.
Solution: Also require using the full name for ":return", ":enddef", ":continue", ":export" and ":import".
Files: src/ex_cmds.h, src/ex_docmd.c, src/errors.h, src/userfunc.c, src/testdir/test_vim9_script.vim

Patch 8.2.4515

Problem: Old substitute syntax is still supported.
Solution: Disallow using backslash after ":s" in Vim9 script.
Files: src/ex_cmds.c, src/errors.h, src/testdir/test_substitute.vim

Patch 8.2.4516 (after 8.2.4515)

Problem: Build failure without the +eval feature.
Solution: Move error message outside of #ifdef.
Files: src/errors.h

Patch 8.2.4517

Problem: MS-Windows: cannot specify location of sodium library.
Solution: Allow for using a path for SODIUM. (Ken Takata, closes #9896)
Files: src/Make_cyg_ming.mak

Patch 8.2.4518

Problem: The binary tag search feature is always enabled.
Solution: Remove the #ifdefs. Add a few more tests. (Yegappan Lakshmanan, closes #9893)
Files: src/evalfunc.c, src/feature.h, src/tag.c, src/version.c, src/testdir/test_tagjump.vim, src/testdir/test_taglist.vim

Patch 8.2.4519

Problem: Vim9: Can still use ":fini" and ":finis" for ":finish".
Solution: Require using ":finish".
Files: src/ex_cmds.h, src/testdir/test_vim9_script.vim

Patch 8.2.4520

Problem: Using wrong highlight for cursor line number.
Solution: Take filler lines into account when using CursorLineNr. (closes #9897)
Files: src/drawline.c, src/testdir/test_diffmode.vim, src/testdir/dumps/Test_diff_with_cursorline_number_01.dump, src/testdir/dumps/Test_diff_with_cursorline_number_02.dump

Patch 8.2.4521 (after 8.2.4520)

Problem: Build failure without the +diff feature. (John Marriott)
Solution: Define filler+lines if not declaring it.
Files: src/drawline.c

Patch 8.2.4522 (after 8.2.4492)

Problem: GUI test fails with Motif. (Dominique Pellé)
Solution: Remove using an invalid value for **'guifontset'**.
Files: src/testdir/test_gui.vim

Patch 8.2.4523

Problem: When gvim is started maximized the **'window'** option isn't set properly. (Christian J. Robinson)
Solution: Check if **'windows'** was already set or not. (Ken Takata, closes #9904)
Files: src/term.c

Patch 8.2.4524

Problem: MS-Windows: cannot build with some sodium libraries.
Solution: Make the DLL name configurable. Add build instructions. (Ken Takata, closes #9905)
Files: src/INSTALLpc.txt, src/Make_cyg_ming.mak, src/Make_mvc.mak, src/crypt.c

Patch 8.2.4525

Problem: Some GUI tests don't work on Athena.
Solution: Skip tests that won't work. (Yegappan Lakshmanan, closes #9902)
Files: src/testdir/test_gui.vim

Patch 8.2.4526

Problem: Vim9: cannot set variables to a null value.
Solution: Add null_list, null_job, etc.
Files: runtime/doc/vim9.txt, src/eval.c, src/proto/eval.pro, src/vim9expr.c, src/vim9script.c, src/vim9instr.c, src/vim9compile.c, src/vim9execute.c, src/vim9.h, src/vim9type.c, src/evalvars.c, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_disassemble.vim, src/testdir/test_vim9_func.vim, src/testdir/test_expr.vim

Patch 8.2.4527

Problem: The Athena GUI is old and does not work well.
Solution: Remove the Athena GUI from configure to find out who still wants support for this GUI.
Files: src/configure.ac, src/auto/configure, src/Makefile

Patch 8.2.4528

Problem: Crash when using null_function for a partial.
Solution: Don't call fname_trans_sid() with NULL. (closes #9908)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.4529

Problem: Vim9: comparing partial with function fails.
Solution: Support this comparison. Avoid a crash. (closes #9909)
Add more test cases.
Files: src/vim9instr.c, src/userfunc.c, src/vim9type.c, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vimscript.vim

Patch 8.2.4530

Problem: Making comparison with null work changes legacy behavior.

Solution: Only use the better comparison in Vim9 script. (closes #9910)
Files: src/typval.c, src/testdir/test_expr.vim

Patch 8.2.4531

Problem: LGTM warnings for condition always true and buffer size too small.
Solution: Remove the useless condition. Make the buffer larger. (Goc Dundar, closes #9914)
Files: src/charset.c, src/term.c

Patch 8.2.4532

Problem: Suspending with **CTRL-Z** does not work on OpenBSD.
Solution: Adjust #ifdef for SIGTSTP. (Stuart Henderson, closes #9912)
Files: src/os_unix.c

Patch 8.2.4533

Problem: Vim9: no test that after assigning null the type is still checked.
Solution: Add a test.
Files: src/testdir/test_vim9_assign.vim

Patch 8.2.4534

Problem: Vim9: "is" operator with empty string and null returns true.
Solution: Consider empty string and null to be different for "is".
Files: src/typval.c, src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4535

Problem: Filename modifier ":8" removes the filename.
Solution: Use strncpy() instead of vim_strncpy(). (Christian Brabandt, closes #9918, closes #8600)
Files: src/filepath.c, src/testdir/test_shortpathname.vim

Patch 8.2.4536 (after 8.2.4534)

Problem: Debugger test fails when breaking on expression.
Solution: Compare strings with "==" instead of "is".
Files: src/debugger.c

Patch 8.2.4537

Problem: Output from linter and language server shows up in git.
Solution: Add patterns to .gitignore. (Goc Dundar, closes #9925)
Files: .gitignore

Patch 8.2.4538

Problem: The find_tags_in_file() function is too long.
Solution: Refactor into smaller functions. (Yegappan Lakshmanan, closes #9920)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.2.4539

Problem: When comparing special v:none and v:null are handled the same when compiling.
Solution: Pass more information so that v:none can be handled differently at compile time. (issue #9923)
Files: src/vim9instr.c, src/vim9compile.c, src/globals.h, src/testdir/test_vim9_expr.vim

Patch 8.2.4540

Problem: Line number for error is off by one.
Solution: Remember the line number of the comparison. (closes #9923)
Files: src/eval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4541

Problem: Crash in debugger when a variable is not available in the current block.
Solution: Check for a NULL name. (closes #9926)
Files: src/vim9execute.c, src/testdir/test_debugger.vim

Patch 8.2.4542

Problem: Vim9: "break" inside try/catch not handled correctly.
Solution: First jump to :endtry. (closes #9927)
Files: src/vim9cmds.c, src/vim9.h, src/testdir/test_vim9_script.vim

Patch 8.2.4543

Problem: Coverity warning for refactored tag search code.
Solution: Avoid the warnings. Update comments. Add one more test case. (Yegappan Lakshmanan, closes #9928)
Files: src/tag.c, src/testdir/test_tagjump.vim

Patch 8.2.4544

Problem: Coverity warnings for not using returned value.
Solution: Assign to vim_ignored.
Files: src/tag.c

Patch 8.2.4545

Problem: MS-Windows: the installed icon is low resolution.
Solution: Use a better icon. Install vim.ico. (Christian Brabandt, closes #9931, closes #9930)
Files: Filelist, nsis/gvim.nsi, src/vim.ico, runtime/bitmaps/vim.ico

Patch 8.2.4546

Problem: Duplicate #undef.
Solution: Remove one #undef. (closes #9932)
Files: src/regexp_nfa.c

Patch 8.2.4547

Problem: The neXTaw GUI is old and does not work well.
Solution: Remove the neXTaw GUI from configure to find out who still wants support for this GUI.
Files: src/configure.ac, src/auto/configure, src/Makefile

Patch 8.2.4548

Problem: Script-local function is deleted when used in a funcref.
Solution: Do not consider a function starting with "<SNR>" reference counted. (closes #9916, closes #9820)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.4549

Problem: Cannot build with Motif and editres. (Tony Mechelynck)
Solution: Fix configure mistake.
Files: src/configure.ac, src/auto/configure

Patch 8.2.4550

Problem: Motif: cannot set the color of the scrollbar thumb.
Solution: Remove #ifdef.
Files: src/gui_motif.c

Patch 8.2.4551

Problem: When mapping <Esc> terminal codes are not recognized.
Solution: Specifically recognize a mapping with just <Esc> and check for terminal codes even though there is no partial mapping.
(closes #9903)
Files: src/getchar.c, src/testdir/test_termcodes.vim

Patch 8.2.4552

Problem: In a :def function "put = expr" does not work.
Solution: Skip over white space. (closes #9936)
Files: src/vim9cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4553

Problem: Linear tag search is a bit slow.
Solution: Remove a vim_ftell() call. (Yegappan Lakshmanan, closes #9937)
Files: src/tag.c, src/testdir/test_taglist.vim

Patch 8.2.4554

Problem: Vim9: using null values not sufficiently tested.
Solution: Add more tests. Fix uncovered problem.
Files: src/vim9type.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.4555

Problem: getmousepos() returns the wrong column. (Ernie Rael)
Solution: Limit to the text size, not the number of bytes.
Files: src/mouse.c, src/testdir/test_functions.vim

Patch 8.2.4556

Problem: Test fails without the +job or +channel feature. (Dominique Pellé)
Solution: Adjust #ifdefs. Pass on skip flag. (closes #9942)
Files: src/eval.c, src/vim9compile.c

Patch 8.2.4557

Problem: Confusing comment about '**cursorlineopt**'.
Solution: Adjust comment. (closes #9939) Add parenthesis around logical OR.
Files: src/drawline.c

Patch 8.2.4558

Problem: Motif: using default colors does not work as expected.
Solution: Do not try to store the default colors, use the resources.
(closes #9933)
Files: src/gui_motif.c, src/gui.h

Patch 8.2.4559 (after 8.24555)

Problem: getmousepos() returns the screen column. (Ernie Rael)
Solution: Return the text column, as documented.

Files: src/mouse.c, src/testdir/test_functions.vim

Patch 8.2.4560

Problem: Suspending with **CTRL-Z** does not work on DragonFlyBSD.

Solution: Adjust #ifdef. (Ozaki Kiichi, closes #9943)

Files: src/os_unix.c

Patch 8.2.4561

Problem: Build failure with some combination of features. (John Marriott)

Solution: Adjust #ifdef.

Files: src/mouse.c

Patch 8.2.4562

Problem: Linear tag search is not optimal.

Solution: Improve linear tag search performance. (Yegappan Lakshmanan, closes #9944)

Files: src/tag.c

Patch 8.2.4563

Problem: "z=" in Visual mode may go beyond the end of the line.

Solution: Adjust "badlen".

Files: src/spellsuggest.c, src/testdir/test_spell.vim

Patch 8.2.4564

Problem: Running test leaves file behind. (Dominique Pellé)

Solution: Run the profiling in a separate Vim instance. (closes #9952)

Files: src/testdir/test_vim9_script.vim

Patch 8.2.4565

Problem: No command line completion for :breakadd and :breakdel.

Solution: Add completion for :breakadd and :breakdel. (Yegappan Lakshmanan, closes #9950)

Files: runtime/doc/builtin.txt, src/cmdexpand.c, src/spellsuggest.c, src/usercmd.c, src/vim.h, src/testdir/test_cmdline.vim, src/testdir/test_writefile.vim

Patch 8.2.4566

Problem: Check for existing buffer in session file does not work for files in the home directory.

Solution: Use fnamemodify(). (James Cherti, closes #9945) Add a test.

Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4567

Problem: Bracketed paste doesn't work well in Visual linewise mode.

Solution: Handle linewise Visual mode differently. (closes #9947)

Files: src/normal.c, src/testdir/test_paste.vim

Patch 8.2.4568

Problem: getmousepos() does not compute the column below the last line.

Solution: Also compute the column when the mouse is below the last line. (Sean Dewar, closes #9946)

Files: src/mouse.c, src/testdir/test_functions.vim

Patch 8.2.4569

Problem: Coverity warning for not using a return value.
Solution: Add "(void)".
Files: src/popupwin.c

Patch 8.2.4570

Problem: No command line completion for :profile and :profdel.
Solution: Implement completion. (Yegappan Lakshmanan, closes #9955)
Files: src/cmdexpand.c, src/profiler.c, src/testdir/test_cmdline.vim,
src/testdir/test_profile.vim

Patch 8.2.4571

Problem: Not all gdb files are recognized.
Solution: Add a few more patterns for gdb. (Jade Lovelace, closes #9956)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4572

Problem: Vim9: return type "any" is sometimes changed to first returned type. (Virginia Senioria)
Solution: Do not change the return type if declared as "any". (closes #9949)
Files: src/vim9cmds.c, src/testdir/test_vim9_func.vim

Patch 8.2.4573

Problem: A nested function (closure) is compiled for debugging without context.
Solution: Check if a nested function is marked for debugging before compiling it. Give an error when trying to compile a closure without its context. (closes #9951)
Files: src/vim9compile.c, src/vim9execute.c, src/proto/vim9execute.pro,
src/vim9expr.c, src/errors.h

Patch 8.2.4574

Problem: Vim9: test for profiling fails.
Solution: Mark function for profiling earlier to avoid E1271.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.4575

Problem: Vim9: test for profiling still fails.
Solution: Update flags for profiling and breakpoints when obtaining the compile type. Do not set the FC_CLOSURE flag for a toplevel function.
Files: src/vim.h, src/vim9compile.c, src/proto/vim9compile.pro,
src/eval.c, src/vim9execute.c, src/vim9expr.c, src/vim9instr.c,
src/vim9.h

Patch 8.2.4576

Problem: Vim9: error for comparing with null can be annoying.
Solution: Allow comparing anything with null. (closes #9948)
Files: src/vim9instr.c, src/typval.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4577

Problem: Message test is flaky. (Elimar Riesebieter)
Solution: Trigger the autocommand event only after startup is finished.
Files: src/testdir/test_messages.vim

Patch 8.2.4578

Problem: No warning when an autoload script for completion function has an error.
Solution: Do not ignore errors when a function name is given with a dot or '#' character. (closes #9958)
Files: src/eval.c, src/testdir/test_cmdline.vim

Patch 8.2.4579

Problem: Cannot use page-up and page-down in the command line completion popup menu.
Solution: Check for to page-up and page-down keys. (Yegappan Lakshmanan, closes #9960)
Files: src/cmdexpand.c, src/ex_getln.c, src/spellsuggest.c, src/vim.h, src/testdir/test_cmdline.vim, src/testdir/dumps/Test_wildmenu_pum_42.dump, src/testdir/dumps/Test_wildmenu_pum_43.dump, src/testdir/dumps/Test_wildmenu_pum_44.dump, src/testdir/dumps/Test_wildmenu_pum_45.dump, src/testdir/dumps/Test_wildmenu_pum_46.dump, src/testdir/dumps/Test_wildmenu_pum_47.dump, src/testdir/dumps/Test_wildmenu_pum_48.dump, src/testdir/dumps/Test_wildmenu_pum_49.dump, src/testdir/dumps/Test_wildmenu_pum_50.dump

Patch 8.2.4580

Problem: Vim9: incorrect error for shadowing variable.
Solution: Do not pass the context when compiling a referenced function.
Files: src/vim9expr.c, src/testdir/test_vim9_func.vim

Patch 8.2.4581

Problem: Null types not fully tested.
Solution: Add some more tests using null types.
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.4582

Problem: Useless code handling a type declaration.
Solution: Remove the code and give an error.
Files: src/eval.c, src/errors.h, src/testdir/test_vim9_script.vim, src/testdir/dumps/Test_misplaced_type.dump

Patch 8.2.4583 (after 8.2.4582)

Problem: Screendump test fails.
Solution: Check that making a screendump is possible.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.4584 (after 8.2.4578)

Problem: Error for using autoload function in custom completion.
Solution: Do not check for errors when using an autoload function. (closes #9962)
Files: src/eval.c, src/testdir/test_cmdline.vim

Patch 8.2.4585

Problem: Cannot use keypad page-up/down for completion menu.
Solution: Recognize the keypad keys. (Yegappan Lakshmanan, closes #9963)

Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.4586

Problem: Vim9: no error for using lower case name for "func" argument.
(Ernie Rael)

Solution: Check the name as soon as the type is known.

Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.4587

Problem: Vim9: double free after unpacking a list.

Solution: Make a copy of the value instead of moving it. (closes #9968)

Files: src/vim9execute.c, src/testdir/test_vim9_script.vim

Patch 8.2.4588

Problem: Mapping with key code after other matching mapping does not work.

Solution: Change ">" to ">=". (closes #9903)

Files: src/getchar.c, src/testdir/test_termcodes.vim

Patch 8.2.4589

Problem: Cannot index the g: dictionary.

Solution: Recognize using "g:[key]". (closes #9969)

Files: src/ex_docmd.c, src/eval.c, src/vim9compile.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.4590

Problem: Vim9: range type check has wrong offset.

Solution: Adjust offset for CHECKTYPE. Remove other type check.

Files: src/vim9compile.c, src/vim9execute.c,
src/testdir/test_vim9_assign.vim

Patch 8.2.4591

Problem: Cursor line not updated when a callback moves the cursor.

Solution: Check if the cursor moved. (closes #9970)

Files: src/main.c, src/drawscreen.c, src/proto/drawscreen.pro,
src/testdir/test_cursorline.vim,
src/testdir/dumps/Test_cursorline_callback_1.dump

Patch 8.2.4592

Problem: Search continues after giving E1204.

Solution: Return failure after giving E1204. (closes #9972)

Files: src/regexp_nfa.c

Patch 8.2.4593

Problem: Unnecessary call to redraw_later().

Solution: Remove the call to redraw_later() in op_yank(). (closes #9971)

Files: src/register.c

Patch 8.2.4594

Problem: Need to write script to a file to be able to source them.

Solution: Make ":source" use lines from the current buffer. (Yegappan
Lakshmanan et al., closes #9967)

Files: runtime/doc/repeat.txt, runtime/doc/todo.txt, src/alloc.c,
src/digraph.c, src/eval.c, src/ex_cmds.h, src/scriptfile.c,
src/proto/scriptfile.pro, src/vim9script.c,

src/testdir/test_source.vim

Patch 8.2.4595

Problem: X11: using --remote-wait may keep the CPU busy.
Solution: Set the timeout for select() on every call. (Jacopo Secchiero, closes #9973)
Files: src/if_xcmdsrv.c

Patch 8.2.4596

Problem: Installing tutor binary may fail.
Solution: Fix the dependency. (Sergei Trofimovich, closes #9978)
Files: src/Makefile

Patch 8.2.4597

Problem: LuaV_debug() not covered by tests.
Solution: Add a test. (Dominique Pellé, closes #9980)
Files: src/testdir/test_lua.vim

Patch 8.2.4598

Problem: Profile completion test sometimes fails.
Solution: Delete the .res file before running tests.
Files: src/testdir/runtest.vim

Patch 8.2.4599

Problem: GTK: get assertion errors when scrolling a split window.
Solution: Use GDK_IS_DRAWABLE() on the scrollbar window. (closes #9982)
Files: src/gui_gtk.c

Patch 8.2.4600

Problem: Vim9: not enough test coverage for executing :def function.
Solution: Add a few more tests. Fix inconsistencies.
Files: src/vim9execute.c, src/evalvars.c, src/proto/evalvars.pro, src/testdir/test_listdict.vim, src/testdir/test_vim9_assign.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.4601

Problem: Vim9: not enough test coverage for executing :def function.
Solution: Add a few more tests.
Files: src/testdir/test_vim9_script.vim, src/testdir/test_vim9_func.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.4602

Problem: Vim9: not enough test coverage for executing :def function.
Solution: Add a few more tests. Fix uncovered problem. Remove dead code.
Files: src/vim9execute.c, src/vim9.h, src/vim9instr.c, src/proto/vim9instr.pro, src/vim9compile.c, src/testdir/test_vim9_script.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.4603

Problem: Sourcing buffer lines is too complicated.
Solution: Simplify the code. Make it possible to source Vim9 script lines. (Yegappan Lakshmanan, closes #9974)
Files: runtime/doc/repeat.txt, src/ex_docmd.c, src/proto/scriptfile.pro, src/scriptfile.c, src/structs.h, src/testdir/test_source.vim

Patch 8.2.4604

Problem: Error for redefining a script item may be confusing.
Solution: Put quotes around the name.
Files: src/errors.h

Patch 8.2.4605

Problem: Error for arguments of remote_expr() even when the +clientserver feature is not included.
Solution: Move #ifdef.
Files: src/clientserver.c

Patch 8.2.4606 (after 8.2.4605)

Problem: Test fails because of changed error message.
Solution: Update the expected error message
Files: src/testdir/test_vim9_import.vim

Patch 8.2.4607

Problem: Sourcing buffer lines may lead to errors for conflicts.
Solution: Add the ++clear argument. (Yegappan Lakshmanan, closes #9991)
Files: runtime/doc/repeat.txt, src/scriptfile.c, src/vim9script.c, src/proto/vim9script.pro, src/testdir/test_source.vim

Patch 8.2.4608

Problem: getcompletion() does not work properly when 'wildoptions' contains "fuzzy".
Solution: Do not use addstar(). (Yegappan Lakshmanan, closes #9992, closes #9986)
Files: runtime/doc/builtin.txt, src/cmdexpand.c, src/testdir/test_cmdline.vim

Patch 8.2.4609

Problem: :unhide does not check for failing to close a window.
Solution: When closing a window fails continue with the next one. Do not try closing the autocmd window. (closes #9984)
Files: src/buffer.c, src/window.c, src/proto/window.pro, src/testdir/test_autocmd.vim

Patch 8.2.4610

Problem: Some conditions are always true.
Solution: Remove the useless conditions. (closes #9993)
Files: src/clientserver.c, src/drawline.c, src/drawscreen.c, src/ex_cmds.c, src/fileio.c, src/message.c, src/misc2.c, src/ops.c, src/sign.c, src/spell.c, src/vim9cmds.c, src/window.c

Patch 8.2.4611

Problem: Typos in tests; one lua line not covered by test.
Solution: Fix typos. Add test case. (Dominique Pellé, closes #9994)
Files: src/testdir/test_breakindent.vim, src/testdir/test_crypt.vim, src/testdir/test_cursorline.vim, src/testdir/test_digraph.vim, src/testdir/test_gui.vim, src/testdir/test_lua.vim, src/testdir/test_regexp_latin.vim, src/testdir/test_signals.vim, src/testdir/test_spell.vim, src/testdir/test_statusline.vim, src/testdir/test_vim9_disassemble.vim,

src/testdir/test_vim9_expr.vim, src/testdir/test_vimscript.vim

Patch 8.2.4612

Problem: Vim9: cannot use a recursive call in a nested function. (Sergey Vlasov)

Solution: Define the funcref before compiling the function. (closes #9989)

Files: src/vim9compile.c, src/vim9instr.c, src/proto/vim9instr.pro,
src/vim9expr.c, src/testdir/test_vim9_func.vim

Patch 8.2.4613

Problem: Return type of swapfile_unchanged() is wrong.

Solution: Use "int". (closes #10000 Yeah!)

Files: src/memline.c

Patch 8.2.4614

Problem: Redrawing too much when '**cursorline**' is set and jumping around.

Solution: Rely on win_update() to redraw the current and previous cursor line, do not mark lines as modified. (closes #9996)

Files: src/drawline.c, src/drawscreen.c, src/move.c, src/proto/move.pro,
src/option.c

Patch 8.2.4615

Problem: Mapping with escaped bar does not work in :def function. (Sergey Vlasov)

Solution: Do not remove the backslash. (closes #10002)

Files: src/ex_docmd.c, src/proto/ex_docmd.pro, src/syntax.c,
src/vim9cmds.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4616

Problem: Vim9: Declarations in a {} block of a user command do not use Vim9 rules if defined in a legacy script. (Yegappan Lakshmanan)

Solution: Pretend the script is Vim9 script.

Files: src/usercmd.c, src/testdir/test_usercommands.vim

Patch 8.2.4617

Problem: No completion for :scriptnames.

Solution: Implement :scriptnames completion. (Yegappan Lakshmanan, closes #10005)

Files: runtime/doc/builtin.txt, src/cmdexpand.c, src/ex_cmds.h,
src/scriptfile.c, src/usercmd.c, src/vim.h,
src/testdir/test_cmdline.vim, src/testdir/test_quickfix.vim

Patch 8.2.4618

Problem: Command line completion does not recognize single letter commands.

Solution: Use the condition from find_ex_command().

Files: src/ex_docmd.c

Patch 8.2.4619

Problem: Mapping is cancelled when mouse moves and popup is visible.

Solution: Only generate mouse moved events when a popup may use them. (closes #10004)

Files: src/gui.c, src/globals.h, src/popupwin.c

Patch 8.2.4620 (after 8.2.4618)

Problem: Two letter substitute commands don't work. (Yegappan Lakshmanan)
Solution: Invert condition.
Files: src/ex_docmd.c

Patch 8.2.4621

Problem: Crash when using the tabline right-click menu.
Solution: Use XtPointer for XmUserData. (closes #10009)
Files: src/gui_motif.c

Patch 8.2.4622

Problem: Vim9: Crash with :execute and :finish. (Sergey Vlasov)
Solution: Check for NULL. (closes #10011)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.4623

Problem: Coverity warns for using uninitialized field.
Solution: Initialize the field to zero.
Files: src/ex_docmd.c

Patch 8.2.4624

Problem: Old Coverity warning for resource leak.
Solution: Close the file if memory allocation fails.
Files: src/diff.c

Patch 8.2.4625

Problem: Old Coverity warning for resource leak.
Solution: Call FreeWild() if expanding matches did not fail.
Files: src/help.c

Patch 8.2.4626

Problem: Visual area not fully updated when removing sign in Visual mode while scrolling.
Solution: Adjust check for topline. (closes #10017)
Files: src/drawscreen.c, src/testdir/test_display.vim, src/testdir/dumps/Test_display_scroll_update_visual.dump

Patch 8.2.4627

Problem: flatten() does not use maxdepth correctly.
Solution: Use a recursive implementation. (closes #10020)
Files: src/list.c, src/testdir/test_flatten.vim

Patch 8.2.4628

Problem: Not enough testing for 2/3 letter substitute commands.
Solution: Add more tests. (Yegappan Lakshmanan, closes #10019)
Files: src/testdir/test_cmdline.vim, src/testdir/test_substitute.vim

Patch 8.2.4629

Problem: flattennew() makes a deep copy unnecessarily.
Solution: Use a shallow copy. (issue #10012)
Files: src/list.c

Patch 8.2.4630

Problem: **'cursorline'** not always updated with **'cursorlineopt'** is "screenline".

Solution: Call `check_redraw_cursorline()` more often. (closes #10013)
Files: `src/normal.c`, `src/edit.c`, `src/testdir/test_cursorline.vim`,
`src/testdir/dumps/Test_cursorline_screenline_1.dump`,
`src/testdir/dumps/Test_cursorline_screenline_2.dump`

Patch 8.2.4631

Problem: Crash when switching window in `BufWipeout` autocommand.
Solution: Put any buffer in the window to avoid it being NULL.
(closes #10024)
Files: `src/window.c`, `src/buffer.c`, `src/testdir/test_autocmd.vim`

Patch 8.2.4632

Problem: Using freed memory in `flatten()`.
Solution: Clear `typval` after recursing into list.
Files: `src/list.c`

Patch 8.2.4633

Problem: Visual range does not work before command modifiers.
Solution: Move Visual range to after command modifiers.
Files: `src/ex_docmd.c`, `src/testdir/test_source.vim`

Patch 8.2.4634

Problem: Vim9: cannot initialize a variable to `null_list`.
Solution: Give negative count to `NEWLIST`. (closes #10027)
Also fix inconsistencies in comparing with null values.
Files: `src/vim9instr.c`, `src/proto/vim9instr.pro`, `src/vim9.h`,
`src/vim9compile.c`, `src/vim9expr.c`, `src/vim9execute.c`,
`src/evalvars.c`, `src/typval.c`, `src/testdir/test_vim9_expr.vim`,
`src/testdir/test_vim9_builtin.vim`,
`src/testdir/test_vim9_disassemble.vim`

Patch 8.2.4635 (after 8.2.4634)

Problem: Tests using null list or dict fail.
Solution: Only use the new rules for Vim9 script.
Files: `src/evalvars.c`

Patch 8.2.4636 (after 8.2.4633)

Problem: Not using Visual range.
Solution: Put the command pointer back to the range.
Files: `src/ex_docmd.c`

Patch 8.2.4637

Problem: Warning for using uninitialized variable. (Tony Mechelynck)
Solution: Initialize it.
Files: `src/ex_docmd.c`

Patch 8.2.4638

Problem: Superfluous check if a redraw is needed for **'cursorline'**.
Solution: Remove `check_redraw_cursorline()`. (closes #10030, closes #10029)
Files: `src/drawscreen.c`, `src/proto/drawscreen.pro`, `src/edit.c`,
`src/main.c`, `src/normal.c`, `src/move.c`,
`src/testdir/dumps/Test_cursorcolumn_callback_1.dump`,
`src/testdir/dumps/Test_relativenumber_callback_1.dump`,
`src/testdir/test_highlight.vim`, `src/testdir/test_number.vim`

Patch 8.2.4639

Problem: Not sufficient parenthesis in preprocessor macros.
Solution: Add more parenthesis. (closes #10031)
Files: src/globals.h, src/gui.h, src/if_py_both.h, src/macros.h,
src/option.h, src/regexp.h, src/spell.h, src/structs.h, src/vim.h,
src/vim9.h

Patch 8.2.4640

Problem: Some boolean options use "long" instead of "int".
Solution: Adjust the type. (James McCoy, closes #10033)
Files: src/option.h

Patch 8.2.4641

Problem: May mark the wrong window for redrawing.
Solution: Use redraw_win_later(). (closes #10032)
Files: src/move.c

Patch 8.2.4642

Problem: Vim9: in :def function script var cannot be null.
Solution: Only initialize a script variable when not set to a null value.
(closes #10034)
Files: src/vim9execute.c, src/vim9type.c, src/globals.h, src/evalvars.c,
src/vim.h, src/vim9script.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4643

Problem: Vim9: variable may be locked unintentionally.
Solution: Clear "v_lock". (closes #10036)
Files: src/vim9execute.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4644

Problem: Redrawing too often when '**relativenumber**' is set.
Solution: Only redraw when the cursor line changed. (Lewis Russell,
closes #10040)
Files: src/change.c, src/drawscreen.c, src/structs.h

Patch 8.2.4645

Problem: '**shortmess**' changed when session does not store options.
Solution: Save and restore '**shortmess**' if needed. (James Cherti,
closes #10037)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4646

Problem: Using buffer line after it has been freed in old regexp engine.
Solution: After getting mark get the line again.
Files: src/regexp_bt.c, src/testdir/test_regexp_latin.vim

Patch 8.2.4647

Problem: "source" can read past end of copied line.
Solution: Add a terminating NUL.
Files: src/scriptfile.c, src/testdir/test_source.vim

Patch 8.2.4648

Problem: Handling LSP messages is a bit slow.

Solution: Included support for LSP messages. (Yegappan Lakshmanan, closes #10025)
Files: runtime/doc/channel.txt, src/channel.c, src/job.c, src/json.c, src/proto/json.pro, src/structs.h, src/testdir/test_channel.vim, src/testdir/test_channel_lsp.py

Patch 8.2.4649

Problem: Various formatting problems.
Solution: Improve the code formatting.
Files: src/mark.c, src/quickfix.c, src/regexp_nfa.c, src/register.c, src/testdir/test_filechanged.vim, src/gui_athena.c, src/gui_motif.c, src/os_unix.c

Patch 8.2.4650

Problem: "import autoload" only works with using **'runtimepath'**.
Solution: Also support a relative and absolute file name.
Files: runtime/doc/vim9.txt, src/structs.h, src/scriptfile.c, src/proto/scriptfile.pro, src/vim9script.c, src/vim9expr.c, src/vim9.h, src/vim9execute.c, src/vim9instr.c, src/proto/vim9instr.pro, src/vim.h, src/userfunc.c, src/proto/userfunc.pro, src/testdir/test_vim9_import.vim, src/testdir/test_vim9_disassemble.vim

Patch 8.2.4651 (after 8.2.4650)

Problem: Test fails because path differs.
Solution: Only compare the tail of the path.
Files: src/testdir/test_vim9_disassemble.vim

Patch 8.2.4652 (after 8.2.4650)

Problem: Leaking memory if assignment fails.
Solution: Clear assigned value on failure.
Files: src/vim9execute.c

Patch 8.2.4653

Problem: "import autoload" does not check the file name.
Solution: Give an error if the file is not readable. (closes #10049)
Files: src/filepath.c, src/proto/filepath.pro, src/errors.h, src/ex_cmds.c, src/ex_docmd.c, src/spellfile.c, src/testdir/test_vim9_import.vim

Patch 8.2.4654 (after 8.2.4653)

Problem: Missing changes for import check.
Solution: Add missing changes.
Files: src/vim9script.c

Patch 8.2.4655

Problem: Command line completion popup menu positioned wrong when using a terminal window.
Solution: Position the popup menu differently when editing the command line. (Yegappan Lakshmanan, closes #10050, closes #10035)
Files: src/popupmenu.c, src/testdir/test_cmdline.vim, src/testdir/test_terminal.vim, src/testdir/dumps/Test_wildmenu_pum_term_01.dump

Patch 8.2.4656

Problem: Vim9: can't use items from "import autoload" with autoload directory name.
Solution: Let sn_autoload_prefix overrule sn_import_autoload. (closes #10054)
Files: src/structs.h, src/vim9instr.c, src/vim9expr.c, src/vim9script.c, src/testdir/test_vim9_import.vim

Patch 8.2.4657

Problem: Errors for functions are sometimes hard to read.
Solution: Use printable_func_name() in more places.
Files: src/vim9execute.c, src/userfunc.c, src/proto/userfunc.pro, src/vim9expr.c, src/eval.c, src/vim9instr.c, src/vim9type.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4658

Problem: Org-mode files are not recognized.
Solution: Add patterns to recognize "org" files. (closes #10046)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4659

Problem: Invalid memory access when using printable function name.
Solution: Adjust copied name length.
Files: src/userfunc.c

Patch 8.2.4660

Problem: Cursorcolumn is sometimes not correct.
Solution: Recompute the cursor column when entering Insert mode and the cursor is on a character wider than a screen cell. (closes #10057)
Files: src/edit.c, src/testdir/test_highlight.vim, src/testdir/dumps/Test_cursorcolumn_insert_on_tab_1.dump, src/testdir/dumps/Test_cursorcolumn_insert_on_tab_2.dump

Patch 8.2.4661

Problem: Coverity warning for using uninitialized variable.
Solution: Initialize variable to NULL.
Files: src/vim9expr.c

Patch 8.2.4662

Problem: No error for using out of range list index.
Solution: Check list index at script level like in compiled function. (closes #10051)
Files: src/vim.h, src/evalvars.c, src/list.c, src/proto/list.pro, src/eval.c, src/vim9execute.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4663

Problem: Occasional crash when running the GUI tests.
Solution: Check that the line index is not too high. (closes #8681)
Files: src/screen.c

Patch 8.2.4664

Problem: Elvish files are not recognized.
Solution: Recognize .elv files. (Bruno Roque, closes #10058)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4665

Problem: Popup with "minwidth" and scrollbar not updated properly.
Solution: Adjust the computation if the window width. (closes #10061)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_previewpopup_4.dump,
src/testdir/dumps/Test_popupwin_previewpopup_5.dump,
src/testdir/dumps/Test_popupwin_previewpopup_7.dump,
src/testdir/dumps/Test_popupwin_previewpopup_8.dump,
src/testdir/dumps/Test_popupwin_previewpopup_9.dump,
src/testdir/dumps/Test_popupwin_previewpopup_10.dump,
src/testdir/dumps/Test_popupwin_drag_minwidth_1.dump,
src/testdir/dumps/Test_popupwin_drag_minwidth_2.dump,
src/testdir/dumps/Test_popupwin_drag_minwidth_3.dump

Patch 8.2.4666

Problem: Vim9: assignment not recognized in skipped block.
Solution: When skipping assume identifier exists. (closes #10059)
Files: src/vim9compile.c, src/proto/vim9compile.pro, src/vim9cmds.c,
src/testdir/test_vim9_cmd.vim, src/testdir/test_vim9_script.vim

Patch 8.2.4667

Problem: expandcmd() fails on an error.
Solution: On failure return the command unmodified. (Yegappan Lakshmanan,
closes #10063)
Files: runtime/doc/builtin.txt, src/evalfunc.c,
src/testdir/test_expand.vim

Patch 8.2.4668

Problem: Buffer allocation failures insufficiently tested.
Solution: Add tests for memory allocation failures. (Yegappan Lakshmanan,
closes #10064)
Files: src/alloc.h, src/buffer.c, src/popupwin.c, src/window.c,
src/testdir/test_buffer.vim, src/testdir/test_swap.vim

Patch 8.2.4669

Problem: In compiled code len('string') is not inlined.
Solution: Compute the length at compile time if possible. (closes #10065)
Files: src/evalfunc.c, src/proto/evalfunc.pro, src/vim9expr.c,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4670

Problem: Memory allocation failures for new tab page not tested.
Solution: Add tests with failing memory allocation. (Yegappan Lakshmanan,
closes #10067)
Files: src/alloc.h, src/blob.c, src/buffer.c, src/window.c,
src/testdir/test_blob.vim, src/testdir/test_buffer.vim,
src/testdir/test_tabpage.vim, src/testdir/test_window_cmd.vim

Patch 8.2.4671

Problem: **'wildignorecase'** is sometimes not used for glob().
Solution: Also use **'wildignorecase'** when there are no wildcards.
(closes #10066, closes #8350)
Files: src/filepath.c, src/testdir/test_functions.vim

Patch 8.2.4672

Problem: Using :normal with Ex mode may make :substitute hang.
Solution: When getting an empty line behave like 'q' was typed.
(closes #10070)
Files: src/ex_cmds.c, src/testdir/test_normal.vim

Patch 8.2.4673

Problem: Redrawing a vertically split window is slow when using **CTRL-F** and **CTRL-B**.
Solution: When deciding on USE_REDRAW bail out if scrolling more than three lines. (issue #8002)
Files: src/screen.c

Patch 8.2.4674

Problem: Cannot force getting MouseMove events.
Solution: Add the '**mousemoveevent**' option with implementation for the GUI.
(Ernie Rael, closes #10044)
Files: runtime/doc/gui.txt, runtime/doc/options.txt,
runtime/doc/testing.txt, src/gui.c, src/option.h,
src/optiondefs.h, src/testing.c, src/testdir/test_gui.vim

Patch 8.2.4675

Problem: No error for missing expression after :elseif. (Ernie Rael)
Solution: Check for missing expression. (closes #10068)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.4676 (after 8.2.4675)

Problem: Test fails with different error.
Solution: Add argument to :elseif.
Files: src/testdir/test_vimscript.vim

Patch 8.2.4677

Problem: The Athena GUI support is outdated.
Solution: Remove the Athena GUI code.
Files: Filelist, src/Makefile, src/proto.h, src/clipboard.c,
src/gui_athena.c, src/proto/gui_athena.pro, src/gui_at_sb.c,
src/gui_at_sb.h, src/gui_at_fs.c, src/gui_motif.c, src/evalfunc.c,
src/gui.c, src/gui_beval.c, src/gui_x11.c, src/if_mzsch.c,
src/main.c, src/menu.c, src/mouse.c, src/version.c, src/feature.h,
src/gui.h, src/structs.h, src/vim.h, src/testdir/gui_init.vim,
src/testdir/setup_gui.vim, src/testdir/test_clientserver.vim,
src/testdir/test_edit.vim, src/testdir/test_gui.vim,
src/testdir/test_highlight.vim, src/testdir/test_quotestar.vim,
src/testdir/test_startup.vim, runtime/doc/gui.txt,
runtime/doc/gui_x11.txt

Patch 8.2.4678

Problem: Vim9: not all code is tested.
Solution: Add a few more tests.
Files: src/vim9execute.c, src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_import.vim, src/testdir/test_vim9_cmd.vim

Patch 8.2.4679

Problem: Cannot have expandcmd() give an error message for mistakes.
Solution: Add an optional argument to give errors. Fix memory leak when expanding files fails. (Yegappan Lakshmanan, closes #10071)
Files: runtime/doc/builtin.txt, src/evalfunc.c, src/filepath.c, src/testdir/test_expand.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.4680

Problem: Build failure without +postscript.
Solution: Use another error message.
Files: src/vim9execute.c, src/testdir/test_vim9_import.vim

Patch 8.2.4681

Problem: Build fails with a combination of features.
Solution: Remove #ifdef for alloc_clear_id(). (John Marriott)
Files: src/alloc.c

Patch 8.2.4682

Problem: Vim9: can use :unlockvar for const variable. (Ernie Rael)
Solution: Check whether the variable is a const.
Files: src/ex_docmd.c, src/evalvars.c, src/vim9script.c, src/proto/vim9script.pro, src/eval.c, src/userfunc.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4683

Problem: Verbose check with dict_find() to see if a key is present.
Solution: Add dict_has_key(). (Yegappan Lakshmanan, closes #10074)
Files: src/channel.c, src/dict.c, src/evalwindow.c, src/filepath.c, src/highlight.c, src/json.c, src/match.c, src/popupwin.c, src/proto/dict.pro, src/quickfix.c, src/search.c, src/sign.c, src/tag.c, src/terminal.c, src/testing.c, src/textprop.c, src/time.c

Patch 8.2.4684

Problem: Cannot open a channel on a Unix domain socket.
Solution: Add Unix domain socket support. (closes #10062)
Files: runtime/doc/channel.txt, src/channel.c, src/testdir/check.vim, src/testdir/shared.vim, src/testdir/test_channel.py, src/testdir/test_channel.vim, src/testdir/test_channel_unix.py, src/testdir/test_cmdline.vim

Patch 8.2.4685

Problem: When a swap file is found for a popup there is no dialog and the buffer is loaded anyway.
Solution: Silently load the buffer read-only. (closes #10073)
Files: runtime/doc/popup.txt, src/memline.c, src/popupwin.c, src/vim.h, src/buffer.c, src/testdir/test_popupwin.vim

Patch 8.2.4686

Problem: Configure doesn't find the Motif library with Cygwin.
Solution: Check for libXm.dll.a. (Kelvin Lee, closes #10077)
Files: src/configure.ac, src/auto/configure

Patch 8.2.4687

Problem: "vimgrep /%v/ *" may cause a crash.

Solution: When compiling the pattern with the old engine fails, restore the regprog of the new engine instead of leaving it NULL.
(closes #10079)

Files: src/regexp.c

Patch 8.2.4688

Problem: New regexp engine does not give an error for "%v".

Solution: Check for a value argument. (issue #10079)

Files: src/regexp_nfa.c, src/errors.h, src/regexp_bt.c,
src/testdir/test_regexp_latin.vim

Patch 8.2.4689

Problem: Using <Cmd> in a mapping does not work for mouse keys in Insert mode. (Sergey Vlasov)

Solution: When reading the <Cmd> argument do not use the stuff buffer.
(closes #10080)

Files: src/getchar.c

Patch 8.2.4690

Problem: Channel tests fail on MS-Windows.

Solution: Check if the AF_UNIX attribute exists. (closes #10083)

Files: src/testdir/test_channel.py, src/testdir/test_channel_unix.py

Patch 8.2.4691 (after 8.2.4689)

Problem: Solution for <Cmd> in a mapping causes trouble.

Solution: Use another solution: put back **CTRL-O** after reading the <Cmd> sequence.

Files: src/getchar.c

Patch 8.2.4692

Problem: No test for what 8.2.4691 fixes.

Solution: Add a test. Use a more generic solution. (closes #10090)

Files: src/getchar.c, src/mouse.c, src/testdir/test_mapping.vim

Patch 8.2.4693 (after 8.2.4688)

Problem: new regexp does not accept pattern "%>0v".

Solution: Do accept digit zero.

Files: src/regexp_bt.c, src/regexp_nfa.c,
src/testdir/test_regexp_latin.vim

Patch 8.2.4694

Problem: Avoidance of #elif causes more preproc nesting.

Solution: Use #elif where it is useful. (Ozaki Kiichi, closes #10081)

Files: src/option.c, src/optiondefs.h, src/optionstr.c, src/version.c

Patch 8.2.4695

Problem: JSON encoding could be faster.

Solution: Optimize encoding JSON strings. (closes #10086)

Files: src/json.c, src/testdir/test_json.vim

Patch 8.2.4696

Problem: delete() with "rf" argument does not report a failure.

Solution: Return -1 if the directory could not be removed. (closes #10078)

Files: src/fileio.c, src/testdir/test_functions.vim

Patch 8.2.4697

Problem: Vim9: crash when adding a duplicate key to a dictionary.
Solution: Clear the stack item when it has been moved into the dictionary.
(closes #10087)
Files: src/vim9execute.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4698

Problem: Vim9: script variable has no flag that it was set.
Solution: Add a flag that it was set, to avoid giving it a value when used.
(closes #10088)
Files: src/structs.h, src/vim9script.c, src/vim9execute.c,
src/evalvars.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4699

Problem: Hard to reproduce hang when reading from a channel.
Solution: Check for readahead before starting to wait. (closes #10093,
closes #7781, closes #6364)
Files: src/channel.c

Patch 8.2.4700

Problem: Buffer remains active if a WinClosed event throws an exception.
Solution: Ignore aborting() when closing the buffer. (closes #10097)
Files: src/window.c, src/testdir/test_autocmd.vim

Patch 8.2.4701

Problem: Kuka Robot Language files not recognized.
Solution: Recognize *.src and *.dat files. (Patrick Meiser-Knosowski,
closes #10096)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim,
runtime/autoload/dist/ft.vim

Patch 8.2.4702

Problem: C++ scope labels are hard-coded.
Solution: Add '**cinscopedcls**' to define the labels. (Rom Praschan,
closes #10109)
Files: runtime/doc/indent.txt, runtime/doc/options.txt,
runtime/doc/quickref.txt, runtime/optwin.vim, src/buffer.c,
src/cindent.c, src/option.c, src/option.h, src/optiondefs.h,
src/optionstr.c, src/structs.h, src/testdir/test_cindent.vim

Patch 8.2.4703 (after 8.2.4702)

Problem: Memory leak in handling '**cinscopedcls**'.
Solution: Free the memory before returning.
Files: src/cindent.c

Patch 8.2.4704

Problem: Using "else" after return or break increases indent.
Solution: Remove "else" and reduce indent. (Goc Dundar, closes #10099)
Files: src/fileio.c, src/memline.c, src/option.c, src/syntax.c

Patch 8.2.4705

Problem: reg_executing may not be cleared.

Solution: Reset `reg_executing` later. (closes #10111, closes #10110)
Files: `src/ex_docmd.c`, `src/getchar.c`, `src/globals.h`, `src/structs.h`,
`src/testdir/test_registers.vim`

Patch 8.2.4706

Problem: Buffer remains active if a `WinClosed` event throws an exception when there are multiple tabpages.

Solution: Ignore `aborting()` when closing the buffer. (closes #10101)
Files: `src/window.c`, `src/testdir/test_autocmd.vim`

Patch 8.2.4707

Problem: Redrawing could be a bit more efficient.

Solution: Optimize redrawing. (closes #10105)
Files: `src/change.c`, `src/edit.c`, `src/testdir/test_highlight.vim`,
`src/testdir/dumps/Test_cursorcolumn_insert_on_tab_3.dump`

Patch 8.2.4708

Problem: PHP test files are not recognized.

Solution: Add the `*.phpt` pattern. (Julien Voisin, closes #10112)
Files: `runtime/filetype.vim`, `src/testdir/test_filetype.vim`

Patch 8.2.4709

Problem: After `:redraw` the statusline highlight might be used.

Solution: Clear the screen attribute after redrawing the screen.
(closes #10108)

Files: `src/ex_docmd.c`

Patch 8.2.4710

Problem: Smart indenting does not work after completion.

Solution: Set `"can_si"`. (Christian Brabandt, closes #10113, closes #558)
Files: `src/edit.c`, `src/testdir/test_ins_complete.vim`

Patch 8.2.4711

Problem: When `'insertmode'` is set `:edit` from `<Cmd>` mapping misbehaves.

Solution: Don't set `"need_start_insertmode"` when already in Insert mode.
(closes #10116)

Files: `src/ex_cmds.c`, `src/testdir/test_edit.vim`

Patch 8.2.4712

Problem: Only get profiling information after exiting.

Solution: Add `"profile dump"` and `"profile stop"`. (Marco Hinz, Yegappan Lakshmanan, closes #10107)

Files: `runtime/doc/repeat.txt`, `src/profiler.c`,
`src/testdir/test_profile.vim`

Patch 8.2.4713

Problem: Plugins cannot track text scrolling.

Solution: Add the `WinScrolled` event. (closes #10102)

Files: `runtime/doc/autocmd.txt`, `src/autocmd.c`, `src/proto/autocmd.pro`,
`src/edit.c`, `src/gui.c`, `src/main.c`, `src/structs.h`, `src/vim.h`,
`src/window.c`, `src/proto/window.pro`, `src/testdir/test_autocmd.vim`

Patch 8.2.4714

Problem: Using `g:filetype_dat` and `g:filetype_src` not tested.

Solution: Add a test. (Patrick Meiser-Knosowski, closes #10117)
Files: src/testdir/test_filetype.vim

Patch 8.2.4715

Problem: Vagrantfile not recognized.
Solution: Recognize Vagrantfile as ruby. (Julien Voisin, closes #10119)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4716

Problem: Memory allocation failure not tested when defining a function.
Solution: Add a test. (Yegappan Lakshmanan, closes #10127)
Files: src/alloc.c, src/alloc.h, src/proto/alloc.pro, src/userfunc.c,
src/testdir/test_user_func.vim, src/testdir/test_vim9_func.vim

Patch 8.2.4717

Problem: For TextYankPost v:event does not contain information about the operation being inclusive or not.
Solution: Add "inclusive" to v:event. (Justin M. Keyes, Yegappan Lakshmanan, closes #10125)
Files: runtime/doc/autocmd.txt, src/register.c,
src/testdir/test_autocmd.vim

Patch 8.2.4718

Problem: @@@ in the last line sometimes drawn in the wrong place.
Solution: Make sure the column is valid. (closes #10130)
Files: src/drawscreen.c, src/screen.c, src/testdir/test_display.vim
src/testdir/dumps/Test_display_lastline_1.dump,
src/testdir/dumps/Test_display_lastline_2.dump,
src/testdir/dumps/Test_display_lastline_3.dump,
src/testdir/dumps/Test_display_lastline_4.dump

Patch 8.2.4719

Problem: ">" marker sometimes not displayed in the jumplist.
Solution: If the buffer no longer exists show "-invalid-". (Christian Brabandt, closes #10131, closes #10100)
Files: runtime/doc/motion.txt, src/mark.c, src/testdir/Make_all.mak,
src/testdir/test_alot.vim, src/testdir/test_jumplist.vim,
src/testdir/test_jumps.vim

Patch 8.2.4720

Problem: ABB Rapid files are not recognized properly.
Solution: Add checks for ABB Rapid files. (Patrick Meiser-Knosowski, closes #10104)
Files: runtime/autoload/dist/ft.vim, runtime/doc/filetype.txt,
runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4721

Problem: Cooklang files are not recognized.
Solution: recognize *.cook files. (Goc Dundar, closes #10120)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4722

Problem: When a recording is ended with a mapped key that key is also recorded.

Solution: Remember the previous `last_recorded_len`. (closes #10122)
Files: `src/getchar.c`, `src/testdir/test_registers.vim`

Patch 8.2.4723

Problem: The `ModeChanged` autocmd event is inefficient.
Solution: Avoid allocating memory. (closes #10134) Rename `trigger_modechanged()` to `may_trigger_modechanged()`.
Files: `src/misc1.c`, `src/proto/misc1.pro`, `src/edit.c`, `src/ex_docmd.c`, `src/ex_getln.c`, `src/insexpand.c`, `src/normal.c`, `src/terminal.c`, `src/autocmd.c`

Patch 8.2.4724

Problem: Current instance of last search pattern not easily spotted.
Solution: Add `CurSearch` highlighting. (closes #10133)
Files: `runtime/doc/options.txt`, `runtime/doc/syntax.txt`, `src/highlight.c`, `src/match.c`, `src/normal.c`, `src/optiondefs.h`, `src/structs.h`, `src/vim.h`, `src/normal.c`, `src/testdir/test_search.vim`, `src/testdir/dumps/Test_hlsearch_cursearch_multiple_line.dump`, `src/testdir/dumps/Test_hlsearch_cursearch_single_line_1.dump`, `src/testdir/dumps/Test_hlsearch_cursearch_single_line_2.dump`, `src/testdir/dumps/Test_hlsearch_cursearch_single_line_3.dump`

Patch 8.2.4725 (after 8.2.4724)

Problem: Unused variable in tiny build.
Solution: Add `#ifdef`.
Files: `src/normal.c`

Patch 8.2.4726

Problem: Cannot use `expand()` to get the script name.
Solution: Support `expand('<script>')`. (closes #10121)
Files: `runtime/doc/cmdline.txt`, `src/errors.h`, `src/ex_docmd.c`, `src/scriptfile.c`, `src/vim.h`, `src/testdir/test_expand.vim`

Patch 8.2.4727

Problem: Unused code.
Solution: Remove code and add `#ifdefs`. (Dominique Pellé, closes #10136)
Files: `runtime/doc/editing.txt`, `runtime/doc/eval.txt`, `runtime/doc/vim9.txt`, `src/errors.h`, `src/option.c`, `src/search.c`, `src/proto/search.pro`

Patch 8.2.4728

Problem: No test that `v:event` cannot be modified.
Solution: Add a test. (closes #10139)
Files: `src/testdir/test_autocmd.vim`

Patch 8.2.4729

Problem: `HEEx` and `Surface` templates do not need a separate filetype.
Solution: Use `Elixir` for the similar filetypes. (Aaron Tinio, closes #10124)
Files: `runtime/filetype.vim`, `src/testdir/test_filetype.vim`

Patch 8.2.4730

Problem: MS-Windows GUI: cannot use `CTRL-/`.
Solution: Handle the `WM_KEYUP` event. (Yasuhiro Matsumoto, closes #10141)
Files: `src/gui_w32.c`

Patch 8.2.4731

Problem: The changelist index is not remembered per buffer.
Solution: Keep the changelist index per window and buffer. (closes #10135, closes #2173)
Files: src/buffer.c, src/evalfunc.c, src/structs.h, src/testdir/test_changelist.vim

Patch 8.2.4732

Problem: Duplicate code to free fuzzy matches.
Solution: Bring back fuzmatch_str_free().
Files: src/search.c, src/proto/search.pro, src/cmdexpand.c

Patch 8.2.4733 (after 8.2.4729)

Problem: HEEEx and Surface do need a separate filetype.
Solution: Revert 8.2.4729. (closes #10147)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4734

Problem: getcharpos() may change a mark position.
Solution: Copy the mark position. (closes #10148)
Files: src/eval.c, src/testdir/test_cursor_func.vim

Patch 8.2.4735

Problem: Quickfix tests can be a bit hard to read.
Solution: Use heredoc instead of strings and line continuation. (Yegappan Lakshmanan, closes #10145)
Files: src/testdir/test_quickfix.vim

Patch 8.2.4736

Problem: Build problem for Cygwin with Motif.
Solution: Undefine ControlMask. (Kelvin Lee, closes #10152)
Files: src/mbyte.c

Patch 8.2.4737

Problem: // in JavaScript string recognized as comment.
Solution: Only check for linecomment if 'cindent' is set. (closes #10151)
Files: src/change.c, src/testdir/test_textformat.vim

Patch 8.2.4738

Problem: Esc on commandline executes command instead of abandoning it.
Solution: Save and restore KeyTyped when removing the popup menu. (closes #10154)
Files: src/cmdexpand.c, src/testdir/test_cmdline.vim, src/testdir/dumps/Test_wildmenu_with_pum_foldexpr_1.dump, src/testdir/dumps/Test_wildmenu_with_pum_foldexpr_2.dump

Patch 8.2.4739

Problem: Accessing freed memory after WinScrolled autocmd event.
Solution: Check the window pointer is still valid. (closes #10156)
Remove the argument from may_trigger_winscrolled().
Files: src/window.c, src/proto/window.pro, src/edit.c, src/gui.c, src/main.c, src/testdir/test_autocmd.vim

Patch 8.2.4740

Problem: When expand() fails there is no error message.
Solution: When **'verbose'** is set give an error message.
Files: runtime/doc/builtin.txt, src/evalfunc.c,
src/testdir/test_expand.vim

Patch 8.2.4741 (after 8.2.4740)

Problem: Startup test fails.
Solution: Avoid an error for verbose expansion. Fix that the "@verbose"
command modifier doesn't work.
Files: runtime/syntax/syntax.vim, runtime/syntax/synload.vim,
src/structs.h, src/ex_docmd.c, src/testdir/test_excmd.vim

Patch 8.2.4742

Problem: There is no way to start logging very early in startup.
Solution: Add the --log argument. Include the date in the start message in
the log file. Avoid a duplicate message when forking. Log an
executed shell command.
Files: runtime/doc/starting.txt, runtime/doc/channel.txt,
src/main.c, src/channel.c, src/os_unix.c, src/os_win32.c,
src/testdir/test_startup.vim

Patch 8.2.4743

Problem: Clang 14 is available on CI.
Solution: Switch from clang 13 to 14. (closes #10157)
Files: .github/workflows/ci.yml

Patch 8.2.4744

Problem: A terminal window can't use the bell.
Solution: Add bell support for the terminal window. (closes #10178)
Files: runtime/doc/options.txt, src/gui_w32.c, src/option.h,
src/optionstr.c, src/terminal.c

Patch 8.2.4745 (after 8.2.4744)

Problem: Using wrong flag for using bell in the terminal.
Solution: Change to use BO_TERM.
Files: src/terminal.c, src/misc1.c

Patch 8.2.4746

Problem: Supercollider filetype not recognized.
Solution: Match file extensions and check file contents to detect
supercollider. (closes #10142)
Files: runtime/filetype.vim, runtime/autoload/dist/ft.vim,
src/testdir/test_filetype.vim

Patch 8.2.4747

Problem: No filetype override for .sys files.
Solution: Add g:filetype_sys. (Patrick Meiser-Knosowski, closes #10181)
Files: runtime/doc/filetype.txt, runtime/autoload/dist/ft.vim,
src/testdir/test_filetype.vim

Patch 8.2.4748

Problem: Cannot use an imported function in a mapping.
Solution: Recognize <SID>name.Func.

Files: runtime/doc/vim9.txt, src/term.c, src/vim9execute.c,
src/proto/vim9execute.pro, src/scriptfile.c,
src/testdir/test_vim9_import.vim

Patch 8.2.4749

Problem: `<script>` is not expanded in autocmd context.

Solution: Add the context to the pattern struct. (closes #10144)
Rename AutoPatCmd to AutoPatCmd_T.

Files: src/autocmd.c, src/proto/autocmd.pro, src/scriptfile.c,
src/structs.h, src/testdir/test_expand.vim

Patch 8.2.4750

Problem: Small pieces of dead code.

Solution: Remove the dead code. (Goc Dundar, closes #10190) Rename the
qftf_cb struct member to avoid confusion.

Files: src/ex_cmds.c, src/misc1.c, src/optionstr.c, src/quickfix.c

Patch 8.2.4751 (after 8.2.4748)

Problem: Mapping `<SID>name.Func` does not work for script in autoload
directory.

Solution: Use the # form for a script in the autoload directory.
(closes #10186)

Files: src/term.c, src/testdir/test_vim9_import.vim

Patch 8.2.4752

Problem: Wrong `'statusline'` value can cause illegal memory access.

Solution: Properly check the value. (closes #10192)

Files: src/optionstr.c, src/testdir/test_options.vim

Patch 8.2.4753

Problem: Error from setting an option is silently ignored.

Solution: Handle option value errors better. Fix uses of N_().

Files: src/option.c, src/proto/option.pro, src/optionstr.c,
src/channel.c, src/crypt.c, src/diff.c, src/edit.c,
src/eval.c, src/evalfunc.c, src/evalvars.c, src/ex_cmds2.c,
src/ex_docmd.c, src/ex_getln.c, src/getchar.c, src/gui.c,
src/gui_gtk_x11.c, src/help.c, src/highlight.c, src/if_tcl.c,
src/main.c, src/memline.c, src/message_test.c,
src/popupmenu.c, src/quickfix.c, src/scriptfile.c, src/spell.c,
src/spellfile.c, src/term.c, src/undo.c, src/vim9script.c

Patch 8.2.4754

Problem: Still using cached values after unsetting some known environment
variables.

Solution: Take care of the side effects. (closes #10194)

Files: src/evalfunc.c, src/evalvars.c, src/misc1.c, src/proto/misc1.pro,
src/vim9execute.c, src/optionstr.c, src/testdir/test_envir.vim

Patch 8.2.4755

Problem: Cannot use `<SID>FuncRef` in completion spec.

Solution: Dereference a function name in another way. (closes #10197)

Files: src/eval.c, src/testdir/test_vim9_import.vim

Patch 8.2.4756 (after 8.2.4754)

Problem: Build error without the +eval feature.
Solution: Adjust #ifdefs.
Files: src/misc1.c

Patch 8.2.4757

Problem: List of libraries to suppress lsan errors is outdated.
Solution: Add another library. (closes #10201)
Files: src/testdir/lsan-suppress.txt

Patch 8.2.4758

Problem: When using an LSP channel want to get the message ID.
Solution: Have ch_sendexpr() return the ID. (Yegappan Lakshmanan, closes #10202)
Files: runtime/doc/channel.txt, src/channel.c, src/evalfunc.c, src/testdir/test_channel.vim

Patch 8.2.4759

Problem: CurSearch highlight does not work for multi-line match.
Solution: Check cursor position before adjusting columns. (closes #10133)
Files: src/structs.h, src/match.c, src/testdir/test_search.vim, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line.dump, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line_1.dump, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line_2.dump, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line_3.dump, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line_4.dump, src/testdir/dumps/Test_hlsearch_cursearch_multiple_line_5.dump

Patch 8.2.4760

Problem: Using matchfuzzy() on a long list can take a while.
Solution: Add a limit to the number of matches. (Yasuhiro Matsumoto, closes #10189)
Files: runtime/doc/builtin.txt, src/search.c, src/testdir/test_matchfuzzy.vim

Patch 8.2.4761

Problem: Documentation for using LSP messages is incomplete.
Solution: Update the documentation. (Yegappan Lakshmanan, closes #10206)
Files: runtime/doc/channel.txt

Patch 8.2.4762

Problem: Using freed memory when using synstack() and synID() in WinEnter.
Solution: Check using the syntax window. (closes #10204)
Files: src/syntax.c, src/testdir/test_syntax.vim

Patch 8.2.4763

Problem: Using invalid pointer with "V:" in Ex mode.
Solution: Correctly handle the command being changed to "+".
Files: src/ex_docmd.c, src/testdir/test_ex_mode.vim

Patch 8.2.4764

Problem: CI uses an older gcc version.
Solution: Use GCC 11. (closes #10185)
Files: .github/workflows/ci.yml, src/testdir/lsan-suppress.txt

Patch 8.2.4765

Problem: Function matchfuzzy() sorts too many items.
Solution: Only put matches in the array. (Yegappan Lakshmanan, closes #10208)
Files: src/search.c

Patch 8.2.4766

Problem: KRL files using "deffct" not recognized.
Solution: Adjust the pattern used for matching. (Patrick Meiser-Knosowski, closes #10200)
Files: runtime/autoload/dist/ft.vim, src/testdir/test_filetype.vim

Patch 8.2.4767

Problem: Openscad files are not recognized.
Solution: Add a filetype pattern. (Niklas Adam, closes #10199)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4768

Problem: CI: codecov upload sometimes does not work.
Solution: Use action v3 instead of v2. (closes #10209)
Files: .github/workflows/ci.yml

Patch 8.2.4769

Problem: Build warning with UCRT.
Solution: Adjust #ifdef for _wenviron. (John Marriott)
Files: src/evalfunc.c

Patch 8.2.4770

Problem: Cannot easily mix expression and heredoc.
Solution: Support `=expr` in heredoc. (Yegappan Lakshmanan, closes #10138)
Files: runtime/doc/eval.txt, src/evalvars.c, src/userfunc.c, src/testdir/test_let.vim, src/testdir/test_vim9_assign.vim

Patch 8.2.4771

Problem: Coverity warns for not checking return value.
Solution: Check return value of rettv_dict_alloc().
Files: src/channel.c

Patch 8.2.4772

Problem: Old Coverity warning for not checking ftell() return value.
Solution: Check return value of fseek() and ftell().
Files: src/misc1.c

Patch 8.2.4773

Problem: Build failure without the +eval feature.
Solution: Use other error message. Avoid warnings.
Files: src/misc1.c, src/cindent.c, src/term.c

Patch 8.2.4774

Problem: Crash when using a number for lambda name.
Solution: Check the type of the lambda reference.
Files: src/eval.c, src/errors.h, src/testdir/test_lambda.vim

Patch 8.2.4775

Problem: SpellBad highlighting does not work in Konsole.
Solution: Do not keep t_8u defined for Konsole. Redraw when t_8u is reset.
(closes #10177)
Files: src/term.c

Patch 8.2.4776

Problem: GTK: 'lines' and 'columns' may change during startup.
Solution: Ignore stale GTK resize events. (Ernie Rael, closes #10179)
Files: src/gui_gtk_x11.c

Patch 8.2.4777 (after 8.2.4775)

Problem: Screenshot tests fail because of a redraw.
Solution: Do not output t_8u before receiving termresponse. Redraw only
when t_8u is not reset and termresponse is received.
Files: src/term.c

Patch 8.2.4778

Problem: Pacman files use dosini filetype.
Solution: Use conf instead. (Chaoren Lin, closes #10213)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4779

Problem: lsan suppression is too version specific.
Solution: Leave out the version number. (Christian Brabandt, closes #10214)
Files: src/testdir/lsan-suppress.txt

Patch 8.2.4780

Problem: Parsing an LSP message fails when it is split.
Solution: Collapse the received data before parsing. (Yegappan Lakshmanan,
closes #10215)
Files: runtime/doc/channel.txt, src/channel.c,
src/testdir/test_channel.vim, src/testdir/test_channel_lsp.py

Patch 8.2.4781

Problem: Maxima files are not recognized.
Solution: Add patterns to detect Maxima files. (Doron Behar, closes #10211)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4782

Problem: Accessing freed memory.
Solution: Clear evalarg after checking for trailing characters.
(issue #10218)
Files: src/userfunc.c, src/testdir/test_lambda.vim

Patch 8.2.4783

Problem: Coverity warns for leaking memory.
Solution: Use another strategy freeing "theline".
Files: src/evalvars.c

Patch 8.2.4784

Problem: Lamba test with timer is flaky.
Solution: Adjust sleep time on retry.
Files: src/testdir/test_lambda.vim

Patch 8.2.4785

Problem: Visual mode not stopped early enough if win_gotoid() goes to another buffer. (Sergey Vlasov)
Solution: Stop Visual mode before jumping to another buffer. (closes #10217)
Files: src/evalwindow.c, src/testdir/test_vim9_builtin.vim,
src/testdir/dumps/Test_win_gotoid_1.dump,
src/testdir/dumps/Test_win_gotoid_2.dump,
src/testdir/dumps/Test_win_gotoid_3.dump

Patch 8.2.4786 (after 8.2.4785)

Problem: Test for win_gotoid() in Visual mode fails on Mac.
Solution: Skip the test on MacOS.
Files: src/testdir/test_vim9_builtin.vim

Patch 8.2.4787

Problem: prop_find() does not find the right property.
Solution: Fix the scan order. (closes #10220)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.4788

Problem: Large payload for LSP message not tested.
Solution: Add a test with a large LSP payload. (Yegappan Lakshmanan, closes #10223)
Files: src/channel.c, src/testdir/test_channel.vim,
src/testdir/test_channel_lsp.py

Patch 8.2.4789

Problem: The cursor may be in the in wrong place when using :redraw while editing the cmdline.
Solution: When editing the command line let :redraw update the command line too. (closes #10210)
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim,
src/testdir/dumps/Test_redraw_in_autocmd_1.dump,
src/testdir/dumps/Test_redraw_in_autocmd_2.dump

Patch 8.2.4790

Problem: Lilypond filetype not recognized.
Solution: Add patterns for lilypond. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4791

Problem: Autocmd events triggered in different order when reusing an empty buffer.
Solution: Call buff_freeall() earlier. (Charlie Groves, closes #10198)
Files: src/buffer.c, src/testdir/test_autocmd.vim

Patch 8.2.4792

Problem: Indent operator creates an undo entry for every line.
Solution: Create one undo entry for all lines. (closes #10227)
Files: src/indent.c, src/testdir/test_indent.vim

Patch 8.2.4793

Problem: Recognizing Maxima filetype even though it might be another.
Solution: Remove *.mc and *.dem patterns from Maxima files

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4794

Problem: Compiler warning for not initialized variable.

Solution: Initialize the variable. (John Marriott)

Files: src/indent.c

Patch 8.2.4795

Problem: **'cursorbind'** scrolling depends on whether **'cursorline'** is set.

Solution: Always call validate_cursor(). (Christian Brabandt, closes #10230, closes #10014)

Files: src/move.c, src/testdir/README.txt,
src/testdir/test_cursorline.vim,
src/testdir/dumps/Test_hor_scroll_1.dump,
src/testdir/dumps/Test_hor_scroll_2.dump,
src/testdir/dumps/Test_hor_scroll_3.dump,
src/testdir/dumps/Test_hor_scroll_4.dump

Patch 8.2.4796 (after 8.2.4795)

Problem: File left behind after running cursorline tests.

Solution: Uncomment the line that deletes the file.

Files: src/testdir/test_cursorline.vim

Patch 8.2.4797

Problem: getwininfo() may get outdated values.

Solution: Make sure w_botline is up-to-date. (closes #10226)

Files: src/evalwindow.c, src/testdir/test_bufwintabinfo.vim

Patch 8.2.4798

Problem: t_8u option was reset even when set by the user.

Solution: Only reset t_8u when using the default value. (closes #10239)

Files: src/term.c

Patch 8.2.4799

Problem: Popup does not use correct topline.

Solution: Also add one when firstline is negative. (closes #10229)

Files: src/popupwin.c, src/testdir/test_popupwin.vim

Patch 8.2.4800 (after 8.2.4798)

Problem: Missing test update for adjusted t_8u behavior.

Solution: Update and extend the test.

Files: src/testdir/test_termcodes.vim

Patch 8.2.4801 (after 8.2.4795)

Problem: Fix for cursorbind fix not fully tested.

Solution: Add another test case. (Christian Brabandt, closes #10240)

Files: src/testdir/test_cursorline.vim,
src/testdir/dumps/Test_hor_scroll_5.dump

Patch 8.2.4802

Problem: Test is not cleaned up.

Solution: Make test clean up after itself. Avoid NUL. (closes #10233)

Files: src/testdir/test_autocmd.vim

Patch 8.2.4803

Problem: WinScrolled not always triggered when scrolling with the mouse.
Solution: Add calls to may_trigger_winscrolled(). (closes #10246)
Files: src/mouse.c, src/testdir/test_autocmd.vim

Patch 8.2.4804

Problem: Expression in heredoc doesn't work for compiled function.
Solution: Implement compiling the heredoc expressions. (Yegappan Lakshmanan, closes #10232)
Files: runtime/doc/eval.txt, src/evalvars.c, src/proto/evalvars.pro, src/ex_getln.c, src/vim9compile.c, src/proto/vim9compile.pro, src/testdir/test_vim9_assign.vim

Patch 8.2.4805

Problem: CurSearch used for all matches in current line.
Solution: Don't use the non-zero line count. (closes #10247)
Files: src/match.c, src/testdir/test_search.vim, src/testdir/dumps/Test_hlsearch_cursearch_single_line_1.dump, src/testdir/dumps/Test_hlsearch_cursearch_single_line_2.dump, src/testdir/dumps/Test_hlsearch_cursearch_single_line_2a.dump, src/testdir/dumps/Test_hlsearch_cursearch_single_line_2b.dump

Patch 8.2.4806

Problem: A mapping using <LeftDrag> does not start Select mode.
Solution: When checking for starting select mode with the mouse also do this when there is typeahead. (closes #10249)
Files: src/normal.c

Patch 8.2.4807

Problem: Processing key events in Win32 GUI is not ideal.
Solution: Improve processing of key events. (closes #10155)
Files: src/gui_w32.c

Patch 8.2.4808

Problem: Unused item in engine struct.
Solution: Remove "expr". Add comment with tags.
Files: src/regexp.h

Patch 8.2.4809

Problem: Various things not properly tested.
Solution: Add various test cases. (Yegappan Lakshmanan, closes #10259)
Files: src/testdir/test_blob.vim, src/testdir/test_debugger.vim, src/testdir/test_listdict.vim, src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_import.vim, src/testdir/test_vim9_script.vim

Patch 8.2.4810 (after 8.2.4808)

Problem: Missing changes in one file.
Solution: Also change the struct initializers.
Files: src/regexp.c

Patch 8.2.4811 (after 8.2.4807)

Problem: Win32 GUI: caps lock doesn't work.
Solution: Handle VK_CAPITAL. (closes #10260, closes #10258)
Files: src/gui_w32.c

Patch 8.2.4812

Problem: Unused struct item.
Solution: Remove "lines" match_T. Simplify the code. (closes #10256)
Files: src/match.c, src/structs.h

Patch 8.2.4813

Problem: Pasting text while indent folding may mess up folds.
Solution: Adjust the way folds are split. (Brandon Simmons, closes #10254)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.4814

Problem: Possible to leave a popup window with win_gotoid().
Solution: Give an error when trying to leave a popup window with win_gotoid(). (closes #10253)
Files: src/evalwindow.c, src/testdir/test_terminal3.vim

Patch 8.2.4815 (after 8.2.4776)

Problem: Cannot build with older GTK version.
Solution: Use gtk_window_get_size() instead of gdk_window_get_width() and gdk_window_get_height(). (Ernie Rael, closes #10257)
Files: src/gui_gtk_x11.c

Patch 8.2.4816

Problem: Still using older codecov app in some places of CI.
Solution: Use v3.1.0. (closes #10209)
Files: .github/workflows/ci.yml

Patch 8.2.4817

Problem: Win32 GUI: modifiers are not always used.
Solution: Handle more modifiers. (closes #10269)
Files: src/gui_w32.c

Patch 8.2.4818 (after 8.2 4806)

Problem: No test for what 8.2.4806 fixes.
Solution: Add a test. (closes #10272)
Files: src/testdir/test_mapping.vim

Patch 8.2.4819

Problem: Unmapping simplified keys also deletes other mapping.
Solution: Only unmap a mapping with m_simplified set. (closes #10270)
Files: src/map.c, src/testdir/test_mapping.vim

Patch 8.2.4820

Problem: No simple programmatic way to find a specific mapping.
Solution: Add getmappings(). (Ernie Rael, closes #10273)
Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/map.c, src/proto/map.pro, src/testdir/test_maparg.vim

Patch 8.2.4821

Problem: Crash when imported autoload script was deleted.
Solution: Initialize local variable. (closes #10274) Give a more meaningful error message.
Files: src/eval.c, src/vim9script.c, src/testdir/test_vim9_import.vim

Patch 8.2.4822

Problem: Setting ufunc to NULL twice.
Solution: Set ufunc to NULL in find_exported(). (closes #19275)
Files: src/eval.c, src/vim9script.c

Patch 8.2.4823

Problem: Concatenating more than 2 strings in a :def function is inefficient.
Solution: Add a count to the CONCAT instruction. (closes #10276)
Files: src/vim9.h, src/vim9cmds.c, src/vim9compile.c, src/vim9execute.c, src/vim9expr.c, src/vim9instr.c, src/proto/vim9instr.pro, src/testdir/test_vim9_disassemble.vim

Patch 8.2.4824

Problem: Expression is evaluated multiple times.
Solution: Evaluate expression once and store the result. (closes #10278)
Files: src/map.c

Patch 8.2.4825

Problem: Can only get a list of mappings.
Solution: Add the optional {abbr} argument. (Ernie Rael, closes #10277)
Rename to maplist(). Rename test file.
Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/map.c, src/proto/map.pro, src/testdir/test_maparg.vim, src/testdir/test_map_functions.vim, src/testdir/Make_all.mak

Patch 8.2.4826

Problem: .cshtml files are not recognized.
Solution: Use html filetype for .cshtml files. (Julien Voisin, closes #10212)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4827

Problem: Typo in variable name. (Gabriel Dupras)
Solution: Rename the variable.
Files: src/map.c

Patch 8.2.4828

Problem: Fix for unmapping simplified key not fully tested.
Solution: Add a test case. (closes #10292)
Files: src/map.c, src/testdir/test_mapping.vim

Patch 8.2.4829

Problem: A key may be simplified to NUL.
Solution: Use K_ZERO instead. Use macros instead of hard coded values. (closes #10290)
Files: src/getchar.c, src/misc2.c, src/testdir/test_termcodes.vim

Patch 8.2.4830

Problem: Possible endless loop if there is unused typeahead.
Solution: Only loop when the typeahead changed.
Files: src/channel.c

Patch 8.2.4831

Problem: Crash when using maparg() and unmapping simplified keys.
Solution: Do not keep a mapblock pointer. (closes #10294)
Files: src/map.c, src/testdir/test_map_functions.vim

Patch 8.2.4832

Problem: Passing zero instead of NULL to a pointer argument.
Solution: Use NULL. (closes #10296)
Files: src/getchar.c, src/term.c

Patch 8.2.4833

Problem: Failure of mapping not checked for.
Solution: Check return value of ins_typebuf(). (closes #10299)
Files: src/getchar.c, src/term.c, src/testdir/test_termcodes.vim

Patch 8.2.4834

Problem: Vim9: some lines not covered by tests.
Solution: Add a few more tests. Remove dead code.
Files: src/vim9execute.c, src/vim9instr.c, src/vim9.h,
src/testdir/test_vim9_expr.vim

Patch 8.2.4835

Problem: Vim9: some lines not covered by tests.
Solution: Add a few more tests. Fix disassemble output.
Files: src/vim9execute.c, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4836

Problem: Vim9: some lines not covered by tests.
Solution: Remove dead code. Add disassemble tests.
Files: src/vim9execute.c, src/vim9.h,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4837 (after patch 8.2.0919)

Problem: Modifiers not simplified when timed out or using feedkeys() with 'n' flag.
Solution: Adjust how mapped flag and timeout are used. (closes #10305)
Files: src/getchar.c, src/testdir/test_paste.vim,
src/testdir/test_termcodes.vim

Patch 8.2.4838

Problem: Checking for absolute path is not trivial.
Solution: Add isabsolute_path(). (closes #10303)
Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/evalfunc.c,
src/filepath.c, src/proto/filepath.pro,
src/testdir/test_functions.vim

Patch 8.2.4839

Problem: Compiler warning for unused argument.
Solution: Add "UNUSED".
Files: src/gui_gtk_x11.c

Patch 8.2.4840

Problem: Heredoc expression evaluated even when skipping.

Solution: Don't evaluate when "skip" is set. (closes #10306)
Files: src/evalvars.c, src/testdir/test_let.vim

Patch 8.2.4841

Problem: Empty string considered an error for expand() when 'verbose' is set. (Christian Brabandt)

Solution: Do not give an error for an empty result. (closes #10307)
Files: src/evalfunc.c, src/ex_docmd.c, src/proto/ex_docmd.pro, src/filepath.c, src/testdir/test_expand_func.vim

Patch 8.2.4842 (after 8.2.4841)

Problem: expand("%:~") is not empty when there is no buffer name.

Solution: When ignoring errors still return NULL. (closes #10311)
Files: src/ex_docmd.c, src/testdir/test_expand_func.vim

Patch 8.2.4843 (after 8.2.4807)

Problem: Win32 GUI: Treating CTRL + ALT as AltGr is not backwards compatible. (Axel Bender)

Solution: Make a difference between left and right menu keys. (closes #10308)

Files: src/gui_w32.c

Patch 8.2.4844

Problem: <C-S-I> is simplified to <S-Tab>.

Solution: Do not simplify CTRL if there is also SHIFT. (closes #10313)
Files: src/getchar.c, src/testdir/test_gui.vim

Patch 8.2.4845

Problem: Duplicate code.

Solution: Move code below if/else. (closes #10314)
Files: src/misc1.c

Patch 8.2.4846 (after 8.2.4844)

Problem: Termcodes test fails.

Solution: use CTRL-SHIFT-V to insert an unsimplified key. (closes #10316)
Files: runtime/doc/cmdline.txt, src/edit.c, src/getchar.c, src/testdir/test_gui.vim

Patch 8.2.4847

Problem: Crash when using uninitialized function pointer.

Solution: Check for NULL pointer. (closes #10319, closes #10319)
Files: src/eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.4848

Problem: Local completion with mappings and simplification not working.

Solution: Fix local completion <C-N>/<C-P> mappings not ignored if keys are not simplified. (closes #10323)

Files: src/getchar.c, src/testdir/test_popup.vim

Patch 8.2.4849

Problem: Gleam filetype not detected.

Solution: Add a pattern for Gleam files. (Mathias Jean Johansen, closes #10326)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4850

Problem: Mksession mixes up "tabpages" and "curdir" arguments.
Solution: Correct logic for storing tabpage in session. (closes #10312)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4851

Problem: Compiler warning for uninitialized variable.
Solution: Use another variable to decide to restore option values.
Files: src/session.c

Patch 8.2.4852

Problem: ANSI color index to RGB value not correct.
Solution: Convert the cterm index to ANSI index. (closes #10321,
closes #9836)
Files: src/term.c

Patch 8.2.4853

Problem: CI with FreeBSD is a bit outdated.
Solution: Use 12.3 instead of 12.1. (closes #10333)
Files: .cirrus.yml

Patch 8.2.4854

Problem: Array size does not match usage.
Solution: Make array size 3 instead of 4. (Christian Brabandt, closes #10336)
Files: src/term.c

Patch 8.2.4855

Problem: Robot files are not recognized.
Solution: Add patterns for robot files. (Zoe Roux, closes #10339)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4856

Problem: MinGW compiler complains about unknown escape sequence.
Solution: Avoid using a backslash in path. (Christian Brabandt,
closes #10337)
Files: .github/workflows/ci.yml

Patch 8.2.4857

Problem: Yaml indent for multiline is wrong.
Solution: Adjust patterns. (closes #10328, closes #8740)
Files: runtime/indent/yaml.vim, runtime/indent/testdir/yaml.in,
runtime/indent/testdir/yaml.ok

Patch 8.2.4858

Problem: K_SPECIAL may be escaped twice.
Solution: Avoid double escaping. (closes #10340)
Files: src/highlight.c, src/misc2.c, src/proto/misc2.pro, src/term.c,
src/typval.c, src/testdir/test_eval_stuff.vim,
src/testdir/test_feedkeys.vim, src/testdir/test_functions.vim,
src/testdir/test_mapping.vim

Patch 8.2.4859

Problem: wget2 files are not recognized.

Solution: Add patterns to recognize wget2. (Doug Kearns)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.4860

Problem: MS-Windows: always uses current directory for executables.
Solution: Check the NoDefaultCurrentDirectoryInExePath environment variable.
(Yasuhiro Matsumoto, closes #10341)
Files: runtime/doc/builtin.txt, src/os_win32.c,
src/testdir/test_functions.vim

Patch 8.2.4861

Problem: It is not easy to restore saved mappings.
Solution: Make mapset() accept a dict argument. (Ernie Rael, closes #10295)
Files: runtime/doc/builtin.txt, src/errors.h, src/evalfunc.c, src/map.c,
src/typval.c, src/proto/typval.pro,
src/testdir/test_map_functions.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4862

Problem: Vim9: test may fail when run with valgrind.
Solution: Wait longer for callback if needed.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.4863

Problem: Accessing freed memory in test without the +channel feature.
(Dominique Pellé)
Solution: Do not generate PUSHCHANNEL or PUSHJOB if they are not
implemented. (closes #10350)
Files: src/vim9instr.c, src/errors.h, src/vim9compile.c,
src/testdir/test_vim9_script.vim

Patch 8.2.4864 (after 8.2.4863)

Problem: Vim9: script test fails.
Solution: Remove "if" around declaration.
Files: src/testdir/test_vim9_script.vim

Patch 8.2.4865

Problem: :startinsert right after :stopinsert does not work when popup menu
is still visible.
Solution: Use ins_compl_active() instead of pum_visible(). (closes #10352)
Files: src/edit.c, src/testdir/test_ins_complete.vim

Patch 8.2.4866

Problem: Duplicate code in "get" functions.
Solution: Use get_var_from() for getwinvar(), gettabvar(), gettabwinvar()
and getbufvar(). (closes #10335)
Files: src/evalvars.c

Patch 8.2.4867

Problem: Listing of mapping with K_SPECIAL is wrong.
Solution: Adjust escaping of special characters. (closes #10351)
Files: src/map.c, src/message.c, src/testdir/test_mapping.vim

Patch 8.2.4868

Problem: When closing help window autocmds triggered for the wrong window.
Solution: Figure out the new current window earlier. (closes #10348)
Files: src/window.c, src/testdir/test_help.vim

Patch 8.2.4869

Problem: Expression in command block does not look after NL.
Solution: Skip over NL to check what follows. (closes #10358)
Files: src/eval.c, src/proto/eval.pro, src/vim9script.c,
src/testdir/test_usercommands.vim

Patch 8.2.4870

Problem: Vim9: expression in :substitute is not compiled.
Solution: Use an INSTR instruction if possible. (closes #10334)
Files: src/evalfunc.c, src/regexp.c, src/vim9execute.c, src/vim9expr.c,
src/testdir/test_vim9_builtin.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4871

Problem: Vim9: in :def function no error for using a range with a command
that does not accept one.
Solution: Check for the command to accept a range. (closes #10330)
Files: src/vim9compile.c, src/testdir/test_vim9_script.vim

Patch 8.2.4872

Problem: Vim9: no error for using an expression only at the script level
when followed by an empty line.
Solution: Do not check the line number but whether something follows.
(closes #10357)
Files: src/ex_eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4873

Problem: Vim9: using "else" differs from using "endif/if !cond".
Solution: Leave the block and enter another one. (closes #10320)
Files: src/ex_eval.c, src/testdir/test_vim9_script.vim

Patch 8.2.4874

Problem: Win32 GUI: horizontal scroll wheel not handled properly.
Solution: Also handle WM_MOUSEHWHEEL. (closes #10309)
Files: src/gui_w32.c

Patch 8.2.4875

Problem: MS-Windows: some .exe files are not recognized.
Solution: Parse APPEXECLINK junctions. (closes #10302)
Files: src/os_mswin.c, src/proto/os_mswin.pro, src/os_win32.c,
src/os_win32.h, src/testdir/test_functions.vim

Patch 8.2.4876

Problem: MS-Windows: Shift-BS results in strange character in powershell.
Solution: Add K_S_BS. (Christian Brabandt, closes #10283, closes #10279)
Files: src/edit.c, src/keymap.h, src/term.c, src/testdir/shared.vim,
src/testdir/test_edit.vim

Patch 8.2.4877

Problem: MS-Windows: Using Normal colors for termguicolors causes problems.

Solution: Do not use Normal colors to set `sg_gui_fg` and `sg_gui_bg`.
(Christian Brabandt, closes #10317, closes #10241)
Files: `src/highlight.c`

Patch 8.2.4878

Problem: Valgrind warning for using uninitialized variable.
Solution: Initialize the type of `newtv`.
Files: `src/strings.c`

Patch 8.2.4879

Problem: Screendump test may fail when using valgrind.
Solution: Wait longer for the first screendump.
Files: `src/testdir/test_vim9_builtin.vim`, `src/testdir/screendump.vim`

Patch 8.2.4880

Problem: Vim9: misplaced `elseif` causes invalid memory access.
Solution: Check `cs_idx` not to be negative.
Files: `src/ex_eval.c`

Patch 8.2.4881

Problem: "P" in Visual mode still changes some registers.
Solution: Make "P" in Visual mode not change any register. (Shougo Matsushita, closes #10349)
Files: `runtime/doc/change.txt`, `runtime/doc/index.txt`,
`runtime/doc/visual.txt`, `src/normal.c`, `src/testdir/test_visual.vim`

Patch 8.2.4882

Problem: Cannot make '**breakindent**' use a specific column.
Solution: Add the "column" entry in '**breakindentopt**'. (Christian Brabandt, closes #10362, closes #10325)
Files: `runtime/doc/options.txt`, `src/indent.c`, `src/structs.h`,
`src/testdir/test_breakindent.vim`

Patch 8.2.4883

Problem: String interpolation only works in heredoc.
Solution: Support interpolated strings. Use syntax for heredoc consistent with strings, similar to C#. (closes #10327)
Files: `runtime/doc/eval.txt`, `src/errors.h`, `src/eval.c`, `src/evalvars.c`,
`src/proto/evalvars.pro`, `src/typval.c`, `src/proto/typval.pro`,
`src/vim9compile.c`, `src/proto/vim9compile.pro`, `src/vim9expr.c`,
`src/testdir/test_debugger.vim`, `src/testdir/test_expr.vim`,
`src/testdir/test_let.vim`, `src/testdir/test_vim9_assign.vim`,
`src/testdir/test_vim9_disassemble.vim`

Patch 8.2.4884

Problem: Test fails without the job/channel feature. (Dominique Pellé)
Solution: Add condition.
Files: `src/testdir/test_vim9_script.vim`

Patch 8.2.4885 (after 8.2.4884)

Problem: Test fails with the job/channel feature.
Solution: Move check for job/channel separately.
Files: `src/testdir/test_vim9_script.vim`

Patch 8.2.4886

Problem: Vim9: redir in skipped block seen as assignment.
Solution: Check for valid assignment.
Files: src/ex_docmd.c, src/testdir/test_vim9_assign.vim

Patch 8.2.4887

Problem: Channel log does not show invoking a timer callback.
Solution: Add a ch_log() call.
Files: src/time.c

Patch 8.2.4888

Problem: Line number of lambda ignores line continuation.
Solution: Use the line number of where the arguments are. Avoid outputting "... " twice. (closes #10364)
Files: src/userfunc.c

Patch 8.2.4889

Problem: CI only tests with FreeBSD 12.
Solution: Also test with FreeBSD 13. (closes #10366)
Files: .cirrus.yml

Patch 8.2.4890

Problem: Inconsistent capitalization in error messages.
Solution: Make capitalization consistent. (Doug Kearns)
Files: src/errors.h

Patch 8.2.4891

Problem: Vim help presentation could be better.
Solution: Add an imported file for extra Vim help support. Show highlight names in the color they have.
Files: Filelist, runtime/import/dist/vimhelp.vim

Patch 8.2.4892

Problem: Test failures because of changed error messages.
Solution: Adjust the expected error messages.
Files: src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_func.vim, src/testdir/test_vim9_script.vim,
src/testdir/test_expand.vim, src/testdir/test_tcl.vim,
src/testdir/test_vimscript.vim

Patch 8.2.4893 (after 8.2.4891)

Problem: Distributed import files are not installed.
Solution: Add rules to Makefile and NSIS.
Files: src/Makefile, nsis/gvim.nsi

Patch 8.2.4894

Problem: MS-Windows: not using italics.
Solution: Use italics. Simplify the code. (closes #10359)
Files: src/term.c

Patch 8.2.4895

Problem: Buffer overflow with invalid command with composing chars.
Solution: Check that the whole character fits in the buffer.

Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.4896 (after 8.2.4869)
Problem: Expression in command block does not look after NL when command is typed.
Solution: Skip over NL also when not in a script. (closes #10358)
Files: src/eval.c, src/testdir/test_usercommands.vim

Patch 8.2.4897
Problem: Comment inside an expression in lambda ignores the rest of the expression.
Solution: Truncate the line at the comment. (closes #10367)
Files: src/eval.c, src/testdir/test_lambda.vim

Patch 8.2.4898
Problem: Coverity complains about pointer usage.
Solution: Move code for increment/decrement.
Files: src/vim9compile.c

Patch 8.2.4899
Problem: With latin1 encoding **CTRL-W** might go before the start of the command line.
Solution: Check already being at the start of the command line.
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.4900
Problem: Vim9 expression test fails without the job feature.
Solution: Add a check for the job feature. (Dominique Pellé, closes #10373)
Files: src/testdir/test_vim9_expr.vim

Patch 8.2.4901
Problem: NULL pointer access when using invalid pattern.
Solution: Check for failed regexp program.
Files: src/buffer.c, src/testdir/test_buffer.vim

Patch 8.2.4902
Problem: Mouse wheel scrolling is inconsistent.
Solution: Use the MS-Windows system setting. (closes #10368)
Files: runtime/doc/scroll.txt, src/gui_w32.c, src/mouse.c, src/proto/mouse.pro, src/testing.c, src/testdir/test_gui.vim

Patch 8.2.4903
Problem: Cannot get the current cmdline completion type and position.
Solution: Add getcmdcompltype() and getcmdscreenpos(). (Shougo Matsushita, closes #10344)
Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/cmdexpand.c, src/proto/cmdexpand.pro, src/evalfunc.c, src/ex_getln.c, src/proto/ex_getln.pro, src/usercmd.c, src/proto/usercmd.pro, src/testdir/test_cmdline.vim

Patch 8.2.4904
Problem: codecov includes MS-Windows install files.
Solution: Ignore dosinst.c and uninstall.c.
Files: .codecov.yml

Patch 8.2.4905

Problem: codecov includes MS-Windows install header file.
Solution: Ignore dosinst.h.
Files: .codecov.yml

Patch 8.2.4906

Problem: MS-Windows: cannot use transparent background.
Solution: Make transparent background work with **'termguicolors'** and NONE background color. (Yasuhiro Matsumoto, closes #10310, closes #7162)
Files: runtime/doc/options.txt, src/os_win32.c, src/term.c

Patch 8.2.4907

Problem: Some users do not want a line comment always inserted.
Solution: Add the '/' flag to **'formatoptions'** to not repeat the comment leader after a statement when using "o".
Files: runtime/doc/change.txt, src/option.h, src/change.c, src/testdir/test_textformat.vim

Patch 8.2.4908

Problem: No text formatting for // comment after a statement.
Solution: format a comment when the 'c' flag is in **'formatoptions'**.
Files: src/textformat.c, src/testdir/test_textformat.vim

Patch 8.2.4909

Problem: MODE_ enum entries names are too generic.
Solution: use CH_MODE_.
Files: src/structs.h, src/channel.c, src/job.c, src/terminal.c

Patch 8.2.4910

Problem: Imperfect coding.
Solution: Make code nicer.
Files: src/ex_getln.c

Patch 8.2.4911

Problem: The mode #defines are not clearly named.
Solution: Prepend MODE_. Renumber them to put the mapped modes first.
Files: src/vim.h, src/autocmd.c, src/buffer.c, src/change.c, src/charset.c, src/cindent.c, src/clipboard.c, src/debugger.c, src/digraph.c, src/drawline.c, src/drawscreen.c, src/edit.c, src/evalfunc.c, src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c, src/fold.c, src/getchar.c, src/globals.h, src/gui.c, src/gui_gtk.c, src/gui_w32.c, src/gui_xim.c, src/indent.c, src/insexpand.c, src/macros.h, src/main.c, src/map.c, src/menu.c, src/message.c, src/misc1.c, src/misc2.c, src/mouse.c, src/netbeans.c, src/normal.c, src/ops.c, src/option.c, src/os_unix.c, src/os_win32.c, src/popupmenu.c, src/search.c, src/tag.c, src/screen.c, src/term.c, src/terminal.c, src/textformat.c, src/window.c

Patch 8.2.4912

Problem: Using execute() to define a lambda doesn't work. (Ernie Rael)
Solution: Put the getline function in evalarg. (closes #10375)
Files: src/eval.c, src/evalfunc.c, src/proto/evalfunc.pro,

src/testdir/test_vim9_func.vim

Patch 8.2.4913

Problem: Popup_hide() does not always have effect.
Solution: Add the POPF_HIDDEN_FORCE flag. (closes #10376)
Files: src/popupwin.c, src/vim.h, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popup_prop_not_visible_01a.dump,
src/testdir/dumps/Test_popup_prop_not_visible_01b.dump

Patch 8.2.4914

Problem: String interpolation in :def function may fail.
Solution: Do not terminate the expression. (closes #10377)
Files: src/vim9compile.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4915

Problem: Sometimes the cursor is in the wrong position.
Solution: When the cursor moved to another line, recompute w_botline.
(closes #9736)
Files: src/move.c

Patch 8.2.4916 (after 8.2.4915)

Problem: Mouse in Insert mode test fails.
Solution: Fix the test and check relevant positions.
Files: src/testdir/test_edit.vim

Patch 8.2.4917

Problem: Fuzzy expansion of option names is not right.
Solution: Pass the fuzzy flag down the call chain. (Christian Brabandt,
closes #10380, closes #10318)
Files: src/cmdexpand.c, src/option.c, src/proto/option.pro,
src/testdir/test_options.vim

Patch 8.2.4918

Problem: Conceal character from matchadd() displayed too many times.
Solution: Check the syntax flag. (closes #10381, closes #7268)
Files: src/drawline.c, src/testdir/test_matchadd_conceal.vim

Patch 8.2.4919

Problem: Can add invalid bytes with :spellgood.
Solution: Check for a valid word string.
Files: src/mbyte.c, src/spellfile.c, src/errors.h,
src/testdir/test_spell_utf8.vim

Patch 8.2.4920 (after 8.2.4902)

Problem: MS-Windows GUI: unused variables.
Solution: Delete the variables. (John Marriott)
Files: src/gui_w32.c

Patch 8.2.4921

Problem: Spell test fails because of new illegal byte check.
Solution: Remove the test.
Files: src/testdir/test_spell.vim

Patch 8.2.4922 (after 8.2.4916)

Problem: Mouse test fails on MS-Windows.
Solution: Set **'mousemodel'** to "extend".
Files: src/testdir/test_edit.vim

Patch 8.2.4923

Problem: Test checks for terminal feature unnecessarily.
Solution: Remove CheckRunVimInTerminal. (closes #10383)
Files: src/testdir/test_matchadd_conceal.vim

Patch 8.2.4924

Problem: maparg() may return a string that cannot be reused.
Solution: use msg_outtrans_special() instead of str2special().
(closes #10384)
Files: src/message.c, src/option.c, src/testdir/test_map_functions.vim,
src/testdir/test_mapping.vim, src/testdir/test_options.vim

Patch 8.2.4925

Problem: Trailing backslash may cause reading past end of line.
Solution: Check for NUL after backslash.
Files: src/textobject.c, src/testdir/test_textobjects.vim

Patch 8.2.4926

Problem: #ifdef for crypt feature around too many lines.
Solution: Move code outside of #ifdef. (closes #10388)
Files: src/option.c

Patch 8.2.4927

Problem: Return type of remove() incorrect when using three arguments.
Solution: Use first argument type when there are three arguments.
(closes #10387)
Files: src/evalfunc.c, src/testdir/test_vim9_builtin.vim

Patch 8.2.4928

Problem: Various white space and cosmetic mistakes.
Solution: Change spaces to tabs, improve comments.
Files: src/bufwrite.c, src/channel.c, src/cindent.c, src/crypt.c,
src/debugger.c, src/digraph.c, src/edit.c, src/evalwindow.c,
src/ex_cmds.c, src/ex_docmd.c, src/ex_getln.c, src/fileio.c,
src/filepath.c, src/gui.c, src/highlight.c, src/indent.c,
src/insexpand.c, src/job.c, src/keymap.h, src/macros.h,
src/menu.c, src/misc1.c, src/misc2.c, src/mouse.c, src/move.c,
src/normal.c, src/ops.c, src/option.c, src/option.h, src/search.c,
src/session.c, src/spellsuggest.c, src/structs.h, src/tag.c,
src/term.c, src/terminal.c, src/textformat.c, src/typval.c,
src/ui.c, src/userfunc.c, src/vim.h, src/vim9.h,
src/vim9compile.c, src/vim9execute.c, src/window.c,
src/testdir/test_cursorline.vim, src/os_unix.c, src/if_lua.c,
src/if_py_both.h, src/os_amiga.c, src/os_win32.c, src/os_mswin.c,
src/os_vms.c, src/os_vms_conf.h

Patch 8.2.4929

Problem: Off-by-one error in statusline item.
Solution: Subtract one less. (closes #10394, closes #5599)
Files: src/buffer.c, src/testdir/test_statusline.vim,

src/testdir/dumps/Test_statusline_hl.dump

Patch 8.2.4930

Problem: Interpolated string expression requires escaping.
Solution: Do not require escaping in the expression.
Files: runtime/doc/eval.txt, src/typval.c, src/proto/typval.pro,
src/dict.c, src/eval.c, src/evalvars.c, src/proto/evalvars.pro,
src/vim9compile.c, src/proto/vim9compile.pro, src/vim9expr.c,
src/vim9instr.c, src/alloc.c, src/proto/alloc.pro,
src/testdir/test_expr.vim, src/testdir/test_let.vim

Patch 8.2.4931

Problem: Crash with sequence of Perl commands.
Solution: Move PUTBACK to another line. (closes #10386)
Files: src/if_perl.xs

Patch 8.2.4932

Problem: Not easy to filter the output of maplist().
Solution: Add mode_bits to the dictionary. (Ernie Rael, closes #10356)
Files: runtime/doc/builtin.txt, src/map.c,
src/testdir/test_map_functions.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4933

Problem: A few more capitalization mistakes in error messages.
Solution: Adjust capitalization. (Doug Kearns)
Files: src/errors.h

Patch 8.2.4934

Problem: String interpolation fails when not evaluating.
Solution: Skip the expression when not evaluating. (closes #10398)
Files: src/typval.c, src/evalvars.c, src/proto/evalvars.pro,
src/testdir/test_vim9_expr.vim

Patch 8.2.4935

Problem: With **'foldmethod'** "indent" some lines are not included in the fold. (Oleg Koshovetc)
Solution: Fix it. (Brandon Simmons, closes #10399, closes #3214)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.4936

Problem: MS-Windows: mouse coordinates for scroll event are wrong.
Solution: Convert coordinates to the text area coordinates. (closes #10400)
Files: src/gui_w32.c

Patch 8.2.4937 (after 8.2.4931)

Problem: No test for what 8.2.4931 fixes.
Solution: Add a test that triggers a valgrind error.
Files: src/testdir/test_perl.vim

Patch 8.2.4938

Problem: Crash when matching buffer with invalid pattern.
Solution: Check for NULL regprog.
Files: src/buffer.c, src/testdir/test_buffer.vim

Patch 8.2.4939

Problem: matchfuzzypos() with "matchseq" does not have all positions.
Solution: Also add a position for white space. (closes #10404)
Files: runtime/doc/builtin.txt, src/search.c,
src/testdir/test_matchfuzzy.vim

Patch 8.2.4940

Problem: Some code is never used.
Solution: Remove dead code. Add a few more test cases.
Files: src/vim9expr.c, src/proto/vim9expr.pro, src/vim9compile.c,
src/testdir/test_vim9_builtin.vim, src/testdir/test_vim9_expr.vim

Patch 8.2.4941

Problem: '[' and ']' marks may be wrong after undo.
Solution: Adjust the '[' and ']' marks if needed. (closes #10407, closes #1281)
Files: src/undo.c, src/testdir/test_undo.vim

Patch 8.2.4942

Problem: Error when setting 'filetype' in help file again.
Solution: Deal with text property type already existing. (closes #10409)
Files: runtime/import/dist/vimhelp.vim

Patch 8.2.4943

Problem: Changing 'switchbuf' may have no effect.
Solution: Handle 'switchbuf' in didset_string_options(). (Sean Dewar,
closes #10406)
Files: src/optionstr.c, src/testdir/test_options.vim

Patch 8.2.4944

Problem: Text properties are wrong after "cc". (Axel Forsman)
Solution: Pass the deleted byte count to inserted_bytes(). (closes #10412,
closes #7737, closes #5763)
Files: src/change.c, src/testdir/test_textprop.vim

Patch 8.2.4945

Problem: Inconsistent use of white space.
Solution: Use Tabs and Spaces consistently.
Files: src/os_amiga.c, src/if_py_both.h, src/os_win32.c, src/os_mswin.c,
src/os_vms.c, src/os_vms_conf.h

Patch 8.2.4946

Problem: Vim9: some code not covered by tests.
Solution: Add a few more test cases. Remove dead code.
Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4947

Problem: Text properties not adjusted when accepting spell suggestion.
Solution: Adjust text properties when text changes. (closes #10414)
Files: src/spell.c, src/spellsuggest.c, src/testdir/test_textprop.vim

Patch 8.2.4948

Problem: Cannot use Perl heredoc in nested :def function. (Virginia

Senioria)
Solution: Only concatenate heredoc lines when not in a nested function.
(closes #10415)
Files: src/userfunc.c, src/testdir/test_vim9_func.vim

Patch 8.2.4949

Problem: Vim9: some code not covered by tests.
Solution: Add a few more test cases. Fix double error message.
Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4950

Problem: Text properties position wrong after shifting text.
Solution: Adjust the text properties when shifting a block of text.
(closes #10418)
Files: src/ops.c, src/testdir/test_textprop.vim

Patch 8.2.4951

Problem: Smart indenting done when not enabled.
Solution: Check option values before setting can_si. (closes #10420)
Files: src/indent.c, src/proto/indent.pro, src/change.c, src/edit.c,
src/ops.c, src/testdir/test_smartindent.vim

Patch 8.2.4952

Problem: GUI test will fail if color scheme changes.
Solution: Reduce the test for now.
Files: src/testdir/test_gui.vim

Patch 8.2.4953

Problem: With '**smartindent**' inserting '}' after completion goes wrong.
Solution: Check the cursor is in indent. (closes #10420)
Files: src/indent.c, src/testdir/test_smartindent.vim

Patch 8.2.4954

Problem: Inserting line breaks text property spanning more than one line.
Solution: Check TP_FLAG_CONT_PREV and TP_FLAG_CONT_NEXT. (closes #10423)
Files: src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.4955

Problem: Text property in wrong position after auto-indent.
Solution: Adjust text property columns. (closes #10422, closes #7719)
Files: src/change.c, src/testdir/test_textprop.vim

Patch 8.2.4956

Problem: Reading past end of line with "gf" in Visual block mode.
Solution: Do not include the NUL in the length.
Files: src/normal.c, src/testdir/test_gf.vim

Patch 8.2.4957

Problem: Text properties in a wrong position after a block change.
Solution: Adjust the properties columns. (closes #10427)
Files: src/ops.c, src/testdir/test_textprop.vim

Patch 8.2.4958

Problem: A couple conditions are always true.

Solution: Remove the conditions. (Goc Dundar, closes #10428)
Files: src/evalfunc.c, src/quickfix.c

Patch 8.2.4959

Problem: Using NULL regexp program.
Solution: Check for regexp program becoming NULL in more places.
Files: src/buffer.c, src/testdir/test_buffer.vim

Patch 8.2.4960

Problem: Text properties that cross line boundary are not correctly updated for a deleted line.
Solution: Correct computing location of text property entry. (Paul Ollis, closes #10431, closes #10430)
Files: src/memline.c, src/testdir/test_textprop.vim

Patch 8.2.4961

Problem: Build error with a certain combination of features.
Solution: Adjust #if. (John Marriott)
Files: src/memline.c

Patch 8.2.4962

Problem: Files show up in git status.
Solution: Adjust the list of ignored files. Clean up more test files. (Shane xb Qian, closes #9929)
Files: .gitignore, src/testdir/Makefile

Patch 8.2.4963

Problem: Expanding path with "/"**" may overrun end of buffer.
Solution: Use vim_snprintf().
Files: src/filepath.c

Patch 8.2.4964

Problem: MS-Windows GUI: mouse event test is flaky.
Solution: Add a short delay after generating a mouse event.
Files: src/testdir/test_gui.vim

Patch 8.2.4965

Problem: GUI: testing mouse move event depends on screen cell size.
Solution: Multiply the row and column with the screen cell size.
Files: runtime/doc/testing.txt, src/testing.c, src/testdir/test_gui.vim

Patch 8.2.4966

Problem: MS-Windows GUI: mouse event test gets extra event.
Solution: Ignore one move event.
Files: src/testdir/test_gui.vim

Patch 8.2.4967 (after 8.2.4966)

Problem: MS-Windows GUI: mouse event test sometimes fails.
Solution: Ignore one move event only if there is an extra event.
Files: src/testdir/test_gui.vim

Patch 8.2.4968

Problem: Reading past end of the line when C-indenting.
Solution: Check for NUL.

Files: src/cindent.c, src/testdir/test_cindent.vim

Patch 8.2.4969

Problem: Changing text in Visual mode may cause invalid memory access.

Solution: Check the Visual position after making a change.

Files: src/change.c, src/edit.c, src/misc2.c, src/proto/misc2.pro,
src/testdir/test_visual.vim

Patch 8.2.4970

Problem: "eval 123" gives an error, "eval 'abc'" does not.

Solution: Also give an error when evaluating only a string. (closes #10434)

Files: src/ex_eval.c, src/testdir/test_vim9_cmd.vim

Patch 8.2.4971

Problem: Vim9: interpolated string seen as range.

Solution: Recognize an interpolated string at the start of a command line.
(closes #10434)

Files: src/ex_docmd.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4972

Problem: Vim9: compilation fails when using dict member when skipping.

Solution: Do not generate ISN_USEDICT when skipping. (closes #10433)

Files: src/vim9expr.c, src/testdir/test_vim9_expr.vim

Patch 8.2.4973

Problem: Vim9: type error for list unpack mentions argument.

Solution: Mention variable. (close #10435)

Files: src/vim9.h, src/vim9execute.c, src/vim9instr.c,
src/proto/vim9instr.pro, src/vim9compile.c,
src/testdir/test_vim9_script.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4974

Problem: ":so" command may read after end of buffer.

Solution: Compute length of text properly.

Files: src/scriptfile.c, src/testdir/test_source.vim

Patch 8.2.4975

Problem: Recursive command line loop may cause a crash.

Solution: Limit recursion of getcmdline().

Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.4976

Problem: Coverity complains about not restoring a saved value.

Solution: Restore value before handling error.

Files: src/vim9execute.c

Patch 8.2.4977

Problem: Memory access error when substitute expression changes window.

Solution: Disallow changing window in substitute expression.

Files: src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.2.4978

Problem: No error if engine selection atom is not at the start.

Solution: Give an error. (Christian Brabandt, closes #10439)
Files: runtime/doc/pattern.txt, src/errors.h, src/regexp_bt.c,
src/regexp_nfa.c, src/testdir/test_regexp_latin.vim

Patch 8.2.4979

Problem: Accessing freed memory when line is flushed.
Solution: Make a copy of the pattern to search for.
Files: src/window.c, src/testdir/test_tagjump.vim

Patch 8.2.4980

Problem: When '**shortmess**' contains 'A' loading a session may still warn for
an existing swap file. (Melker Österberg)
Solution: Keep the 'A' flag to '**shortmess**' in the session file.
(closes #10443)
Files: src/session.c, src/testdir/test_mksession.vim

Patch 8.2.4981

Problem: It is not possible to manipulate autocommands.
Solution: Add functions to add, get and set autocommands. (Yegappan
Lakshmanan, closes #10291)
Files: runtime/doc/autocmd.txt, runtime/doc/builtin.txt,
runtime/doc/usr_41.txt, src/autocmd.c, src/evalfunc.c,
src/proto/autocmd.pro, src/testdir/test_autocmd.vim,
src/testdir/test_vim9_builtin.vim

Patch 8.2.4982

Problem: Colors in terminal window are not 100% correct.
Solution: Use g:terminal_ansi_colors as documented. (closes #10429,
closes #7227 closes #10347)
Files: src/job.c, src/option.c, src/proto/term.pro,
src/terminal.c, src/proto/terminal.pro, src/term.c,
src/testdir/test_functions.vim, src/testdir/test_terminal.vim

Patch 8.2.4983 (after 8.2.4982)

Problem: Colors test fails in the GUI.
Solution: Reset g:terminal_ansi_colors.
Files: src/testdir/test_functions.vim

Patch 8.2.4984

Problem: Dragging statusline fails for window with winbar.
Solution: Fix off-by-one error. (closes #10448)
Files: src/mouse.c, src/testdir/test_winbar.vim

Patch 8.2.4985

Problem: PVS warns for possible array underrun.
Solution: Add a check for a positive value. (Goc Dundar, closes #10451)
Files: src/spell.c

Patch 8.2.4986

Problem: Some github actions are outdated.
Solution: Update CodeQL to v2, update checkout to v3. (closes #10450)
Files: .github/workflows/ci.yml, .github/workflows/codeql-analysis.yml,
.github/workflows/coverity.yml

Patch 8.2.4987

Problem: After deletion a small fold may be closable.
Solution: Check for a reverse range. (Brandon Simmons, closes #10457)
Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.4988

Problem: Textprop in wrong position when replacing multi-byte chars.
Solution: Adjust textprop position. (closes #10461)
Files: src/change.c, src/testdir/test_textprop.vim

Patch 8.2.4989

Problem: Cannot specify a function name for :defcompile.
Solution: Implement a function name argument for :defcompile.
Files: runtime/doc/vim9.txt, src/userfunc.c, src/proto/userfunc.pro,
src/vim9execute.c, src/ex_cmds.h, src/testdir/test_vim9_cmd.vim,
src/testdir/test_vim9_disassemble.vim

Patch 8.2.4990 (after 8.2.4989)

Problem: Memory leak when :defcompile fails.
Solution: Free fname when returning early.
Files: src/userfunc.c

Patch 8.2.4991

Problem: No test for what patch 8.1.0535 fixes.
Solution: Add a test. (closes #10462)
Files: src/testdir/test_fold.vim

Patch 8.2.4992 (after 8.2.4989)

Problem: Compiler warning for possibly uninitialized variable. (Tony Mechelynck)
Solution: Initialize variable in the caller instead of in the function.
Files: src/userfunc.c, src/vim9execute.c

Patch 8.2.4993

Problem: smart/C/lisp indenting is optional, which makes the code more complex, while it only reduces the executable size a bit.
Solution: Graduate FEAT_CINDENT, FEAT_SMARTINDENT and FEAT_LISP.
Files: runtime/doc/builtin.txt, runtime/doc/indent.txt,
runtime/doc/options.txt, runtime/doc/various.txt, src/feature.h,
src/buffer.c, src/change.c, src/cindent.c, src/charset.c,
src/edit.c, src/evalfunc.c, src/indent.c, src/insexpand.c,
src/main.c, src/mouse.c, src/ops.c, src/option.c, src/optionstr.c,
src/register.c, src/search.c, src/textformat.c, src/version.c,
src/option.h, src/optiondefs.h, src/structs.h, src/globals.h,
src/testdir/test_edit.vim

Patch 8.2.4994

Problem: Tests are using legacy functions.
Solution: Convert a few tests to use :def functions.
Files: src/testdir/test_cindent.vim

Patch 8.2.4995 (after 8.2.4992)

Problem: Still a compiler warning for possibly uninitialized variable. (Tony Mechelynck)

Solution: Initialize variables.
Files: src/vim9execute.c

Patch 8.2.4996 (after 8.2.4969)

Problem: setbuflin() may change Visual selection. (Qiming Zhao)
Solution: Disable Visual mode when using another buffer. (closes #10466)
Files: src/evalbuffer.c, src/testdir/test_buflin.vim

Patch 8.2.4997

Problem: Python: changing hidden buffer can cause the display to be messed up.
Solution: Do not mark changed lines when using another buffer. (Paul Ollis, closes #10437, closes #7972)
Files: src/if_py_both.h, src/testdir/test_python3.vim

Patch 8.2.4998

Problem: Vim9: crash when using multiple funcref().
Solution: Check if varargs type is NULL. (closes #10467)
Files: src/vim9type.c, src/testdir/test_vim9_func.vim

Patch 8.2.4999

Problem: Filetype test table is not properly sorted.
Solution: Sort by filetype. (Doug Kearns)
Files: src/testdir/test_filetype.vim

Patch 8.2.5000

Problem: No patch for documentation updates.
Solution: Update documentation files.
Files: runtime/doc/arabic.txt, runtime/doc/autocmd.txt,
runtime/doc/builtin.txt, runtime/doc/change.txt,
runtime/doc/channel.txt, runtime/doc/cmdline.txt,
runtime/doc/diff.txt, runtime/doc/digraph.txt,
runtime/doc/editing.txt, runtime/doc/eval.txt,
runtime/doc/filetype.txt, runtime/doc/fold.txt,
runtime/doc/ft_ada.txt, runtime/doc/ft_ps1.txt,
runtime/doc/ft_raku.txt, runtime/doc/ft_rust.txt,
runtime/doc/ft_sql.txt, runtime/doc/gui.txt,
runtime/doc/gui_w32.txt, runtime/doc/helphelp.txt,
runtime/doc/help.txt, runtime/doc/if_cscop.txt,
runtime/doc/if_lua.txt, runtime/doc/if_perl.txt,
runtime/doc/if_pyth.txt, runtime/doc/if_tcl.txt,
runtime/doc/indent.txt, runtime/doc/index.txt,
runtime/doc/insert.txt, runtime/doc/intro.txt,
runtime/doc/map.txt, runtime/doc/mbyte.txt,
runtime/doc/message.txt, runtime/doc/motion.txt,
runtime/doc/netbeans.txt, runtime/doc/options.txt,
runtime/doc/os_dos.txt, runtime/doc/os_vms.txt,
runtime/doc/os_win32.txt, runtime/doc/pattern.txt,
runtime/doc/pi_netrw.txt, runtime/doc/pi_zip.txt,
runtime/doc/popup.txt, runtime/doc/print.txt,
runtime/doc/quickfix.txt, runtime/doc/quickref.txt,
runtime/doc/remote.txt, runtime/doc/repeat.txt,
runtime/doc/rileft.txt, runtime/doc/scroll.txt,
runtime/doc/sign.txt, runtime/doc/spell.txt,

runtime/doc/sponsor.txt, runtime/doc/starting.txt,
runtime/doc/syntax.txt, runtime/doc/tabpage.txt,
runtime/doc/tagsrch.txt, runtime/doc/terminal.txt,
runtime/doc/term.txt, runtime/doc/testing.txt,
runtime/doc/textprop.txt, runtime/doc/tips.txt,
runtime/doc/todo.txt, runtime/doc/uganda.txt,
runtime/doc/undo.txt, runtime/doc/usr_02.txt,
runtime/doc/usr_04.txt, runtime/doc/usr_05.txt,
runtime/doc/usr_06.txt, runtime/doc/usr_08.txt,
runtime/doc/usr_09.txt, runtime/doc/usr_12.txt,
runtime/doc/usr_20.txt, runtime/doc/usr_29.txt,
runtime/doc/usr_40.txt, runtime/doc/usr_41.txt,
runtime/doc/usr_45.txt, runtime/doc/usr_46.txt,
runtime/doc/usr_50.txt, runtime/doc/usr_51.txt,
runtime/doc/usr_52.txt, runtime/doc/usr_90.txt,
runtime/doc/usr_toc.txt, runtime/doc/various.txt,
runtime/doc/version5.txt, runtime/doc/version6.txt,
runtime/doc/version7.txt, runtime/doc/version8.txt,
runtime/doc/version9.txt, runtime/doc/vi_diff.txt,
runtime/doc/vim9.txt, runtime/doc/visual.txt,
runtime/doc/windows.txt, runtime/doc/tags, runtime/doc/Makefile

Patch 8.2.5001

Problem: Checking translations affects the search pattern history.
Solution: Use "keeppatterns". (Doug Kearns)
Files: src/po/check.vim

Patch 8.2.5002

Problem: deletebuflines() may change Visual selection.
Solution: Disable Visual mode when using another buffer. (closes #10469)
Files: src/evalbuffer.c, src/testdir/test_buflines.vim

Patch 8.2.5003

Problem: Cannot do bitwise shifts.
Solution: Add the >> and << operators. (Yegappan Lakshmanan, closes #8457)
Files: runtime/doc/eval.txt, src/errors.h, src/eval.c, src/structs.h,
src/vim.h, src/vim9execute.c, src/vim9expr.c,
src/testdir/test_expr.vim, src/testdir/test_vim9_disassemble.vim,
src/testdir/test_vim9_expr.vim

Patch 8.2.5004

Problem: Right shift on negative number does not work as documented.
Solution: Use a uvarnumber_T type cast.
Files: runtime/doc/eval.txt, src/eval.c, src/vim9expr.c,
src/vim9execute.c, src/charset.c, src/testdir/test_expr.vim

Patch 8.2.5005 (after 8.2.5003)

Problem: Compiler warning for uninitialized variable. (John Marriott)
Solution: Initialize the pointer to NULL.
Files: src/vim9expr.vim

Patch 8.2.5006 (after 8.2.5003)

Problem: Asan warns for undefined behavior.
Solution: Cast the shifted value to unsigned.

Files: src/eval.c, src/vim9expr.c, src/vim9execute.c

Patch 8.2.5007

Problem: Spell suggestion may use uninitialized memory. (Zdenek Dohnal)

Solution: Avoid going over the end of the word.

Files: src/spellsuggest.c, src/testdir/test_spell_utf8.vim

Patch 8.2.5008

Problem: When '**formatoptions**' contains "/" wrongly wrapping a long trailing comment.

Solution: Pass the OPENLINE_FORMAT flag.

Files: src/change.c, src/vim.h, src/textformat.c,
src/testdir/test_textformat.vim

Patch 8.2.5009

Problem: Fold may not be closable after appending.

Solution: Set the fd_small flag to MAYBE. (Brandon Simmons, closes #10471)

Files: src/fold.c, src/testdir/test_fold.vim

Patch 8.2.5010

Problem: The terminal debugger uses various global variables.

Solution: Add a dictionary to hold the terminal debugger preferences.

Files: runtime/doc/terminal.txt,
runtime/pack/dist/opt/termdebug/plugin/termdebug.vim

Patch 8.2.5011

Problem: Replacing an autocommand requires several lines.

Solution: Add the "replace" flag to autocmd_add(). (Yegappan Lakshmanan, closes #10473)

Files: runtime/doc/autocmd.txt, runtime/doc/builtin.txt, src/autocmd.c,
src/testdir/test_autocmd.vim

Patch 8.2.5012

Problem: Cannot select one character inside ().

Solution: Do not try to extend the area if it is empty. (closes #10472, closes #6616)

Files: src/textobject.c, src/testdir/test_textobjects.vim

Patch 8.2.5013

Problem: After text formatting the cursor may be in an invalid position.

Solution: Correct the cursor position after formatting.

Files: src/textformat.c, src/testdir/test_textformat.vim

Patch 8.2.5014

Problem: Byte offsets are wrong when using text properties.

Solution: Make sure text properties do not affect the byte counts.
(Paul Ollis, closes #10474)

Files: src/memline.c, src/textprop.c, src/testdir/test_textprop.vim

Patch 8.2.5015

Problem: Hoon and Moonscript files are not recognized.

Solution: Add filetype patterns. (Goc Dundar, closes #10478)

Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.5016

Problem: Access before start of text with a put command.
Solution: Check the length is more than zero.
Files: src/register.c, src/testdir/test_put.vim

Patch 8.2.5017

Problem: Gcc 12.1 warns for uninitialized variable.
Solution: Initialize the variable. (closes #10476)
Files: src/evalvars.c

Patch 8.2.5018

Problem: Vim9: some code is not covered by tests.
Solution: Delete dead code.
Files: src/vim9instr.c, src/proto/vim9instr.pro, src/vim9compile.c,
src/vim9expr.c,

Patch 8.2.5019

Problem: Cannot get the first screen column of a character.
Solution: Let virtcol() optionally return a list. (closes #10482,
closes #7964)
Files: runtime/doc/builtin.txt, src/evalfunc.c,
src/testdir/test_functions.vim, src/testdir/test_vim9_builtin.vim

Patch 8.2.5020

Problem: Using **'imstatusfunc'** and **'imactivatefunc'** breaks **'foldopen'**.
Solution: Save and restore the KeyTyped flag. (closes #10479)
Files: src/gui_xim.c, src/testdir/test_iminsert.vim

Patch 8.2.5021

Problem: Build fails with normal features and +terminal. (Dominique Pellé)
Solution: Add #ifdefs. (closes #10484)
Files: src/terminal.c

Patch 8.2.5022

Problem: **'completefunc'**/**'omnifunc'** error does not end completion.
Solution: Check if there was an error or exception. (closes #10486,
closes #4218)
Files: src/insexpand.c, src/testdir/test_ins_complete.vim

Patch 8.2.5023

Problem: Substitute overwrites allocated buffer.
Solution: Disallow undo when in a substitute command.
Files: src/normal.c, src/undo.c, src/testdir/test_substitute.vim

Patch 8.2.5024

Problem: Using freed memory with "]d".
Solution: Copy the pattern before searching.
Files: src/normal.c, src/testdir/test_tagjump.vim

Patch 8.2.5025

Problem: Vim9: a few lines not covered by tests.
Solution: Add a few tests.
Files: src/vim9script.c, src/testdir/test_vim9_assign.vim,

src/testdir/test_vim9_import.vim

Patch 8.2.5026

Problem: Vim9: a few lines not covered by tests.
Solution: Delete dead code. Add a few test cases. make "12->func()" work.
Files: src/vim9type.c, src/ex_docmd.c, src/proto/ex_docmd.pro,
src/vim9compile.c, src/testdir/test_vim9_assign.vim,
src/testdir/test_vim9_func.vim

Patch 8.2.5027

Problem: Error for missing :endif when an exception was thrown. (Dani Dickstein)
Solution: Do not give an error when aborting. (closes #10490)
Files: src/ex_docmd.c, src/testdir/test_trycatch.vim

Patch 8.2.5028

Problem: Syntax regexp matching can be slow.
Solution: Adjust the counters for checking the timeout to check about once per msec. (closes #10487, closes #2712)
Files: src/regexp_bt.c, src/regexp_nfa.c

Patch 8.2.5029

Problem: "textlock" is always zero.
Solution: Remove "textlock" and rename "textwinlock" to "textlock". (closes #10489)
Files: runtime/doc/insert.txt, runtime/doc/tags, src/beval.c,
src/change.c, src/edit.c, src/errors.h, src/eval.c, src/ex_cmds.c,
src/ex_getln.c, src/proto/ex_getln.pro, src/globals.h,
src/indent.c, src/insexpand.c, src/map.c, src/register.c,
src/undo.c, src/window.c, src/testdir/test_edit.vim,
src/testdir/test_ins_complete.vim, src/testdir/test_popup.vim,
src/testdir/test_quickfix.vim

Patch 8.2.5030

Problem: autocmd_add() can only handle one event and pattern.
Solution: Support a list of events and patterns. (Yegappan Lakshmanan, closes #10483)
Files: runtime/doc/builtin.txt, src/autocmd.c, src/errors.h,
src/testdir/test_autocmd.vim

Patch 8.2.5031

Problem: Cannot easily run the benchmarks.
Solution: Have "make benchmark" in the src directory work.
Files: src/Makefile, src/testdir/Makefile

Patch 8.2.5032

Problem: Python 3 test fails without the GUI.
Solution: Check the balloon_eval feature is available.
Files: src/testdir/test_python3.vim

Patch 8.2.5033 (after 8.2.5030)

Problem: Build error with +eval but without +quickfix. Warning for uninitialized variable.
Solution: Adjust #ifdefs. (John Marriott)

Files: src/errors.h, src/autocmd.c

Patch 8.2.5034

Problem: There is no way to get the byte index from a virtual column.

Solution: Add virtcol2col(). (Yegappan Lakshmanan, closes #10477, closes #10098)

Files: runtime/doc/builtin.txt, runtime/doc/usr_41.txt, src/evalfunc.c, src/move.c, src/proto/move.pro, src/testdir/test_cursor_func.vim

Patch 8.2.5035

Problem: When splitting a window the changelist position moves.

Solution: Set the changelist index a bit later. (closes #10493)

Files: src/window.c, src/testdir/test_changelist.vim, src/testdir/test_normal.vim

Patch 8.2.5036 (after 8.2.5028)

Problem: Using two counters for timeout check in NFA engine.

Solution: Use only one counter. Tune the counts based on guessing.

Files: src/regexp_nfa.c

Patch 8.2.5037

Problem: Cursor position may be invalid after "0;" range.

Solution: Check the cursor position when it was set by ";" in the range.

Files: src/ex_docmd.c, src/testdir/test_excmd.vim

Patch 8.2.5038

Problem: A finished terminal in a popup window does not show a scrollbar.

Solution: Show the scrollbar if the terminal job is finished. (closes #10497)

Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_poptermscroll_1.dump, src/testdir/dumps/Test_popupwin_poptermscroll_2.dump, src/testdir/dumps/Test_popupwin_poptermscroll_3.dump

Patch 8.2.5039

Problem: Confusing error if first argument of popup_create() is wrong.

Solution: Give a more informative error.

Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/errors.h, src/testdir/dumps/Test_popup_settext_07.dump

Patch 8.2.5040

Problem: Scrollbar thumb in scrolled popup not visible.

Solution: Show at least one thumb character. (fixes 10492)

Files: src/popupwin.c, src/testdir/test_popupwin.vim, src/testdir/dumps/Test_popupwin_scroll_13.dump

Patch 8.2.5041

Problem: Cannot close a terminal popup with "NONE" job.

Solution: Adjust the conditions for whether a job is running. (closes #10498)

Files: src/buffer.c, src/terminal.c, src/proto/terminal.pro, src/undo.c, src/testdir/test_popupwin.vim

Patch 8.2.5042

Problem: Scrollbar thumb in tall scrolled popup not visible.
Solution: Show at least one thumb character. (fixes 10492)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_scroll_13.dump

Patch 8.2.5043

Problem: Can open a cmdline window from a substitute expression.
Solution: Disallow opening a command line window when text or buffer is locked.
Files: src/buffer.c, src/ex_getln.c, src/proto/ex_getln.pro,
src/window.c, src/testdir/test_substitute.vim

Patch 8.2.5044 (after 8.2.5043)

Problem: Command line test fails.
Solution: Also beep when cmdline win can't be opened because of locks.
Make the test not beep. Make the test pass on MS-Windows.
Files: src/ex_getln.c, src/testdir/test_substitute.vim

Patch 8.2.5045

Problem: Can escape a terminal popup window when the job is finished.
Solution: Only check for a finished job where it is relevant.
(closes #10253)
Files: src/popupwin.c, src/testdir/test_popupwin.vim,
src/testdir/dumps/Test_popupwin_poptermscroll_1.dump,
src/testdir/dumps/Test_popupwin_poptermscroll_2.dump,
src/testdir/dumps/Test_popupwin_poptermscroll_3.dump,
src/testdir/dumps/Test_popupwin_poptermscroll_4.dump

Patch 8.2.5046

Problem: vim_regsub() can overwrite the destination.
Solution: Pass the destination length, give an error when it doesn't fit.
Files: src/regexp.h, src/regexp.c, src/proto/regexp.pro, src/eval.c,
src/ex_cmds.c

Patch 8.2.5047

Problem: CurSearch highlight is often wrong.
Solution: Remember the last highlighted position and redraw when needed.
Files: src/globals.h, src/match.c, src/drawscreen.c, src/change.c,
src/testdir/test_search.vim,
src/testdir/dumps/Test_hlsearch_cursearch_changed_1.dump

Patch 8.2.5048

Problem: When using XIM the gui test may fail.
Solution: Only use --not-a-term when not using XIM.
Files: src/testdir/test_gui.vim

Patch 8.2.5049

Problem: Insufficient tests for autocommands.
Solution: Add a few more tests. (Yegappan Lakshmanan, closes #10507)
Files: src/autocmd.c, src/testdir/gen_opt_test.vim,
src/testdir/test_autocmd.vim, src/testdir/test_cmdline.vim

Patch 8.2.5050

Problem: Using freed memory when searching for pattern in path.

Solution: Make a copy of the line.
Files: src/search.c, src/testdir/test_tagjump.vim

Patch 8.2.5051

Problem: Check for autocmd_add() event argument is confusing.
Solution: Make the code more straightforward.
Files: src/autocmd.c

Patch 8.2.5052

Problem: CI checkout step title is a bit cryptic.
Solution: Add a better title. (closes #10509)
Files: .github/workflows/ci.yml, .github/workflows/coverity.yml,
.github/workflows/codeql-analysis.yml

Patch 8.2.5053

Problem: Cannot have a comment halfway an expression in an autocmd command block.
Solution: When skipping over the NL also skip over comments. (closes #10519)
Files: src/eval.c, src/testdir/test_autocmd.vim

Patch 8.2.5054

Problem: No good filetype for conf files similar to dosini.
Solution: Add the confini filetype. (closes #10518)
Files: runtime/filetype.vim, src/testdir/test_filetype.vim

Patch 8.2.5055

Problem: Statusline is not updated when terminal title changes.
Solution: Redraw the status line when the title changes. (issue #10425)
Files: src/terminal.c

Patch 8.2.5056

Problem: The channel log only contains some of the raw terminal output.
Solution: Add the "o" flag to log all terminal output. Use it for "--log".
Files: runtime/doc/channel.txt, runtime/doc/starting.txt, src/main.c,
src/channel.c, src/vim.h, src/term.c, src/edit.c, src/normal.c,
src/optionstr.c

Patch 8.2.5057

Problem: Using gettimeofday() for timeout is very inefficient.
Solution: Set a platform dependent timer. (Paul Ollis, closes #10505)
Files: src/auto/configure, src/config.h.in, src/configure.ac,
src/drawscreen.c, src/errors.h, src/evalfunc.c, src/ex_cmds.c,
src/ex_getln.c, src/match.c, src/os_mac.h, src/os_macosx.m,
src/os_unix.c, src/os_win32.c, src/proto/os_unix.pro,
src/proto/os_win32.pro, src/proto/regexp.pro, src/quickfix.c,
src/regexp.c, src/regexp.h, src/regexp_bt.c, src/regexp_nfa.c,
src/screen.c, src/search.c, src/structs.h, src/syntax.c,
src/testdir/test_hlsearch.vim, src/testdir/test_search.vim,
src/testdir/test_syntax.vim

Patch 8.2.5058

Problem: input() does not handle composing characters properly.
Solution: Use mb_cpnr2char_adv() instead of mb_ptr2char_adv().
(closes #10527)

Files: src/getchar.c, src/testdir/test_functions.vim

Patch 8.2.5059

Problem: Autoconf 2.71 produces many obsolete warnings.

Solution: Replace obsolete macros with non-obsolete ones, where the functionality does not change. (issue #10528)

Files: src/configure.ac, src/auto/configure

Patch 8.2.5060 (after 8.2.5059)

Problem: Running configure fails.

Solution: Remove line break.

Files: src/configure.ac, src/auto/configure

Patch 8.2.5061

Problem: C89 requires signal handlers to return void.

Solution: Drop RETSIGTYPE and hard-code a void return value.

Files: src/configure.ac, src/auto/configure, src/if_cscope.c, src/os_unix.c, src/pty.c, src/os_mac.h, src/os_vms_conf.h, src/config.h.in, src/osdef1.h.in

Patch 8.2.5062

Problem: Coverity warns for dead code.

Solution: Remove the dead code.

Files: src/os_unix.c, src/match.c

Patch 8.2.5063

Problem: Error for a command may go over the end of IObuff.

Solution: Truncate the message.

Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.5064

Problem: No test for what 8.1.0052 fixes.

Solution: Add a test. (closes #10531)

Files: src/getchar.c, src/testdir/test_mapping.vim

Patch 8.2.5065

Problem: Wrong return type for main() in tee.c.

Solution: Use "int" instead of "void". Remove unused variable.

Files: src/tee/tee.c

Patch 8.2.5066

Problem: Can specify multispace listchars only for whole line.

Solution: Add "leadmultispace". (Christian Brabandt, closes #10496)

Files: runtime/doc/options.txt, src/drawline.c, src/message.c, src/screen.c, src/structs.h, src/window.c, src/testdir/test_listchars.vim

Patch 8.2.5067

Problem: Timer_create is not available on every Mac system. (Hisashi T Fujinaka)

Solution: Adjust #ifdef.

Files: src/os_unix.c

Patch 8.2.5068

Problem: Gcc 12.1 warning when building tee.
Solution: Change type to size_t. (John Marriott)
Files: src/tee/tee.c

Patch 8.2.5069

Problem: Various warnings from clang on MS-Windows.
Solution: Fix the code to avoid the warnings. (Yegappan Lakshmanan, closes #10538)
Files: src/dosinst.c, src/fileio.c, src/gui_w32.c, src/os_mswin.c, src/os_win32.c

Patch 8.2.5070

Problem: Unnecessary code.
Solution: Remove code that isn't needed. (closes #10534)
Files: src/message.c, src/screen.c

Patch 8.2.5071

Problem: With some Mac OS version clockid_t is redefined.
Solution: Adjust #ifdefs. (Ozaki Kiichi, closes #10549)
Files: src/os_mac.h

Patch 8.2.5072

Problem: Using uninitialized value and freed memory in spell command.
Solution: Initialize "attr". Check for empty line early.
Files: src/spell.c, src/testdir/test_spell_utf8.vim

Patch 8.2.5073

Problem: Clang on MS-Windows produces warnings.
Solution: Avoid the warnings. (Yegappan Lakshmanan, closes #10546)
Files: src/dosinst.c, src/dosinst.h, src/gui_dwrite.cpp, src/gui_w32.c, src/iscypty.c, src/libvterm/src/vterm_internal.h, src/mbyte.c, src/os_win32.c, src/os_win32.h, src/term.c, src/xdiff/xinclude.h

Patch 8.2.5074

Problem: Spell test fails on MS-Windows.
Solution: Do not change **'encoding'**
Files: src/testdir/test_spell_utf8.vim

Patch 8.2.5075

Problem: Clang gives an out of bounds warning.
Solution: adjust conditional expression (John Marriott)
Files: src/ui.c

Patch 8.2.5076

Problem: Unnecessary code.
Solution: Remove code and replace with function call. (closes #10552)
Files: src/drawline.c, src/getchar.c

Patch 8.2.5077

Problem: Various warnings from clang on MS-Windows.
Solution: Avoid the warnings. (Yegappan Lakshmanan, closes #10553)
Files: src/dosinst.c, src/dosinst.h, src/filepath.c, src/gui_w32.c, src/misc1.c, src/os_win32.c

Patch 8.2.5078

Problem: Substitute test has a one second delay.
Solution: Use ":silent!". Add another test case. (closes #10558)
Files: src/testdir/test_substitute.vim

Patch 8.2.5079

Problem: DirChanged autocommand may use freed memory. (Shane-XB Qian)
Solution: Free the memory later. (closes #10555)
Files: src/ex_docmd.c, src/testdir/test_autocmd.vim

Patch 8.2.5080

Problem: When indenting gets out of hand it is hard to stop.
Solution: When line gets too long set got_int.
Files: src/indent.c

Patch 8.2.5081

Problem: Autocmd test fails on MS-Windows.
Solution: Set shellslash to get forward slashes.
Files: src/testdir/test_autocmd.vim

Patch 8.2.5082 (after 8.2.5080)

Problem: Retab test fails.
Solution: Disable the test for now.
Files: src/testdir/test_retab.vim

Patch 8.2.5083

Problem: Autocmd test still fails on MS-Windows.
Solution: Change backward to forward slashes.
Files: src/testdir/test_autocmd.vim

Patch 8.2.5084

Problem: When the GUI shows a dialog tests get stuck.
Solution: Add the --gui-dialog-file argument.
Files: runtime/doc/starting.txt, src/Make_mvc.mak, src/gui.c, src/main.c, src/message.c, src/os_mswin.c, src/proto/gui.pro, src/proto/main.pro, src/structs.h, src/testdir/Make_dos.mak, src/testdir/Make_ming.mak, src/testdir/Makefile, src/testdir/runtest.vim, src/testdir/shared.vim

Patch 8.2.5085

Problem: Gcc gives warning for signed/unsigned difference.
Solution: Use a different pointer type. (John Marriott)
Files: src/os_mswin.c

Patch 8.2.5086

Problem: CI runs on Windows 2019.
Solution: Switch to Windows 2022. (closes #10566)
Files: .github/workflows/ci.yml

Patch 8.2.5087

Problem: Cannot build with clang on MS-Windows.
Solution: Add support for building with clang. (Yegappan Lakshmanan, closes #10557)
Files: src/GvimExt/Make_ming.mak, src/INSTALLpc.txt,

src/Make_cyg_ming.mak

Patch 8.2.5088

Problem: Value of cmod_verbose is a bit complicated to use.
Solution: Use zero for not set, value + 1 when set. (closes #10564)
Files: src/ex_docmd.c, src/ex_getln.c, src/globals.h, src/structs.h

Patch 8.2.5089

Problem: Some functions return a different value on failure.
Solution: Initialize the return value earlier. (Yegappan Lakshmanan, closes #10568)
Files: src/autocmd.c, src/dict.c, src/evalfunc.c, src/list.c

Patch 8.2.5090

Problem: MS-Windows: vim.def is no longer used.
Solution: Delete vim.def. (Ken Takata, closes #10569)
Files: Filelist, Makefile, src/vim.def

Patch 8.2.5091

Problem: Terminal test fails with some shell commands.
Solution: Disable setting the window title. (closes #10530)
Files: src/testdir/test_terminal.vim

Patch 8.2.5092

Problem: Using "'<,'>" in Ex mode may compare unrelated pointers.
Solution: Set eap->cmd to "+" only later.
Files: src/ex_docmd.c

Patch 8.2.5093

Problem: Error message for unknown command may mention the command twice. (Malcolm Rowe)
Solution: Add the did_append_cmd flag. (closes #10570)
Files: src/ex_docmd.c

Patch 8.2.5094

Problem: MS-Windows GUI: empty command may cause a dialog.
Solution: Delete the dialog file. Improve the message.
Files: src/testdir/runtest.vim, src/testdir/test_ex_mode.vim

Patch 8.2.5095

Problem: Terminal test still fails with some shell commands.
Solution: Disable setting the window title in the Vim instance running in a terminal window. (closes #10530)
Files: src/testdir/test_terminal.vim

Patch 8.2.5096 (after 8.2.5095)

Problem: Terminal test still fails with some shell commands.
Solution: Add missing "call". (closes #10530)
Files: src/testdir/test_terminal.vim

Patch 8.2.5097

Problem: Using uninitialized memory when using '**listchars**'.
Solution: Use the length returned by mb_char2bytes(). (closes #10576)
Files: src/message.c

Patch 8.2.5098

Problem: Spelldump test sometimes hangs.
Solution: Catch the problem of the spell file not being found to avoid hanging in the download dialog.
Files: src/testdir/test_spell.vim

Patch 8.2.5099

Problem: Some terminal tests are not retried.
Solution: Mark terminal tests as flaky.
Files: src/testdir/test_terminal.vim

Patch 8.2.5100

Problem: Memory usage tests are not retried.
Solution: Mark memory usage tests as flaky.
Files: src/testdir/test_memory_usage.vim

Patch 8.2.5101

Problem: MS-Windows with MinGW: \$CC may be "cc" instead of "gcc".
Solution: Set \$CC if it is not matching "clang". (Yegappan Lakshmanan, closes #10578)
Files: src/INSTALLpc.txt, src/Make_cyg_ming.mak

Patch 8.2.5102

Problem: Interrupt not caught in test.
Solution: Consider an exception thrown in the current try/catch when got_int is set. Also catch early exit when not using try/catch.
Files: src/indent.c, src/testing.c, src/testdir/test_retab.vim, src/testdir/runtest.vim

Patch 8.2.5103

Problem: Build fails with small features.
Solution: Add #ifdef. Skip test on MS-Windows.
Files: src/indent.c, src/testdir/test_retab.vim

Patch 8.2.5104 (after 8.2.5103)

Problem: Test hangs on MS-Windows.
Solution: Skip another test on MS-Windows.
Files: src/testdir/test_retab.vim

Patch 8.2.5105 (after 8.2.5104)

Problem: Test still hangs on MS-Windows.
Solution: Skip "nocatch" test the right way.
Files: src/testdir/test_retab.vim

Patch 8.2.5106

Problem: Default cmdwin mappings are re-mappable.
Solution: Make the default mappings not re-mappable. (closes #10580) Use symbols for the first do_map() argument.
Files: src/vim.h, src/ex_getln.c, src/map.c, src/proto/map.pro, src/digraph.c, src/netbeans.c

Patch 8.2.5107

Problem: Some callers of rettv_list_alloc() check for not OK. (Christ van

Willegen)
Solution: Use "==" instead of "!=" when checking the return value.
Files: src/evalbuffer.c, src/channel.c, src/cmdexpand.c, src/evalfunc.c,
src/evalwindow.c, src/insexpand.c, src/job.c, src/list.c,
src/map.c, src/menu.c, src/mouse.c, src/move.c, src/sign.c,
src/textprop.c, src/term.c, src/time.c

Patch 8.2.5108

Problem: Retab test disabled because it hangs on MS-Windows.
Solution: Also set got_int at the other place an overlong text is detected.
Files: src/indent.c, src/testdir/test_retab.vim

Patch 8.2.5109

Problem: Mode not updated after **CTRL-O CTRL-C** in Insert mode.
Solution: Set redraw_mode and use it. (closes #10581)
Files: src/main.c, src/normal.c, src/testdir/test_normal.vim,
src/testdir/dumps/Test_mode_updated_1.dump

Patch 8.2.5110

Problem: Icon filetype not recognized from the first line.
Solution: Add a check for the first line. (Doug Kearns)
Files: runtime/autoload/dist/script.vim, src/testdir/test_filetype.vim

Patch 8.2.5111

Problem: No test for --gui-dialog-file.
Solution: Add a test.
Files: src/testdir/test_gui.vim

Patch 8.2.5112 (after 8.2.5111)

Problem: Gui test hangs on MS-Windows.
Solution: Use "!start" to start Vim.
Files: src/testdir/test_gui.vim

Patch 8.2.5113

Problem: Timer becomes invalid after fork/exec, :gui gives errors. (Gabriel Dupras)
Solution: Delete the timer before forking. (closes #10584)
Files: src/os_unix.c, src/proto/os_unix.pro, src/gui.c

Patch 8.2.5114

Problem: Time limit on searchpair() does not work properly.
Solution: Set the time limit once instead of for each regexp. (closes #10562)
Files: src/search.c, src/evalfunc.c, src/testdir/test_search.vim

Patch 8.2.5115

Problem: Search timeout is overrun with some patterns.
Solution: Check for timeout in more places. Make the flag volatile and atomic. Use assert_inrange() to see what happened.
Files: src/regexp_nfa.c, src/regexp_bt.c, src/regexp.c, src/os_unix.c,
src/proto/os_unix.pro, src/testdir/test_search.vim

Patch 8.2.5116

Problem: "limit" option of matchfuzzy() not always respected.
Solution: Remove "else". (Kazuyuki Miyagi, closes #10586)

Files: runtime/doc/builtin.txt, src/search.c,
src/testdir/test_matchfuzzy.vim

Patch 8.2.5117

Problem: Crash when calling a Lua callback from a :def function. (Bohdan Makohin)

Solution: Handle FC_CFUNC in call_user_func_check(). (closes #10587)

Files: src/userfunc.c, src/testdir/test_lua.vim

Patch 8.2.5118

Problem: MS-Windows: sending a message to another Vim may hang if that Vim is halted.

Solution: Add a timeout to serverSendToVim(). (Ken Takata, closes #10585)

Files: runtime/pack/dist/opt/editexisting/plugin/editexisting.vim,
src/os_mswin.c

Patch 8.2.5119

Problem: CI uses cache v2.

Solution: Use cache v3. (closes #10588)

Files: .github/workflows/ci.yml

Patch 8.2.5120

Problem: Searching for quotes may go over the end of the line.

Solution: Check for running into the NUL.

Files: src/textobject.c

Patch 8.2.5121

Problem: Interrupt test sometimes fails.

Solution: Use a different file name.

Files: src/testdir/test_interrupt.vim

Patch 8.2.5122

Problem: Lisp indenting my run over the end of the line.

Solution: Check for NUL earlier.

Files: src/indent.c, src/testdir/test_indent.vim

Patch 8.2.5123

Problem: Using invalid index when looking for spell suggestions.

Solution: Do not decrement the index when it is zero.

Files: src/spellsuggest.c, src/testdir/test_spell.vim

Patch 8.2.5124

Problem: When syntax timeout test fails it does not show the time.

Solution: Use assert_inrange().

Files: src/testdir/test_syntax.vim

Patch 8.2.5125

Problem: MS-Windows: warnings from MinGW compiler.

Solution: Use "volatile". (Yasuhiro Matsumoto, closes #10589) Initialize variable.

Files: src/os_win32.c, src/proto/os_win32.pro, src/map.c

Patch 8.2.5126

Problem: Substitute may overrun destination buffer.

Solution: Disallow switching buffers in a substitute expression.
Files: src/ex_docmd.c, src/testdir/test_substitute.vim

Patch 8.2.5127

Problem: Using assert_true() does not show value on failure.
Solution: Use assert_inrange(). (closes #10593)
Files: src/testdir/test_channel.vim, src/testdir/test_hlsearch.vim

Patch 8.2.5128

Problem: Syntax highlighting disabled when using synID() in searchpair()
skip expression and it times out. (Jaehwang Jung)
Solution: Add the redrawtime_limit_set flag. (closes #10562)
Files: src/globals.h, src/drawscreen.c, src/syntax.c

Patch 8.2.5129

Problem: Timeout handling is not optimal.
Solution: Avoid setting timeout_flag twice. Adjust the pointer when
stopping the regexp timeout. Adjust variable name.
Files: src/os_unix.c, src/os_win32.c, src/regexp.c

Patch 8.2.5130

Problem: Edit test for mode message fails when using valgrind.
Solution: Use WaitForAssert(). Run beep test later.
Files: src/testdir/test_edit.vim

Patch 8.2.5131

Problem: Timeout implementation is not optimal.
Solution: Further improvements for timeouts. Add a test for searchpair()
timeout. (partly by Paul Ollis)
Files: src/configure.ac, src/auto/configure,
src/testdir/test_hlsearch.vim, src/testdir/test_search.vim

Patch 8.2.5132

Problem: :mkview test doesn't test much.
Solution: Save the view with the folds closed. (James McCoy, closes #10596)
Files: src/testdir/test_mksession.vim

Patch 8.2.5133

Problem: MacOS: build fails.
Solution: Remove "#if 0" from timer_delete().
Files: src/os_macosx.m

Patch 8.2.5134

Problem: Function has confusing name.
Solution: Rename tgetent_error() to invoke_tgetent().
Files: src/term.c

Patch 8.2.5135

Problem: Running configure gives warnings for main() return type.
Solution: Specify "int" return type. Avoid a few more warnings.
Files: src/configure.ac, src/auto/configure

Patch 8.2.5136

Problem: Debugger test fails when run with valgrind.

Solution: Wait longer when using valgrind.
Files: src/testdir/shared.vim, src/testdir/test_debugger.vim,
src/testdir/test_search.vim

Patch 8.2.5137

Problem: Cannot build without the +channel feature. (Dominique Pellé)
Solution: Add #ifdef around ch_log() calls. (closes #10598)
Files: src/os_unix.c, src/regexp_nfa.c, src/regexp_bt.c

Patch 8.2.5138

Problem: Various small issues.
Solution: Various small improvements.
Files: src/filepath.c, src/job.c, src/mark.c, src/move.c,
src/popupwin.c, src/testdir/test_filetype.vim

Patch 8.2.5139

Problem: TIME_WITH_SYS_TIME is no longer supported by autoconf.
Solution: Always include time.h.
Files: src/os_unix.h

Patch 8.2.5140

Problem: Seachpair timeout test is flaky.
Solution: Mark the test as flaky so it is retried.
Files: src/testdir/test_search.vim

Patch 8.2.5141

Problem: Using "volatile int" in a signal handler might be wrong.
Solution: Use "volatile sig_atomic_t".
Files: src/os_unix.c, src/proto/os_unix.pro, src/os_win32.c,
src/proto/os_win32.pro, src/regexp.c,

Patch 8.2.5142

Problem: Startup test fails if there is a status bar at the top of the
screen. (Ernie Rael)
Solution: Use a larger vertical offset in the test.
Files: src/testdir/test_startup.vim

Patch 8.2.5143

Problem: Some tests fail when using valgrind. Spurious leak reports.
Solution: Use WaitForAssert(). Avoid failing fork/exec. Skip tests where a
job is killed when running valgrind.
Files: src/testdir/test_iminsert.vim, src/testdir/test_popup.vim,
src/testdir/test_cscope.vim, src/testdir/test_channel.vim

Patch 8.2.5144

Problem: With '**lazyredraw**' set completion menu may be displayed wrong.
Solution: When the popup menu is visible do not insert a screen line.
(closes #10601)
Files: src/screen.c

Patch 8.2.5145

Problem: Exit test causes spurious valgrind reports.
Solution: Skip test. Add CheckNotValgrind.
Files: src/testdir/test_exit.vim, src/testdir/check.vim,

src/testdir/test_channel.vim

Patch 8.2.5146

Problem: Memory leak when substitute expression nests.
Solution: Use an array of expression results.
Files: src/alloc.c, src/regexp.c, src/proto/regexp.pro,
src/errors.h, src/ex_cmds.c, src/testdir/test_substitute.vim

Patch 8.2.5147

Problem: Flaky test always fails on retry.
Solution: Delete the created function.
Files: src/testdir/test_search.vim

Patch 8.2.5148

Problem: Invalid memory access when using an expression on the command line.
Solution: Make sure the position does not go negative.
Files: src/ex_getln.c, src/testdir/test_cmdline.vim

Patch 8.2.5149 (after 8.2.5148)

Problem: Cannot build without the +eval feature. (Tony Mechelynck)
Solution: Add #ifdefs.
Files: src/ex_getln.c

Patch 8.2.5150

Problem: Read past the end of the first line with ":0;{'".
Solution: When on line zero check the column is valid for line one.
Files: src/ex_docmd.c, src/testdir/test_cmdline.vim

Patch 8.2.5151

Problem: Reading beyond the end of the line with lisp indenting.
Solution: Avoid going over the NUL at the end of the line.
Files: src/indent.c, src/testdir/test_lispwords.vim

Patch 8.2.5152

Problem: search() gets stuck with "c" and skip evaluates to true.
Solution: Reset the SEARCH_START option. (closes #10608)
Files: src/evalfunc.c, src/testdir/test_syntax.vim

Patch 8.2.5153

Problem: "make uninstall" does not remove colors/lists.
Solution: Add a line to the Makefile. (closes #10609)
Files: src/Makefile

Patch 8.2.5154

Problem: Still mentioning version8, some cosmetic issues.
Solution: Prefer mentioning version9, cosmetic improvements.
Files: src/version.c, src/if_tcl.c, src/regexp.c,
src/testdir/test_gui.vim, src/os_unix.c, Filelist, src/Makefile

Patch 8.2.5155

Problem: In diff mode windows may get out of sync. (Gary Johnson)
Solution: Avoid that the other window scrolls for '**cursorbind**'.
Files: src/move.c, src/testdir/test_diffmode.vim,
src/testdir/dumps/Test_diff_scroll_1.dump,

src/testdir/dumps/Test_diff_scroll_2.dump

Patch 8.2.5156

Problem: Search timeout test often fails with FreeBSD.
Solution: Double the maximum time.
Files: src/testdir/test_search.vim

Patch 8.2.5157

Problem: MS-Windows GUI: CTRL-key combinations do not always work.
Solution: Handle special key combinations better. (closes #10613,
closes #10602, closes #10579)
Files: src/gui_w32.c

Patch 8.2.5158

Problem: TSTP and INT signal tests are not run with valgrind.
Solution: Sleep a bit longer. (closes #10614)
Files: src/testdir/test_signals.vim

Patch 8.2.5159 (after 8.2.5157)

Problem: Fix for CTRL-key combinations causes more problems than it solves.
Solution: Roll back the change.
Files: src/gui_w32.c

Patch 8.2.5160

Problem: Accessing invalid memory after changing terminal size.
Solution: Adjust cmdline_row and msg_row to the value of Rows.
Files: src/term.c

Patch 8.2.5161

Problem: Might still access invalid memory.
Solution: Add extra check for negative value.
Files: src/message.c

Patch 8.2.5162

Problem: Reading before the start of the line with BS in Replace mode.
Solution: Check the cursor column is more than zero.
Files: src/edit.c

Patch 8.2.5163

Problem: Crash when deleting buffers in diff mode.
Solution: Recompute diffs later. Skip window without a valid buffer.
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.2.5164

Problem: Invalid memory access after diff buffer manipulations.
Solution: Use zero offset when change removes all lines in a diff block.
Files: src/diff.c, src/testdir/test_diffmode.vim

Patch 8.2.5165

Problem: Import test fails because 'diffexpr' isn't reset.
Solution: Reset 'diffexpr'.
Files: src/testdir/test_vim9_import.vim

Patch 8.2.5166

Problem: Test for DiffUpdated fails.
Solution: Also accept a count of two.
Files: src/testdir/test_diffmode.vim

Patch 8.2.5167

Problem: get(Fn, 'name') on funcref returns special byte code.
Solution: Use the printable name.
Files: src/evalfunc.c, src/testdir/test_getvar.vim

Patch 8.2.5168

Problem: Cannot build with Python 3.11.
Solution: Adjust define for _PyObject_TypeCheck. (Zdenek Dohnal, closes #10627)
Files: src/if_python3.c

Patch 8.2.5169

Problem: Nested :source may use NULL pointer.
Solution: Do not use the NULL pointer.
Files: src/eval.c, src/testdir/test_vimscript.vim

Patch 8.2.5170

Problem: Tiny issues.
Solution: Tiny improvements.
Files: src/misc1.c, src/screen.c, src/vim.h

Patch 8.2.5171

Problem: Dependencies and proto files are outdated.
Solution: Update dependencies and proto files. Avoid errors.
Files: src/Makefile, src/gui_w32.c, src/os_mswin.c, src/proto/buffer.pro, src/proto/cmdexpand.pro, src/proto/getchar.pro, src/proto/help.pro, src/proto/mbyte.pro, src/proto/option.pro, src/proto/screen.pro, src/proto/syntax.pro, src/proto/textformat.pro, src/proto/textobject.pro, src/proto/time.pro, src/proto/gui_x11.pro, src/proto/if_tcl.pro, src/proto/os_mswin.pro

Patch 8.2.5172

Problem: "make menu" still uses legacy script.
Solution: make menu generation script use Vim9 script, fix errors.
Files: runtime/makemenu.vim

VERSION 9.1

version-9.1 version9.1 vim-9.1

This section is about improvements made between version 9.0 and 9.1. The release 9.1 is dedicated to Vim's Benevolent dictator for life [Bram-Moolenaar](#) .

This release has hundreds of bug fixes, there are a few new features and there are many minor improvements.

Vim9 classes

Support for classes and objects in a Vim9 script are added. This is described in `vim9-class`. The following features are supported:

- Defining classes and instantiating objects
- Multiple constructors
- Class variables and methods
- Object variables and methods
- Public and protected access for variables and methods
- Constant and final object/class variables
- Extending classes (single inheritance)
- Interfaces
- Abstract classes
- Exporting classes

Support for creating a type alias for an existing type is added.

Virtual text

Support for adding `virtual-text` to a buffer is added. This is useful for language server features (e.g. inlay hints)

Smooth Scroll

Support for scrolling text using screen lines instead of file lines is added. Refer to the `'smoothscroll'` option.

The EditorConfig (`editorconfig-install`) and the JSON formatting (`ft-json-plugin`) plugins are included.

OpenVMS x86_64 platform port: <http://www.polarhome.com/vim/>

Other improvements

`new-other-9.1`

- Support for undercurl (`t_Ce`), double underline (`t_Us`), dotted underline (`t_ds`) and dashed underline (`t_Ds`) termcap entries and highlight modes (`highlight-term`).
- The `'fillchars'` option is now a global-local option and allows to specify the character used for the last window line.
- The `:vertical` and `:horizontal` modifiers can be used with `":wincmd ="` command to equalize windows vertically or horizontally.
- `:defer` command to defer the execution of a function till the end of a function.
- Support for closing a tab page using the middle mouse button.
- Sound support in MacOS.
- The `prop_remove()` function can now remove multiple text properties.
- The `items()` function now supports a List or a String argument.
- The `repeat()` function now supports blob repeat.
- Command line completion support for `:runtime` command.
- Support for custom completion using `getcompletion()`.
- Command-line completion support for option values.
- Support for multiple `"%="` items in `'statusline'`.
- Support for the `PmenuKind`, `PmenuKindSel`, `PmenuExtra` and `PmenuExtraSel` highlight groups.
- Text properties can override `'cursorline'` highlight.
- Support for echoing messages in a popup window `:echowindow`.

- Control scroll behavior when splitting windows **'splitkeep'**.
- Support for mouse scrolling in the MS-Windows console.
- Support for using different keyboard protocols **'keyprotocol'** (xterm `modifyOtherKeys` and Kitty terminal protocol (CSI u) support).
- Support for UTF-16 string index.
- Quickfix list items can have associated custom user data.
- `reverse()` supports reversing a string.
- **'switchbuf'** is used by more commands.
- `undotree()` can be used with any buffer.
- `printf()` supports positional arguments.
- `col()`, `charcol()` and `virtcol()` can be used with different windows.
- Support for Python3 stable ABI (`python-stable`).
- Use `jumplist` like a stack **'jumpoptions'**.
- Support for writing extended attributes in Linux (`xattr`).
- **'errorformat'** supports parsing a buffer number ("%b").
- Support for building the python interface using Python 3.12.
- `matchaddpos()` is no longer limited to 8 positions.
- `readblob()` allows to read part of a file.
- **'shortmess'** allows to suppress "scanning" messages.
- Add the XChaCha20v2 encryption method **'cryptmethod'**.
- Add support for testing syntax files.
- Support "*" for filename expansion with bash using globstar setting.
- Add quadruple prime digraph using 4'.
- Better high dpi support on Windows.
- Various fixes for unsafe memory access, memory leaks, buffer overflows and potential crashes.
- Termdebug: Support for the variables window (`termdebug_variables_window`), moving up/down the stack frames (`termdebug-frames`) and setting temporary breakpoints (`:Tbreak`).
- xxd: support for using colors in the hex dump output (`xxd -R`).
- xxd: reversing a bit dump (`xxd -r`).
- xxd: customize the variable name used in the C include output (`xxd -n`).

Changed

changed-9.1

- The features `++builtin_terms`, `+cmdline_info`, `+cmdwin`, `+file_in_path`, `+float`, `+path_extra`, `+textobjects`, `+wildignore` and `+wildmenu` are available in all the builds.
- Support for Windows-XP is dropped.
- Support for VisVim is removed.
- The "small" and "big" builds are dropped in favor of "tiny" and "huge" builds.
- Mention the detected **'keyprotocol'** on ":verbose map" when listing mappings.
- The optional Content-Type header is removed from the LSP messages.
- `charidx()` returns the character length when the index is one more than the last byte index.
- Ctrl-Q works like Ctrl-V in replace mode.
- `popup_filter_menu()` now wraps around.
- `popup_create()` now aborts on an error.
- `g<End>` now jumps to the last non-blank character.
- dot and hyphen are supported in highlight group names.
- `executable()` resolves symlinks on MS-Windows.
- Ruby 1.8 support is dropped.
- **'ttyfast'** is always set.

- `virtcol2col()` returns the first byte for a multi-byte character.
- `i_CTRL-0` resets Select Mode.
- `mode()` returns more submodes.
- `r` with **CTRL-C** on a visual area replaces using **CTRL-C** consistently.
- The `matchparen` plugin no longer uses hard coded match id 3.
- The Statusline no longer uses hard coded values `"^"` and `"="` if the highlighting groups for the statusline have been cleared.
- `'maxfuncdepth'` setting is also used for the maximum callback depth.
- Migrate to `autoconf 2.71`.
- Start using C99 feature (declare variable in for loops).

Added

added-9.1

Various syntax, indent and other plugins were added.

Functions:

<code>err_teapot()</code>	produce error 418 or 503
<code>getbufoneline()</code>	get a single line from the specified buffer
<code>getcellwidths()</code>	get character cell width overrides
<code>getmouseshape()</code>	get name of the current mouse shape
<code>getscriptinfo()</code>	get list of sourced vim scripts
<code>indexof()</code>	index in a List or Blob of a true expression
<code>instanceof()</code>	check if a variable is an instance of a given class
<code>keytrans()</code>	translate internal key codes to be usable with <code>:map</code>
<code>popup_findecho()</code>	get window ID for popup used for <code>:echowindow</code>
<code>setcmdline()</code>	set the current command line
<code>strutf16len()</code>	number of UTF-16 code units in a string
<code>swapfilelist()</code>	list of existing swap files in <code>'directory'</code>
<code>test_mswin_event()</code>	generate an MS-Windows event for testing
<code>utf16idx()</code>	UTF-16 index of a byte in a string

Autocommands:

<code>TextChangedT</code>	after a change was made to the text in Terminal mode
<code>WinResized</code>	after a window in the current tab page is resized

Commands:

<code>:abstract</code>	define a Vim9 abstract class
<code>:class</code>	start of a class specification
<code>:defer</code>	call function when current function is done
<code>:echowindow</code>	same as <code>:echomsg</code> , but use a popup window
<code>:endinterface</code>	end of an interface specification
<code>:endclass</code>	end of a class specification
<code>:horizontal</code>	following window command works horizontally
<code>:interface</code>	start of an interface specification
<code>:public</code>	prefix for a class or object member
<code>:static</code>	prefix for a class member or function
<code>:this</code>	prefix for an object member
<code>:type</code>	create a type alias

Options:

'endoffile'	write CTRL-Z at end of the file
'jumpoptions'	specifies how jumping is done
'keyprotocol'	what keyboard protocol to use for what terminal
'lispoptions'	changes how Lisp indenting is done
'showcmdloc'	where to show (partial) command
'smoothscroll'	scroll by screen lines when 'wrap' is set
'splitkeep'	determines scroll behavior for split windows

PATCHES

[patches-9.1](#) [bug-fixes-9.1](#)
[patches-after-9.0](#)

The list of patches that got included since 9.0.0. This includes all the new features, but does not include runtime file changes (syntax, indent, ftpplugin, documentation, etc.)

Note: authorship and the list of changed files is left out and only visible through `git log` for each commit.

Patch 9.0.0001

Problem: Travis CI is no longer used.

Solution: Delete the Travis CI configuration. (Hugo Osvaldo Barrera, closes #10636)

Patch 9.0.0002

Problem: Map functionality outside of map.c.

Solution: Move f_hasmapto() to map.c. Rename a function. (closes #10611)

Patch 9.0.0003

Problem: Functions are global while they could be local.

Solution: Add "static". Add a few tests. (Yegappan Lakshmanan, closes #10612)

Patch 9.0.0004

Problem: Plural messages not translated properly.

Solution: Use ngettext() in a few more places. (Matvey Tarasov, closes #10606)

Patch 9.0.0005

Problem: Hare files are not recognized.

Solution: Add a filetype pattern. (Hugo Osvaldo Barrera, closes #10630)

Patch 9.0.0006

Problem: Not all Visual Basic files are recognized.

Solution: Change detection of *.cls files. (Doug Kearns)

Patch 9.0.0007

Problem: No support for double, dotted and dashed underlines.

Solution: Add the termcap entries and highlight modes. (closes #9553)

Patch 9.0.0008

Problem: Cannot specify the variable name for "xxd -i".

Solution: Add the "-name" argument. (David Gow, closes #10599)

Patch 9.0.0009

Problem: Going past the end of a menu item with only modifier.

Solution: Check for NUL.

Patch 9.0.0010

Problem: Returning 0 for has('patch-9.0.0') is inconsistent.

Solution: Make it return 1. (closes #10640)

Patch 9.0.0011

Problem: Reading beyond the end of the line with put command.

Solution: Adjust the end mark position.

Patch 9.0.0012

Problem: Signature files not detected properly.

Solution: Add a function to better detect signature files. (Doug Kearns)

Patch 9.0.0013

Problem: Reproducing memory access errors can be difficult.

Solution: When testing, copy each line to allocated memory, so that valgrind can detect accessing memory before and/or after it. Fix uncovered problems.

Patch 9.0.0014

Problem: Missing part of the test override change.

Solution: Add the missing part.

Patch 9.0.0015

Problem: With EXITFREE defined terminal menus are not cleared.

Solution: Also clear terminal menus. Remove condition that is always true. (closes #10641)

Patch 9.0.0016

Problem: Comparing line pointer for '**breakindent**' is not reliable.

Solution: Make a copy of the line.

Patch 9.0.0017

Problem: Accessing memory beyond the end of the line.

Solution: Stop Visual mode when closing a window.

Patch 9.0.0018

Problem: Going over the end of the typeahead.

Solution: Put a NUL after the typeahead.

Patch 9.0.0019

Problem: Timers test not run where possible.

Solution: Adjust platform checks. (closes #10645)

Patch 9.0.0020

Problem: With some completion reading past end of string.

Solution: Check the length of the string.

Patch 9.0.0021

Problem: Invalid memory access when adding word with a control character to the internal spell word list.

Solution: Disallow adding a word with control characters or a trailing slash.

Patch 9.0.0022

Problem: Spell test fails.

Solution: Expect new error is given.

Patch 9.0.0023

Problem: On Solaris timer_create() exists but does not work.

Solution: Adjust the configure check to run the test program. (closes #10647)

Patch 9.0.0024

Problem: May access part of typeahead buf that isn't filled.

Solution: Check length of typeahead.

Patch 9.0.0025

Problem: Accessing beyond allocated memory when using the cmdline window in Ex mode.

Solution: Use "*" instead of "<,>" for Visual mode.

Patch 9.0.0026

Problem: Accessing freed memory with diff put.

Solution: Bail out when diff pointer is no longer valid.

Patch 9.0.0027

Problem: The command line test is getting quite big.

Solution: Move command line window tests to a separate file.

Patch 9.0.0028

Problem: MS-Windows: tests fail if there is a stray "runtime" directory.

Solution: Only use a "runtime" directory if it contains "defaults.vim".

Patch 9.0.0029

Problem: The bitmaps/vim.ico file is not in the distribution.

Solution: Add it back to the distribution. Adjust the build rules to have it end up in the right place.

Patch 9.0.0030

Problem: Matchfuzzy test depends on path of current directory.

Solution: Use fnamemodify() to remove the path. (Robin Becker, closes #10650)

Patch 9.0.0031

Problem: <cmud> of user command does not have correct verbose value.

Solution: Use the value from the command modifier. (closes #10651)

Patch 9.0.0032

Problem: In the quickfix window 'cursorline' overrules QuickFixLine highlighting.

Solution: Combine the attributes. Add a test. (closes #10654)

Patch 9.0.0033

Problem: On a Belgian keyboard CTRL-[does not work.

Solution: Handle GDK_KEY_dead_circumflex. (Anton Sharonov, closes #10658)

Patch 9.0.0034

Problem: Spell tests do not always clear the word list.

Solution: Clear the word list in TearDown(). (closes #10659)

Patch 9.0.0035

Problem: Spell dump may go beyond end of an array.

Solution: Limit the word length.

Patch 9.0.0036

Problem: **'fillchars'** cannot have window-local values.

Solution: Make **'fillchars'** global-local. (closes #5206)

Patch 9.0.0037

Problem: Build error.

Solution: Add missing change.

Patch 9.0.0038

Problem: **'listchars'** test fails.

Solution: Use window-local value after setting the global value

Patch 9.0.0039

Problem: Not all systems have GDK_KEY_dead_circumflex. (Hisashi T Fujinaka)

Solution: Add an #ifdef.

Patch 9.0.0040

Problem: Use of set_chars_option() is confusing.

Solution: Add "apply" argument to store the result or not. Merge similar code.

Patch 9.0.0041

Problem: A couple of filetype patterns do not have "*" before "/etc".

Solution: Add the star. (Jonas Strittmatter, closes #10662)

Patch 9.0.0042

Problem: Missing change for filetype detection.

Solution: Include change to detect guile from shebang line.

Patch 9.0.0043

Problem: Insufficient testing for bracket commands.

Solution: Add a few more tests. (closes #10668)

Patch 9.0.0044

Problem: Typos in comments, wrapping lines.

Solution: Adjust comments. Wrap lines.

Patch 9.0.0045

Problem: Reading past end of completion with a long line and **'infercase'** set.

Solution: Allocate the string if needed.

Patch 9.0.0046

Problem: Reading past end of completion with duplicate match.

Solution: Check string length

Patch 9.0.0047

Problem: Using freed memory with recursive substitute.

Solution: Always make a copy for reg_prev_sub.

Patch 9.0.0048

Problem: Cursor in wrong column with mouse click after concealed text.

Solution: Store the text column when drawing text.

Patch 9.0.0049

Problem: Csv and tsv files are not recognized.

Solution: Add patterns fo csv and tsv files. (Leandro Lourenci, closes #10680)

Patch 9.0.0050

Problem: Split else-of is confusing.

Solution: Join the lines. (closes #10696)

Patch 9.0.0051

Problem: Using **CTRL-C** with :append may hang Vim.

Solution: Reset got_int. (closes #10729, closes #10728)

Patch 9.0.0052

Problem: "zG" may throw an error if invalid character follows.

Solution: Pass the word length to valid_spell_word(). (Ken Takata, closes #10737)

Patch 9.0.0053

Problem: E1281 not tested with the old regexp engine.

Solution: Loop over the values of '**regexp**'. (Dominique Pellé, closes #10695)

Patch 9.0.0054

Problem: Compiler warning for size_t to int conversion.

Solution: Add type cast. (Mike Williams, closes #10741)

Patch 9.0.0055

Problem: Bitbake files are not detected.

Solution: Add bitbake filetype detection by file name and contents. (Gregory Anders, closes #10697)

Patch 9.0.0056

Problem: Wrong line number reported when :cexpr fails in :def function.

Solution: Set line_number before executing :cexpr. (closes #10735)

Patch 9.0.0057

Problem: has('patch-xxx') returns true.

Solution: Check for digit. (closes #10751)

Patch 9.0.0058

Problem: Win32: cannot test low level events.
Solution: Add "sendevent" to test_gui_event(). (Yegappan Lakshmanan, closes #10679)

Patch 9.0.0059

Problem: Test file has wrong name.
Solution: Rename the file. Various small fixes. (closes #10674)

Patch 9.0.0060

Problem: Accessing uninitialized memory when completing long line.
Solution: Terminate string with NUL.

Patch 9.0.0061

Problem: ml_get error with nested autocommand.
Solution: Also check line numbers for a nested autocommand. (closes #10761)

Patch 9.0.0062

Problem: Compiler warnings for signed/unsigned char.
Solution: Add type casts. (John Marriott)

Patch 9.0.0063

Problem: Too many type casts for dict_get functions.
Solution: Change the key argument from "char_u *" to "char *".

Patch 9.0.0064

Problem: Confusing error when using "q:" in command line window.
Solution: Check for the situation and give a better error message. (closes #10756)

Patch 9.0.0065

Problem: Cross-compiling doesn't work because of timer_create check.
Solution: Use AC_CACHE_CHECK(). (Richard Purdie, closes #10777)

Patch 9.0.0066

Problem: Switching window unnecessarily when getting buffer options.
Solution: Do not switch window when getting buffer options. (closes #10767)

Patch 9.0.0067

Problem: Cannot show virtual text.
Solution: Initial changes for virtual text support, using text properties.

Patch 9.0.0068

Problem: Build fails with tiny features.
Solution: Add #ifdef.

Patch 9.0.0069

Problem: Leaking memory when using text prop with inserted text.
Solution: Clear the growarray with text.

Patch 9.0.0070

Problem: Using utfc_ptr2char_len() when length is negative.
Solution: Check value of length. (closes #10760)

Patch 9.0.0071

Problem: Command overlaps with printed text in scrollbar.
Solution: Clear until end-of-line and use correct message chunk.
(closes #10765, closes #10764)

Patch 9.0.0072

Problem: Compiler warning for uninitialized variable.
Solution: Initialize it. (John Marriott)

Patch 9.0.0073

Problem: Too many files recognized as bsd.
Solution: Use pattern "*.bsd" instead of "*bsd". (Martin Tournoij,
closes #10783)

Patch 9.0.0074

Problem: Coverity warns for double free.
Solution: Reset cts_text_prop_count when freeing cts_text_props.

Patch 9.0.0075

Problem: Some compilers warn for using an uninitialized variable. (Tony
Mechelynck)
Solution: Initialize the variable.

Patch 9.0.0076

Problem: No test for what patch 8.1.1424 fixes.
Solution: Add a test. (closes #10789)

Patch 9.0.0077

Problem: When switching window in autocmd the restored cursor position may
be wrong.
Solution: Do not restore the cursor if it was not set. (closes #10775)

Patch 9.0.0078

Problem: Star register is changed when deleting and both "unnamed" and
"unnamedplus" are in **'clipboard'**.
Solution: Make the use of the star register work as documented. (Ernie Rael,
closes #10669)

Patch 9.0.0079

Problem: Error in autoload script not reported for **'foldexpr'**.
Solution: Reset "emsg_off" when auto-loading a script. (closes #10685)

Patch 9.0.0080

Problem: Compiler warning for size_t to int conversion.
Solution: Add type casts. (Mike Williams, closes #10795)

Patch 9.0.0081

Problem: Command line completion of user command may have duplicates.
(Dani Dickstein)
Solution: Skip global user command if an identical buffer-local one is
defined. (closes #10797)

Patch 9.0.0082

Problem: Cannot interrupt global command from command line.
Solution: Reset got_int in another place. (closes #10739)

Patch 9.0.0083

Problem: ModeChanged event not triggered when leaving the cmdline window.
Solution: Call may_trigger_modechanged(). (closes #10791)

Patch 9.0.0084

Problem: Using "terraform" filetype for .tfvars file is bad.
Solution: use "terraform-vars", so that different completion and other mechanisms can be used. (Radek Simko, closes #10755)

Patch 9.0.0085

Problem: ":write" fails after ":file name" and the ":edit".
Solution: Reset BF_NOTEDITED when using ":edit". (closes #10790)

Patch 9.0.0086

Problem: Tabline is not redrawn when entering command line.
Solution: Set "redraw_tabline". (closes #10771)

Patch 9.0.0087

Problem: MS-Windows: CTRL-[on Belgian keyboard does not work like Esc.
Solution: Figure out what the key code means. (Anton Sharonov, closes #10687, closes #10454)

Patch 9.0.0088

Problem: Pattern for detecting bitbake files is not sufficient.
Solution: Adjust the pattern. (Gregory Anders, closes #10743)

Patch 9.0.0089

Problem: Fuzzy argument completion doesn't work for shell commands.
Solution: Check for cmdidx not being CMD_bang. (Yegappan Lakshmanan, closes #10769)

Patch 9.0.0090

Problem: No error when assigning bool to a string option with setwinvar().
Solution: Give an error (closes #10766)

Patch 9.0.0091

Problem: Duplicate error number.
Solution: Use unique error number.

Patch 9.0.0092

Problem: Plugins cannot change v:completed_item.
Solution: Make v:completed_item writeable. (Shougo Matsushita, closes #10801)

Patch 9.0.0093

Problem: Sway config files are recognized as i3config.
Solution: Recognize swayconfig separately. (James Eapen, closes #10672)

Patch 9.0.0094

Problem: Cursor restored unexpected with nested autocommand.
Solution: Do not restore the cursor when it was moved intentionally. (closes #10780)

Patch 9.0.0095

Problem: Conditions are always true.
Solution: Remove useless conditions. (closes #10802)

Patch 9.0.0096

Problem: Flag "new_value_allocated" is always true.
Solution: Remove "new_value_allocated". (closes #10792)

Patch 9.0.0097

Problem: Long quickfix line is truncated for :clist.
Solution: Allocate a buffer if needed.

Patch 9.0.0098

Problem: missing include file in timer_create configure check.
Solution: Include stdlib.h.

Patch 9.0.0099

Problem: Scrollback can be wrong after redrawing the command line.
Solution: Clear unfinished scrollback when redrawing. (closes #10807)

Patch 9.0.0100

Problem: Get hit-enter prompt for system() when '!' is in 'guioptions'.
Solution: Do not call wait_return() when not redrawing. (closes #3327)

Patch 9.0.0101

Problem: Invalid memory access in diff mode with "dp" and undo.
Solution: Make sure the line number does not go below one.

Patch 9.0.0102

Problem: Reading past end of line with insert mode completion.
Solution: Check text length.

Patch 9.0.0103

Problem: If running configure with cached results -lrt may be missing.
Solution: Use two cache variables, one without and one with -lrt.
(closes #10799) Swap checks to avoid adding -lrt unnecessarily.

Patch 9.0.0104

Problem: Going beyond allocated memory when evaluating string constant.
Solution: Properly skip over <Key> form.

Patch 9.0.0105

Problem: Illegal memory access when pattern starts with illegal byte.
Solution: Do not match a character with an illegal byte.

Patch 9.0.0106

Problem: Illegal byte regexp test doesn't fail when fix is reversed.
Solution: Make sure illegal bytes end up in sourced script file.

Patch 9.0.0107

Problem: Condition always has the same value.
Solution: Remove the condition.

Patch 9.0.0108

Problem: Configure check for timer_create may give wrong error.
Solution: Give a warning instead of an error.

Patch 9.0.0109

Problem: Writing over the end of a buffer on stack when making list of spell suggestions.

Solution: Make sure suggested word is not too long. (closes #10812)

Patch 9.0.0110

Problem: Help tag generation picks up words in code examples.

Solution: Skip over examples. (Carlo Teubner, closes #10813)

Patch 9.0.0111

Problem: "nocombine" is missing from synIDattr().

Solution: Add "nocombine". (Muni Tanjim, closes #10816)

Patch 9.0.0112

Problem: MS-Windows: test fails because file already exists.

Solution: Wait a little while until the file is gone.

Patch 9.0.0113

Problem: has() is not strict about parsing the patch version.

Solution: Check the version more strictly. (Ken Takata, closes #10752)

Patch 9.0.0114

Problem: The command line takes up space even when not used.

Solution: Allow for 'cmdheight' to be set to zero. (Shougo Matsushita, closes #10675, closes #940)

Patch 9.0.0115

Problem: When 'cmdheight' is zero pressing ':' may scroll a window.

Solution: Add the made_cmdheight_nonzero flag and set 'scrolloff' to zero.

Patch 9.0.0116

Problem: Virtual text not displayed if 'signcolumn' is "yes".

Solution: Set c_extra and c_final to NUL.

Patch 9.0.0117

Problem: Text of removed textprop with text is not freed.

Solution: Free the text when the property is removed. Reduce the array size to ignore NULLs at the end.

Patch 9.0.0118

Problem: No test for what patch 9.0.0155 fixes.

Solution: Add a test. Fix typos. (closes #10822)

Patch 9.0.0119

Problem: Tiny chance that creating a backup file fails.

Solution: Check for EEXIST error. (Ken Takata, closes #10821)

Patch 9.0.0120

Problem: MS-Windows GUI: cannot use AltGr + Space.

Solution: Check for VK_MENU instead of VK_LMENU. (Anton Sharonov, closes #10820, closes #10753)

Patch 9.0.0121

Problem: Cannot put virtual text after or below a line.

Solution: Add "text_align" and "text_wrap" arguments.

Patch 9.0.0122

Problem: Breakindent test fails.

Solution: Fix condition.

Patch 9.0.0123

Problem: Cannot build with small features.

Solution: Add #ifdef.

Patch 9.0.0124

Problem: Code has more indent than needed.

Solution: Use continue and return statements. (closes #10824)

Patch 9.0.0125

Problem: Cursor positioned wrong with virtual text after the line.

Solution: Clear cts_with_trailing.

Patch 9.0.0126

Problem: Expanding file names fails in directory with more than 255 entries.

Solution: Use an int instead of char_u to count. (John Drouhard, closes #10818)

Patch 9.0.0127

Problem: Unused variable.

Solution: Remove the variable. (closes #10829)

Patch 9.0.0128

Problem: Coverity complains about possible double free.

Solution: Clear the pointer to avoid warnings.

Patch 9.0.0129

Problem: Compiler warning for int/size_t usage.

Solution: Add a type cast. (Mike Williams, closes #10830)

Patch 9.0.0130

Problem: Cursor position wrong when inserting around virtual text.

Solution: Update the cursor position properly.

Patch 9.0.0131

Problem: Virtual text with Tab is not displayed correctly.

Solution: Change any Tab to a space.

Patch 9.0.0132

Problem: Multi-byte characters in virtual text not handled correctly.

Solution: Count screen cells instead of bytes.

Patch 9.0.0133

Problem: Virtual text after line moves to joined line. (Yegappan Lakshmanan)

Solution: When joining lines only keep virtual text after the last line.

Patch 9.0.0134

Problem: No test for text property with column zero.

Solution: Add a test. Add message to assert for no open popups.

Patch 9.0.0135

Problem: Comment about tabpage line above the wrong code.

Solution: Move the comment. (closes #10836)

Patch 9.0.0136

Problem: After **CTRL-Left**-mouse click a mouse scroll also has CTRL.

Solution: Reset orig_mouse_code also for wheel events. (closes #10840)

Patch 9.0.0137

Problem: Debugger test may fail when \$CWD is very long.

Solution: Skip the test if the directory name is too long. (James McCoy, closes #10837)

Patch 9.0.0138

Problem: Not enough characters accepted for '**spellfile**'.

Solution: Add vim_is_fname_char() and use it for '**spellfile**'.

Patch 9.0.0139

Problem: Truncating virtual text after a line not implemented.
Cursor positioning wrong with Newline in the text.

Solution: Implement truncating. Disallow control characters in the text.
(closes #10842)

Patch 9.0.0140

Problem: execute() does not use the "legacy" command modifier.

Solution: pass the command modifier in sticky_cmdmod_flags. (Kota Kato, closes #10845)

Patch 9.0.0141

Problem: "delmenu" does not remove autocommands. Running menu test function alone fails.

Solution: Delete autocommands Make sure there is at least one menu.
(closes #10848)

Patch 9.0.0142

Problem: Crash when adding and removing virtual text. (Ben Jackson)

Solution: Check that the text of the text property still exists.

Patch 9.0.0143

Problem: Cursor positioned after virtual text in empty line.

Solution: Keep cursor in the first column. (closes #10786)

Patch 9.0.0144

Problem: Text property cannot override '**cursorline**' highlight.

Solution: Add the "override" flag to prop_type_add(). (closes #5533, closes #8225).

Patch 9.0.0145

Problem: Substitute that joins lines drops text properties.
Solution: Move text properties of the last line to the new line.

Patch 9.0.0146

Problem: Missing part of change for "override" flag.
Solution: Add the missing change.

Patch 9.0.0147

Problem: Cursor positioned wrong after two text properties with virtual text and "below" alignment. (Tim Pope)
Solution: Do not stop after a text property using MAXCOL. (closes #10849)

Patch 9.0.0148

Problem: A "below" aligned text property gets '**showbreak**' displayed.
Solution: Do not use '**showbreak**' before or in virtual text. (issue #10851)

Patch 9.0.0149

Problem: Test for fuzzy completion fails sometimes.
Solution: Use a more specific file name to minimize the chance of matching a random directory name. (closes #10854)

Patch 9.0.0150

Problem: Error for using #{ in an expression is a bit confusing.
Solution: Mention that this error is only given for an expression.
Avoid giving the error more than once. (closes #10855)

Patch 9.0.0151

Problem: A "below" aligned text property does not work with '**nowrap**'.
Solution: Start a new screen line to display the virtual text.
(closes #10851)

Patch 9.0.0152

Problem: Warning for unused argument in small build.
Solution: Add "UNUSED".

Patch 9.0.0153

Problem: No fold and sign column for virtual text with "below" align and '**nowrap**'.
Solution: Go back to draw state WL_START when moving to the next line.
(closes #10851)

Patch 9.0.0154

Problem: Text properties wrong after splitting a line.
Solution: Check for text properties after the line. (closes #10857)

Patch 9.0.0155

Patch 9.0.0156

Problem: Giving E1170 only in an expression is confusing.
Solution: Give E1170 for any "#{ comment". (closes #10855)

Patch 9.0.0157

Problem: '**showbreak**' displayed below truncated "after" text prop.
Solution: Suppress '**showbreak**' when "after" prop doesn't wrap.

Patch 9.0.0158

Problem: With **'nowrap'** "below" property not displayed correctly.

Solution: Adjust virtual text with **'nowrap'**, do not truncate.

Patch 9.0.0159

Problem: Cannot build with small features.

Solution: Check for E1170 only with FEAT_EVAL.

Patch 9.0.0160

Problem: Some diff mode tests fail.

Solution: Only advance "ptr" when a text property follows.

Patch 9.0.0161

Problem: Warning for uninitialized variable. (Tony Mechelynck)

Solution: Initialize line_attr_save.

Patch 9.0.0162

Problem: Text property "below" gets indent if **'breakindent'** is set. (Tim Pope)

Solution: Do not put indent before text property. (closes #10859)

Patch 9.0.0163

Problem: Text property not adjusted for text inserted with "p".

Solution: Adjust column and length of text properties.

Patch 9.0.0164

Problem: Using freed memory with put command.

Solution: Get byte offset before replacing the line.

Patch 9.0.0165

Problem: Looking up a text property type by ID is slow.

Solution: Keep an array of property types sorted on ID.

Patch 9.0.0166

Problem: When using text properties the line text length is computed twice.

Solution: If the text length was already computed don't do it again.

Patch 9.0.0167

Problem: Checking for text properties could be a bit more efficient.

Solution: Return early when there are no text properties. Update TODO items.

Patch 9.0.0168

Problem: Cursor positioned wrong with two virtual text properties close together. (Ben Jackson)

Solution: Add the original size, not the computed one. (closes #10864)

Patch 9.0.0169

Problem: Insufficient testing for line2byte() with text properties.

Solution: Add tests with a lot of text.

Patch 9.0.0170

Problem: Various minor code formatting issues.

Solution: Improve code formatting.

Patch 9.0.0171

Problem: Quickfix line highlight is overruled by **'cursorline'**.

Solution: Reverse the combination of attributes. (closes #10654)

Patch 9.0.0172

Problem: Trying to allocate zero bytes.

Solution: Do not allocate the proptype array when there are none.
(closes #10867)

Patch 9.0.0173

Problem: Assert fails only on MS-Windows.

Solution: Disable the assert for now.

Patch 9.0.0174

Problem: No error for using "#{ comment" in a compiled function.

Solution: Make error checking for "#{ consistent. (closes #10855)

Patch 9.0.0175

Problem: Spell checking for capital not working with trailing space.

Solution: Do not calculate cap_col at the end of the line. (Christian
Brabandt, closes #10870, issue #10838)

Patch 9.0.0176

Problem: Checking character options is duplicated and incomplete.

Solution: Move checking to check_chars_options(). (closes #10863)

Patch 9.0.0177

Problem: Cursor position wrong with **'virtualedit'** and mouse click after end
of the line. (Hermann Mayer)

Solution: Do not use ScreenCols[] when **'virtualedit'** is active.
(closes #10868)

Patch 9.0.0178

Problem: Cursor position wrong with virtual text before Tab.

Solution: Use the byte length, not the cell with, to compare the column.
Correct tab size after text prop. (closes #10866)

Patch 9.0.0179

Problem: Cursor position wrong with wrapping virtual text in empty line.

Solution: Adjust handling of an empty line. (closes #10875)

Patch 9.0.0180

Problem: Stray logfile appears when running tests.

Solution: Remove ch_logfile() calls.

Patch 9.0.0181

Problem: Textprop test with line2byte() fails on MS-Windows.

Solution: Fix updating chunks in ml_delete_int().

Patch 9.0.0182

Problem: Quarto files are not recognized.

Solution: Recognize quarto files by the extension. (Jonas Strittmatter,

closes #10880)

Patch 9.0.0183

Problem: Extra space after virtual text when **'linebreak'** is set.
Solution: Do not count virtual text when getting linebreak value.
(closes #10884)

Patch 9.0.0184

Problem: Virtual text prop highlight continues after truncation.
Solution: Recompute the length of attributes.

Patch 9.0.0185

Problem: Virtual text does not show if there is a text prop at same position. (Ben Jackson)
Solution: Fix the sorting of properties. (closes #10879)

Patch 9.0.0186

Problem: Virtual text without highlighting does not show. (Ben Jackson)
Solution: Use a text property when it has highlighting or when it has text.
(closes #10878)

Patch 9.0.0187

Problem: Command line height changes when maximizing window height.
Solution: Do not change the command line height. (closes #10885)

Patch 9.0.0188

Problem: Strange effects when using virtual text with "text_align" and non-zero column. (Martin Tournoij)
Solution: Give an error. (closes #10888)

Patch 9.0.0189

Problem: Invalid memory access for text prop without highlight.
Solution: Check for a valid highlight ID.

Patch 9.0.0190

Problem: The way **'cmdheight'** can be made zero is inconsistent.
Solution: Only make **'cmdheight'** zero when setting it explicitly, not when resizing windows. (closes #10890)

Patch 9.0.0191

Problem: Messages test fails; window size incorrect when **'cmdheight'** is made smaller.
Solution: Properly cleanup after test with cmdheight zero. Resize windows correctly when **'cmdheight'** gets smaller.

Patch 9.0.0192

Problem: Possible invalid memory access when **'cmdheight'** is zero. (Martin Tournoij)
Solution: Avoid going over the end of w_lines[] when w_height is Rows.
(closes #10882)

Patch 9.0.0193

Problem: Search and match highlight interfere with virtual text highlight. (Ben Jackson)

Solution: Check for match highlight after text properties. Reset and restore search highlight when showing virtual text. (closes #10892)

Patch 9.0.0194

Problem: Cursor displayed in wrong position after removing text prop. (Ben Jackson)

Solution: Invalidate the cursor position. (closes #10898)

Patch 9.0.0195

Problem: Metafun files are not recognized.

Solution: Add filetype detection patterns.

Patch 9.0.0196

Problem: Finding value in list may require a for loop.

Solution: Add indexof(). (Yegappan Lakshmanan, closes #10903)

Patch 9.0.0197

Problem: Astro files are not detected.

Solution: Add a pattern to match Astro files. (Emilia Zapata, closes #10904)

Patch 9.0.0198

Problem: ml_get error when switching buffer in Visual mode.

Solution: End Visual mode when switching buffer. (closes #10902)

Patch 9.0.0199

Problem: Cursor position wrong with two right-aligned virtual texts.

Solution: Add the padding for right-alignment. (issue #10906)

Patch 9.0.0200

Problem: cursor in a wrong position if **'wrap'** is off and using two right aligned text props in one line.

Solution: Count an extra line for a right aligned text property after a below or right aligned text property. (issue #10909)

Patch 9.0.0201

Problem: CursorLine highlight overrules virtual text highlight.

Solution: Let extra attribute overrule line attribute. (closes #10909)

Patch 9.0.0202

Problem: Code and help for indexof() is not ideal.

Solution: Refactor the code, improve the help. (Yegappan Lakshmanan, closes #10908)

Patch 9.0.0203

Problem: Confusing variable name.

Solution: Use "prim_aep" instead of "spell_aep".

Patch 9.0.0204

Problem: indexof() may leak memory.

Solution: Free allocated values. (Yegappan Lakshmanan, closes #10916)

Patch 9.0.0205

Problem: Cursor in wrong position when inserting after virtual text. (Ben

Jackson)
Solution: Put the cursor after the virtual text, where the text will be inserted. (closes #10914)

Patch 9.0.0206

Problem: Redraw flags are not named specifically.
Solution: Prefix "UPD_" to the flags, for UPDate_screen().

Patch 9.0.0207

Problem: Stacktrace not shown when debugging.
Solution: Set msg_scroll in msg_source(). (closes #10917)

Patch 9.0.0208

Problem: The override flag has no effect for virtual text. (Ben Jackson)
Solution: Make the override flag work. (closes #10915)

Patch 9.0.0209

Problem: Build error with small features.
Solution: Add #ifdef.

Patch 9.0.0210

Problem: **'list'** mode does not work properly with virtual text.
Solution: Show the "\$" at the right position. (closes #10913)

Patch 9.0.0211

Problem: Invalid memory access when compiling :lockvar.
Solution: Don't read past the end of the line.

Patch 9.0.0212

Problem: Invalid memory access when compiling :unlet.
Solution: Don't read past the end of the line.

Patch 9.0.0213

Problem: Using freed memory with error in assert argument.
Solution: Make a copy of the error.

Patch 9.0.0214

Problem: Splitting a line may duplicate virtual text. (Ben Jackson)
Solution: Don't duplicate a text property with virtual text. Make auto-indenting work better. (closes #10919)

Patch 9.0.0215

Problem: Not passing APC_INDENT flag.
Solution: Pass the flag where it's needed.

Patch 9.0.0216

Problem: Undo earlier test sometimes fails on MS-Windows.
Solution: Use another file name.

Patch 9.0.0217

Problem: **'shellslash'** works differently when sourcing a script again.
Solution: Use the name from the script item. (closes #10920)

Patch 9.0.0218

Problem: Reading before the start of the line.
Solution: When displaying "\$" check the column is not negative.

Patch 9.0.0219

Problem: Cannot make a funcref with "s:func" in a def function in legacy script.
Solution: Allow for using a lower case function name after "s:". (Kota Kato, closes #10926)

Patch 9.0.0220

Problem: Invalid memory access with for loop over NULL string.
Solution: Make sure mb_ptr2len() consistently returns zero for NUL.

Patch 9.0.0221

Problem: Accessing freed memory if compiling nested function fails.
Solution: Mess up the variable name so that it won't be found.

Patch 9.0.0222

Problem: No good reason why text objects are only in larger builds.
Solution: Graduate +textobjects.

Patch 9.0.0223

Problem: Typo in diffmode test.
Solution: Fix the typo. (closes #10932)

Patch 9.0.0224

Problem: Using NULL pointer when skipping compiled code.
Solution: Check for skipping.

Patch 9.0.0225

Problem: Using freed memory with multiple line breaks in expression.
Solution: Free eval_tofree later.

Patch 9.0.0226

Problem: job_start() test may fail under valgrind.
Solution: Wait until the job is running.

Patch 9.0.0227

Problem: Cannot read error message when abort() is called.
Solution: Output a newline before calling abort().

Patch 9.0.0228

Problem: Crash when pattern looks below the last line.
Solution: Consider invalid lines to be empty. (closes #10938)

Patch 9.0.0229

Problem: Vim9: error message for missing type is not clear.
Solution: Mention the context. (issue #10944)

Patch 9.0.0230

Problem: No error for comma missing in list in :def function.
Solution: Check for missing comma. (closes #10943)

Patch 9.0.0231

Problem: Expanding "*" may loop forever with directory links.
Solution: Check for being interrupted. (closes #10946)

Patch 9.0.0232

Problem: Test with BufNewFile autocmd is flaky.
Solution: Use another file name.

Patch 9.0.0233

Problem: Removing multiple text properties takes many calls.
Solution: Pass a list to prop_remove(). (Ben Jackson, closes #10945)

Patch 9.0.0234

Problem: Cannot make difference between the end of :normal and a character in its argument.
Solution: Add the "typebuf_was_empty" flag. (closes #10950)

Patch 9.0.0235

Problem: 'autoshelldir' does not work with chunked response.
Solution: Collect chunks before parsing OSC 7. (closes #10949)

Patch 9.0.0236

Problem: Popup menu not removed when 'wildmenu' reset while it is visible.
Solution: Do not check p_wmnu, only pum_visible(). (closes #10953)

Patch 9.0.0237

Problem: Mac: cannot build if dispatch.h is not available.
Solution: Add #ifdef. (Evan Miller, closes #10954)

Patch 9.0.0238

Problem: Shift-Tab shows matches on cmdline when 'wildmenu' is off.
Solution: Only show matches when 'wildmode' contains "list". (closes #10951)

Patch 9.0.0239

Problem: Build failure without the +wildmenu feature.
Solution: Move parenthesis.

Patch 9.0.0240

Problem: Crash when using ":mkspell" with an empty .dic file.
Solution: Check for an empty word tree.

Patch 9.0.0241

Problem: "make install" does not install shared syntax file. (James McCoy)
Solution: Install and uninstall the shared syntax files. (closes #10956)

Patch 9.0.0242

Problem: "make install" still fails. (Wilhelm Payne)
Solution: Also add the directory to installrtbase. (Dominique Pellé)

Patch 9.0.0243

Problem: Text properties "below" sort differently on MS-Windows.
Solution: Use the ID as a tie breaker. (closes #10958)

Patch 9.0.0244

Problem: Cannot easily get the list of sourced scripts.

Solution: Add the `getscriptinfo()` function. (Yegappan Lakshmanan, closes #10957)

Patch 9.0.0245

Problem: Mechanism to prevent recursive screen updating is incomplete.

Solution: Add `"redraw_not_allowed"` and set it in `build_stl_str_hl()`. (issue #10952)

Patch 9.0.0246

Problem: Using freed memory when `'tagfunc'` deletes the buffer.

Solution: Make a copy of the tag name.

Patch 9.0.0247

Problem: Cannot add padding to virtual text without highlight.

Solution: Add the `"text_padding_left"` argument. (issue #10906)

Patch 9.0.0248

Problem: Duplicate code in finding a script in the execution stack.

Solution: Reduce duplicate code. (closes #10961)

Patch 9.0.0249

Problem: No test for what 9.0.0234 fixes.

Solution: Add a test. (issue #10950)

Patch 9.0.0250

Problem: Slightly inconsistent error messages.

Solution: Make it `"Using a Float"`. (closes #10959)

Patch 9.0.0251

Problem: Test output shows up in git.

Solution: Ignore the `"failed"` directory. (Yao-Ching Huang, closes #10969)

Patch 9.0.0252

Problem: Cursor in wrong place after virtual text.

Solution: Do not change the length of a virtual text property. (closes #10964)

Patch 9.0.0253

Problem: A symlink to an autoload script results in two entries in the list of scripts, items expected in one are actually in the other.

Solution: Have one script item refer to the actually sourced one. (closes #10960)

Patch 9.0.0254

Problem: Typo in function name.

Solution: Rename the function. (closes #10971)

Patch 9.0.0255

Problem: Build failure without the eval feature.

Solution: Add `#ifdef`.

Patch 9.0.0256

Problem: Compiler warning for uninitialized variables.

Solution: Initialize the variables.

Patch 9.0.0257

Problem: "->" in ":scriptnames" output not tested yet.

Solution: Add a check.

Patch 9.0.0258

Problem: MS-Windows installer skips syntax/shared.

Solution: Use "File /r" in the installer script. (Ken Takata, closes #10972)

Patch 9.0.0259

Problem: Crash with mouse click when not initialized.

Solution: Check tabPageIdxs[] is not NULL.

Patch 9.0.0260

Problem: Using freed memory when using 'quickfixtextfunc' recursively.

Solution: Do not allow for recursion.

Patch 9.0.0261

Problem: bufload() reads a file even if the name is not a file name. (Cyker Way)

Solution: Do not read the file when the buffer name is not a file name. (closes #10975)

Patch 9.0.0262

Problem: Build failure without the +quickfix feature.

Solution: Add #ifdef.

Patch 9.0.0263

Problem: Too many #ifdefs.

Solution: Make some functions always available.

Patch 9.0.0264

Problem: CI still runs on Ubuntu 18.04.

Solution: Run CI on Ubuntu 20.04. (closes #10582)

Patch 9.0.0265

Problem: No good reason why the "gf" command is not in the tiny version.

Solution: Graduate the file_in_path feature.

Patch 9.0.0266

Problem: Compiler warning for unused argument.

Solution: Add UNUSED.

Patch 9.0.0267

Problem: Coverity workflow still uses Ubuntu 18.04.

Solution: Use Ubuntu 20.04

Patch 9.0.0268

Problem: Build error without the +eval feature.

Solution: Remove #ifdef.

Patch 9.0.0269

Problem: getscriptinfo() does not include the version. Cannot select entries by script name.

Solution: Add the "version" item and the "name" argument. (Yegappan Lakshmanan, closes #10962)

Patch 9.0.0270

Problem: Some values of **'path'** and **'tags'** do not work in the tiny version.
Solution: Graduate the +path_extra feature.

Patch 9.0.0271

Problem: Using INIT() in non-header files.
Solution: Remove INIT(). (closes #10981)

Patch 9.0.0272

Problem: BufReadCmd not triggered when loading a "nofile" buffer. (Maxim Kim)
Solution: Call readfile() but bail out before reading a file. (closes #10983)

Patch 9.0.0273

Problem: Konsole termresponse not recognized.
Solution: Handle Konsole like libvterm, set **'ttymouse'** to "sgr". (closes #10990)

Patch 9.0.0274

Problem: Netrw plugin does not show remote files.
Solution: Do read a file when **'buftype'** is "acwrite". (closes #10983)

Patch 9.0.0275

Problem: BufEnter not triggered when using ":edit" in "nofile" buffer.
Solution: Let readfile() return NOTDONE. (closes #10986)

Patch 9.0.0276

Problem: **'buftype'** values not sufficiently tested.
Solution: Add and extend tests with **'buftype'** values. (closes #10988)

Patch 9.0.0277

Problem: Coverity CI: update-alternatives not needed with Ubuntu 20.04.
Solution: Remove update-alternatives for Lua. (closes #10987)

Patch 9.0.0278

Problem: The +wildignore feature is nearly always available.
Solution: Graduate +wildignore for consistency.

Patch 9.0.0279

Problem: The tiny version has the popup menu but not **'wildmenu'**.
Solution: Graduate the wildmenu feature.

Patch 9.0.0280

Problem: The builtin termcap list depends on the version.
Solution: Always include all termcap entries. Remove duplicate lines.

Patch 9.0.0281

Problem: Build failure without the +eval feature.
Solution: Add #ifdef.

Patch 9.0.0282

Problem: A nested timeout stops the previous timeout.
Solution: Ignore any nested timeout.

Patch 9.0.0283

Problem: Cannot complete "syn list @cluster".
Solution: Recognize and handle "list @". (Björn Linse, closes #10990)

Patch 9.0.0284

Problem: Using static buffer for multiple completion functions.
Solution: Use one buffer in expand_T.

Patch 9.0.0285

Problem: It is not easy to change the command line from a plugin.
Solution: Add setcmdline(). (Shougo Matsushita, closes #10869)

Patch 9.0.0286

Problem: Using freed memory when location list changed in autocmd.
Solution: Return QF_ABORT and handle it. (Yegappan Lakshmanan, closes #10993)

Patch 9.0.0287

Problem: Irix systems no longer exist.
Solution: Remove references to Irix. (Yegappan Lakshmanan, closes #10994)

Patch 9.0.0288

Problem: When '**cmdheight**' is zero some messages are not displayed.
Solution: Use a popup notification window.

Patch 9.0.0289

Problem: Invalid memory write.
Solution: Do not put NUL in a static string.

Patch 9.0.0290

Problem: Compiler warning for variable set but not used.
Solution: Add #ifdef.

Patch 9.0.0291

Problem: Test failing.
Solution: Run test with cmdheight=0 last.

Patch 9.0.0292

Problem: Test causes another test to fail.
Solution: Redraw to remove the popup window

Patch 9.0.0293

Problem: Messages window not hidden when starting a command line.
Solution: Hide the messages window. (closes #10996)

Patch 9.0.0294

Problem: Crash when '**cmdheight**' is 0 and popup_clear() used.
Solution: Reset "message_win" when the message popup is cleared. Close the popup when '**cmdheight**' is non-zero. Add a screendump test.

Patch 9.0.0295

Problem: GUI drop files test sometimes fails.

Solution: Mark the test as flaky.

Patch 9.0.0296

Problem: Message in popup is shortened unnecessary.

Solution: Do not use **'showcmd'** and **'ruler'** for a message in the popup.
Set the timer when un hiding the message popup.

Patch 9.0.0297

Problem: Cursor position wrong after right aligned virtual text. (Iizuka Masashi)

Solution: Take the width of the column offset into account. (closes #10997)
Also fix virtual text positioning.

Patch 9.0.0298

Problem: Compiler warning for size_t to int conversion.

Solution: Add a type cast. (Wilhelm Payne, closes #11000)

Patch 9.0.0299

Problem: Error messages for setcmdline() could be better.

Solution: Use more specific error messages. (Yegappan Lakshmanan, closes #10995)

Patch 9.0.0300

Problem: **'cptions'** tests are flaky.

Solution: Use a different file name for each test.

Patch 9.0.0301

Problem: The message window popup is delayed after an error message.

Solution: Do not set msg_on_display when using the message window.

Patch 9.0.0302

Problem: CI for Coverity is bothered by deprecation warnings.

Solution: Ignore deprecation warnings. (closes #11002)

Patch 9.0.0303

Problem: It is not easy to get information about a script.

Solution: Make getscriptinfo() return the version. When selecting a specific script return functions and variables. (Yegappan Lakshmanan, closes #10991)

Patch 9.0.0304

Problem: WinScrolled is not triggered when only skipcol changes.

Solution: Add w_last_skipcol and use it. (closes #10998)

Patch 9.0.0305

Problem: CI lists useless deprecation warnings.

Solution: Ignore deprecation warnings. (closes #11003)

Patch 9.0.0306

Problem: Buffer write message is two lines in message popup window.

Solution: Overwrite message if "msg_scroll" is off.

Patch 9.0.0307

Problem: :echomsg doesn't work properly with cmdheight=0.
Solution: Improve scrolling and displaying.

Patch 9.0.0308

Problem: When cmdheight is zero the attention prompt doesn't show.
Solution: Do not use the message window for a prompt.

Patch 9.0.0309

Problem: Invalid memory access when cmdheight is zero.
Solution: Check index in w_lines is smaller than Rows.

Patch 9.0.0310

Problem: Output of :messages disappears when cmdheight is zero.
Solution: Do not use the messages window for :messages. Make Esc close the messages window.

Patch 9.0.0311

Problem: Test for hit-Enter prompt fails.
Solution: Only reset cmdline_row when 'cmdheight' is zero.

Patch 9.0.0312

Problem: Test for cmdheight zero fails.
Solution: Do not close the messages window for **CTRL-C**.

Patch 9.0.0313

Problem: Using common name in tests leads to flaky tests.
Solution: Rename files and directories to be more specific.

Patch 9.0.0314

Problem: VDM files are not recognized.
Solution: Add patterns for VDM files. (Alessandro Pezzoni, closes #11004)

Patch 9.0.0315

Problem: Shell command is displayed in message window.
Solution: Do not echo the shell command in the message window.

Patch 9.0.0316

Problem: Screen flickers when 'cmdheight' is zero.
Solution: Redraw over existing text instead of clearing.

Patch 9.0.0317

Problem: When updating the whole screen a popup may not be redrawn.
Solution: Mark the screen and windows for redraw also when not clearing.
Also mark popup windows for redraw.

Patch 9.0.0318

Problem: Clearing screen causes flicker.
Solution: Do not clear but redraw in more cases. Add () to "wait_return".

Patch 9.0.0319

Problem: Godot shader files are not recognized.
Solution: Add patterns for "gdshader". (Maxim Kim, closes #11006)

Patch 9.0.0320

Problem: Command line type of CmdlineChange differs from getcmdtype().
Solution: Use the same type. (closes #11005)

Patch 9.0.0321

Problem: Cannot use the message popup window directly.
Solution: Add ":echowindow".

Patch 9.0.0322

Problem: Crash when no errors and 'quickfixtextfunc' is set.
Solution: Do not handle errors if there aren't any.

Patch 9.0.0323

Problem: Using common name in tests leads to flaky tests.
Solution: Rename files and directories to be more specific.

Patch 9.0.0324

Problem: MS-Windows: resolve() test fails.
Solution: Revert renaming the directory.

Patch 9.0.0325

Problem: MS-Windows: completion test fails.
Solution: Adjust directory prefix.

Patch 9.0.0326

Problem: Some changes for cmdheight=0 are not needed.
Solution: Revert resize behavior if height is greater than the available space. (Shougo Matsushita, closes #11008)

Patch 9.0.0327

Problem: items() does not work on a list. (Sergey Vlasov)
Solution: Make items() work on a list. (closes #11013)

Patch 9.0.0328

Problem: OLD_DIGRAPHS is unused.
Solution: Remove OLD_DIGRAPHS. Also drop HPUX_DIGRAPHS.

Patch 9.0.0329

Problem: ":highlight" hangs when 'cmdheight' is zero.
Solution: Add to msg_col when using the message window. (closes #11014)

Patch 9.0.0330

Problem: Method tests fail.
Solution: Adjust for change of items().

Patch 9.0.0331

Problem: Cannot use items() on a string.
Solution: Make items() work on a string. (closes #11016)

Patch 9.0.0332

Problem: Overwrite check may block BufWriteCmd.
Solution: Do not use overwrite check when 'buftype' is "acwrite".
(closes #11011)

Patch 9.0.0333

Problem: Method test fails.

Solution: Adjust test for items() now working on string.

Patch 9.0.0334

Problem: Test does not properly clean up.

Solution: Fix typo in argument of delete(). (Dominique Pellé, closes #11010)

Patch 9.0.0335

Problem: Checks for Dictionary argument often give a vague error message.

Solution: Give a useful error message. (Yegappan Lakshmanan, closes #11009)

Patch 9.0.0336

Problem: Tests are flaky because of using a common file name.

Solution: Rename files and directories to be more unique.

Patch 9.0.0337

Problem: Flicker when resetting cmdline_row after updating the screen.

Solution: Do not update cmdline_row. (issue #11017)

Patch 9.0.0338

Problem: Return value of list_append_list() not always checked.

Solution: Check return value and handle failure.

Patch 9.0.0339

Problem: No check if the return value of XChangeGC() is NULL.

Solution: Only use the return value when it is not NULL. (closes #11020)

Patch 9.0.0340

Problem: The `'cmdheight'` zero support causes too much trouble.

Solution: Revert support for `'cmdheight'` being zero.

Patch 9.0.0341

Problem: mapset() does not restore `<Nop>` mapping properly.

Solution: Use an empty string for `<Nop>`. (closes #11022)

Patch 9.0.0342

Problem: `":wincmd ="` equalizes in two directions.

Solution: Make `":vertical wincmd ="` equalize vertically only and `":horizontal wincmd ="` equalize horizontally only.

Patch 9.0.0343

Problem: ColorScheme autocommand triggered when colorscheme is not found.
(Romain Lafourcade)

Solution: Only trigger ColorScheme when loading the colorscheme succeeds.
(closes #11024)

Patch 9.0.0344

Problem: MS-Windows: background color wrong in Console.

Solution: Figure out the default console background color. (Yasuhiro Matsumoto, issue #10310)

Patch 9.0.0345

Problem: Error message for list argument could be clearer.

Solution: Include the argument number. (Yegappan Lakshmanan, closes #11027)

Patch 9.0.0346

Problem: :horizontal modifier not fully supported.

Solution: Also use :horizontal for completion and user commands.
(closes #11025)

Patch 9.0.0347

Problem: MS-Windows: cannot set cursor shape in Windows Terminal.

Solution: Make cursor shape work with Windows Terminal. (Ken Takata,
closes #11028, closes #6576)

Patch 9.0.0348

Problem: MS-Windows: GUI mouse move event test is flaky.

Solution: Wait for a little while for the first move event.

Patch 9.0.0349

Problem: Filetype of *.sil files not well detected.

Solution: Inspect the file contents to guess the filetype.

Patch 9.0.0350

Problem: :echowindow does not work in a compiled function.

Solution: Handle the expression at compile time.

Patch 9.0.0351

Problem: Message window may obscure the command line.

Solution: Reduce the maximum height of the message window.

Patch 9.0.0352

Problem: using :echowindow in a timer clears part of message

Solution: Do not use msg_clr_eos().

Patch 9.0.0353

Problem: Missing entry in switch.

Solution: Add ISN_ECHOWINDOW.

Patch 9.0.0354

Problem: MS-Windows: starting a python server for test sometimes fails.

Solution: Increase the waiting time for the port.

Patch 9.0.0355

Problem: Check for uppercase char in autoload name is wrong, it checks the
name of the script.

Solution: Remove the check. (closes #11031)

Patch 9.0.0356

Problem: :echowindow sets the in_echowindow flag too early.

Solution: Set in_echowindow only when outputting the text. (Yasuhiro
Matsumoto, closes #11033)

Patch 9.0.0357

Problem: **'linebreak'** interferes with text property highlight if there is
syntax highlighting.

Solution: Check the text prop attributes after combining with syntax

attributes. (closes #11035)

Patch 9.0.0358

Problem: **'breakindent'** does not indent non-lists with "breakindentopt=list:-1".

Solution: Adjust indent computation. (Maxim Kim, closes #11038)

Patch 9.0.0359

Problem: Error message for wrong argument type is not specific.

Solution: Include more information in the error. (Yegappan Lakshmanan, closes #11037)

Patch 9.0.0360

Problem: Crash when invalid line number on :for is ignored.

Solution: Do not check breakpoint for non-existing line.

Patch 9.0.0361

Problem: Removing a listener may result in a memory leak and remove subsequent listeners.

Solution: Init the "prev" pointer only once. (Yegappan Lakshmanan, closes #11039)

Patch 9.0.0362

Problem: Expanding ":e %" does not work for remote files.

Solution: If the "%" or "#" file does not exist add the expansion anyway.

Patch 9.0.0363

Problem: Common names in test files causes tests to be flaky.

Solution: Use more specific names.

Patch 9.0.0364

Problem: Clang static analyzer gives warnings.

Solution: Avoid the warnings. (Yegappan Lakshmanan, closes #11043)

Patch 9.0.0365

Problem: File name used in test is unusual.

Solution: Rename it. (Dominique Pellé, closes #11044)

Patch 9.0.0366

Problem: Cannot use import->Func() in lambda. (Israel Chauca Fuentes)

Solution: Adjust how an expression in a lambda is parsed. (closes #11042)

Patch 9.0.0367

Problem: Coverity complains about dropping sign of character.

Solution: Add explicit type cast.

Patch 9.0.0368

Problem: Old Coverity warning for using NULL pointer.

Solution: Bail out if dictionary allocation fails.

Patch 9.0.0369

Problem: A failing flaky test doesn't mention the time.

Solution: Add the time for debugging. Improve error message.

Patch 9.0.0370

Problem: Cleaning up afterwards can make a function messy.
Solution: Add the :defer command.

Patch 9.0.0371

Problem: Compiler warning for uninitialized variable.
Solution: Initialize the variable. (John Marriott)

Patch 9.0.0372

Problem: MS-Windows: "%T" time format does not appear to work.
Solution: Use "%H:%M:%S" instead.

Patch 9.0.0373

Problem: Coverity warns for NULL check and unused return value.
Solution: Remove the NULL check, it was already checked earlier. Add (void) to ignore the return value.

Patch 9.0.0374

Problem: Coverity still complains about dropping sign of character.
Solution: Add intermediate variable.

Patch 9.0.0375

Problem: The footer feature is unused.
Solution: Remove FEAT_FOOTER and code.

Patch 9.0.0376

Problem: Clang warns for dead assignments.
Solution: Adjust the code. (Yegappan Lakshmanan, closes #11048)

Patch 9.0.0377

Problem: Argument assignment does not work.
Solution: Skip over "=".

Patch 9.0.0378

Problem: Compiler warning for uninitialized variable. (Tony Mechelynck)
Solution: Initialize it.

Patch 9.0.0379

Problem: Cleaning up after writefile() is a hassle.
Solution: Add the 'D' flag to defer deleting the written file. Very useful in tests.

Patch 9.0.0380

Problem: Deleting files in tests is a hassle.
Solution: Use the new 'D' flag of writefile().

Problem: Deleting files in tests is a hassle.
Solution: Use the new 'D' flag of writefile().

Patch 9.0.0381

Problem: Writefile test leaves files behind.
Solution: Fix the file names of files to be deleted. (Dominique Pellé, closes #11056)

Patch 9.0.0382

Problem: Freeing the wrong string on failure.
Solution: Adjust the argument. Reorder the code.

Patch 9.0.0383

Problem: Coverity complains about unused value.
Solution: Use the value.

Patch 9.0.0384

Problem: Coverity still complains about using return value of `getc()`.
Solution: Check for EOF.

Patch 9.0.0385

Problem: GUI: when **CTRL-D** is mapped in Insert mode it gets inserted.
(Yasuhiro Matsumoto)
Solution: Also recognize modifier starting with CSI. (closes #11057)

Patch 9.0.0386

Problem: Some code blocks are nested too deep.
Solution: Bail out earlier. (Yegappan Lakshmanan, closes #11058)

Patch 9.0.0387

Problem: repeating a `<ScriptCmd>` mapping does not use the right script context.
Solution: When using a mapping put `<SID>{sid};` in the redo buffer.
(closes #11049)

Patch 9.0.0388

Problem: The `do_arg_all()` function is too long.
Solution: Split the function in smaller parts. (Yegappan Lakshmanan, closes #11062)

Patch 9.0.0389

Problem: Crash when `'tagfunc'` closes the window.
Solution: Bail out when the window was closed.

Patch 9.0.0390

Problem: Cannot use a partial with `:defer`.
Solution: Add the partial arguments before the other arguments. Disallow using a dictionary.

Patch 9.0.0391

Problem: Using separate `delete()` call instead of `writefile()` 'D' flag.
Solution: Use the `writefile` 'D' flag.

Patch 9.0.0392

Problem: Inverted condition is a bit confusing.
Solution: Remove the `!"` and swap the blocks. (Ken Takata)

Patch 9.0.0393

Problem: Signals test often fails on FreeBSD.
Solution: Use separate files for Suspend and Resume. (Ken Takata, closes #11065)

Patch 9.0.0394

Problem: Cygwin: multibyte characters may be broken in terminal window.
Solution: Adjust how to read and write on the channel. (Ken Takata, closes #11063)

Patch 9.0.0395

Problem: Clang warnings for function prototypes.
Solution: Remove incomplete function prototypes. (closes #11068)

Patch 9.0.0396

Problem: :findrepl does not escape '&' and '~' properly.
Solution: Escape depending on the value of '**magic**'. (closes #11067)

Patch 9.0.0397

Problem: :defer not tested with exceptions and ":qa!".
Solution: Test :defer works when exceptions are thrown and when ":qa!" is used. Invoke the deferred calls on exit.

Patch 9.0.0398

Problem: Members of funccall_T are inconsistently named.
Solution: Use the "fc_" prefix for all members.

Patch 9.0.0399

Problem: Using :defer in expression funcref not tested.
Solution: Add a test. Fix uncovered problems.

Patch 9.0.0400

Problem: GUI test sometimes hangs on CI.
Solution: Delete a test file explicitly. (Ken Takata, closes #11072)

Patch 9.0.0401

Problem: CI uses older clang version.
Solution: Switch from clang 14 to 15. (closes #11066)

Patch 9.0.0402

Problem: Javascript module files are not recognized.
Solution: Recognize "*.jasm" files as Javascript. (Brett Holman, closes #11069)

Patch 9.0.0403

Problem: '**equalalways**' may be off when '**laststatus**' is zero.
Solution: call last_status() before win_equal(). (Luuk van Baal, closes #11070)

Patch 9.0.0404

Problem: Crash when passing invalid arguments to assert_fails().
Solution: Check for NULL string.

Patch 9.0.0405

Problem: Arguments in a partial not used by a :def function.
Solution: Put the partial arguments on the stack.

Patch 9.0.0406

Problem: Deferred functions not invoked when partial func exits.
Solution: Create a funccall_T when calling a :def function.

Patch 9.0.0407

Problem: matchstr() does match column offset. (Yasuhiro Matsumoto)
Solution: Accept line number zero. (closes #10938)

Patch 9.0.0408

Problem: GUI test sometimes fails on MS-Windows.
Solution: Make sure Vim is the foreground window. (Ken Takata, closes #11077)

Patch 9.0.0409

Problem: #{g:x} was seen as a curly-braces expression.
Solution: Do never see #{} as a curly-braces expression. (closes #11075)

Patch 9.0.0410

Problem: Struct member cts_lnum is unused.
Solution: Delete it.

Patch 9.0.0411

Problem: Only created files can be cleaned up with one call.
Solution: Add flags to mkdir() to delete with a deferred function.
Expand the writefile() name to a full path to handle changing directory.

Patch 9.0.0412

Problem: Compiler warning for unused argument.
Solution: Add UNUSED.

Patch 9.0.0413

Problem: ASAN reports a memory leak.
Solution: Free the string received from the server. (Ken Takata, closes #11080)

Patch 9.0.0414

Problem: matchstr() still does not match column offset when done after a text search.
Solution: Only use the line number for a multi-line search. Fix the test. (closes #10938)

Patch 9.0.0415

Problem: On MS-Windows some tests are flaky.
Solution: Add sleeps, disable swapfile, mark test as flaky. (Ken Takata, closes #11082)

Patch 9.0.0416

Problem: ml_get error when appending lines in popup window.
Solution: Only update w_topline when w_buffer matches curbuf. (closes #11074)

Patch 9.0.0417

Problem: Jsonnet files are not recognized.
Solution: Add a pattern for Jsonnet files. (Cezary Drożak, closes #11073, closes #11081)

Patch 9.0.0418

Problem: Manually deleting temp test files.

Solution: Use the 'D' flag of writefile() and mkdir().

Patch 9.0.0419

Problem: The :defer command does not check the function argument count and types.

Solution: Check the function arguments when adding a deferred function.

Patch 9.0.0420

Problem: Function went missing.

Solution: Add the function back.

Patch 9.0.0421

Problem: MS-Windows makefiles are inconsistently named.

Solution: Use consistent names. (Ken Takata, closes #11088)

Patch 9.0.0422

Problem: Not enough testing of the :all command.

Solution: Add more testing. (Yegappan Lakshmanan, closes #11091)

Patch 9.0.0423

Problem: "for" and "while" not recognized after :vim9cmd and :legacy.
(Emanuele Torre)

Solution: Recognize all the command modifiers. (closes #11087)
Add a test to check the list of modifiers.

Patch 9.0.0424

Problem: gitattributes files are not recognized.

Solution: Add patterns to match gitattributes files. (closes #11085)

Patch 9.0.0425

Problem: Autocmd test is a bit flaky on MS-Windows.

Solution: Add a bit more sleeping. (Ken Takata, closes #11095)

Patch 9.0.0426

Problem: Failed flaky tests reports only start time.

Solution: Also report the end time.

Patch 9.0.0427

Problem: Drupal theme files are not recognized.

Solution: Use php filetype for Drupl theme files. Remove trailing spaces.
(Rodrigo Aguilera, closes #11096)

Patch 9.0.0428

Problem: Autocmd test uses common file name.

Solution: Use unique name to reduce flakiness.

Patch 9.0.0429

Problem: Not all keys are tested for the MS-Windows GUI.

Solution: Add more key codes to the list. (Yegappan Lakshmanan,
closes #11097)

Patch 9.0.0430

Problem: Cannot use repeat() with a blob.

Solution: Implement blob repeat. (closes #11090)

Patch 9.0.0431

Problem: Current mode shows in message window.

Solution: Reset in_echowindow before redrawing. (issue #11094)

Patch 9.0.0432

Problem: Crash when using for loop variable in closure.

Solution: Check that the variable wasn't deleted. (issue #11094)

Patch 9.0.0433

Problem: Coverity warns for not checking allocation failure.

Solution: Check that allocating a list or blob succeeded.

Patch 9.0.0434

Problem: gitignore files are not recognized.

Solution: Add patterns for the gitignore filetype. (closes #11102)

Patch 9.0.0435

Problem: Compiler warning for uninitialized variable.

Solution: Initialize it.

Patch 9.0.0436

Problem: CI: running tests in parallel causes flakiness.

Solution: Reorganize the MS-Windows runs. (Ken Takata, closes #11101)

Patch 9.0.0437

Problem: No error when a custom completion function returns something else than the expected list.

Solution: Give an error. (closes #11100)

Patch 9.0.0438

Problem: Cannot put virtual text above a line.

Solution: Add the "above" value for "text_align".

Patch 9.0.0439

Problem: Cursor wrong if inserting before line with virtual text above.

Solution: Add the width of the "above" virtual text to the cursor position. (issue #11084)

Patch 9.0.0440

Problem: Crash when using mkdir() with "R" flag in compiled function.

Solution: Reserve a variable for deferred function calls. Handle more than one argument.

Patch 9.0.0441

Problem: Closure in for loop test fails on some systems.

Solution: Do not wait for the ruler to show up. (issue #11106)

Patch 9.0.0442

Problem: Virtual text "above" doesn't handle line numbers.

Solution: Take the left column offset into account. (issue #11084)

Also make padding work.

Patch 9.0.0443

Problem: Blueprint files are not recognized.

Solution: Add a pattern for blueprint files. (Gabriele Musco, closes #11107)

Patch 9.0.0444

Problem: Trying to declare g:variable gives confusing error.

Solution: Give a better error message. (closes #11108)

Patch 9.0.0445

Problem: When opening/closing window text moves up/down.

Solution: Add the '**splitscroll**' option. When off text will keep its position as much as possible.

Patch 9.0.0446

Problem: Message window may be positioned too low.

Solution: Compute cmdline_row before computing the position.

Patch 9.0.0447

Problem: Using :echowin while at the hit-enter prompt causes problems.

Solution: Do not prompt for :echowin. Postpone showing the message window. Start the timer when the window is displayed.

Patch 9.0.0448

Problem: SubRip files are not recognized.

Solution: Add a pattern for SubRip. (closes #11113)

Patch 9.0.0449

Problem: There is no easy way to translate a string with a key code into a readable string.

Solution: Add the keytrans() function. (closes #11114)

Patch 9.0.0450

Problem: Return value of argument check functions is inconsistent.

Solution: Return OK/FAIL instead of TRUE/FALSE. (closes #11112)

Patch 9.0.0451

Problem: Virtual text "above" does not work with '**nowrap**'.

Solution: Do wrap the line after. (closes #11084)

Patch 9.0.0452

Problem: Visual highlighting extends into virtual text prop.

Solution: Do not highlight what isn't actually selected. Fix ordering of stored text props.

Patch 9.0.0453

Problem: On an AZERTY keyboard digit keys get the shift modifier.

Solution: Remove the shift modifier from digit keys. (closes #11109)

Patch 9.0.0454

Problem: Incorrect color for modeless selection with GTK.

Solution: Use simple inversion instead of XOR. (closes #11111)

Patch 9.0.0455

Problem: A few problems with **'scroll'**.

Solution: Fix **'scroll'** problems. (Luuk van Baal, closes #11117)

Patch 9.0.0456

Problem: Function called at debug prompt is also debugged.

Solution: Reset the debug level while entering the debug command.
(closes #11118)

Patch 9.0.0457

Problem: Substitute prompt does not highlight an empty match.

Solution: Highlight at least one character.

Patch 9.0.0458

Problem: Splitting a line with a text prop "above" moves it to a new line below.

Solution: Keep an "above" text prop above the first line.

Patch 9.0.0459

Problem: Vim9: block in for loop doesn't behave like a code block.

Solution: Use a new block ID for each loop at the script level.

Patch 9.0.0460

Problem: Loop variable can't be found.

Solution: Adjust block_id of the loop variable each round.

Patch 9.0.0461

Problem: **'scroll'** is not always updated.

Solution: Call win_init_size() at the right place.

Patch 9.0.0462

Problem: ASAN warning for integer overflow.

Solution: Check for tp_col to be MAXCOL.

Patch 9.0.0463

Problem: Command line test leaves directory behind.

Solution: Use the "R" flag on the first mkdir(). (Dominique Pellé,
closes #11127)

Patch 9.0.0464

Problem: With virtual text "above" indenting doesn't work well.

Solution: Ignore text properties while adjusting indent. (issue #11084)

Patch 9.0.0465

Problem: Cursor moves when cmdwin is closed when **'scroll'** is off.

Solution: Temporarily set **'scroll'** when jumping back to the original window. (closes #11128)

Patch 9.0.0466

Problem: Virtual text wrong after adding line break after line.

Solution: Pass an "eol" flag to where text properties are adjusted.
(closes #11131)

Patch 9.0.0467

Problem: Build failure.
Solution: Add missing change.

Patch 9.0.0468

Problem: Execution stack underflow without the +eval feature. (Dominique Pellé)

Solution: Add to execution stack without FEAT_EVAL. (closes #11135)

Patch 9.0.0469

Problem: Cursor moves if cmdwin is closed when 'splitscroll' is off.

Solution: Skip win_fix_cursor if called when cmdwin is open or closing. (Luuk van Baal, closes #11134)

Patch 9.0.0470

Problem: In a :def function all closures in a loop get the same variables.

Solution: When in a loop and a closure refers to a variable declared in the loop, prepare for making a copy of variables for each closure.

Patch 9.0.0471

Problem: No test for what patch 9.0.0469 fixes.

Solution: Add a test. (closes #11140)

Patch 9.0.0472

Problem: Virtual text "below" doesn't show in list mode.

Solution: Reset lcs_eol_one when displaying text property.

Patch 9.0.0473

Problem: fullcommand() only works for the current script version.

Solution: Add an optional argument for the script version.

Patch 9.0.0474

Problem: fullcommand() test failure.

Solution: Update function table.

Patch 9.0.0475

Problem: Not using deferred delete in tests.

Solution: Use deferred delete more often.

Patch 9.0.0476

Problem: Varargs does not work for replacement function of substitute().

Solution: Check the varargs flag of the function. (closes #11142)

Patch 9.0.0477

Problem: Missing dependency may cause crashes on incomplete build.

Solution: Add dependency.

Patch 9.0.0478

Problem: Test for 'splitscroll' takes too much time.

Solution: Only test some of the combinations. (Luuk van Baal, closes #11139)

Patch 9.0.0479

Problem: In a :def function all closures in a loop get the same variables.

Solution: Use a separate list of variables for LOADOUTER and SAVEOUTER.

Patch 9.0.0480

Problem: Cannot use a :def varargs function with substitute().

Solution: Use has_varargs(). (closes #11146)

Patch 9.0.0481

Problem: In a :def function all closures in a loop get the same variables.

Solution: Use a separate list of variables for LOADOUTER and STOREOUTER.
Not copied at end of loop yet.

Patch 9.0.0482

Problem: "g0" moves to wrong location with virtual text "above".

Solution: Compensate for the extra columns. (closes #11141) Also fix "g\$"

Patch 9.0.0483

Problem: Illegal memory access when replacing in virtualedit mode.

Solution: Check for replacing NUL after Tab.

Patch 9.0.0484

Problem: In a :def function all closures in a loop get the same variables.

Solution: Add ENDLLOOP at break, continue and return if needed.

Patch 9.0.0485

Problem: In a :def function all closures in a loop get the same variables.

Solution: Make a copy of loop variables used in a closure.

Patch 9.0.0486

Problem: Text scrolled with '**nosplitscroll**', autocmd win opened and help window closed.

Solution: Skip win_fix_scroll() in more situations. (Luuk van Baal, closes #11150)

Patch 9.0.0487

Problem: Using freed memory with combination of closures.

Solution: Do not use a partial after it has been freed through the funcstack.

Patch 9.0.0488

Problem: Cursor in wrong position with virtual text "above" and '**showbreak**'.

Solution: Take the first character column into account. (closes #11149)

Patch 9.0.0489

Problem: Using "end_lnum" with virtual text causes problems.

Solution: Disallow using "end_lnum" with virtual text. (closes #11151)
Also disallow "end_col" and "length".

Patch 9.0.0490

Problem: Using freed memory with cmdwin and BufEnter autocmd.

Solution: Make sure pointer to b_p_iminsert is still valid.

Patch 9.0.0491

Problem: No good reason to build without the float feature.

Solution: Remove configure check for float and "#ifdef FEAT_FLOAT".

Patch 9.0.0492

Problem: Cmdwin test fails on MS-Windows.
Solution: Skip test on MS-Windows.

Patch 9.0.0493

Problem: Perl test fails.
Solution: Remove remaining FEAT_EVAL.

Patch 9.0.0494

Problem: Small build misses float function declarations.
Solution: Adjust #ifdefs.

Patch 9.0.0495

Problem: Closure doesn't work properly in nested loop.
Solution: Save variables up to the outer loop.

Patch 9.0.0496

Problem: No good reason to keep supporting Windows-XP.
Solution: Drop Windows-XP support. (Ken Takata, closes #11089)

Patch 9.0.0497

Problem: LyRiCs files are not recognized.
Solution: Add a pattern to detect LyRiCs files. (closes #11155)

Patch 9.0.0498

Problem: Various small issues.
Solution: Various small fixes.

Patch 9.0.0499

Problem: In :def function list created after const is locked.
Solution: Reset v_lock. (closes #11154)

Patch 9.0.0500

Problem: When quitting the cmdline window with **CTRL-C** it remains visible.
Solution: Redraw to avoid confusion. Adjust the error message.
(closes #11152) Adjust the cursor position after **CTRL-C**.

Patch 9.0.0501

Problem: Warning for using uninitialized value in mouse test.
Solution: Clear ScreenCols when allocating it. (Dominique Pellé)

Patch 9.0.0502

Problem: A closure in a nested loop in a :def function does not work.
Solution: Use an array of loopvars, one per loop level.

Patch 9.0.0503

Problem: Build failure.
Solution: Add missing changes.

Patch 9.0.0504

Problem: still a Build failure.
Solution: Add another missing changes. Avoid compiler warning.

Patch 9.0.0505

Problem: Various problems with **'nosplitscroll'**.
Solution: Fix **'nosplitscroll'** problems. (Luuk van Baal, closes #11166)

Patch 9.0.0506
Problem: Line number argument for :badd does not work.
Solution: Set the last cursor position in the new buffer. (closes #11161)

Patch 9.0.0507
Problem: Command line cleared when using :redrawstatus in CmdlineChanged autocommand event.
Solution: Postpone the redraw. (closes #11162)

Patch 9.0.0508
Problem: When the channel test fails there is no clue why.
Solution: Add info about the job status. (Ken Takata, closes #11175)

Patch 9.0.0509
Problem: Confusing error for "saveas" command with "nofile" buffer.
Solution: Give a clearer error message. (closes #11171)

Patch 9.0.0510
Problem: Chatito files are not recognized.
Solution: Add a pattern for Chatito files. (closes #11174)

Patch 9.0.0511
Problem: Unnecessary scrolling for message of only one line.
Solution: Only set msg_scroll when needed. (closes #11178)

Patch 9.0.0512
Problem: Cannot redraw the status lines when editing a command.
Solution: Only postpone the redraw when messages have scrolled.
(closes #11170)

Patch 9.0.0513
Problem: May not be able to use a pattern at the debug prompt.
Solution: Temporarily disable the timeout. (closes #11164)

Patch 9.0.0514
Problem: Terminal test sometimes hangs.
Solution: Add a bit more information to the test output. (issue #11179)

Patch 9.0.0515
Problem: Virtual text highlight starts too early when **'number'** is set.
Solution: Set column offset when wrapping. (issue #11138)

Patch 9.0.0516
Problem: Virtual text "above" highlights gap after it.
Solution: Do not highlight the gap. (closes #11138)

Patch 9.0.0517
Problem: When at the command line :redrawstatus does not work well.
Solution: Only update the statuslines instead of the screen. (closes #11180)

Patch 9.0.0518

Problem: Virtual text highlight starts too early with 'nowrap' and 'number' set.

Solution: Add the offset to the attribute skip count. (issue #11138)

Patch 9.0.0519

Problem: The win_line() function is much too long.

Solution: Move the code to draw the line number to a separate function.

Patch 9.0.0520

Problem: Declaring a loop variable at the start of a block is clumsy.

Solution: Declare the variable inside the loop in a few places to see if this works.

Patch 9.0.0521

Problem: Compiler warns for unused argument in small version.

Solution: Add UNUSED.

Patch 9.0.0522

Problem: Build fails on Appveyor.

Solution: Select Visual Studio 2015 for the build tools.

Patch 9.0.0523

Problem: more compiler warnings for arguments in small version

Solution: Adjust #ifdefs.

Patch 9.0.0524

Problem: Build instructions for MS-Windows are outdated.

Solution: Remove instructions for old MSVC versions.

Patch 9.0.0525

Problem: Manually deleting temp test files.

Solution: Add the 'D' flag to writefile().

Patch 9.0.0526

Problem: MS-Windows: still some support for XP and old compilers.

Solution: Remove XP support and mention of old compilers. (Ken Takata, closes #11183)

Patch 9.0.0527

Problem: Long sign text may overflow buffer.

Solution: Use a larger buffer. Prevent for overflow.

Patch 9.0.0528

Problem: MS-Windows: no batch files for more recent MSVC versions.

Solution: Add batch files for 2017, 2019 and 2022. (Ken Takata, closes #11184)

Patch 9.0.0529

Problem: Appveyor setup contains outdated lines.

Solution: Remove outdated lines. (Ken Takata, closes #11182)

Patch 9.0.0530

Problem: Using freed memory when autocmd changes mark.

Solution: Copy the mark before editing another buffer.

Patch 9.0.0531

Problem: The win_line() function is much too long.

Solution: Move code to separate functions.

Patch 9.0.0532

Problem: Edit test is flaky when run under valgrind.

Solution: Send some text to the terminal to trigger a redraw.

Patch 9.0.0533

Problem: The win_line() function is much too long.

Solution: Move code to separate functions.

Patch 9.0.0534

Problem: Line number is displayed at virtual text "above".

Solution: Show the line number at the text line.

Patch 9.0.0535

Problem: Closure gets wrong value in for loop with two loop variables.

Solution: Correctly compute the number of loop variables to clear.

Patch 9.0.0536

Problem: CI: codecov action update available.

Solution: Update Codecov 3.1.0 to 3.3.1. (closes #11188)

Patch 9.0.0537

Problem: The do_set() function is much too long.

Solution: Move setting of a string option to a separate function.

Patch 9.0.0538

Problem: Manually deleting test temp files.

Solution: Add the 'D' flag to writefile().

Patch 9.0.0539

Problem: Long message test can be flaky.

Solution: Wait for more prompt instead of ruler.

Patch 9.0.0540

Problem: Assigning stack variable to argument confuses Coverity.

Solution: Use a local pointer, also makes the code simpler.

Patch 9.0.0541

Problem: Terminal pwd test fails with a very long path name.

Solution: Join two lines.

Patch 9.0.0542

Problem: MSVC build still has support for 2012 edition.

Solution: Drop MSVC 2012 support. (Ken Takata, closes #11191)

Patch 9.0.0543

Problem: Insufficient testing for assert and test functions.

Solution: Add a few more tests. (Yegappan Lakshmanan, closes #11190)

Patch 9.0.0544

Problem: Minor issues with setting a string option.
Solution: Adjust the code, add a test. (closes #11192)

Patch 9.0.0545

Problem: When a test is slow and CI times out there is no time info.
Solution: Add the elapsed time to the "Executing" message.

Patch 9.0.0546

Problem: Supporting Ruby 1.8 makes code complicated.
Solution: Drop Ruby 1.8 support, it is ancient. (Ken Takata, closes #11195)

Patch 9.0.0547

Problem: Looping over empty out_loop[] entries.
Solution: Store the array size.

Patch 9.0.0548

Problem: reduce() with a compiled lambda could be faster.
Solution: Call eval_expr_typval() instead of call_func() directly.

Patch 9.0.0549

Problem: Duplicated code in calling a :def function.
Solution: Simplify the code.

Patch 9.0.0550

Problem: Crash when closing a tabpage and buffer is NULL.
Solution: Adjust how autocommands are triggered when closing a window.
(closes #11198, closes #11197)

Patch 9.0.0551

Problem: Mode message is delayed when :echowin was used. (Maxim Kim)
Solution: Save and restore msg_didout in :echowin. (closes #11193)

Patch 9.0.0552

Problem: Crash when using NUL in buffer that uses :source.
Solution: Don't get a next line when skipping over NL.

Patch 9.0.0553

Problem: No error for "|" after "{" in lambda.
Solution: Check for invalid "|". (closes #11199)

Patch 9.0.0554

Problem: Using freed memory when command follows lambda.
Solution: Don't free what is still in use. (closes #11201)

Patch 9.0.0555

Problem: Scrolling with 'nosplitscroll' in callback changing curwin.
Solution: Invalidate w_cline_row in the right place. (Luuk van Baal,
closes #11185)

Patch 9.0.0556

Problem: Leaking memory with nested functions.
Solution: Free saved pointer.

Patch 9.0.0557

Problem: Valgrind reports possibly leaked memory.
Solution: Move the problematic test function to the "fails" test file to avoid obscuring real memory leaks.

Patch 9.0.0558

Problem: Coverity warns for possibly using NULL pointer.
Solution: Only use "evalarg" when not NULL.

Patch 9.0.0559

Problem: Timer test may get stuck at hit-enter prompt.
Solution: Feed some more characters.

Patch 9.0.0560

Problem: Elapsed time since testing started is not visible.
Solution: Show the elapsed time while running tests.

Patch 9.0.0561

Problem: When a test gets stuck it just hangs forever.
Solution: Set a timeout of 30 seconds.

Patch 9.0.0562

Problem: HSL playlist files are not recognized.
Solution: Add a pattern to recognize HSL playlist files. (Benoît Ryder, closes #11204)

Patch 9.0.0563

Problem: Timer_info() test fails.
Solution: Ignore test timeout timer. Don't use test_null_job() when not available.

Patch 9.0.0564

Problem: A few tests keep failing on MacOS M1.
Solution: Add a test check CheckNotMacM1. Fix timer tests.

Patch 9.0.0565

Problem: Cscope test causes problems when code for test timeout timer is included (even when commented out).
Solution: Disable part of the cscope test for now.

Patch 9.0.0566

Problem: Nim files are not recognized.
Solution: Add patterns for Nim files. (Nbiba Bedis, closes #11205)

Patch 9.0.0567

Problem: **'completeopt'** "longest" is not used for complete().
Solution: Also use "longest" for complete(). (Bjorn Linse, closes #11206)

Patch 9.0.0568

Problem: Autocmd code is indented more than needed.
Solution: Break out sooner. (Yegappan Lakshmanan, closes #11208)
Also in user function code.

Patch 9.0.0569

Problem: Cannot easily get out when using "vim file | grep word".

Solution: Without changes let **CTRL-C** exit Vim. Otherwise give a message on stderr. (closes #11209)

Patch 9.0.0570

Problem: CI for Windows is still using codecov action 3.1.0.

Solution: Use action 3.1.1. (closes #11212)

Patch 9.0.0571

Problem: MS-Windows: **CTRL-C** can make Vim exit.

Solution: Check the not-a-term argument.

Patch 9.0.0572

Problem: Insert complete tests leave a mapping behind.

Solution: Use a buffer-local mapping. (closes #11211)

Patch 9.0.0573

Problem: Outdated dependencies go unnoticed.

Solution: Use github Dependabot. (closes #11213)

Patch 9.0.0574

Problem: Timer garbage collect test hangs on Mac M1.

Solution: Properly check for Mac M1 and skip the test.

Patch 9.0.0575

Problem: The getchar() function behaves strangely with bracketed paste.

Solution: Do not handle paste-start in getchar(). (issue #11172)

Patch 9.0.0576

Problem: Unused loop variables.

Solution: Use a while loop instead. (closes #11214)

Patch 9.0.0577

Problem: Buffer underflow with unexpected :finally.

Solution: Check CSF_TRY can be found.

Patch 9.0.0578

Problem: One timer test fails on Mac M1.

Solution: Skip the test on Mac M1.

Patch 9.0.0579

Problem: Using freed memory when **'tagfunc'** wipes out buffer that holds **'complete'**.

Solution: Make a copy of the option. Make sure cursor position is valid.

Patch 9.0.0580

Problem: No CI running for MacOS on M1.

Solution: Add a cirrus CI task. (closes #11203)

Patch 9.0.0581

Problem: Adding a character for incsearch fails at end of line.

Solution: Only check cursor line number.

Patch 9.0.0582

Problem: Channel cwd test fails on Cirrus CI.

Solution: Also remove /private from the expected directory.

Patch 9.0.0583

Problem: Only recognizing .m3u8 files is inconsistent.

Solution: Also match .m3u files. (issue #11204)

Patch 9.0.0584

Problem: Cscope test with wrong executable name fails.

Solution: Use /bin/sh to execute the command. (Yegappan Lakshmanan)

Patch 9.0.0585

Problem: When long message test fails the error message is not visible.

Solution: Dump more lines.

Patch 9.0.0586

Problem: Missing change in test.

Solution: Add the test change.

Patch 9.0.0587

Problem: Unicode tables are outdated.

Solution: Update to Unicode release 15. (Christian Brabandt, closes #11220)

Patch 9.0.0588

Problem: MorphOS build is broken.

Solution: Add "-lm" to LDFLAGS and "-noixemul" to CFLAGS. (Ola Söder, closes #11222)

Patch 9.0.0589

Problem: On AmigaOS4 the pid is available but the task address is used.

Solution: Use getpid(). (Ola Söder, closes #11224)

Patch 9.0.0590

Problem: After exiting Insert mode spelling is not checked in the next line.

Solution: When spelling is enabled redraw the next line after exiting Insert mode in case the spell highlight needs updating.

Patch 9.0.0591

Problem: Message window popup shows on only one tab page. (Naruhiko Nishino)

Solution: Show the message window popup on all tab pages. (closes #11231)

Patch 9.0.0592

Problem: Display not cleared when scrolling back in messages, a background color is set and t_ut is empty.

Solution: Clear to the end of the display if needed. (closes #8973)

Patch 9.0.0593

Problem: CI actions have too many permissions.

Solution: Restrict permissions to what is required. (closes #11223)

Patch 9.0.0594

Problem: Makefile error message causes a shell error.

Solution: Put the message in single quotes. (closes #11232)

Patch 9.0.0595

Problem: Extra newline in messages after a verbose shell message.
Solution: Output the newline with `msg_putchar_attr()`. (closes #11233)
Make it possible to filter a screendump before comparing it.

Patch 9.0.0596

Problem: CI on Mac M1 has the channel feature disabled.
Solution: Include the channel feature.

Patch 9.0.0597

Problem: Cannot close a tab page with the middle mouse button.
Solution: Support closing a tab page with the middle mouse button, like many other programs. (closes #10746)

Patch 9.0.0598

Problem: Using negative array index with negative width window.
Solution: Make sure the window width does not become negative.

Patch 9.0.0599

Problem: Latexmkrc files are not recognized.
Solution: Use Perl filetype for latexmkrc files. (closes #11241)

Patch 9.0.0600

Problem: GYP files are not recognized.
Solution: Recognize GYP files. (closes #11242)

Patch 9.0.0601

Problem: Too much indent.
Solution: Return out early from a function. (Yegappan Lakshmanan, close #11238)

Patch 9.0.0602

Problem: New TypeScript extensions are not recognized.
Solution: Recognize `.mts` and `.cts` files. (closes #11237)

Patch 9.0.0603

Problem: With `'nosplitscroll'` folds are not handled correctly.
Solution: Take care of closed folds when moving the cursor. (Luuk van Baal, closes #11234)

Patch 9.0.0604

Problem: Luacheckrc file is not recognized.
Solution: Use lua filetype for luacheckrc. (closes #11236)

Patch 9.0.0605

Problem: Dump file missing.
Solution: Add the missing dump file. (issue #11234)

Patch 9.0.0606

Problem: `system()` opens a terminal window when using the GUI and `"!"` is in `'guioptions'`.
Solution: Do not use a terminal window when the `SHELL_SILENT` flag is used. (closes #11202)

Patch 9.0.0607

Problem: Verbose echo message test fails on Mac OS.

Solution: Skip the test on Mac OS.

Patch 9.0.0608

Problem: With spell checking, deleting a full stop at the end of a line does not update SpellCap at the start of the next line.

Solution: Update the next line when characters have been deleted. Also when using undo.

Patch 9.0.0609

Problem: Blockedit test fails because of wrong indent.

Solution: Adjust the expected text temporarily

Patch 9.0.0610

Problem: Global interrupt test fails when run under valgrind.

Solution: Use TermWait().

Patch 9.0.0611

Problem: Tests delete files with a separate delete() call.

Solution: Use deferred delete.

Patch 9.0.0612

Problem: Blockedit test passes with wrong result.

Solution: Add a "vim9script" line to make indenting work.

Patch 9.0.0613

Problem: Running source tests leaves file behind.

Solution: Add the "D" flag to writefile(). (Dominique Pellé, closes #11243)
Also do this for other writefile() calls and drop delete().

Patch 9.0.0614

Problem: SpellFileMissing autocmd may delete buffer.

Solution: Disallow deleting the current buffer to avoid using freed memory.

Patch 9.0.0615

Problem: Using reduce() on a list from range() is a bit slow.

Solution: Avoid materializing the list.

Patch 9.0.0616

Problem: Spell test fails because error message changed.

Solution: Adjust expected error message.

Patch 9.0.0617

Problem: Calling function for reduce() has too much overhead.

Solution: Only call clear_tv() when needed.

Patch 9.0.0618

Problem: Calling function for reduce() has too much overhead.

Solution: Do not create a funccall_T every time.

Patch 9.0.0619

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0620

Problem: matchaddpos() can only add up to 8 matches.

Solution: Allocate the array of positions. (closes #11248)

Patch 9.0.0621

Problem: Filetype test leaves file behind.

Solution: Add deferred delete flag to writefile(). (Dominique Pellé, closes #11249)

Patch 9.0.0622

Problem: matchaddpos() can get slow when adding many matches.

Solution: Update the next available match ID when manually picking an ID and remove check if the available ID can be used. (idea by Rick Howe)

Patch 9.0.0623

Problem: Error for modifying a const is not detected at compile time.

Solution: Add TTFLAG_CONST and check for it in add() and extend().

Patch 9.0.0624

Problem: Leaking argument type array.

Solution: Add allocated memory to type_gap.

Patch 9.0.0625

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0626

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0627

Problem: "const" and "final" both make the type a constant. (Daniel Steinberg)

Solution: Only have "const" make the type a constant.

Patch 9.0.0628

Problem: Coverity warns for not checking return value.

Solution: Check the return value and simplify the code.

Patch 9.0.0629

Problem: Get an error for using const only when executing.

Solution: Check for const at compile time for filter(), map(), remove(), reverse(), sort() and uniq().

Patch 9.0.0630

Problem: In Vim9 script a numbered function cannot be called.

Solution: Do not require "g:" before a numbered function name. (closes #11254)

Patch 9.0.0631

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0632

Problem: Calling a function from an "expr" option has too much overhead.

Solution: Add call_simple_func() and use it for **'foldexpr'**

Patch 9.0.0633

Problem: FEAT_TITLE was removed but is still used.

Solution: Remove FEAT_TITLE. (Naruhiko Nishino, closes #11256)

Patch 9.0.0634

Problem: Evaluating "expr" options has more overhead than needed.

Solution: Use call_simple_func() for **'foldtext'**, **'includeexpr'**, **'printexpr'**,
"expr" of **'spellsuggest'**, **'diffexpr'**, **'patchexpr'**, **'balloonexpr'**,
'formatexpr', **'indentexpr'** and **'charconvert'**.

Patch 9.0.0635

Problem: Build error and compiler warnings.

Solution: Add missing change. Add type casts.

Patch 9.0.0636

Problem: Underline color does not work in terminals that don't send a
termresponse.

Solution: Do output t_8u if it was set explicitly. (closes #11253)

Patch 9.0.0637

Problem: Syntax of commands in Vim9 script depends on +eval feature.

Solution: Use same syntax with and without the +eval feature.

Patch 9.0.0638

Problem: Popup menu highlight wrong on top of preview popup. (Yegappan
Lakshmanan)

Solution: Also check for the popup menu in screen_line().

Patch 9.0.0639

Problem: Checking for popup in screen_char() is too late, the attribute has
already been changed.

Solution: Move check for popup to where screen_char() is called.

Patch 9.0.0640

Problem: Cannot scroll by screen line if a line wraps.

Solution: Add the **'smoothscroll'** option. Only works for **CTRL-E** and **CTRL-Y**
so far.

Patch 9.0.0641

Problem: Missing part of the new option code.

Solution: Add missing WV_SMS.

Patch 9.0.0642

Problem: Breakindent test fails.

Solution: Correct logic for resetting need_showbreak.

Patch 9.0.0643

Problem: Smoothscroll test fails.

Solution: Check if skipcol changed.

Patch 9.0.0644

Problem: **'smoothscroll'** is not copied to a new window on `:split`.

Solution: Copy the option value. Add a test.

Patch 9.0.0645

Problem: **CTRL-Y** does not stop at line 1. (John Marriott)

Solution: Stop at line 1 when **'smoothscroll'** is not set. (closes #11261)

Patch 9.0.0646

Problem: with **'smoothscroll'** set **CTRL-E** does not work properly when **'foldmethod'** is set to "indent". (Yee Cheng Chin)

Solution: Merge the code for scrolling with folds and **'smoothscroll'**. (closes #11262)

Patch 9.0.0647

Problem: The **'splitscroll'** option is not a good name.

Solution: Rename **'splitscroll'** to **'splitkeep'** and make it a string option, also supporting "topline". (Luuk van Baal, closes #11258)

Patch 9.0.0648

Problem: When using powershell input redirection does not work.

Solution: Use a different shell command for powershell. (Yegappan Lakshmanan, closes #11257)

Patch 9.0.0649

Problem: No indication when the first line is broken for **'smoothscroll'**.

Solution: Show "<<<" in the first line.

Patch 9.0.0650

Problem: Some tests are failing.

Solution: Adjust for "<<<" showing up.

Patch 9.0.0651

Problem: Build fails without the +conceal feature.

Solution: Rename called function.

Patch 9.0.0652

Problem: **'smoothscroll'** not tested with **'number'** and "n" in **'cpo'**.

Solution: Add tests, fix uncovered problem.

Patch 9.0.0653

Problem: BS and DEL do not work properly in an interactive shell. (Gary Johnson)

Solution: Adjust the length for replaced codes.

Patch 9.0.0654

Problem: Breakindent test fails.

Solution: Temporarily accept wrong result.

Patch 9.0.0655

Problem: passing modifier codes to a shell running in the GUI. (Gary Johnson)

Solution: Include modifier codes into the key and drop the modifiers.

Patch 9.0.0656

Problem: Cannot specify another character to use instead of '@' at the end of the window.

Solution: Add "lastline" to **'fillchars'**. (Martin Tournoij, closes #11264, closes #10963)

Patch 9.0.0657

Problem: Too many #ifdefs.

Solution: Graduate the +cmdwin feature. Now the tiny and small builds are equal, drop the small build. (Martin Tournoij, closes #11268)

Patch 9.0.0658

Problem: Tiny build fails on Mac OS.

Solution: Define FEAT_CLIPBOARD only for normal build.

Patch 9.0.0659

Problem: Wrong type of comment in SetSyn() function.

Solution: Use Vim9 comment. (closes #11278)

Patch 9.0.0660

Problem: Mapping with CTRL keys does not work in the GUI.

Solution: Recognize CSI next to K_SPECIAL. (closes #11275, closes #11270)

Patch 9.0.0661

Problem: Multi-byte "lastline" item in **'fillchars'** does not work properly when the window is two columns wide.

Solution: Compute the text length correctly. (closes #11280)

Patch 9.0.0662

Problem: Concealed characters do not work correctly.

Solution: Subtract boguscols instead of adding them. (closes #11273)

Patch 9.0.0663

Problem: Tests check for +cmdwin feature which is always present.

Solution: Remove the checks. (closes #11287)

Patch 9.0.0664

Problem: Bad redrawing with spell checking, using "C" and "\$" in **'cpo'**.

Solution: Do not redraw the next line when "\$" is in **'cpo'**. (closes #11285)

Patch 9.0.0665

Problem: Setting **'cmdheight'** has no effect if last window was resized.

Solution: Do apply **'cmdheight'** when told to. Use the frame height instead of the cmdline_row. (closes #11286)

Patch 9.0.0666

Problem: Spacing-combining characters handled as composing, causing text to take more space than expected.

Solution: Handle characters marked with "Mc" not as composing. (closes #11282)

Patch 9.0.0667

Problem: ml_get error when **'splitkeep'** is "screen". (Marius Gedminas)

Solution: Check the botline is not too large. (Luuk van Baal, closes #11293, closes #11292)

Patch 9.0.0668

Problem: CI on Mac M1 only uses clang

Solution: Also run with gcc. (closes #11263)

Patch 9.0.0669

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0670

Problem: No space for command line when there is a tabline.

Solution: Correct computation of where the command line should be. (closes #11295)

Patch 9.0.0671

Problem: Negative topline using **CTRL-Y** with **'smoothscroll'** and **'diff'**. (Ernie Rael)

Solution: Only use **'smoothscroll'** when **'wrap'** is set.

Patch 9.0.0672

Problem: Cursor line only partly shows with **'smoothscroll'** and **'scrolloff'** zero.

Solution: Do not use **'smoothscroll'** when adjusting the bottom of the window. (closes #11269)

Patch 9.0.0673

Problem: First line not scrolled properly with **'smoothscroll'** and **'scrolloff'** zero and using "k".

Solution: Make sure the cursor position is visible.

Patch 9.0.0674

Problem: Build error with tiny version.

Solution: Use PLINES_NOFILL macro.

Patch 9.0.0675

Problem: Search test screendump is outdated.

Solution: Update the screendump for improved display.

Patch 9.0.0676

Problem: CI on Mac M1 with gcc actually uses clang.

Solution: Remove the gcc task. (Ozaki Kiichi, closes #11297)

Patch 9.0.0677

Problem: Breakindent test accepts wrong result.

Solution: Fix the number column and adjust the expected text.

Patch 9.0.0678

Problem: Using exclamation marks on :function.

Solution: Use :func and :endfunc as usual.

Patch 9.0.0679

Problem: Tests failing with **'smoothscroll'**, **'number'** and "n" in **'cpo'**.

Solution: Do not count number column in topline if columns are skipped.

Patch 9.0.0680

Problem: Tests failing with **'breakindent'**, **'number'** and **"n"** in **'cpo'**.

Solution: Do count the number column in topline if **'breakindent'** is set.

Patch 9.0.0681

Problem: **"<<<"** shows for **'smoothscroll'** even when **'showbreak'** is set.

Solution: When **'showbreak'** is set do not display **"<<<"**.

Patch 9.0.0682

Problem: Crash when popup with deleted timer is closed. (Igbanam Ogbulujah)

Solution: Check the timer still exists. (closes #11301)

Patch 9.0.0683

Problem: Cannot specify a time for **:echowindow**.

Solution: A count can be used to specify the display time. Add **popup_findecho()**.

Patch 9.0.0684

Problem: Skipped **:exe** command fails compilation on MS-Windows.

Solution: Adjust return value when skipping.

Patch 9.0.0685

Problem: **FORTIFY_SOURCE** causes a crash in Vim9 script.

Solution: Use a pointer to the first char. (Yee Cheng Chin, closes #11302)

Patch 9.0.0686

Problem: The right **ALT** key does not work on some MS-Windows keyboards.

Solution: Adjust the modifiers based on **GetKeyState()**. (Anton Sharonov, closes #11300)

Patch 9.0.0687

Problem: **"export def"** does not work in a nested block.

Solution: Do not handle **"export"** with a separate function but in the same command stack. (closes #11304)

Patch 9.0.0688

Problem: Debugger does not display the whole command.

Solution: Set **ea.cmd** before checking for a breakpoint.

Patch 9.0.0689

Problem: Compiler warning for unused function.

Solution: Add **#ifdef**. (John Marriott)

Patch 9.0.0690

Problem: Buffer size for expanding tab not correctly computed.

Solution: Correctly use size of end character.

Patch 9.0.0691

Problem: **lalloc(0)** error in **listchars** test.

Solution: Skip generating text for tab if **tab_len** is zero.

Patch 9.0.0692

Problem: PoE filter files are not recognized.

Solution: Add a pattern to detect PoE filter files. (closes #11305)

Patch 9.0.0693

Problem: browse() first argument cannot be a bool.

Solution: Use tv_get_bool_chk() instead of tv_get_number_chk().
(closes #11308)

Patch 9.0.0694

Problem: No native sound support on Mac OS.

Solution: Add sound support for Mac OS. (Yee Cheng Chin, closes #11274)

Patch 9.0.0695

Problem: Failing check for dictionary type for const any.

Solution: Check for any type properly. (closes #11310)

Patch 9.0.0696

Problem: It is unclear if the +rightleft and +arabic features are actively being used.

Solution: Disable the features, await feedback.

Patch 9.0.0697

Problem: Cursor in wrong position with Visual substitute.

Solution: When restoring '**linebreak**' mark the virtual column as invalid.
(closes #11309, closes #11311)

Patch 9.0.0698

Problem: VisVim is outdated, does not work with current Visual Studio.

Solution: Remove VisVim. (Martin Tournoij)

Patch 9.0.0699

Problem: Tiny build fails.

Solution: Add #ifdef.

Patch 9.0.0700

Problem: There is no real need for a "big" build.

Solution: Move common features to "normal" build, less often used features to the "huge" build. (Martin Tournoij, closes #11283)

Patch 9.0.0701

Problem: With '**smoothscroll**' the cursor position is not adjusted in a long line.

Solution: Move the cursor further up or down in the line.

Patch 9.0.0702

Problem: Incomplete testing cursor position after change with '**linebreak**' set.

Solution: Add a test and move test cases together. (closes #11313)

Patch 9.0.0703

Problem: Failing check for argument type for const any.

Solution: Check for any type properly. (closes #11316)

Patch 9.0.0704

Problem: CI runs "tiny" and "small" builds, which are the same.

Solution: Remove the "small" build. (Naruhiko Nishino, closes #11315)

Patch 9.0.0705

Problem: Virtual text truncation does not take padding into account.

Solution: Subtract the padding from the available space. (closes #11318)

Patch 9.0.0706

Problem: :help in a narrow window always opens at the top.

Solution: Respect '**splitbelow**'. (closes #11319)

Patch 9.0.0707

Problem: With '**smoothscroll**' and '**scrolloff**' non-zero the cursor position is not properly adjusted in a long line.

Solution: Move the cursor further up or down in the line.

Patch 9.0.0708

Problem: :confirm does not work properly for a terminal buffer.

Solution: Handle :confirm for a terminal buffer differently. (Yee Cheng Chin, closes #11312)

Patch 9.0.0709

Problem: Virtual text "after" not correct with '**nowrap**'.

Solution: Do not display "after" text prop on the next line when '**wrap**' is off.

Patch 9.0.0710

Problem: Quitting/unloading/hiding a terminal buffer does not always work properly.

Solution: Avoid that ":q!" leaves an empty buffer behind. ":bunload!" also kills the job and unloads the buffer. ":hide" does not unload the buffer. (Yee Cheng Chin, closes #11323)

Patch 9.0.0711

Problem: SubStation Alpha files are not recognized.

Solution: Add patterns for SubStation Alpha files. (closes #11332)

Patch 9.0.0712

Problem: Wrong column when calling setcursorcharpos() with zero lnum.

Solution: Set the line number before calling buf_charidx_to_byteidx(). (closes #11329)

Patch 9.0.0713

Problem: <amatch> of MenuPopup event is expanded like a file name.

Solution: Do not expand <amatch> for MenuPopup. (closes #11328)

Patch 9.0.0714

Problem: With '**nowrap**' two virtual text below not displayed correctly.

Solution: Set text_prop_follows before continuing. Correct for number column. (closes #11333)

Patch 9.0.0715

Problem: Wrong argument for append() gives two error messages.

Solution: When getting an error for a number argument don't try using it as a string. (closes #11335)

Patch 9.0.0716

Problem: With **'nowrap'** virtual text "after" does not scroll left.

Solution: Skip part of the virtual text that is left of the window.
(closes #11320) Fix going beyond the last column of the window.

Patch 9.0.0717

Problem: Compiler warning for unused variable in tiny build.

Solution: Add #ifdefs.

Patch 9.0.0718

Problem: Extra empty line between two virtual text "below" when **'wrap'** and **'number'** are set.

Solution: Reset "before" when there is no text in the screen line.
(closes #11334)

Patch 9.0.0719

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0720

Problem: MS-Windows GUI may have pixel dust from antialiasing.

Solution: When a character changes also redraw the next one. (issue #8532)

Patch 9.0.0721

Problem: Virtual text "above" with padding not displayed correctly.

Solution: Take padding into account when truncating. (closes #11340)

Patch 9.0.0722

Problem: Virtual text "after" does not show with **'list'** set.

Solution: Do not break out of the loop when another text prop follows.
(closes #11337)

Patch 9.0.0723

Problem: Extra empty line below virtual text when **'list'** is set.

Solution: Do not reset lcs_eol_one but set text_prop_follows. (closes #11339)

Patch 9.0.0724

Problem: Closure in compiled function gets same variable in block.

Solution: At the end of a block to not always reset the variable count.
(issue #11094)

Patch 9.0.0725

Problem: Virtual text "after" wraps to next line even when **'wrap'** is off and **'list'** is set.

Solution: Do not use the minimum width when **'wrap'** is off. (issue #11336)

Patch 9.0.0726

Problem: Looping over list of lists and changing the list contents works in Vim9 script, not in a compiled function.

Solution: Mark the loop variable final instead of const. (closes #11347)

Patch 9.0.0727

Problem: Help in the repository differs from patched version too much.
Solution: Make a patch for a few help files.

Patch 9.0.0728

Problem: extend() test fails.
Solution: Item is final, not const.

Patch 9.0.0729

Problem: The rightleft and arabic features are disabled.
Solution: Re-enable the features, some users want to use the functionality.

Patch 9.0.0730

Problem: Startup test fails with right-left feature.
Solution: Do not delete test file too early.

Patch 9.0.0731

Problem: clang-tidy configuration files are not recognized.
Solution: Recognize clang-tidy files as yaml. (closes #11350)

Patch 9.0.0732

Problem: No check for white space before and after "=<<". (Doug Kearns)
Solution: Check for white space in Vim9 script. (closes #11351)

Patch 9.0.0733

Problem: Use of strftime() is not safe.
Solution: Check the return value of strftime(). Use a larger buffer and correctly pass the available space. (Dominique Pellé, closes #11348)

Patch 9.0.0734

Problem: Cursor position invalid when scrolling with **'smoothscroll'** set. (Ernie Rael)
Solution: Add w_valid_skipcol and clear flags when it changes. Adjust w_skipcol after moving the cursor.

Patch 9.0.0735

Problem: Breakindent and scrolloff tests fail.
Solution: Temporarily skip the assertions.

Patch 9.0.0736

Problem: Quickfix listing does not handle very long messages.
Solution: Use a growarray instead of a fixed size buffer. (Yegappan Lakshmanan, closes #11357)

Patch 9.0.0737

Problem: Lisp word only recognized when a space follows.
Solution: Also match a word at the end of a line. Rename the test. Use a compiled function to avoid backslashes.

Patch 9.0.0738

Problem: Cannot suppress completion "scanning" messages.
Solution: Add the "C" flag in **'shortmess'**. (Bjorn Linse, closes #11354)

Patch 9.0.0739

Problem: Mouse column not correctly used for popup_setpos.

Solution: Adjust off-by-one error and handle Visual line selection properly.
(Yee Cheng Chin, closes #11356)

Patch 9.0.0740

Problem: prop_add_list() gives multiple errors for invalid argument.

Solution: Only give one error message.

Patch 9.0.0741

Problem: Cannot specify an ID for each item with prop_add_list(). (Sergey Vlasov)

Solution: Add an optional fifth number to the item. (closes #11360)

Patch 9.0.0742

Problem: Reading past end of the line when compiling a function with errors.

Solution: Do not return an invalid pointer. Fix skipping redirection.

Patch 9.0.0743

Problem: Starting cscope on Unix does not quote the arguments correctly. (Gary Johnson)

Solution: Move the final quote after the arguments.

Patch 9.0.0744

Problem: In script in autoload dir exported variable is not found. (Doug Kearns)

Solution: Find the variable with the "script#" prefix. (closes #11361)

Patch 9.0.0745

Problem: Wrong cursor position when using "gj" and "gk" in a long line.

Solution: Adjust computations for the cursor position and skipcol. Re-enable tests that pass now, disable failing breakindent test.

Patch 9.0.0746

Problem: Breakindent test cases are commented out.

Solution: Adjust expected result to slightly different behavior. Correct computations for cursor position.

Patch 9.0.0747

Problem: Too many #ifdefs.

Solution: Graduate the +cmdline_info feature. (Martin Tournoij, closes #11330)

Patch 9.0.0748

Problem: Kitty may send key without modifiers with CSI u code.

Solution: Handle CSI u code without modifiers. (Trygve Aaberge, closes #11364)

Patch 9.0.0749

Problem: Alloc/free of buffer for each quickfix entry is inefficient.

Solution: Use a shared grow array. (Yegappan Lakshmanan, closes #11365)

Patch 9.0.0750

Problem: Crash when popup closed in callback. (Maxim Kim)
Solution: In syntax_end_parsing() check that syn_block is valid.

Patch 9.0.0751

Problem: **'scrolloff'** does not work well with **'smoothscroll'**.
Solution: Make positioning the cursor a bit better. Rename functions.

Patch 9.0.0752

Problem: Rprofile files are not recognized.
Solution: Recognize Rprofile files as "r". (closes #11369)

Patch 9.0.0753

Problem: Some Ex commands are not in the help index.
Solution: Add the missing commands. Add a script to check all Ex commands are in the help index. (Yee Cheng Chin, closes #11371)

Patch 9.0.0754

Problem: **'indentexpr'** overrules lisp indenting in one situation.
Solution: Add "else" to keep the lisp indent. (issue #11327)

Patch 9.0.0755

Problem: Huge build on macos always fails on CI.
Solution: Temporarily disable the perl interface.

Patch 9.0.0756

Problem: No autocmd event for changing text in a terminal window.
Solution: Add TextChangedT. (Shougo Matsushita, closes #11366)

Patch 9.0.0757

Problem: Line number not visible with **'smoothscroll'**, **'nu'** and **'rnu'**.
Solution: Put the ">>>" after the line number instead of on top.

Patch 9.0.0758

Problem: "precedes" from **'listchars'** overwritten by <<< for **'smoothscroll'**.
Solution: Keep the "precedes" character.

Patch 9.0.0759

Problem: Huge build on macos does not use Perl.
Solution: Re-enable the Perl interface using "dynamic". (closes #11375)

Patch 9.0.0760

Problem: Display test for **'listchars'** "precedes" fails.
Solution: Correct the expected result.

Patch 9.0.0761

Problem: Cannot use **'indentexpr'** for Lisp indenting.
Solution: Add the **'lispoptions'** option.

Patch 9.0.0762

Problem: Build failure.
Solution: Add missing change.

Patch 9.0.0763

Problem: MS-Windows: warning for using int for size_t.

Solution: Declare variable as size_t.

Patch 9.0.0764

Problem: Indent and option tests fail.

Solution: Change OP_INDENT. Add entry to options test table.

Patch 9.0.0765

Problem: With a Visual block a put command column may go negative.

Solution: Check that the column does not become negative.

Patch 9.0.0766

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0767

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0768

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0769

Problem: Too many delete() calls in tests.

Solution: Use deferred delete where possible.

Patch 9.0.0770

Problem: Quickfix commands may keep memory allocated.

Solution: Free memory when it's a bit much. (Yegappan Lakshmanan, closes #11379)

Patch 9.0.0771

Problem: Cannot always tell the difference between tex and rexx files.

Solution: Recognize tex by a leading backslash. (Martin Tournouij, closes #11380)

Patch 9.0.0772

Problem: The libvterm code is outdated.

Solution: Include libvterm changes from revision 790 to 801.

Patch 9.0.0773

Problem: Huge build on MacOS uses dynamic Perl.

Solution: Use built-in Perl, uninstall the brew one. (closes #11382)

Patch 9.0.0774

Problem: The libvterm code is outdated.

Solution: Include libvterm changes from revision 802 to 817. Revert some changes made for C89.

Patch 9.0.0775

Problem: MS-Windows: mouse scrolling not supported in the console.

Solution: Add event handling for mouse scroll events. (Christopher Plewright, closes #11374)

Patch 9.0.0776

Problem: MSVC can't have field name "small".
Solution: Rename small to smallfont.

Patch 9.0.0777

Problem: Code is indented too much.
Solution: Use an early return. (Yegappan Lakshmanan, closes #11386)

Patch 9.0.0778

Problem: Indexing of unknown const type fails during compilation.
Solution: Check for "any" properly. (closes #11389)

Patch 9.0.0779

Problem: lsl and lm3 file extensions are not recognized.
Solution: Add *.lsl and *.lm3 patterns. (Doug Kearns, closes #11384)

Patch 9.0.0780

Problem: **'scroll'** value computed in unexpected location.
Solution: Compute **'scroll'** when the window height is changed. (Luuk van Baal, closes #11387)

Patch 9.0.0781

Problem: Workaround to rename "small" to "smallfont" is clumsy.
Solution: Undefine "small" after including windows.h. (Ken Takata)

Patch 9.0.0782

Problem: OpenVPN files are not recognized.
Solution: Add patterns for OpenVPN files. (closes #11391)

Patch 9.0.0783

Problem: "!!" doesn't do anything but does update the previous command.
Solution: Do not have "!!" change the previous command. (Martin Tournoij, closes #11372)

Patch 9.0.0784

Problem: Text prop "above" not displayed correctly with **'number'** and "n" in **'cpo'**.
Solution: Draw the line number column until the line text is reached.

Patch 9.0.0785

Problem: Memory leak with empty shell command.
Solution: Free the allocated memory when bailing out.

Patch 9.0.0786

Problem: User command does not get number from :tab modifier.
Solution: Include the number. (closes #11393, closes #6901)

Patch 9.0.0787

Problem: MS-Windows: mouse scrolling in terminal misbehaves without dll.
Solution: Add #ifdef as a temporary solution. (Christopher Plewright, closes #11392)

Patch 9.0.0788

Problem: ModeChanged autocmd not executed when Visual mode is ended with

CTRL-C.

Solution: Do not trigger the autocmd when got_int is set. (closes #11394)

Patch 9.0.0789

Problem: Dummy buffer ends up in a window.

Solution: Disallow navigating to a dummy buffer.

Patch 9.0.0790

Problem: Test for dummy buffer does not always produce the E86 error.

Solution: Do not check if the error is produced.

Patch 9.0.0791

Problem: At the hit-Enter prompt the End and Home keys may not work.

Solution: Use the special "@" code for End and Home, like it was done for the cursor keys in patch 8.2.2246. (Trygve Aaberge, closes #11396)

Patch 9.0.0792

Problem: MS-Windows: compiler complains about unused function.

Solution: Add #ifdef. (John Marriott)

Patch 9.0.0793

Problem: MS-Windows: mouse scroll events only work with the dll.

Solution: Accept CSI codes for MS-Windows without the GUI. (Christopher Plewright, closes #11401)

Patch 9.0.0794

Problem: There is no way to find out if an escape sequence with modifyOtherKeys has been seen.

Solution: Add a notice with ":verbose map".

Patch 9.0.0795

Problem: readblob() always reads the whole file.

Solution: Add arguments to read part of the file. (Ken Takata, closes #11402)

Patch 9.0.0796

Problem: Mapping test fails in some situations.

Solution: Find the line with the verbose information.

Patch 9.0.0797

Problem: Order of assert function arguments is reverted.

Solution: Swap the arguments. (closes #11399)

Patch 9.0.0798

Problem: Clang format configuration files are not recognized.

Solution: Use yaml for Clang format configuration files. (Marwin Glaser, closes #11398)

Patch 9.0.0799

Problem: In compiled function ->() on next line not recognized.

Solution: Also check for "(". (closes #11405)

Patch 9.0.0800

Problem: Compiler complains about repeated typedef.

Solution: Remove one typedef.

Patch 9.0.0801

Problem: The modifyOtherKeys flag is set when it should not.

Solution: Do not handle special key codes with a modifier value above 16 as a modifyOtherKeys value. (issue #11403)

Patch 9.0.0802

Problem: MS-Windows: cannot map console mouse scroll events.

Solution: Change CSI to K_SPECIAL when checking for a mapping. (Christopher Plewright, closes #11410)

Patch 9.0.0803

Problem: readblob() cannot read from character device.

Solution: Use S_ISCHR() to not check the size. (Ken Takata, closes #11407)

Patch 9.0.0804

Problem: Crash when trying to divide the largest negative number by -1.

Solution: Handle this case specifically.

Patch 9.0.0805

Problem: Filetype autocmd may cause freed memory access.

Solution: Set the quickfix-busy flag while filling the buffer.

Patch 9.0.0806

Problem: **'langmap'** works differently when there are modifiers.

Solution: Only apply **'langmap'** to a character where modifiers have no effect. (closes #11395, closes #11404)

Patch 9.0.0807

Problem: With **'smoothscroll'** typing "0" may not go to the first column.

Solution: Recompute w_cline_height when needed. Do not scroll up when it would move the cursor.

Patch 9.0.0808

Problem: jsonnet filetype detection has a typo.

Solution: Change "libjsonnet" to "libsonnet". (Maxime Brunet, closes #11412)

Patch 9.0.0809

Problem: Test for job writing to buffer fails.

Solution: Correct w_topline when deleting a buffer line.

Patch 9.0.0810

Problem: readblob() returns empty when trying to read too much.

Solution: Return what is available.

Patch 9.0.0811

Problem: Error if :echowin is preceded by a command modifier.

Solution: Do not give an error for range when there is a modifier. (closes #11414)

Patch 9.0.0812

Problem: GUI mouse scrollwheel mappings don't work.

Solution: Add check for "gui.in_use". (Christopher Plewright, closes #11418)

Patch 9.0.0813

Problem: Kitty terminal is not recognized.

Solution: Recognize Kitty by the termresponse and then do not set seenModifyOtherKeys, since Kitty doesn't support that. (issue #11413)

Patch 9.0.0814

Problem: Aws config files are not recognized.

Solution: Use "confini" for aws config files. (Justin M. Keyes, closes #11416)

Patch 9.0.0815

Problem: "!!" does not switch to the alternate screen.

Solution: For "!!" don't clear the previous command. (closes #11420, closes #11409)

Patch 9.0.0816

Problem: **CTRL-Z** at end of file is always dropped.

Solution: Add the 'endoffile' option, like the 'endofline' option. (closes #11408, closes #11397)

Patch 9.0.0817

Problem: Build error.

Solution: correct variable name.

Patch 9.0.0818

Problem: "!ls" does not work.

Solution: Do not free memory that is in use.

Patch 9.0.0819

Problem: Still a build error, tests are failing.

Solution: Correct recent changes. Add missing init for 'eof'.

Patch 9.0.0820

Problem: Memory leak with empty shell command.

Solution: Free the empty string.

Patch 9.0.0821

Problem: Memory leak with empty shell command.

Solution: Free the empty string.

Patch 9.0.0822

Problem: Crash when dragging the statusline with a mapping.

Solution: Check for valid window pointer. (issue #11427)

Patch 9.0.0823

Problem: Mouse drag test fails.

Solution: Only reset the mouse click flag when actually switching to another tab page. Disable test that keeps failing.

Patch 9.0.0824

Problem: Crash when using win_move_separator() in other tab page.

Solution: Check for valid window in current tab page.

(closes #11479, closes #11427)

Patch 9.0.0825

Problem: Cannot drag an entry in the tabpage line.

Solution: Clear dragwin instead of got_click. (closes #11483, closes #11482)

Patch 9.0.0826

Problem: If **'endoffline'** is set the **CTRL-Z** may be written in the wrong place.

Solution: Write **CTRL-Z** at the end of the file. Update the help to explain the possibilities better. (Ken Takata, closes #11486)

Patch 9.0.0827

Problem: The **<Home>** key in tmux doesn't work when **'term'** is set to "xterm". (Dominique Pellé)

Solution: Only use '@' in a termcap key entry for "1" when ";" follows. (closes #11429)

Patch 9.0.0828

Problem: Various typos.

Solution: Correct typos. (closes #11432)

Patch 9.0.0829

Problem: Wrong counts in macro comment.

Solution: Update the value counts. (closes #11480)

Patch 9.0.0830

Problem: Compiling with Perl on Mac 12 fails.

Solution: Suppress infinite warnings. (closes #11499)

Patch 9.0.0831

Problem: Compiler warning for redefining HAVE_DUP.

Solution: Undefine HAVE_DUP if needed. (Ozaki Kiichi, closes #11484)

Patch 9.0.0832

Problem: Deprecation warning causes build failure.

Solution: Suppress deprecation warning. (closes #11503)

Patch 9.0.0833

Problem: Mac: no +sound feature in huge build.

Solution: Enable +sound in Mac huge build. (closes #11497)

Patch 9.0.0834

Problem: Warning for missing return type.

Solution: Add "int". (San James, closes #11496)

Patch 9.0.0835

Problem: The window title is not redrawn when **'endoffile'** changes.

Solution: redraw the window title when **'endoffile'** is changed. (Ken Takata, closes #11488)

Patch 9.0.0836

Problem: Wrong error when using extend() with funcref.

Solution: Better check the variable type. (closes #11468, closes #11455)

Patch 9.0.0837

Problem: append() reports failure when not appending anything.

Solution: Only report failure when appending something. (closes #11498)

Patch 9.0.0838

Problem: Compiler warnings for unused variables.

Solution: Adjust #ifdef and remove unused variables. (John Marriott)

Patch 9.0.0839

Problem: Test may fail depending on sequence of events.

Solution: Accept error codes in either order. (Yee Cheng Chin, closes #11510)

Patch 9.0.0840

Problem: Cannot change a slice of a const list. (Takumi KAGIYAMA)

Solution: Remove the const flag from the slice type. (closes #11490)

Patch 9.0.0841

Problem: deletebufline() does not always return 1 on failure.

Solution: Refactor the code to make it work more predictable. (closes #11511)

Patch 9.0.0842

Problem: Unicode range for Apple SF symbols is outdated.

Solution: Update to SF Symbols 4. (Yee Cheng Chin, closes #11474)

Patch 9.0.0843

Problem: VHS tape files are not recognized.

Solution: Add a filetype pattern. (Carlos Alexandro Becker, closes #11452)

Patch 9.0.0844

Problem: Handling 'statusline' errors is spread out.

Solution: Pass the option name to the lower levels so the option can be reset there when an error is encountered. (Luuk van Baal, closes #11467)

Patch 9.0.0845

Problem: Shell command with just space gives strange error.

Solution: Skip white space at start of the argument. (Christian Brabandt, Shane-XB-Qian, closes #11515, closes #11495)

Patch 9.0.0846

Problem: Using assert_fails() may cause hit-enter prompt.

Solution: Set no_wait_return. (closes #11522)

Patch 9.0.0847

Problem: CI: not totally clear what MS-Windows version is used.

Solution: Show the Windows version. (Ken Takata, closes #11524)

Patch 9.0.0848

Problem: Help item for --log argument is not aligned nicely.

Solution: Add a Tab. (Ken Takata, closes #11521)

Patch 9.0.0849

Problem: Terminal mouse test is a bit flaky.

Solution: Add WaitFor() calls. (James McCoy closes #11519) Tune wait times to reduce flakiness.

Patch 9.0.0850

Problem: MS-Windows Terminal has unstable color control.

Solution: Do not try to read the old command prompt colortable, use modern VT sequences. (Christopher Plewright, closes #11450, closes #11373)

Patch 9.0.0851

Problem: Terminal mouse test is still flaky.

Solution: Also use WaitForAssert().

Patch 9.0.0852

Problem: Crypt test is skipped if xxd is not found.

Solution: Find xxd where it was supposed to be build.

Patch 9.0.0853

Problem: Terminal mouse test is still flaky on MacOS M1.

Solution: Also wait for the file to have some contents.

Patch 9.0.0854

Problem: No proper test for what 9.0.0846 fixes.

Solution: Run test in a terminal so that the hit-enter prompt can show up. (closes #11523)

Patch 9.0.0855

Problem: Comment not located above the code it refers to.

Solution: Move the comment. (closes #11527)

Patch 9.0.0856

Problem: MS-Windows: executable not found when running individual test.

Solution: Also look for vimd.exe. (Christopher Plewright, closes #11525)

Patch 9.0.0857

Problem: Selecting MSVC 2017 does not set \$PLATFORM.

Solution: Use \$VSCMD_ARG_TGT_ARCH. (Ken Takata, closes #11485)

Patch 9.0.0858

Problem: "!!sort" in a closed fold sorts too many lines.

Solution: Round to end of fold after adding the line count. (closes #11487)

Patch 9.0.0859

Problem: Compiler warning for unused variable.

Solution: Add #ifdef.

Patch 9.0.0860

Problem: MS-Windows: windres fails with clang 15.0.4.

Solution: Use llvm-windres. (John Marriott)

Patch 9.0.0861

Problem: Solution for "!!sort" in closed fold is not optimal.

Solution: Use a different range instead of the subtle difference in handling a range with an offset. (issue #11487)

Patch 9.0.0862

Problem: Default value of `'endoffile'` is wrong.

Solution: The default must be `'noendoffile'`.

Patch 9.0.0863

Problem: `col()` and `charcol()` only work for the current window.

Solution: Add an optional `winid` argument. (Yegappan Lakshmanan, closes #11466, closes #11461)

Patch 9.0.0864

Problem: Crash when using `"!!"` without a previous shell command.

Solution: Check `"prevcmd"` is not NULL. (closes #11487)

Patch 9.0.0865

Problem: Duplicate arguments are not always detected.

Solution: Expand to full path before comparing arguments. (Nir Lichtman, closes #11505, closes #9402)

Patch 9.0.0866

Problem: No test for what patch 8.2.2207 fixes.

Solution: Add a test case. (closes #11531)

Patch 9.0.0867

Problem: Wildmenu redrawing code is spread out.

Solution: Refactor to move code together. (closes #11528)

Patch 9.0.0868

Problem: MS-Windows: after Vim exits console resizing does not work properly.

Solution: Restore screen behavior checks for various WT and VTP combinations. (Christopher Plewright, closes #11526, closes #11507)

Patch 9.0.0869

Problem: Bogus error when string used after `:elseif`.

Solution: Do not consider a double quote the start of a comment. (closes #11534)

Patch 9.0.0870

Problem: Get E967 when using text property in quickfix window. (Sergey Vlasov)

Solution: Do not add an extra NUL and compute the text length correctly. (closes #11513)

Patch 9.0.0871

Problem: Using freed memory when clearing augroup at more prompt.

Solution: Delay clearing augroup until it's safe. (closes #11441)

Patch 9.0.0872

Problem: Code is indented more than needed.

Solution: Return early. (Yegappan Lakshmanan, closes #11538)

Patch 9.0.0873

Problem: Using freed memory when executing mapclear at the more prompt.
Solution: Do not clear mappings while listing them. (closes #11438)

Patch 9.0.0874

Problem: Using freed memory when executing unmenu at the more prompt.
Solution: Do not clear menus while listing them. (closes #11439)

Patch 9.0.0875

Problem: Using freed memory when executing delfunc at the more prompt.
Solution: Check function list not changed in another place. (closes #11437)

Patch 9.0.0876

Problem: Code is indented more than needed.
Solution: Split ExpandEscape() in two. (Yegappan Lakshmanan, closes #11539)

Patch 9.0.0877

Problem: Using freed memory with :comclear while listing commands.
Solution: Bail out when the command list has changed. (closes #11440)

Patch 9.0.0878

Problem: Coverity warns for dead code.
Solution: Remove the dead code.

Patch 9.0.0879

Problem: Unnecessary nesting in makefile.
Solution: Join "else" and "ifeq". (Ken Takata, closes #11547)

Patch 9.0.0880

Problem: Preprocessor indenting is off.
Solution: Adjust preprocessor indentation. (Ken Takata, closes #11546)

Patch 9.0.0881

Problem: Cannot get the currently showing mouse shape.
Solution: Add getmousetheshape().

Patch 9.0.0882

Problem: Using freed memory after SpellFileMissing autocmd uses bwipe.
Solution: Bail out if the window no longer exists.

Patch 9.0.0883

Problem: A silent mapping may cause dots on the command line.
Solution: Don't show dots for completion if they are not going to be removed again. (closes #11501)

Patch 9.0.0884

Problem: Mouse shape remains in op-pending mode after failed change.
Solution: Reset finish_op and restore it. (closes #11545)

Patch 9.0.0885

Problem: Informational message has an error message number.
Solution: Use a message without an error number. (closes #11530)

Patch 9.0.0886

Problem: Horizontal mouse scroll only works in the GUI.
Solution: Make horizontal mouse scroll also work in a terminal.
(Christopher Plewright, closes #11448)

Patch 9.0.0887

Problem: Cannot easily try out what codes various keys produce.
Solution: Add a script to gather key code information, with an initial list of codes to compare with.

Patch 9.0.0888

Problem: MS-Windows GUI: **CTRL-]** does not work on Swiss keyboard.
Solution: Check the key code and don't consider it as a dead key. (Aedin Louis Xavier, closes #11556)

Patch 9.0.0889

Problem: Keycode check script has a few flaws.
Solution: Sort on terminal name. Ignore XTGETTCAP responses. Check for version and status response. Update entries.

Patch 9.0.0890

Problem: No test for what patch 9.0.0827 fixes.
Solution: Add a test (still doesn't fail when fix is reverted).

Patch 9.0.0891

Problem: Virtual text below after match has wrong highlight.
Solution: Restore search_attr only after the virtual text.
(closes #11446)

Patch 9.0.0892

Problem: May redraw when not needed, causing slow scrolling.
Solution: Do not redraw when w_skipcol doesn't change. When w_skipcol changes only redraw from the top. (issue #11559)

Patch 9.0.0893

Problem: **'smoothscroll'** cursor calculations wrong when **'number'** is set.
Solution: Correct the code that computes the width. (closes #11492)

Patch 9.0.0894

Problem: Virtual text property highlight ignores window background.
Solution: Combine text prop attribute with win_attr into extra_attr.
(closes #11462)

Patch 9.0.0895

Problem: File renamed twice in test; missing feature check.
Solution: Remove a rename() call. Add check for cryptv feature.
(closes #11564)

Patch 9.0.0896

Problem: Test for home key fails when **'term'** is "tmux".
Solution: Only save termcap entries that exist. Adjust code for xHome to what xterm uses. (closes #11566)

Patch 9.0.0897

Problem: Clinical Quality Language files are not recognized.
Solution: Add the "*.cql" pattern. (Matthew Gramigna, closes #11452)

Patch 9.0.0898

Problem: With '**smoothscroll**' cursor is one screen line too far down. (Ernie Rael)

Solution: Add a test that currently has the wrong result so that a fix can be made. (issue #11436)

Patch 9.0.0899

Problem: The builtin terminals are in one long list.

Solution: Refactor into multiple lists and an index of the lists.

Patch 9.0.0900

Problem: Cursor moves too far with '**smoothscroll**'.

Solution: Only move as far as really needed. (Yee Cheng Chin, closes #11504)

Patch 9.0.0901

Problem: Setting w_leftcol and handling side effects is confusing.

Solution: Use a function to set w_leftcol() and handle side effects.

Patch 9.0.0902

Problem: Some mouse scroll code is not in a good place.

Solution: Refactor the code. (Christopher Plewright, closes #11561)

Patch 9.0.0903

Problem: Key code checker doesn't check modifyOtherKeys resource.

Solution: Request the modifyOtherKeys resource value. Drop resource DCS responses.

Patch 9.0.0904

Problem: Various comment and indent flaws.

Solution: Improve comments and indenting.

Patch 9.0.0905

Problem: Virtual text after the line wraps when '**wrap**' is off.

Solution: Only set text_prop_follows when wrapping. (closes #11463)

Patch 9.0.0906

Problem: Mouse scroll code is not optimal.

Solution: Properly organise Normal mode, Insert mode and common code. (Christopher Plewright, closes #11572)

Patch 9.0.0907

Problem: Restoring window after WinScrolled may fail.

Solution: Lock the window layout when triggering WinScrolled.

Patch 9.0.0908

Problem: With '**smoothscroll**' cursor may end up in wrong position.

Solution: Correct the computation of screen lines. (Yee Cheng Chin, closes #11502)

Patch 9.0.0909

Problem: Error message for layout change does not match action.

Solution: Pass the command to where the error is given. (closes #11573)

Patch 9.0.0910

Problem: Setting lines in another buffer may not work well.

Solution: Make sure the buffer being changed has a window. (issue #11558)

Patch 9.0.0911

Problem: With '**smoothscroll**' set mouse click position may be wrong.

Solution: Adjust computations for w_skipcol. (Yee Cheng Chin, closes #11514)

Patch 9.0.0912

Problem: libvterm with modifyOtherKeys level 2 does not match xterm.

Solution: Adjust key code escape sequences to be the same as what xterm sends in modifyOtherKeys level 2 mode. Check the value of no_reduce_keys before using it.

Patch 9.0.0913

Problem: Only a change in the current window triggers the WinScrolled event.

Solution: Trigger WinScrolled if any window scrolled or changed size. (issue #11576)

Patch 9.0.0914

Problem: deletebuflines() may move marks in the wrong window.

Solution: Find a window for the buffer being changed. (closes #11583)

Patch 9.0.0915

Problem: WinScrolled may trigger immediately when defined.

Solution: Initialize the fields in all windows. (closes #11582)

Patch 9.0.0916

Problem: getbuflines() is inefficient for getting a single line.

Solution: Add getbufoneline().

Patch 9.0.0917

Problem: The WinScrolled autocommand event is not enough.

Solution: Add WinResized and provide information about what changed. (closes #11576)

Patch 9.0.0918

Problem: MS-Windows: modifier keys do not work with mouse scroll events.

Solution: Use K_SPECIAL instead of CSI for the modifier keys. (Christopher Plewright, closes #11587)

Patch 9.0.0919

Problem: Build failure with tiny features.

Solution: Adjust #ifdef's.

Patch 9.0.0920

Problem: Cannot find an import prefixed with "s:". (Doug Kearns)

Solution: Skip over the "s:". (closes #11585)

Patch 9.0.0921

Problem: Missing defined(PROTO) in #ifdef.

Solution: Adjust #ifdef so that proto works with different features.
Clean up some preprocessor indenting.

Patch 9.0.0922

Problem: Mermaid files are not recognized.

Solution: Add patterns for Mermaid. (Crag MacEachern)

Patch 9.0.0923

Problem: Second SIGWINCH signal may be ignored.

Solution: When set_shellsize() is busy when called then run the inner code again when it's done. (issue #424)

Patch 9.0.0924

Problem: The first termcap entry of a builtin termcap is not used.

Solution: Remove increment that was previously skipping the KS_NAME entry.

Patch 9.0.0925

Problem: Two conditions are always false.

Solution: Remove the conditions. Update return value types to make clear what could be returned. (closes #11593)

Patch 9.0.0926

Problem: Coverity warns for not using return value of dict_add().

Solution: When dict_add() fails then don't call hash_remove().

Patch 9.0.0927

Problem: Coverity warns for using a NULL pointer.

Solution: Check for memory allocation failure.

Patch 9.0.0928

Problem: Using Ruby LDFLAGS may cause build problems.

Solution: Do not add Ruby LDFLAGS to Vim's LDFLAGS. (Zdenek Dohnal, closes #11592)

Patch 9.0.0929

Problem: Build failure with tiny version. (Tony Mechelynck)

Solution: Add #ifdef.

Patch 9.0.0930

Problem: Cannot debug the Kitty keyboard protocol with TermDebug.

Solution: Add Kitty keyboard protocol support to the libvterm fork. Recognize the escape sequences that the protocol generates. Add the **'keyprotocol'** option to allow the user to specify for which terminal what protocol is to be used, instead of hard-coding this. Add recognizing the kitty keyboard protocol status.

Patch 9.0.0931

Problem: MS-Windows: mouse column limited to 223.

Solution: Use two bytes for each mouse coordinate. Add the mouse position to scroll events. (Christopher Plewright, closes #11597)

Patch 9.0.0932

Problem: Oblivion files are not recognized.

Solution: Recognize Oblivion files and alike as "obse". (closes #11540)

Patch 9.0.0933

Problem: Kitty shows "already at oldest change" on startup.

Solution: When receiving the keyboard protocol state return the ignore key.
(closes #11601)

Patch 9.0.0934

Problem: Various code formatting issues.

Solution: Improve code formatting.

Patch 9.0.0935

Problem: When using dash it may not be recognize as filetype "sh".

Solution: Add checks for "dash". (Eisuke Kawashima, closes #11600)

Patch 9.0.0936

Problem: Wrong type for "isunnamed" returned by getreginfo().

Solution: Use VAR_BOOL instead of VAR_SPECIAL. (closes #11598)

Patch 9.0.0937

Problem: Forked repositories send out useless email.

Solution: When Coverity fails to run just ignore it. (Shane-XB-Qian,
closes #11604)

Patch 9.0.0938

Problem: MS-Windows: debug executable not found when running test.

Solution: Look for vimd.exe. (Christopher Plewright, closes #11602)

Patch 9.0.0939

Problem: Still using simplified mappings when using the kitty keyboard protocol.

Solution: Use the kitty_protocol_state value to decide whether to use simplified mappings. Improve how seenModifyOtherKeys is set and reset.

Patch 9.0.0940

Problem: Crash when typing a letter in a terminal window. (Shane-XB-Qian)

Solution: Use the "vterm" variable instead of getting the terminal pointer from the current buffer. (closes #11608)

Patch 9.0.0941

Problem: CI failures in sound dummy.

Solution: Temporarily disable building sound dummy. (closes #11610)

Patch 9.0.0942

Problem: Workflow Description Language files are not recognized.

Solution: Add a pattern for the "wdl" filetype. (Matt Dunford,
closes #11611)

Patch 9.0.0943

Problem: Pretending to go out of Insert mode when Esc is received has side effects.

Solution: When the kitty keyboard protocol is enabled expect Esc to always be the start of an escape sequence.

Patch 9.0.0944

Problem: **'cursorline'** causes virtual text highlight to continue.
Solution: Save and restore line_attr. (closes #11588)

Patch 9.0.0945

Problem: Failures in the cursorline test.
Solution: Reset extra_attr only after a text property.

Patch 9.0.0946

Problem: CI: Error in Coverity flow is not reported.
Solution: Use another way to avoid errors in a forked repository. (Ken Takata, closes #11609)

Patch 9.0.0947

Problem: Invalid memory access in substitute with function that goes to another file.
Solution: Check for text locked in **CTRL-W** gf.

Patch 9.0.0948

Problem: **'ttyfast'** is set for arbitrary terminals.
Solution: Always set **'ttyfast'**. (closes #11549)

Patch 9.0.0949

Problem: Crash when unletting a variable while listing variables.
Solution: Disallow changing a hashtable while going over the entries. (closes #11435)

Patch 9.0.0950

Problem: The pattern "_s\zs" matches at EOL.
Solution: Make the pattern "_s\zs" match at the start of the next line. (closes #11617)

Patch 9.0.0951

Problem: Trying every character position for a match is inefficient.
Solution: Use the start position of the match ignoring "\zs".

Patch 9.0.0952

Problem: Eclipse preference files are not recognized.
Solution: Add a pattern to use "jproperties" for Eclipse preference files. (closes #11618)

Patch 9.0.0953

Problem: Part of making search more efficient is missing.
Solution: Add the change in searchit().

Patch 9.0.0954

Problem: Cannot detect whether modifyOtherKeys is enabled.
Solution: Use XTQMODKEYS introduced by xterm version 377 to request the modifyOtherKeys level. Update the keycode check results.

Patch 9.0.0955

Problem: Libvterm does not support the XTQMODKEYS request.
Solution: Implement the XTQMODKEYS request and response. Update the keycode check results.

Patch 9.0.0956

Problem: Terminal tests fail when using key with modifier.

Solution: Use the modifyOtherKeys encoding when using RunVimInTerminal().

Patch 9.0.0957

Problem: Tests fail without the terminal feature.

Solution: Move functions to another utility script.

Patch 9.0.0958

Problem: Messages test is flaky.

Solution: Add a short delay.

Patch 9.0.0959

Problem: Error when using the "File Settings / Text Width" menu.

Solution: Use str2nr(). (closes #11624)

Patch 9.0.0960

Problem: Error when using the "Tools / Spelling / Find More Languages" menu.

Solution: Remove "<SID>". Reset "g:menutrans_set_lang_to" when 'encoding' changes. (closes #11625)

Patch 9.0.0961

Problem: Using deletebufline() may jump to another window.

Solution: Do not use a window where the buffer was only in the past. (closes #11594)

Patch 9.0.0962

Problem: Virtual text below cannot be placed below empty lines.

Solution: Add one character. (James Alvarado, closes #11606, closes #11520)

Patch 9.0.0963

Problem: Function name does not match autocmd event name.

Solution: Rename "optionsset" to "optionset". (closes #11630)

Patch 9.0.0964

Problem: Status line of other window not redrawn when dragging it when 'splitkeep' is set to "screen".

Solution: Set w_redr_status earlier. (Luuk van Baal, closes #11635, closes #11632)

Patch 9.0.0965

Problem: Using one window for executing autocommands is insufficient.

Solution: Use up to five windows for executing autocommands.

Patch 9.0.0966

Problem: Some compilers don't allow a declaration after a label.

Solution: Move the declaration to the start of the block. (John Marriott)

Patch 9.0.0967

Problem: Leaking memory from autocmd windows.

Solution: Free window when auc_win is not NULL.

Patch 9.0.0968

Problem: GUI mouse event test is a bit flaky.

Solution: Mark the test case as flaky. Move test function failure checks to a separate test function.

Patch 9.0.0969

Problem: Matchparen highlight is not updated when switching buffers.

Solution: Listen to the BufLeave and the BufWinEnter autocmd events.
(closes #11626)

Patch 9.0.0970

Problem: Coverity warns for uninitialized variable.

Solution: Initialize "ren_ret".

Patch 9.0.0971

Problem: Escape sequences not recognized without the termresponse feature.

Solution: Recognize escape sequences to avoid display mess up.

Patch 9.0.0972

Problem: Build failure on some systems.

Solution: Adjust #ifdefs related to the termresponse feature.

Patch 9.0.0973

Problem: Kitty keyboard protocol key not decoded when it has an unsupported modifier, such as NumLock.

Solution: Accept a key with any modifier. (closes #11638)

Patch 9.0.0974

Problem: Even when Esc is encoded a timeout is used.

Solution: Use K_ESC when an encoded Esc is found.

Patch 9.0.0975

Problem: Virtual text below an empty line is misplaced when 'number' is set.

Solution: Adjust the computations. (closes #11629)

Patch 9.0.0976

Problem: Enabling the kitty keyboard protocol uses push/pop.

Solution: Use the start/stop codes to avoid unpredictable behavior.

Patch 9.0.0977

Problem: It is not easy to see what client-server commands are doing.

Solution: Add channel log messages if ch_log() is available. Move the channel logging and make it available with the +eval feature.

Patch 9.0.0978

Problem: Build errors without the +channel feature. (John Marriott)

Solution: Adjust #ifdefs.

Patch 9.0.0979

Problem: ch_log() text can be hard to find in the log file.

Solution: Prepend "ch_log()" to the text.

Patch 9.0.0980

Problem: The keyboard state response may end up in a shell command.
Solution: Only request the keyboard protocol state when the typeahead is empty, no more commands are following and not exiting. Add the t_RK termcap entry for this.

Patch 9.0.0981

Problem: Build error in tiny version.
Solution: Add #ifdef.

Patch 9.0.0982

Problem: **'cursorline'** not drawn before virtual text below.
Solution: Add the **'cursorline'** attribute to the empty space. (closes #11647)

Patch 9.0.0983

Problem: Stray characters displayed when starting the GUI.
Solution: Add t_RK to the list of terminal options.

Patch 9.0.0984

Problem: GUI: remote_foreground() does not always work. (Ron Aaron)
Solution: For GTK use gtk_window_set_keep_above(). (issue #11641)

Patch 9.0.0985

Problem: When using kitty keyboard protocol function keys may not work. (Kovid Goyal)
Solution: Recognize CSI ending in **[ABCDEFHPQRS]** also when the termcap entries are not specified. (closes #11648)

Patch 9.0.0986

Problem: Build failure with tiny version.
Solution: Add #ifdef.

Patch 9.0.0987

Problem: File missing from list of distributed files.
Solution: Add logfile.pro to list of distributed files.

Patch 9.0.0988

Problem: Using feedkeys() does not show up in a channel log.
Solution: Add ch_log() calls and clean up the code.

Patch 9.0.0989

Problem: Popwin test is more flaky on MacOS.
Solution: Use a longer wait time.

Patch 9.0.0990

Problem: Callback name argument is changed by setqflist().
Solution: Use the expanded function name for the callback, do not store it in the argument. (closes #11653)

Patch 9.0.0991

Problem: Crash when reading help index with various options set. (Marius Gedminas)
Solution: Do not set wlv.c_extra to NUL when wlv.p_extra is NULL. (closes #11651)

Patch 9.0.0992

Problem: Vim9 script: get E1096 when comment follows return.

Solution: Adjust condition for return without expression. (closes #11654)

Patch 9.0.0993

Problem: Display errors when adding or removing text property type.

Solution: Perform a full redraw. Only use text properties for which the type is defined. (closes #11655)

Patch 9.0.0994

Problem: Tests for empty prop type name fail.

Solution: Correct the error number.

Patch 9.0.0995

Problem: Padding before virtual text below is highlighted when **'number'** and **'nowrap'** are set.

Solution: Save and restore n_attr_skip. (closes #11643)

Patch 9.0.0996

Problem: If **'keyprotocol'** is empty "xterm" still uses modifyOtherKeys.

Solution: Remove t_TI, t_RK and t_TE from the "xterm" builtin termcap and let the default value of **'keyprotocol'** add those.

Patch 9.0.0997

Problem: Coverity warns for dead code.

Solution: Don't use ASCII_ISUPPER() for a negative value.

Patch 9.0.0998

Problem: "gk" may reset skipcol when not needed.

Solution: Only reset skipcol if the cursor column is less.

Patch 9.0.0999

Problem: Memory may leak.

Solution: Free the sound callback function name if it was allocated.

Patch 9.0.1000

Problem: With **'smoothscroll'** skipcol may be reset unnecessarily.

Solution: Check the line does actually fit in the window.

Patch 9.0.1001

Problem: Classes are not documented or implemented yet.

Solution: Make the first steps at documenting Vim9 objects, classes and interfaces. Make initial choices for the syntax. Add a skeleton implementation. Add "public" and "this" in the command table.

Patch 9.0.1002

Problem: Command list test fails.

Solution: Add commands added to the list.

Patch 9.0.1003

Problem: Tiny build fails.

Solution: Remove #ifdef from error message.

Patch 9.0.1004

Problem: Suspend test sometimes fails on MacOS.
Solution: Wait a short while for terminal responses.

Patch 9.0.1005

Problem: A failed test may leave a swap file behind.
Solution: Delete the swap file to avoid another test to fail. Use another file name.

Patch 9.0.1006

Problem: Suspend test still sometimes fails on MacOS.
Solution: Wait a little while for terminal responses.

Patch 9.0.1007

Problem: There is no way to get a list of swap file names.
Solution: Add the swapfilelist() function. Use it in the test script to clean up. Remove deleting individual swap files.

Patch 9.0.1008

Problem: Test for swapfilelist() fails on MS-Windows.
Solution: Only check the tail of the path. Mark a test as flaky.

Patch 9.0.1009

Problem: Test for catch after interrupt is flaky on MS-Windows.
Solution: Mark the test as flaky.

Patch 9.0.1010

Problem: Stray warnings for existing swap files.
Solution: Wipe out the buffer until it has no name and no swap file.

Patch 9.0.1011

Problem: ml_get error when using screenpos().
Solution: Give an error for the line number. (closes #11661)

Patch 9.0.1012

Problem: Tests may get stuck in buffer with swap file.
Solution: Bail out when bwipe! doesn't get another buffer.

Patch 9.0.1013

Problem: Suspend test often fails on Mac OS.
Solution: Make t_RP empty.

Patch 9.0.1014

Problem: Zir files are not recognized.
Solution: Add a pattern for Zir files. (closes #11664)

Patch 9.0.1015

Problem: Without /dev/urandom srand() seed is too predictable.
Solution: Use micro seconds and XOR with process ID. (Yasuhiro Matsumoto, closes #11656)

Patch 9.0.1016

Problem: screenpos() does not count filler lines for diff mode.
Solution: Add filler lines. (closes 11658)

Patch 9.0.1017

Problem: Test for srand() fails on MS-Windows.

Solution: Do not expect the same result a second time.

Patch 9.0.1018

Problem: Suspend test still fails on Mac OS.

Solution: Make **'keyprotocol'** empty.

Patch 9.0.1019

Problem: **'smoothscroll'** and virtual text above don't work together.
(Yee Cheng Chin)

Solution: Skip virtual text above when w_skipcol is non-zero.
(closes #11665)

Patch 9.0.1020

Problem: Tests call GetSwapFileList() before it is defined.

Solution: Move the call to after defining the function. (Christopher Plewright)

Patch 9.0.1021

Problem: Test trips over g:name.

Solution: Delete g:name after using it.

Patch 9.0.1022

Problem: Suspend test fails on Mac OS when suspending Vim.

Solution: Make **'keyprotocol'** empty.

Patch 9.0.1023

Problem: MS-Windows: dynamic loading of libsodium doesn't work.

Solution: Add "randombytes_random". (Ken Takata, closes #11667)

Patch 9.0.1024

Problem: CI doesn't use the latest FreeBSD version.

Solution: Go from 12.3 to 12.4. (closes #11423)

Patch 9.0.1025

Problem: WinScrolled is not triggered when filler lines change.

Solution: Add "topfill" to the values that WinScrolled triggers on.
(closes #11668)

Patch 9.0.1026

Problem: type of w_last_topfill is wrong.

Solution: Use "int" instead of "linenr_T". (closes #11670)

Patch 9.0.1027

Problem: LGTM is soon shutting down.

Solution: Remove LGTM from CI. (closes #11671)

Patch 9.0.1028

Problem: Mouse shape test is flaky, especially on Mac OS.

Solution: Instead of starting all timers at the same time, start the next one in the callback of the previous one. (Yee Cheng Chin, closes #11673) Also use "bwipe!" instead of "close!" to avoid swap files remaining.

Patch 9.0.1029

Problem: Autoload directory missing from distribution.

Solution: Add the autoload/zig directory to the list of distributed files.

Patch 9.0.1030

Problem: Using freed memory with the cmdline popup menu.

Solution: Clear the popup menu when clearing the matches. (closes #11677)

Patch 9.0.1031

Problem: Vim9 class is not implemented yet.

Solution: Add very basic class support.

Patch 9.0.1032

Problem: Test fails when terminal feature is missing.

Solution: Use CheckRunVimInTerminal.

Patch 9.0.1033

Problem: Tiny build fails because of conflicting typedef.

Solution: Remove one typedef.

Patch 9.0.1034

Problem: Reporting swap file when windows are split.

Solution: Close extra windows after running a test.

Patch 9.0.1035

Problem: Object members are not being marked as used, garbage collection may free them.

Solution: Mark object members as used. Fix reference counting.

Patch 9.0.1036

Problem: Undo misbehaves when writing from an insert mode mapping.

Solution: Sync undo when writing. (closes #11674)

Patch 9.0.1037

Problem: lalloc(0) error for a class without members.

Solution: Don't allocate room for members if there aren't any.
Don't create the class if there was an error.

Patch 9.0.1038

Problem: Function name does not match what it is used for.

Solution: Include the modifier in the name. (closes #11679)

Patch 9.0.1039

Problem: Using a <Cmd> mapping CmdlineChanged may be triggered twice.

Solution: Count the number of times CmdlineChanged is triggered and avoid doing it twice. (closes #116820)

Patch 9.0.1040

Problem: Test for <Cmd> mapping with CmdlineChanged fails.

Solution: Put back the check for the cmdline length not changing.

Patch 9.0.1041

Problem: Cannot define a method in a class.

Solution: Implement defining an object method. Make calling an object method work.

Patch 9.0.1042

Problem: ASAN gives false alarm about array access.

Solution: Use an intermediate pointer.

Patch 9.0.1043

Problem: Macro has confusing name and is duplicated.

Solution: Use one macro with an understandable name. (closes #11686)

Patch 9.0.1044

Problem: Setting window height using Python may cause errors.

Solution: When setting "curwin" also set "curbuf". (closes #11687)

Patch 9.0.1045

Problem: In a class object members cannot be initialized.

Solution: Support initializing object members. Make "disassemble" work on an object method.

Patch 9.0.1046

Problem: Class method disassemble test fails on MS-Windows.

Solution: Do not match with a specific size.

Patch 9.0.1047

Problem: Matchparen is slow.

Solution: Actually use the position where the match started, not the position where the search started. (closes #11644)

Patch 9.0.1048

Problem: With "screenline" in **'culopt'** cursorline highlight is wrong.

Solution: Apply the priority logic also when "screenline" is in **'culopt'**. (closes #11696)

Patch 9.0.1049

Problem: Crash when opening a very small terminal window.

Solution: Instead of crashing fix the cursor position. (closes #11697)

Patch 9.0.1050

Problem: Using freed memory when assigning to variable twice.

Solution: Make copy of the list type. (closes #11691)

Patch 9.0.1051

Problem: After a failed **CTRL-W**] next command splits window.

Solution: Reset postponed_split. (Rob Pilling, closes #11698)

Patch 9.0.1052

Problem: Using freed memory on exit when EXITFREE is defined.

Solution: Make a deep copy of the type. Make sure TTFLAG_STATIC is not set in the copy.

Patch 9.0.1053

Problem: Default constructor arguments are not optional.

Solution: Use "= v:none" to make constructor arguments optional.

Patch 9.0.1054

Problem: Object member can't get type from initializer.

Solution: If there is no type specified try to use the type of the initializer. Check for a valid type.

Patch 9.0.1055

Problem: Coverity warns for using uninitialized memory.

Solution: Clear the "lhs" field earlier.

Patch 9.0.1056

Problem: Leaking memory when disassembling an object method.

Solution: Free the typval of the class.

Patch 9.0.1057

Problem: Conflict between supercollider and scala filetype detection.

Solution: Do not check for "Class : Method", it can appear in both filetypes. (Chris Kipp, closes #11699)

Patch 9.0.1058

Problem: String value of class and object do not have useful information.

Solution: Add the class name and for the object the member values.

Patch 9.0.1059

Problem: Build failure with some compilers that can't handle a declaration directly after a "case" statement.

Solution: Add a block to put the declarations in.

Patch 9.0.1060

problem: Private and public object members are not implemented yet.

Solution: Implement private and public object members.

Patch 9.0.1061

Problem: Cannot display '**showcmd**' somewhere else.

Solution: Add the '**showcmdloc**' option. (Luuk van Baal, closes #11684)

Patch 9.0.1062

Problem: Some test function names do not match what they are doing.

Solution: Leave out user data for the test that is called "NoUserData". (closes #11703)

Patch 9.0.1063

Problem: When using Kitty a shell command may mess up the key protocol state.

Solution: Output t_te before t_TE. If t_te switches between the main and the alternate screen then deactivating the key protocol by t_TE should happen after switching screen. (issue #11705)

Patch 9.0.1064

Problem: Code for making '**shortmess**' temporarily empty is repeated.

Solution: Add functions for making '**shortmess**' empty and restoring it. (Christian Brabandt, closes #11709)

Patch 9.0.1065

Problem: A shell command switching screens may still have a problem with the kitty keyboard protocol.
Solution: Disable the kitty keyboard protocol both in the current and the alternate screen, if there are indications it might be needed. (issue #11705) Also fix naming.

Patch 9.0.1066

Problem: Test function name is wrong.
Solution: Rename to what is actually being tested. (closes #11712)

Patch 9.0.1067

Problem: In diff mode virtual text is highlighted incorrectly. (Rick Howe)
Solution: Do not use diff attributes for virtual text. (closes #11714)

Patch 9.0.1068

Problem: No information about whether requesting term codes has an effect.
Solution: Add ch_log() calls to report the effect of term code responses. Avoid deleting an entry and then adding back the same one.

Patch 9.0.1069

Problem: Diff mode highlight fails for special characters.
Solution: Adjust condition for setting "diff_hlf".

Patch 9.0.1070

Problem: Reading beyond array size.
Solution: Only use name[0] and name[1], do not use "name" as a string.

Patch 9.0.1071

Problem: Codecov action version is too specific.
Solution: Only use "v3" to automatically use the latest stable version. (closes #11720)

Patch 9.0.1072

Problem: screenpos() column result in fold may be too small.
Solution: Add space of '**number**', sign column, etc. (closes #11715)

Patch 9.0.1073

Problem: Using "xterm-kitty" for '**term**' causes problems.
Solution: Remove the "xterm-" part when '**term**' is set from \$TERM. Detect a few kitty-specific properties based on the version response instead of the terminal name.

Patch 9.0.1074

Problem: Class members are not supported yet.
Solution: Add initial support for class members.

Patch 9.0.1075

Problem: build fails if the compiler doesn't allow for a declaration right after "case".
Solution: Add a block.

Patch 9.0.1076

Problem: ASAN complains about NULL argument.
Solution: Skip memmove() when there is nothing to move.

Patch 9.0.1077

Problem: Can add text property with negative ID before virtual text property.

Solution: Remember that a text property with a negative ID was used and give an appropriate error message. (closes #11725)
Fix index computation.

Patch 9.0.1078

Problem: With the +vartabs feature indent folding may use wrong **'tabstop'**.

Solution: Use the "buf" argument instead of "curbuf".

Patch 9.0.1079

Problem: Leaking memory when defining a user command fails.

Solution: Free "compl_arg" when needed. (closes #11726)

Patch 9.0.1080

Problem: The "kitty" terminfo entry is not widespread, resulting in the kitty terminal not working properly.

Solution: Go back to using "xterm-kitty" and avoid the problems it causes in another way.

Patch 9.0.1081

Problem: Using "->" with split lines does not always work.

Solution: Avoid trying to get another line. (closes #11723)

Patch 9.0.1082

Problem: Some jsonc files are not recognized.

Solution: Add patterns for jsonc and move some from json to jsonc. (closes #11711)

Patch 9.0.1083

Problem: Empty and comment lines in a class cause an error.

Solution: Skip empty and comment lines. (closes #11734)

Patch 9.0.1084

Problem: Code handling low level MS-Windows events cannot be tested.

Solution: Add test_mswin_event() and tests using it. (Christopher Plewright, closes #11622)

Patch 9.0.1085

Problem: Compiler warns for uninitialized variable.

Solution: Initialize the variable. Remove unused function. (John Marriott)

Patch 9.0.1086

Problem: Display wrong in Windows terminal after exiting Vim.

Solution: Apply screen restore fix for Windows 11 also to Windows 10 builds. (Christopher Plewright, closes #11713, closes #11706)

Patch 9.0.1087

Problem: Autocommand test sometimes fails.

Solution: Add a short delay. (James McCoy, closes #11737)

Patch 9.0.1088

Problem: Clang warns for unused variable.
Solution: Adjust #ifdef. (John Marriott)

Patch 9.0.1089

Problem: unnecessary assignment
Solution: Remove the assignment. (Luuk van Baal, closes #1136)

Patch 9.0.1090

Problem: FHIR Shorthand files are not recognized.
Solution: Add a pattern to detect FSH files. (Matthew Gramigna, closes #11738)

Patch 9.0.1091

Problem: Assignment to non-existing member causes a crash. (Yegappan Lakshmanan)
Solution: Give an error message and bail out when a member cannot be found.

Patch 9.0.1092

Problem: Search error message doesn't show used pattern.
Solution: Pass the actually used pattern to where the error message is given. (Rob Pilling, closes #11742)

Patch 9.0.1093

Problem: Using freed memory of object member. (Yegappan Lakshmanan)
Solution: Make a copy of the object member when getting it.

Patch 9.0.1094

Problem: Compiler warning when HAS_MESSAGE_WINDOW is not defined.
Solution: Add UNUSED.

Patch 9.0.1095

Problem: Using freed memory when declaration fails. (Yegappan Lakshmanan)
Solution: After unreferencing an object set the reference to NULL.

Patch 9.0.1096

Problem: Reallocating hashtable when the size didn't change.
Solution: Bail out when the hashtable is already the desired size.

Patch 9.0.1097

Problem: Tests are failing.
Solution: Do clean up a hashtable when at the initial size.

Patch 9.0.1098

Problem: Code uses too much indent.
Solution: Use an early return. (Yegappan Lakshmanan, closes #11747)

Patch 9.0.1099

Problem: Trying to resize a hashtable may cause a problem.
Solution: Do not try to resize a hashtable before adding an item.

Patch 9.0.1100

Problem: A hashtable with many removed items is not cleaned up.
Solution: Re-hash a hashtable even when the size didn't change if too many items were removed.

Patch 9.0.1101

Problem: Unused global variable.

Solution: Remove the variable. (closes #11752)

Patch 9.0.1102

Problem: Complicated use of #ifdef.

Solution: Simplify #ifdef use. (Ken Takata, closes #11745)

Patch 9.0.1103

Problem: jq files are not recognized.

Solution: Add detection of Jq files. (David McDonald, closes #11743)

Patch 9.0.1104

Problem: Invalid memory access when checking function argument types.

Solution: Do not check beyond the number of arguments. (closes #11755)

Patch 9.0.1105

Problem: Code is indented too much.

Solution: Use an early return. (Yegappan Lakshmanan, closes #11756)

Patch 9.0.1106

Problem: Not all postfix files are recognized.

Solution: Recognize main.cf.proto files. (closes #11732)

Patch 9.0.1107

Problem: Float constant not recognized as float.

Solution: Check the vartype instead of comparing with t_float.
(closes #11754)

Patch 9.0.1108

Problem: Type error when using "any" type and adding a number to a float.

Solution: Accept both a number and a float. (closes #11753)

Patch 9.0.1109

Problem: Leaking allocated type.

Solution: Reset the "static" flag in the allocated type copy.

Patch 9.0.1110

Problem: Build fails on Mac OS X 10.4/10.5 .

Solution: Check if the dispatch/dispatch.h header exists. (Evan Miller,
closes #11746)

Patch 9.0.1111

Problem: Termcap entries for RGB colors are not set automatically.

Solution: Always set the termcap entries when +termguicolors is enabled.

Patch 9.0.1112

Problem: test_mswin_event() can hang.

Solution: Add the "execute" argument to process events right away.
(Christopher Plewright, closes #11760)

Patch 9.0.1113

Problem: Users cannot easily try out a PR.

Solution: Add an "artifacts" section to the AppVeyor CI config. (Christian Brabandt, closes #11762)

Patch 9.0.1114

Problem: CI does not use the latest Python version.

Solution: Switch from Python 3.10 to 3.11. (closes #11761)

Patch 9.0.1115

Problem: Code is indented more than needed.

Solution: Use an early return to reduce indenting. (Yegappan Lakshmanan, closes #11758)

Patch 9.0.1116

Problem: Compiler may complain about an unused function.

Solution: Add #ifdef. (John Marriott)

Patch 9.0.1117

Problem: Terminfo entries for bracketed paste are not used.

Solution: Use the newly added terminfo entries for bracketed paste. Correct mixup of output strings and key codes.

Patch 9.0.1118

Problem: Sporadic test failures when using a terminal window.

Solution: Adjust waiting times. (James McCoy, closes #11763)

Patch 9.0.1119

Problem: Type of arguments not checked when calling a partial.

Solution: Give an error for a wrong argument type. (closes #11753)

Patch 9.0.1120

Problem: Tex filetype detection not sufficiently tested.

Solution: Add more test cases for "tex" detection. (Jonas Strittmatter, closes #11765)

Patch 9.0.1121

Problem: Cursor positioning and display problems with **'smoothscroll'** and using "zt", "zb" or "zz".

Solution: Adjust computations and conditions. (Yee Cheng Chin, closes #11764)

Patch 9.0.1122

Problem: Class member access is not fully tested yet.

Solution: Add more tests.

Patch 9.0.1123

Problem: Class function not implemented yet.

Solution: Implement defining and calling a class function.

Patch 9.0.1124

Problem: Virtual text at a column position is truncated at the window edge. (Yegappan Lakshmanan)

Solution: Do not truncated virtual text that is placed at a column.

Patch 9.0.1125

Problem: Memory leak when using class functions.
Solution: Clear and free the array with class functions.

Patch 9.0.1126

Problem: Bracketed paste can be enabled when pasted text is not recognized.
Solution: Output t_BE only when t_PS and t_PE are set.

Patch 9.0.1127

Problem: No error if function argument shadows class member.
Solution: Give an error for shadowing a class member.

Patch 9.0.1128

Problem: Build failure.
Solution: Add type cast. Add missing error messages.

Patch 9.0.1129

Problem: Sporadic Test_range() failure.
Solution: Clear typeahead. Move to a separate function. (issue #22771)

Patch 9.0.1130

Problem: Unexpected output when autoloading a script for an interactive operation.
Solution: Reset "KeyTyped" while loading a script and when handling a nested function. (closes #11773)

Patch 9.0.1131

Problem: Build failure without the +eval feature.
Solution: Move code inside #ifdef.

Patch 9.0.1132

Problem: Code is indented more than needed.
Solution: Use an early return to reduce indentation. (Yegappan Lakshmanan, closes #11769)

Patch 9.0.1133

Problem: Error message names do not match the items.
Solution: Add "_str" when the text contains "%s".

Patch 9.0.1134

Problem: Comparing objects uses identity instead of equality.
Solution: Compare the object values.

Patch 9.0.1135

Problem: Missing function argument.
Solution: Add ignore case flag.

Patch 9.0.1136

Problem: Memory leak when getting class member type from expr.
Solution: Clear the expression result.

Patch 9.0.1137

Problem: Some conditions are always false.
Solution: Remove the useless conditions. (closes #11776)

Patch 9.0.1138

Problem: Crash when expecting varargs but it is something else.

Solution: Only use the member when the type is a list. (closes #11774)

Patch 9.0.1139

Problem: Cannot create a new object in a compiled function.

Solution: Compile the instructions to create a new object.

Patch 9.0.1140

Problem: Cannot call an object method in a compiled function.

Solution: Compile the instructions to invoke an object method.

Patch 9.0.1141

Problem: **'cursorcolumn'** and **'colorcolumn'** wrong after concealing and wrapping line.

Solution: Reset "wlv.vcol_off" after each screen line. (Alexey Radkov, closes #11777)

Patch 9.0.1142

Problem: Crash and/or memory leak when redefining function after error.

Solution: Clear pointer after making a copy. Clear arrays on failure. (closes #11774)

Patch 9.0.1143

Problem: Invalid memory access with bad **'statusline'** value.

Solution: Avoid going over the NUL at the end.

Patch 9.0.1144

Problem: Reading beyond text.

Solution: Add strlen_maxlen() and use it.

Patch 9.0.1145

Problem: Invalid memory access with recursive substitute expression.

Solution: Check the return value of vim_regsub().

Patch 9.0.1146

Problem: MS-Windows: various special keys and modifiers are not mappable.

Solution: Adjust the handling of keys with modifiers. (Christian Plewright, closes #11768)

Patch 9.0.1147

Problem: Cannot access a class member in a compiled function.

Solution: Implement looking up a class member.

Patch 9.0.1148

Problem: Cmdline test fails in the GUI.

Solution: Skip the test when running in the GUI.

Patch 9.0.1149

Problem: Class members may be garbage collected.

Solution: Mark class members as being in use.

Patch 9.0.1150

Problem: :interface is not implemented yet.

Solution: Implement the basics of :interface.

Patch 9.0.1151

Problem: Build failure.

Solution: Add missing part of :interface change.

Patch 9.0.1152

Problem: Class "implements" argument not implemented.

Solution: Implement "implements" argument. Add basic checks for when a class implements an interface.

Patch 9.0.1153

Problem: Build error with some compilers.

Solution: Clear pointer the right way.

Patch 9.0.1154

Problem: Coverity warns for dead code.

Solution: Remove condition that is always true.

Patch 9.0.1155

Problem: Cannot use a class as a type.

Solution: Accept a class and interface name as a type.

Patch 9.0.1156

Problem: Tests fail because of a different error message.

Solution: Don't give an error if a type name can't be found.

Patch 9.0.1157

Problem: "implements" only handles one interface name.

Solution: Handle a comma separated list of names. Check for duplicate names.

Patch 9.0.1158

Problem: Code is indented more than necessary.

Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11787)

Patch 9.0.1159

Problem: Extends argument for class not implemented yet.

Solution: Basic implementation of "extends".

Patch 9.0.1160

Problem: ASAN error for ufunc_T allocated with wrong size.

Solution: Make sure the size can always fit the struct.

Patch 9.0.1161

Problem: Coverity warns for using strcpy().

Solution: Call a function to set the function name.

Patch 9.0.1162

Problem: Configure does not handle all FORTIFY_SOURCE variants.

Solution: Also handle Fedora's default FORTIFY_SOURCE flags. (Zdenek Dohnal, closes #11794)

Patch 9.0.1163

Problem: Compiler warning for implicit size_t/int conversion.
Solution: Add a type cast. (Mike Williams, closes #11795)

Patch 9.0.1164

Problem: Evaluating string expression advances function line.
Solution: Disable function lines while parsing a string expression.
(Hirohito Higashi, closes #11796)

Patch 9.0.1165

Problem: Tests using IPv6 sometimes fail.
Solution: Use getaddrinfo() and use try/catch. (James McCoy,
closes #11783)

Patch 9.0.1166

Problem: Code is indented more than necessary.
Solution: Use an early return where it makes sense. (Yegappan Lakshmanan,
closes #11792)

Patch 9.0.1167

Problem: EditorConfig files do not have their own filetype.
Solution: Add the "editorconfig" filetype. (Gregory Anders, closes #11779)

Patch 9.0.1168

Problem: Code to enable/disable mouse is not from terminfo/termcap.
Solution: Request the "XM" entry and use it to set **'ttymouse'** if possible.

Patch 9.0.1169

Problem: Some key+modifier tests fail on some AppVeyor images.
Solution: Adjust the tests for key movements and fix the revealed bugs.
(Christopher Plewright, closes #11798)

Patch 9.0.1170

Problem: LGTM badge no longer works.
Solution: Remove the LGTM badge. (closes #11799)

Patch 9.0.1171

Problem: Screen is not redrawn after using setcellwidths().
Solution: Redraw the screen when the cell widths have changed. (Yasuhiro
Matsumoto, closes #11800)

Patch 9.0.1172

Problem: When **'selection'** is "exclusive" then "1v" is one char short.
Solution: Add one character when **'selection'** is "exclusive". (closes #11791)

Patch 9.0.1173

Problem: Compiler warning for unused variable on non-Unix systems.
Solution: Move #ifdef. (John Marriott)

Patch 9.0.1174

Problem: Smali files are not recognized.
Solution: Add a pattern for Smali files. (Amaan Qureshi, closes #11801)

Patch 9.0.1175

Problem: The set_ref_in_item() function is too long.
Solution: Use a separate function for more complicated types. (Yegappan Lakshmanan, closes #11802)

Patch 9.0.1176

Problem: smithy files are not recognized.
Solution: Add a pattern for Smithy files. (Chris Kipp, closes #11804)

Patch 9.0.1177

Problem: AppVeyor uses some older tools.
Solution: Switch to Visual Studio 2022 and Python 3.11. (Christopher Plewright, closes #11793)

Patch 9.0.1178

Problem: A child class cannot override functions from a base class.
Solution: Allow overriding and implement "super".

Patch 9.0.1179

Problem: Not all errors around inheritance are tested.
Solution: Add more tests. Fix uncovered problems.

Patch 9.0.1180

Problem: Compiler warnings without the +job feature.
Solution: Adjust #ifdefs. (John Marriott)

Patch 9.0.1181

Problem: Class inheritance and typing insufficiently tested.
Solution: Add more tests. Implement missing behavior.

Patch 9.0.1182

Problem: go checksum files are not recognized.
Solution: Add the name of go checksum files. (Amaan Qureshi, closes #11803)

Patch 9.0.1183

Problem: Code is indented more than necessary.
Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11805)

Patch 9.0.1184

Problem: Interface of an object is not recognized when checking type.
Solution: Use the interface implemented by an object.

Patch 9.0.1185

Problem: Using class from imported script not tested.
Solution: Add tests. Implement what is missing.

Patch 9.0.1186

Problem: Imported class does not work when used twice in a line.
Solution: Fix the type parsing.

Patch 9.0.1187

Problem: Test for using imported class fails.
Solution: Skip over rest of type.

Patch 9.0.1188

Problem: Return value of type() for class and object unclear.
Solution: Add v:t_object and v:t_class.

Patch 9.0.1189

Problem: Invalid memory access with folding and using "L".
Solution: Prevent the cursor from moving to line zero.

Patch 9.0.1190

Problem: AppVeyor runs much slower with MSVC 2022.
Solution: Go back to MSVC 2015. (Christopher Plewright, closes #11810)

Patch 9.0.1191

Problem: Some Bazel files are not recognized.
Solution: Add an extra Bazel pattern. (Keith Smily, closes #11807)

Patch 9.0.1192

Problem: No error when class function argument shadows a member.
Solution: Check for shadowing.

Patch 9.0.1193

Problem: Cannot map <Esc> when using the Kitty key protocol.
Solution: Add a non-simplified mapping for K_ESC. (closes #11811)

Patch 9.0.1194

Problem: Compiler warning for comparing pointer with int.
Solution: Change NULL to zero.

Patch 9.0.1195

Problem: Restoring KeyTyped when building statusline not tested.
Solution: Add a test. Clean up and fix other tests. (closes #11815)

Patch 9.0.1196

Problem: Code is indented more than necessary.
Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11813)

Patch 9.0.1197

Problem: Dump file missing from patch.
Solution: Add missing dump file.

Patch 9.0.1198

Problem: Abstract class not supported yet.
Solution: Implement abstract class and add tests.

Patch 9.0.1199

Problem: Crash when using kitty and using a mapping with <Esc>.
Solution: Do not try setting did_simplify when it is NULL. (closes #11817)

Patch 9.0.1200

Problem: AppVeyor builds with an old Python version.
Solution: Switch from Python 3.8 to 3.11. (Christopher Plewright, closes #11814)

Patch 9.0.1201

Problem: Assignment with operator doesn't work in object method.

Solution: Handle loading the object member. (closes #11820) Add a few more tests.

Patch 9.0.1202

Problem: Crash when iterating over list of objects.

Solution: Do not make a copy of tt_member for object or class. (closes #11823)

Patch 9.0.1203

Problem: Return type of values() is always list<any>.

Solution: Use the member type if possible. (issue #11822)

Patch 9.0.1204

Problem: Expression compiled the wrong way after using an object.

Solution: Generate constants before getting the type.

Patch 9.0.1205

Problem: Crash when handling class that extends another class with more than one object members.

Solution: Correct pointer computations. (closes #11824)

Patch 9.0.1206

Problem: Testing with Python on AppVeyor does not work properly.

Solution: Fix typo. Move most lines to the .bat file. (Christopher Plewright, closes #11828)

Patch 9.0.1207

Problem: Error when object type is expected but getting "any".

Solution: When actual type is "any" use a runtime type check. (closes #11826)

Patch 9.0.1208

Problem: Code is indented more than necessary.

Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11819)

Patch 9.0.1209

Problem: Getting interface member does not always work.

Solution: Convert the index on the interface to the index on the object. (closes #11825)

Patch 9.0.1210

Problem: Compiler complains about declaration after label.

Solution: Move declaration to beginning of block (John Marriott)

Patch 9.0.1211

Problem: Storing value in interface member does not always work.

Solution: Convert the index on the interface to the index on the object.

Patch 9.0.1212

Problem: Cannot read back what setcellwidths() has done.

Solution: Add getcellwidths(). (Kota Kato, closes #11837)

Patch 9.0.1213

Problem: Adding a line below the last one does not expand fold.
Solution: Do not skip mark_adjust() when adding lines below the last one.
(Brandon Simmons, closes #11832, closes #10698)

Patch 9.0.1214

Problem: File left behind after running tests.
Solution: Delete the file. (Dominique Pellé, closes #11839)

Patch 9.0.1215

Problem: Using isalpha() adds dependency on current locale.
Solution: Do not use isalpha() for recognizing a URL or the end of an Ex command. (closes #11835)

Patch 9.0.1216

Problem: Coverity warns for ignoring return value.
Solution: Break out of loop if function fails.

Patch 9.0.1217

Problem: Using an object member in a closure doesn't work.
Solution: Initialize lv_loop_depth. (closes #11840)

Patch 9.0.1218

Problem: Completion includes functions that don't work.
Solution: Skip functions that are not implemented. (Kota Kato, closes #11845)

Patch 9.0.1219

Problem: Handling of FORTIFY_SOURCE flags doesn't match Fedora usage.
Solution: Adjust the "sed" patterns. (Zdenek Dohnal, closes #11847)

Patch 9.0.1220

Problem: Termcap/terminfo entries do not indicate where modifiers might appear.
Solution: Add ";*" for function keys where modifiers are likely to be used.

Patch 9.0.1221

Problem: Code is indented more than necessary.
Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11833)

Patch 9.0.1222

Problem: Terminal tests are flaky on MacOS.
Solution: Add TermWait() calls. (Yegappan Lakshmanan, closes #11852)

Patch 9.0.1223

Problem: Cannot use setcellwidths() below 0x100.
Solution: Also accept characters between 0x80 and 0x100. (Ken Takata, closes #11834)

Patch 9.0.1224

Problem: Cannot call a :def function with a number for a float argument.
Solution: Accept a number as well, convert it to a float.

Patch 9.0.1225

Problem: Reading past the end of a line when formatting text.
Solution: Check for not going over the end of the line.

Patch 9.0.1226

Problem: Spurious empty line when using text properties and virtual text.
Solution: Do not set "text_prop_follows" when the other text property is not virtual text. (closes #11846)

Patch 9.0.1227

Problem: No cmdline completion for :runtime.
Solution: Add completion for :runtime. (closes #11853, closes #11447)
Improve the resulting matches.

Patch 9.0.1228

Problem: Fuzzy menu completion is only tested in the GUI.
Solution: Make fuzzy menu completion test work without GUI.
(closes #11861)

Patch 9.0.1229

Problem: Cap'n Proto files are not recognized.
Solution: Add a pattern and the "capnp" filetype. (Amaan Qureshi,
closes #11862)

Patch 9.0.1230

Problem: Apache thrift files are not recognized.
Solution: Add a pattern for thrift files. (Amaan Qureshi, closes #11859)

Patch 9.0.1231

Problem: Completion of :runtime does not handle {where} argument.
Solution: Parse the {where} argument. (closes #11863)

Patch 9.0.1232

Problem: ColorTable saving and restoring does not work properly.
Solution: Restore ColorTable[16] usage. (Christopher Plewright,
closes #11836)

Patch 9.0.1233

Problem: search() loops forever if "skip" is TRUE for all matches.
Solution: Keep the position of the first match.

Patch 9.0.1234

Problem: The code style has to be checked manually.
Solution: Add basic code style checks in a test. Fix or avoid uncovered problems.

Patch 9.0.1235

Problem: MS-Windows console: not flushing termguicolors.
Solution: Flush termguicolors. (Christopher Plewright, closes #11871)

Patch 9.0.1236

Problem: Code in same_leader() can be simplified.
Solution: Simplify code that is executed only once. (closes #11867)

Patch 9.0.1237

Problem: Code is indented more than necessary.

Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11858)

Patch 9.0.1238

Problem: :runtime completion can be further improved.

Solution: Also complete the `{where}` argument values and adjust the completion for that. (closes #11874)

Patch 9.0.1239

Problem: Cannot have a line break before an object member access.

Solution: Check for "." in next line. (closes #11864)

Patch 9.0.1240

Problem: Cannot access a private object member in a lambda defined inside the class.

Solution: Go up the context stack to find the class. (closes #11866)

Patch 9.0.1241

Problem: Coverity warns for not checking function return value.

Solution: Explicitly ignore the return value.

Patch 9.0.1242

Problem: Code for :runtime completion is not consistent.

Solution: Make code for cmdline expansion more consistent. (closes #11875)

Patch 9.0.1243

Problem: :setglobal cannot use script-local function for "expr" option.

Solution: Use the pointer to the option value properly. (closes #11883)

Patch 9.0.1244

Problem: Cursor briefly displayed in a wrong position when pressing Esc in Insert mode after autoindent was used.

Solution: Do not adjust the cursor position for assumed deleted white space if text is following. (closes #11877)

Patch 9.0.1245

Problem: Code is indented more than necessary.

Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11879)

Patch 9.0.1246

Problem: Code is indented more than necessary.

Solution: Use an early return where it makes sense. (Yegappan Lakshmanan, closes #11887)

Patch 9.0.1247

Problem: Divide by zero with '`smoothscroll`' set and a narrow window.

Solution: Bail out when the window is too narrow.

Patch 9.0.1248

Problem: Cannot export an interface. (Ernie Rael)

Solution: Add the EX_EXPORT flag to :interface. (closes #11884)

Patch 9.0.1249

Problem: Cannot export an abstract class. (Ernie Rael)

Solution: Add the EX_EXPORT flag to :abstract. (closes #11884)

Patch 9.0.1250

Problem: Cannot use an object method with :defer. (Ernie Rael)

Solution: Find the object method and generate code to call it.
(closes #11886)

Patch 9.0.1251

Problem: Checking returned value of ga_grow() is inconsistent.

Solution: Check for FAIL instead of "not OK". (Yegappan Lakshmanan,
closes #11897)

Patch 9.0.1252

Problem: MS-Windows: scrollbar cropped off on Vim exit.

Solution: Don't call SetConsoleScreenBufferInfoEx when using the alternate
screen buffer. (Christopher Plewright, closes #11882)

Patch 9.0.1253

Problem: CI adds repository unnecessarily.

Solution: Remove the line from the workflow. (closes #11900)

Patch 9.0.1254

Problem: Calling a method on an interface does not work.

Solution: At runtime figure out what method to call. (closes #11901)

Patch 9.0.1255

Problem: Changing '**virtualedit**' does not have immediate effect.

Solution: Correct how is checked for a changed value. (closes #11878)

Patch 9.0.1256

Problem: NetworkManager connection files are not recognized.

Solution: Add a pattern for NetworkManager connection files. (closes #11893)

Patch 9.0.1257

Problem: Code style is not check in test scripts.

Solution: Add basic code style check for test files.

Patch 9.0.1258

Problem: Code style test fails.

Solution: Adjust test files.

Patch 9.0.1259

Problem: Diffmode test fails.

Solution: Adjust expected result for adjusted indenting.

Patch 9.0.1260

Problem: Coverity warns for possible NULL pointer usage.

Solution: Change the condition.

Patch 9.0.1261

Problem: Elsa files are not recognized.
Solution: Add a pattern for Elsa files. (Amaan Qureshi, closes #11908)

Patch 9.0.1262

Problem: The did_set_string_option function is too long.
Solution: Split off functionality to individual functions. (Yegappan Lakshmanan, Lewis Russell, closes #11904)

Patch 9.0.1263

Problem: KDL files are not recognized.
Solution: Add a pattern for KDL files. (Amaan Qureshi, closes #11898)

Patch 9.0.1264

Problem: Motif: compiler warning for unused argument.
Solution: Add "UNUSED".

Patch 9.0.1265

Problem: Using an interface method may give a compilation error.
Solution: Do not try to compile the body of a method of an interface. (closes #11885)

Patch 9.0.1266

Problem: Error for space before ": type" is inconsistent.
Solution: Give E1059 in more places. (closes #11868)

Patch 9.0.1267

Problem: The did_set_string_option function is too long.
Solution: Further cleanup of handling string options. (Yegappan Lakshmanan, Lewis Russell, closes #11920)

Patch 9.0.1268

Problem: .clangd and .stylelintrc files don't get a filetype.
Solution: Use yaml for .clangd and json for .stylelintrc files. (Mark Skelton, closes #11916)

Patch 9.0.1269

Problem: Channel test often fails on Mac OS.
Solution: Increase the wait time from one to 15 milliseconds. (D. Ben Knoble, closes #11894)

Patch 9.0.1270

Problem: Crash when using search stat in narrow screen.
Solution: Check length of message. (closes #11921)

Patch 9.0.1271

Problem: Using sizeof() and subtract array size is tricky.
Solution: Use offsetof() instead. (closes #11926)

Patch 9.0.1272

Problem: Typo in pattern for filetype detection.
Solution: Fix the typo. (closes #11924)

Patch 9.0.1273

Problem: "1v" may select block with wrong size. (Evgeni Chasnovski)

Solution: Compute "curswant" in the right line. (closes #11925)

Patch 9.0.1274

Problem: FIRRTL files are not recognized.

Solution: Add a pattern for FIRRTL files. (Amaan Qureshi, closes #11931)

Patch 9.0.1275

Problem: The code for setting options is too complicated.

Solution: Refactor the do_set() function. (Yegappan Lakshmanan, Lewis Russell, closes #11932)

Patch 9.0.1276

Problem: Some mappings with Meta and Shift do not work.

Solution: Apply the Shift modifier to the key. (issue #11913)

Patch 9.0.1277

Problem: Cursor may move with autocmd in Visual mode.

Solution: Restore "Visual_active" before calling check_cursor().
(closes #11939)

Patch 9.0.1278

Problem: go.work.sum files are not recognized.

Solution: Recognize go.work.sum files as the gosum filetype. (Amaan Qureshi, closes #11940)

Patch 9.0.1279

Problem: Display shows lines scrolled down erroneously. (Yishai Lerner)

Solution: Do not change "wl_lnum" at index zero. (closes #11938)

Patch 9.0.1280

Problem: Insufficient testing for what 9.0.1265 fixes.

Solution: Add a couple of test cases. (issue #11885)

Patch 9.0.1281

Problem: Cadence files are not recognized.

Solution: Recognize Cadence files. (Janez Podhostnik, closes #11951)

Patch 9.0.1282

Problem: Ron files are not recognized.

Solution: Recognize Ron files. (Amaan Qureshi, closes #11948)

Patch 9.0.1283

Problem: The code for setting options is too complicated.

Solution: Refactor the do_set() function. (Yegappan Lakshmanan, Lewis Russell, closes #11945)

Patch 9.0.1284

Problem: Compiler warnings for uninitialized variables. (Tony Mechelynck)

Solution: Add variable initializations.

Patch 9.0.1285

Problem: Various small problems.

Solution: Adjust white space and comments.

Patch 9.0.1286

Problem: Coverity warns for using a NULL pointer.

Solution: Bail out when "varp" is NULL.

Patch 9.0.1287

Problem: With the Kitty key protocol Esc with NumLock cannot be mapped.

Solution: Also use K_ESC when there is a modifier. (closes #11811)

Patch 9.0.1288

Problem: FunC files are not recognized.

Solution: Recognize FunC files. (Amaan Qureshi, closes #11949)

Patch 9.0.1289

Problem: A newer version of clang can be used for CI.

Solution: Switch from clang-15 to clang-16. (closes #11577)

Patch 9.0.1290

Problem: **CTRL-N** and -P on cmdline don't trigger CmdlineChanged.

Solution: Jump to cmdline_changed instead of cmdline_not_changed.
(closes #11956)

Patch 9.0.1291

Problem: Move language files are not recognized.

Solution: Recognize Move language files. (Amaan Qureshi, closes #11947)

Patch 9.0.1292

Problem: :defer may call the wrong method for an object. (Ernie Rael)

Solution: When an object is from a class that extends or implements, figure out the method to call at runtime. (closes #11910)

Patch 9.0.1293

Problem: The set_num_option() is too long.

Solution: Move code to separate functions. (Yegappan Lakshmanan, closes #11954)

Patch 9.0.1294

Problem: The set_bool_option() function is too long.

Solution: Move code to separate functions. (Yegappan Lakshmanan, closes #11964)

Patch 9.0.1295

Problem: The option initialization function is too long.

Solution: Move code to separate functions. (Yegappan Lakshmanan, closes #11966)

Patch 9.0.1296

Problem: Calling an object method with arguments does not work. (Ernie Rael)

Solution: Take the argument count into account when looking up the object.
(closes #11911)

Patch 9.0.1297

Problem: Wrong value for \$LC_CTYPE makes the environ test fail.

Solution: Unset \$LC_CTYPE when running tests. (closes #11963)

Patch 9.0.1298

Problem: Inserting a register on the command line does not trigger incsearch or update hlsearch.

Solution: Have cmdline_insert_reg() return CMDLINE_CHANGED when appropriate and handle it correctly. (Ken Takata, closes #11960)

Patch 9.0.1299

Problem: Change for triggering incsearch not sufficiently tested.

Solution: Add a test case. Simplify the code. (closes #11971)

Patch 9.0.1300

Problem: **'statusline'** only supports one "%=" item.

Solution: Add support for multiple "%=" items. (TJ DeVries, Yegappan Lakshmanan, closes #11970, closes #11965)

Patch 9.0.1301

Problem: Virtual text below empty line not displayed.

Solution: Adjust flags and computations. (closes #11959)

Patch 9.0.1302

Problem: On a Belgian keyboard **CTRL-]** does not work.

Solution: Translate **CTRL-]** into **CTRL-]**. (closes #11831)

Patch 9.0.1303

Problem: Motif: scrollbar width/height wrong when maximized.

Solution: Set the width/height when creating the scrollbar. (closes #11946)

Patch 9.0.1304

Problem: "\$" for **'list'** option displayed in wrong position when there are text properties.

Solution: Adjust logic for order of displayed items. (closes #11959)

Patch 9.0.1305

Problem: Cursor in wrong line with virtual text above.

Solution: Count extra line for text property above/below. (closes #11959)

Patch 9.0.1306

Problem: No regression test for solved problem of #11959.

Solution: Add a test, also with **'list'** set. (closes #11959)

Patch 9.0.1307

Problem: Setting **'formatoptions'** with :let doesn't check for errors.

Solution: Pass "errbuf" to set_string_option(). (Yegappan Lakshmanan, closes #11974, closes #11972)

Patch 9.0.1308

Problem: The code for setting options is too complicated.

Solution: Refactor the code for setting options. (Yegappan Lakshmanan, closes #11989)

Patch 9.0.1309

Problem: Scrolling two lines with even line count and **'scrolloff'** set.

Solution: Adjust how the topline is computed. (closes #10545)

Patch 9.0.1310

Problem: **'splitkeep'** test has failures.

Solution: Adjust expected cursor line position.

Patch 9.0.1311

Problem: Coverity warns for using a NULL pointer.

Solution: Use "empty_option" instead of NULL.

Patch 9.0.1312

Problem: Cursor position wrong when splitting window in insert mode.

Solution: Pass the actual mode to win_fix_cursor(). (Luuk van Baal, closes #11999,

Patch 9.0.1313

Problem: Some settings use the current codepage instead of **'encoding'**.

Solution: Adjust how options are initialized. (Ken Takata, closes #11992)

Patch 9.0.1314

Problem: :messages behavior depends on **'fileformat'** of current buffer.

Solution: Pass the buffer pointer to where it is used. (Mirko Ceroni, closes #11995)

Patch 9.0.1315

Problem: Escaping for completion of map command not properly tested.

Solution: Add a few test cases. (closes #12009)

Patch 9.0.1316

Problem: MS-Windows: vimfiles dir created with admin group.

Solution: Use ShellExecAsUser to create the vimfiles directory. (Christopher Plewright, Ken Takata, closes #12000, closes #11888)

Patch 9.0.1317

Problem: Crash when using an unset object variable.

Solution: Give an error instead. (closes #12005)

Patch 9.0.1318

Problem: Code style test fails.

Solution: Remove trailing white space.

Patch 9.0.1319

Problem: PRQL files are not recognized.

Solution: Add a filetype pattern for PRQL files. (Matthias Queitsch, closes #12018)

Patch 9.0.1320

Problem: Checking the type of a null object causes a crash.

Solution: Don't try to get the class of a null object. (closes #12005)
Handle error from calling a user function better.

Patch 9.0.1321

Problem: vimscript test fails where using {expr} syntax.

Solution: Only return FCERR_FAILED in call_user_func() for Vim9 script.

Patch 9.0.1322

Problem: Crash when indexing "any" which is an object.

Solution: Check the index is a number. Do not check the member type of an object. (closes #12019)

Patch 9.0.1323

Problem: Build failure with +eval feature.

Solution: Add missing part for using funcerror_T.

Patch 9.0.1324

Problem: "gj" and "gk" do not move correctly over a closed fold.

Solution: Use the same code as used for "j"/"k" to go to the next/previous line. (Luuk van Baal, closes #12007)

Patch 9.0.1325

Problem: 'colorcolumn' highlight wrong with virtual text above.

Solution: Adjust column of 'colorcolumn' for text property. (closes #12004)

Patch 9.0.1326

Problem: Relative line number not updated with virtual text above.

Solution: Adjust the row for the line number for virtual text above. (closes #12004)

Patch 9.0.1327

Problem: Cursor in wrong position below line with virtual text below ending in multi-byte character.

Solution: When checking for last character take care of multi-byte character.

Patch 9.0.1328

Problem: Error when using "none" for GUI color is confusing.

Solution: Mention that the name should perhaps be "NONE". (closes #1400)

Patch 9.0.1329

Problem: Completion of map includes simplified ones.

Solution: Do not complete simplified mappings. (closes #12013)

Patch 9.0.1330

Problem: Handling new value of an option has a long "else if" chain.

Solution: Use a function pointer. (Yegappan Lakshmanan, closes #12015)

Patch 9.0.1331

Problem: Illegal memory access when using :ball in Visual mode.

Solution: Stop Visual mode when using :ball. (Pavel Mayorov, closes #11923)

Patch 9.0.1332

Problem: Crash when using buffer-local user command in cmdline window. (Karl Yngve Lervåg)

Solution: Use the right buffer to find the user command. (closes #12030, closes #12029)

Patch 9.0.1333

Problem: When redo'ing twice <ScriptCmd> may not get the script ID.

Solution: When "last_used_map" map is not set use "last_used_sid".

(closes #11930)

Patch 9.0.1334

Problem: Using `tt_member` for the class leads to mistakes.
Solution: Add a separate `tt_class` field.

Patch 9.0.1335

Problem: No test for bad use of spaces in help files.
Solution: Add checks for use of spaces in help files. Ignore intentional spaces. (Hirohito Higashi, closes #11952)

Patch 9.0.1336

Problem: Functions without arguments are not always declared properly.
Solution: Use `"(void)"` instead of `"()"`. (Yegappan Lakshmanan, closes #12031)

Patch 9.0.1337

Problem: Yuck files are not recognized.
Solution: Add a filetype pattern for yuck files. (Amaan Qureshi, closes #12033)

Patch 9.0.1338

Problem: `:defcompile` and `:disassemble` can't find class method. (Ernie Rael)
Solution: Make a class name and class.method name work. (closes #11984)

Patch 9.0.1339

Problem: No test for `:disassemble` with class function.
Solution: Add a test.

Patch 9.0.1340

Problem: Coverity warns for using NULL pointer.
Solution: Check that `lhs_type` is not NULL.

Patch 9.0.1341

Problem: Build error with `mzscheme` but without GUI.
Solution: Adjust `#ifdefs`. (Ken Takata, closes #12042) Also fix function argument.

Patch 9.0.1342

Problem: MS-Windows: linking may fail with space in directory name.
Solution: Add quotes. (closes #12050)

Patch 9.0.1343

Problem: Check for OSC escape sequence doesn't work.
Solution: Move square bracket to the right place. (Johan Mattsson, closes #12048)

Patch 9.0.1344

Problem: Check for OSC escape sequence doesn't work.
Solution: Fix typo in index.

Patch 9.0.1345

Problem: Too many "else if" statements for handling options.
Solution: Add more functions to handle options. (Yegappan Lakshmanan, closes #12051)

Patch 9.0.1346

Problem: Starlark files are not recognized.

Solution: Add patterns for Starlark files. (Amaan Qureshi, closes #12049)

Patch 9.0.1347

Problem: "gr **CTRL-O**" stays in Insert mode. (Pierre Ganty)

Solution: Do not set restart_edit when "cmdchar" is 'v'. (closes #12045)

Patch 9.0.1348

Problem: Un-grammar files are not recognized.

Solution: Add patterns for Un-grammar files. (Amaan Qureshi, closes #12034)

Patch 9.0.1349

Problem: "gr" with a count fails.

Solution: Break out of the loop only after using the count.

Patch 9.0.1350

Problem: CPON files are not recognized.

Solution: Add patterns for CPON files. (Amaan Qureshi, closes #12053)

Patch 9.0.1351

Problem: Dhall files are not recognized.

Solution: Add patterns for Dhall files. (Amaan Qureshi, closes #12052)

Patch 9.0.1352

Problem: "ignore" files are outdated.

Solution: Update "ignore" files. (Ken Takata, closes #12056)

Patch 9.0.1353

Problem: Too many "else if" statements to handle option values.

Solution: Add more functions to handle option value changes. (Yegappan Lakshmanan, closes #12058)

Patch 9.0.1354

Problem: "gr **CTRL-G**" stays in virtual replace mode. (Pierre Ganty)

Solution: Prepend **CTRL-V** before control characters. (closes #12045)

Patch 9.0.1355

Problem: No error when declaring a class twice. (Ernie Rael)

Solution: Pass different flags when declaring the class. (closes #12057)

Patch 9.0.1356

Problem: Cannot cancel "gr" with Esc.

Solution: Make "gr<Esc>" do nothing. (closes #12064)

Patch 9.0.1357

Problem: Using null_object results in an internal error. (Ernie Rael)

Solution: Add instructions for pushing an object and class. (closes #12044)

Patch 9.0.1358

Problem: Compilation error with some compilers.

Solution: Avoid using "class" as member name.

Patch 9.0.1359

Problem: Too many "else if" statements in handling options.

Solution: Add more functions for handling option changes. (Yegappan Lakshmanan, closes #12060)

Patch 9.0.1360

Problem: Cue files are not recognized.

Solution: Add patterns for Cue files. (Amaan Qureshi, closes #12067)

Patch 9.0.1361

Problem: extendnew() not sufficiently tested.

Solution: Add a few more test cases for extendnew(). (closes #12075)

Patch 9.0.1362

Problem: ml_get error when going to another tab. (Daniel J. Perry)

Solution: Do not call update_topline() if "curwin" is invalid.
(closes #11907)

Patch 9.0.1363

Problem: Crash when :def function has :break in skipped block. (Ernie Rael)

Solution: Don't generate a jump for a skipped :break. (closes #12077)

Patch 9.0.1364

Problem: Build error with older Mac OS.

Solution: Adjust #ifdef. (Yee Cheng Chin, closes #12074)

Patch 9.0.1365

Problem: Dead test code.

Solution: Remove code that depends on Farsi, which has been removed.
(closes #12084)

Patch 9.0.1366

Problem: Functions for setting options are in random order.

Solution: Sort functions alphabetically. (Yegappan Lakshmanan,
closes #12082)

Patch 9.0.1367

Problem: Divide by zero in zero-width window.

Solution: Check the width is positive.

Patch 9.0.1368

Problem: Bass files are not recognized.

Solution: Add patterns for Bass files. (Amaan Qureshi, closes #12088)

Patch 9.0.1369

Problem: Still some "else if" constructs for setting options.

Solution: Add a few more functions for handling options. (Yegappan Lakshmanan, closes #12090)

Patch 9.0.1370

Problem: Crash when using a NULL object. (Ernie Rael)

Solution: Check for NULL and give an error message. (closes #12083)

Patch 9.0.1371

Problem: Ballooneval interferes with Insert completion.
Solution: Ignore mouse-move events when completing. (closes #12094, closes #12092)

Patch 9.0.1372

Problem: Test for **'toolbariconsizesize'** may fail.
Solution: Only test **'toolbariconsizesize'** when it is supported. (James McCoy, closes #12095)

Patch 9.0.1373

Problem: Wrong text displayed when using both **'linebreak'** and **'list'**.
Solution: Only set "c_extra" to NUL when "p_extra" is not empty. (Hirohito Higashi, closes #12065)

Patch 9.0.1374

Problem: Function for setting options not used consistently.
Solution: Use a function for **'encoding'** and terminal options. (Yegappan Lakshmanan, closes #12099)

Patch 9.0.1375

Problem: Crash when getting member of obj of unknown class.
Solution: Check for NULL class and give an error message. (Ernie Rael, closes #12096)

Patch 9.0.1376

Problem: Accessing invalid memory with put in Visual block mode.
Solution: Adjust the cursor column if needed.

Patch 9.0.1377

Problem: job_status() may return "dead" if the process parent changed.
Solution: Call mch_process_running() to check if the job is still alive.

Patch 9.0.1378

Problem: Illegal memory access when using virtual editing.
Solution: Make sure "startspaces" is not negative.

Patch 9.0.1379

Problem: Functions for handling options are not ordered.
Solution: Put functions in alphabetical order. (Yegappan Lakshmanan, closes #12101)

Patch 9.0.1380

Problem: **CTRL-X** on 2**64 subtracts two. (James McCoy)
Solution: Correct computation for large number. (closes #12103)

Patch 9.0.1381

Problem: ACCESS_ names have a conflict with on some systems.
Solution: Rename by prepending VIM_. (Ola Söder, closes #12105)

Patch 9.0.1382

Problem: Failing test for strtptime() doesn't show returned value.
Solution: Use assert_equal() instead of assert_true().

Patch 9.0.1383

Problem: xxd: combination of little endian and cols fails. (Aapo Rantalainen)
Solution: Round up the space taken by the hex output. (closes #12097)

Patch 9.0.1384
Problem: Setting HOMEBREW_NO_AUTO_UPDATE is not needed with Homebrew version 4.
Solution: Remove setting HOMEBREW_NO_AUTO_UPDATE. (closes #12008)

Patch 9.0.1385
Problem: g'Esc is considered an error.
Solution: Make g'Esc silently abandon the command. (closes #12110)

Patch 9.0.1386
Problem: Options test fails with some window width.
Solution: Adjust what text the test checks with. (closes #12111)

Patch 9.0.1387
Problem: Scrollbar test sporadically fails.
Solution: Mark the scrollbar test as flaky. (Christian Brabandt, closes #12113)

Patch 9.0.1388
Problem: Amiga: not all builds use gethostname().
Solution: Use gethostname() for all builds except AROS. (Ola Söder, closes #12107)

Patch 9.0.1389
Problem: Amiga: a couple of include files are included twice.
Solution: Remove duplicate includes. (Ola Söder, closes #12106)

Patch 9.0.1390
Problem: FOR_ALL_ macros are defined in an unexpected file.
Solution: Move FOR_ALL_ macros to macros.h. Add FOR_ALL_HASHTAB_ITEMS. (Yegappan Lakshmanan, closes #12109)

Patch 9.0.1391
Problem: "clear" macros are not always used.
Solution: Use ALLOC_ONE, VIM_CLEAR, CLEAR_POINTER and CLEAR_FIELD in more places. (Yegappan Lakshmanan, closes #12104)

Patch 9.0.1392
Problem: Using NULL pointer with nested :open command.
Solution: Check that ccline.cmdbuff is not NULL.

Patch 9.0.1393
Problem: Cairo files are not recognized.
Solution: Add a pattern for Cairo files. (Amaan Qureshi, closes #12118)

Patch 9.0.1394
Problem: Unx Tal files are not recognized.
Solution: Add a pattern for Unx Tal files. (Amaan Qureshi, closes #12117)

Patch 9.0.1395

Problem: Odin files are not recognized.
Solution: Add a pattern for Odin files. (Amaan Qureshi, closes #12122)

Patch 9.0.1396
Problem: sort(list, 'N') does not work in Vim9 script context.
Solution: Convert string to number without giving an error. (closes #12061)

Patch 9.0.1397
Problem: Highlight for popupmenu kind and extra cannot be set.
Solution: Add PmenuKind, PmenuKindSel, PmenuExtra and PmenuExtraSel highlight groups and use them. (Gianmaria Bajo, closes #12114)

Patch 9.0.1398
Problem: Profile test repeats the headers many times.
Solution: Put the headers in script variables.

Patch 9.0.1399
Problem: Highlight test script has a few problems.
Solution: Rewrite the script in Vim9 syntax. (closes #10379)

Patch 9.0.1400
Problem: find_file_in_path() is not reentrant.
Solution: Instead of global variables pass pointers to the functions. (closes #12093)

Patch 9.0.1401
Problem: Condition is always true.
Solution: Remove the condition. (closes #12139)

Patch 9.0.1402
Problem: Crash when using null_class.
Solution: Give an error when trying to use a null class.

Patch 9.0.1403
Problem: Unused variables and functions.
Solution: Delete items and adjust #ifdefs. (Dominique Pellé, closes #12145)

Patch 9.0.1404
Problem: Compilation error with some compilers.
Solution: Adjust array initialization. (John Marriott)

Patch 9.0.1405
Problem: Missing check for out-of-memory.
Solution: Check for alloc() returning NULL pointer. (closes #12149)

Patch 9.0.1406
Problem: ILE RPG files are not recognized.
Solution: Add patterns for ILE RPG files. (Andreas Louv, issue #12152)

Patch 9.0.1407
Problem: TableGen files are not recognized.
Solution: Add a pattern for TableGen files. (Amaan Qureshi, closes #12156)

Patch 9.0.1408

Problem: QMLdir files are not recognized.
Solution: Add a pattern for QMLdir files. (Amaan Qureshi, closes #12161)

Patch 9.0.1409

Problem: Racket files are recognized as scheme.
Solution: Recognize rackets files separately. (Gabriel Kakizaki, closes #12164, closes #12162)

Patch 9.0.1410

Problem: MacOS: sed fails on .po files.
Solution: Set \$LANG to "C". (Yee Cheng Chin, closes #12153)

Patch 9.0.1411

Problem: Accuracy of profiling is not optimal.
Solution: Use CLOCK_MONOTONIC if possible. (Ernie Rael, closes #12129)

Patch 9.0.1412

Problem: Pony files are not recognized.
Solution: Add a pattern for Pony files. (Amaan Qureshi, closes #12155)

Patch 9.0.1413

Problem: Compiler warning for unused variable.
Solution: Move variable declaration. (John Marriott)

Patch 9.0.1414

Problem: <M-S-x> in Kitty does not use the Shift modifier.
Solution: Apply the Shift modifier to ASCII letters. (closes #11913)

Patch 9.0.1415

Problem: Crystal files are not recognized.
Solution: Add a pattern for Crystal files. (Amaan Qureshi, closes #12175)

Patch 9.0.1416

Problem: Crash when collection is modified when using filter().
Solution: Lock the list/dict/blob. (Ernie Rael, closes #12183)

Patch 9.0.1417

Problem: ESDL files are not recognized.
Solution: Add a pattern for ESDL files. (Amaan Qureshi, closes #12174)

Patch 9.0.1418

Problem: The included xdiff code is a bit outdated.
Solution: Sync with the latest git xdiff code. (Yee Cheng Chin, closes #12181)

Patch 9.0.1419

Problem: Lean files are not recognized.
Solution: Add a pattern for Lean files. (Amaan Qureshi, closes #12177)

Patch 9.0.1420

Problem: Build failure because SIZE_MAX is not defined.
Solution: Define SIZE_MAX when missing. (John Marriott)

Patch 9.0.1421

Problem: Nu files are not recognized.
Solution: Add a pattern for Nu files. (Amaan Qureshi, closes #12172)

Patch 9.0.1422
Problem: Sage files are not recognized.
Solution: Add a pattern for Sage files. (Amaan Qureshi, closes #12176)

Patch 9.0.1423
Problem: WebAssembly Interface Type files are not recognized.
Solution: Add a pattern for WIT files. (Amaan Qureshi, closes #12173)

Patch 9.0.1424
Problem: Unused macros are defined.
Solution: Remove the unused macros.

Patch 9.0.1425
Problem: "wat" and "wast" files are one filetype.
Solution: Add a separate filetype for "wat" files. (Amaan Qureshi, closes #12165)

Patch 9.0.1426
Problem: Indent wrong after "export namespace" in C++.
Solution: Skip over "inline" and "export" in any order. (Virginia Senioria, closes #12134, closes #12133)

Patch 9.0.1427
Problem: Warning for uninitialized variable. (Tony Mechelynck)
Solution: Add #ifdef.

Patch 9.0.1428
Problem: Cursor in wrong position when leaving insert mode.
Solution: Update the w_valid flags. Position the cursor also when not redrawing. (closes #12137)

Patch 9.0.1429
Problem: Invalid memory access when ending insert mode.
Solution: Check if the insert_skip value is valid.

Patch 9.0.1430
Problem: Livebook files are not recognized.
Solution: Add a pattern for Livebook files. (Mathias Jean Johansen, closes #12203)

Patch 9.0.1431
Problem: getscriptinfo() loops even when specific SID is given.
Solution: Only loop when needed. Give a clearer error message. (closes #12207)

Patch 9.0.1432
Problem: Completion popup in wrong position with virtual text "above".
Solution: Adjust the column. (closes #12210)

Patch 9.0.1433
Problem: On some systems the Lua library is not found.

Solution: Check if a subdirectory for Lua exists. (closes #4475)

Patch 9.0.1434

Problem: Crash when adding package already in `'runtimepath'`.

Solution: Change order for using `'runtimepath'` entries. (closes #12215)

Patch 9.0.1435

Problem: Scrolling too many lines when `'wrap'` and `'diff'` are set.

Solution: Only scroll by screenlines for `'diff'` when `'wrap'` is not set. (closes #12211)

Patch 9.0.1436

Problem: Cannot compare a typed variable with `v:none`.

Solution: Allow for `"x is v:none"` and `"x isnot v:none"`. (issue #12194)

Patch 9.0.1437

Problem: Test fails with different error number.

Solution: Adjust the expected error.

Patch 9.0.1438

Problem: `.fs` files are falsely recognized as forth files.

Solution: Check 100 lines for something that looks like forth. (Johan Kotlinski, closes #12219, closes #11988)

Patch 9.0.1439

Problem: Start Insert mode when accessing a hidden prompt buffer.

Solution: Call `leaving_window()` in `aucmd_restbuf()`. (Thorben Tröbst, closes #12148, closes #12147)

Patch 9.0.1440

Problem: `"rvim"` can execute a shell through `:diffpatch`.

Solution: Disallow the shell `"patch"` command.

Patch 9.0.1441

Problem: MacOS: Python 3 using framework do not set dll name properly.

Solution: Use the framework prefix. (Yee Cheng Chin, closes #12189)

Patch 9.0.1442

Problem: `mapset()` does not restore non-script context.

Solution: Also accept negative sid. (closes #12132)

Patch 9.0.1443

Problem: Ending Insert mode when accessing a hidden prompt buffer.

Solution: Don't stop Insert mode when it was active before. (closes #12237)

Patch 9.0.1444

Problem: Crash when passing NULL to `setcmdline()`. (Andreas Louv)

Solution: Use `tv_get_string()` instead of using `v_string` directly. (closes #12231, closes #12227)

Patch 9.0.1445

Problem: openSUSE: configure doesn't find the Motif library. (Tony Mechelynck)

Solution: Also search in `/usr/lib64`.

Patch 9.0.1446

Problem: Unnecessary checks for the "skip" flag when skipping.

Solution: Remove the unnecessary checks. (closes #12254)

Patch 9.0.1447

Problem: Condition is always true.

Solution: Remove the useless condition. (closes #12253)

Patch 9.0.1448

Problem: Diff test fails on MacOS 13.

Solution: Install GNU diffutils. (Ozaki Kiichi, closes #12258)

Patch 9.0.1449

Problem: Test for prompt buffer is flaky.

Solution: Use WaitForAssert() instead of TermWait(). (Ozaki Kiichi, closes #12247)

Patch 9.0.1450

Problem: MacOS: building fails if clock_gettime() is not available.

Solution: Add a configure check for clock_gettime(). (closes #12242)

Patch 9.0.1451

Problem: Unnecessary redrawing when '`showcmdloc`' is not "last".

Solution: Redraw later when "showcmd_is_clear" is set. (Luuk van Baal, closes #12260)

Patch 9.0.1452

Problem: Code using EVAL_CONSTANT is dead, it is never set.

Solution: Remove EVAL_CONSTANT. (closes #12252)

Patch 9.0.1453

Problem: Typos in source code and tests.

Solution: Fix the typos. (Dominique Pellé, closes #12217)

Patch 9.0.1454

Problem: Code indenting is confused by macros.

Solution: Put semicolon after the macros instead of inside. (Ozaki Kiichi, closes #12257)

Patch 9.0.1455

Problem: C++ 20 modules are not recognized.

Solution: Add patterns to recognize C++ 20 modules as "cpp". (Ben Jackson, closes #12261)

Patch 9.0.1456

Problem: Shortmess test depends on order of test execution.

Solution: Clear messages. (closes #12264)

Patch 9.0.1457

Problem: No regression test for what patch 9.0.1333 fixes.

Solution: Extend existing test to cover the fixed problem. (issue #11930)

Patch 9.0.1458

Problem: Buffer overflow when expanding long file name.
Solution: Use a larger buffer and avoid overflowing it. (Yee Cheng Chin, closes #12201)

Patch 9.0.1459

Problem: Typo in name of type.
Solution: Change funcctal_T to funcctall_T. (closes #12265)

Patch 9.0.1460

Problem: Insufficient testing for getcmdcomplttype().
Solution: Add a few more test cases. (closes #12268)

Patch 9.0.1461

Problem: Ruler not drawn correctly when using '**rulerformat**'.
Solution: Adjust formatting depending on whether the ruler is drawn in the statusline or the command line. (Sean Dewar, closes #12246)

Patch 9.0.1462

Problem: Recursively calling :defer function if it does :qa.
Solution: Clear the defer entry before calling the function. (closes #12266)

Patch 9.0.1463

Problem: Virtual text truncation only works with Unicode '**encoding**'.
Solution: Convert the ellipsis character to '**encoding**' if needed. (Hirohito Higashi, closes #12233)

Patch 9.0.1464

Problem: Strace filetype detection is expensive.
Solution: Match with a cheap pattern first. (Federico Mengozzi, closes #12220)

Patch 9.0.1465

Problem: Haiku build fails.
Solution: Do not include globals.h and proto.h twice. (Ozaki Kiichi, closes #12273)

Patch 9.0.1466

Problem: Cannot use an object member name as a method argument.
Solution: Do not give an error for using an object member name for a method argument. (Hirohito Higashi, closes #12241, closes #12225)
Fix line number for other argument error.

Patch 9.0.1467

Problem: Jenkinsfiles are not recognized as groovy.
Solution: Add a pattern for Jenkinsfiles. (closes #12236)

Patch 9.0.1468

Problem: Recursively calling :defer function if it does :qa in a compiled function.
Solution: Clear the defer entry before calling the function. (closes #12271)

Patch 9.0.1469

Problem: Deferred functions not called from autocommands.
Solution: Also go through the funcctal_stack. (closes #12267)

Patch 9.0.1470

Problem: Deferred functions invoked in unexpected order when using :qa and autocommands.

Solution: Call deferred functions for the current funccal before using the stack. (closes #12278)

Patch 9.0.1471

Problem: Warnings for function declarations.

Solution: Add argument types. (Michael Jarvis, closes #12277)

Patch 9.0.1472

Problem: ":drop fname" may change the last used tab page.

Solution: Restore the last used tab page when :drop has changed it. (closes #12087)

Patch 9.0.1473

Problem: CI does not run sound tests.

Solution: Re-enable sound tests. Use "apt-get" instead of "apt". (Ozaki Kiichi, closes #12280)

Patch 9.0.1474

Problem: CI runs with old version of Ubuntu and tools.

Solution: Update CI to more recent versions. (closes #11092)

Patch 9.0.1475

Problem: Busted configuration files are not recognized.

Solution: Recognize busted configuration files as Lua. (Craig MacEachern, closes #12209)

Patch 9.0.1476

Problem: Lines put in non-current window are not displayed. (Marius Gedminas)

Solution: Don't increment the topline when inserting just above it. (closes #12212)

Patch 9.0.1477

Problem: Crash when recovering from corrupted swap file.

Solution: Check for a valid page count. (closes #12275)

Patch 9.0.1478

Problem: Filetypes for *.v files not detected properly.

Solution: Use the file contents to detect the filetype. (Turiya, closes #12281)

Patch 9.0.1479

Problem: Small source file problems; outdated list of distributed files.

Solution: Small updates to source files and list of distributed files.

Patch 9.0.1480

Problem: Using popup menu may leave text in the command line.

Solution: Clear the command line if the popup menu covered it. (Luuk van Baal, closes #12286)

Patch 9.0.1481

Problem: Decrypting with libsodium may fail if the library changes.
Solution: Add parameters used to the encrypted file header. (Christian Brabandt, closes #12279)

Patch 9.0.1482

Problem: Crash when textprop has a very large "padding" value. (Yegappan Lakshmanan)
Solution: Avoid the "after" count to go negative.

Patch 9.0.1483

Problem: += operator does not work on class member.
Solution: Do not skip as if "this." was used. (Christian Brabandt, closes #12263)

Patch 9.0.1484

Problem: Coverity warns for using invalid array index.
Solution: Add entry for Xchacha, even though it is not used.

Patch 9.0.1485

Problem: no functions for converting from/to UTF-16 index.
Solution: Add UTF-16 flag to existing functions and add strutf16len() and utf16idx(). (Yegappan Lakshmanan, closes #12216)

Patch 9.0.1486

Problem: Parallel make might not work.
Solution: Add missing dependencies. (Samuel Dionne-Riel, closes #12288)

Patch 9.0.1487

Problem: Content-type header for LSP channel not according to spec.
Solution: Use "vscode-jsonrpc" instead of "vim-jsonrpc". (Yegappan Lakshmanan, closes #12295)

Patch 9.0.1488

Problem: xchacha20v2 crypt header is platform dependent.
Solution: Avoid using "size_t". (Ozaki Kiichi, closes #12296)

Patch 9.0.1489

Problem: Crypt with libsodium is not tested on CI.
Solution: Configure testing with libsodium. (Ozaki Kiichi, closes #12297)

Patch 9.0.1490

Problem: The ModeChanged event may be triggered too often.
Solution: Only trigger ModeChanged when no operator is pending.
(closes #12298)

Patch 9.0.1491

Problem: Wrong scrolling with ls=0 and :botright split.
Solution: Add statusline before calling frame_new_height(). (closes #12299)

Patch 9.0.1492

Problem: Using uninitialized memory when argument is missing.
Solution: Check there are sufficient arguments before the base.
(closes #12302)

Patch 9.0.1493

Problem: Popup menu position wrong in window with toolbar.

Solution: Take the window toolbar into account when positioning the popup menu. (closes #12308)

Patch 9.0.1494

Problem: Crash when recovering from corrupted swap file.

Solution: Bail out when the line index looks wrong. (closes #12276)

Patch 9.0.1495

Problem: GTK3: hiding the mouse pointer does not work. (Rory O'Kane)

Solution: Set alpha level to zero. (Kenny Stauffer, closes #12293, closes #3256)

Patch 9.0.1496

Problem: Test restoring register with wrong value.

Solution: Correct name of variable. (closes #12310)

Patch 9.0.1497

Problem: The ruler percentage can't be localized.

Solution: Use a string that can be translated. (Emir Sari, closes #12311)

Patch 9.0.1498

Problem: In a terminal window the cursor may jump around. (Kenny Stauffer)

Solution: Do not move the cursor to the position for terminal-normal mode. (closes #12312)

Patch 9.0.1499

Problem: Using uninitialized memory with fuzzy matching.

Solution: Initialize the arrays used to store match positions.

Patch 9.0.1500

Problem: The falsy operator is not tested properly.

Solution: Add a few more test cases. (closes #12319)

Patch 9.0.1501

Problem: Crash with nested :try and :throw in catch block.

Solution: Jump to :endtry before returning from function. (closes #12245)

Patch 9.0.1502

Problem: No test for deleting the end of a long wrapped line.

Solution: Add a test to check the right text is displayed. (Luuk van Baal, closes #12318)

Patch 9.0.1503

Problem: Lua files are not recognized.

Solution: Add a patter for Lua files. (Amaan Qureshi, closes #12317)

Patch 9.0.1504

Problem: No error when calling remote_startserver() with an empty string.

Solution: Give an error for an empty string. (Hirohito Higashi, closes #12327)

Patch 9.0.1505

Problem: Error when heredoc content looks like heredoc.

Solution: Handle curly expressions. (closes #12325)

Patch 9.0.1506

Problem: Line number not displayed when using **'smoothscroll'**.

Solution: Adjust condition for showing the line number. (closes #12333)

Patch 9.0.1507

Problem: Assert message is confusing with boolean result. `assert_inrange()` replaces message instead of adding it.

Solution: Don't put quotes around expected boolean value. Append message for `assert_inrange()`. (closes #12342, closes #12341)

Patch 9.0.1508

Problem: Catch does not work when lines are joined with a newline.

Solution: Set `"nextcmd"` appropriately. (closes #12348)

Patch 9.0.1509

Problem: Error message lacks mentioning the erroneous argument.

Solution: Specify the argument that the error is for.

Patch 9.0.1510

Problem: Misleading variable name for error message.

Solution: Change `"name"` to `"number"`. (closes #12345)

Patch 9.0.1511

Problem: Crash when using wrong arg types to `assert_match()`.

Solution: Check for NULL pointer. (closes #12349)

Patch 9.0.1512

Problem: Inserting lines when scrolling with **'smoothscroll'** set.

Solution: Adjust line height computation for `w_skipcol`. (Luuk van Baal, closes #12350)

Patch 9.0.1513

Problem: Text scrolls unnecessarily when splitting and **'splitkeep'** is not `"cursor"`.

Solution: Avoid resetting `w_skipcol`. (Luuk van Baal, closes #12334)

Patch 9.0.1514

Problem: Test waits unnecessarily long before checking screendump.

Solution: Remove `TermWait()` call.

Patch 9.0.1515

Problem: `reverse()` does not work for a `String`.

Solution: Implement `reverse()` for a `String`. (Yegappan Lakshmanan, closes #12179)

Patch 9.0.1516

Problem: Cannot use special keys in `<Cmd>` mapping.

Solution: Do allow for special keys in `<Cmd>` and `<ScriptCmd>` mappings. (closes #12326)

Patch 9.0.1517

Problem: MacOS: configure removes -O2 from \$CFLAGS.

Solution: Only adjust \$CFLAGS for gcc. (closes #12351)

Patch 9.0.1518

Problem: Search stats not always visible when searching backwards.

Solution: Do not display the top/bot message on top of the search stats.
(Christian Brabandt, closes #12322, closes #12222)

Patch 9.0.1519

Problem: Global '**filetype**' is set when it is detected from the file content.

Solution: Set the local '**filetype**' option value.

Patch 9.0.1520

Problem: Completion for option name includes all bool options.

Solution: Do not recognize the "noinv" prefix. Prefix "no" or "inv" when appropriate.

Patch 9.0.1521

Problem: Failing redo of command with control characters.

Solution: Use AppendToRedobuffLit() for colon commands. (closes #12354)

Patch 9.0.1522

Problem: Some functions give two error messages.

Solution: Do not give a second error message. (closes #12352)

Patch 9.0.1523

Problem: Some error messages are not marked for translation.

Solution: Surround the messages in _(). (closes #12356)

Patch 9.0.1524

Problem: Passing -1 for bool is not always rejected.

Solution: Check for error in a better way. (closes #12358)

Patch 9.0.1525

Problem: '**smoothscroll**' does not always work properly.

Solution: Do not reset w_skipcol after it was intentionally set. (Luuk van Baal, closes #12360, closes #12199, closes #12323)

Patch 9.0.1526

Problem: Condition is always true.

Solution: Remove unnecessary condition. (closes #12359)

Patch 9.0.1527

Problem: Crash when using negative value for term_cols.

Solution: Check for invalid term_cols value. (Kenta Sato, closes #12362)

Patch 9.0.1528

Problem: Libsodium encryption is only used with "huge" features, even when manually enabled through configure. (Tony Mechelynck)

Solution: Remove the condition on FEAT_HUGE.

Patch 9.0.1529

Problem: Code style test doesn't check for space after "if".

Solution: Add a test for space.

Patch 9.0.1530

Problem: Cursor moves to wrong line when **'foldmethod'** is "diff". (Rick Howe)

Solution: Adjust logic for scrolling. (Luuk van Baal, closes #12364, closes #12218)

Patch 9.0.1531

Problem: Crash when register contents ends up being invalid.

Solution: Check "y_array" is not NULL.

Patch 9.0.1532

Problem: Crash when expanding "~" in substitute causes very long text.

Solution: Limit the text length to MAXCOL.

Patch 9.0.1533

Problem: Test for **'smoothscroll'** is ineffective.

Solution: Change the order of testing "zb" and "zt". (Luuk van Baal, closes #12366)

Patch 9.0.1534

Problem: Test for expanding "~" in substitute takes too long.

Solution: Disable the test for now.

Patch 9.0.1535

Problem: Test commented out in a wrong way.

Solution: Use legacy script comment character.

Patch 9.0.1536

Problem: CI: sound dummy stopped working.

Solution: Temporarily stop using sound dummy.

Patch 9.0.1537

Problem: Message for opening the cmdline window is not translated.

Solution: Add gettext() and scan the defaults script for text to be translated. (closes #12371)

Patch 9.0.1538

Problem: :wqall does not trigger ExitPre. (Bart Libert)

Solution: Move preparations for :qall to a common function. (closes #12374)

Patch 9.0.1539

Problem: Typst filetype is not recognized.

Solution: Distinguish between sql and typst. (Gaetan Lepage, closes #12363)

Patch 9.0.1540

Problem: reverse() on string doesn't work in compiled function.

Solution: Accept string in argument type check. (Yegappan Lakshmanan, closes #12377)

Patch 9.0.1541

Problem: CI: sound dummy is disabled.

Solution: Make sound dummy work again. (closes #12380)

Patch 9.0.1542

Problem: Line not fully displayed if it doesn't fit in the screen.

Solution: Do not reset `s_skipcol` if not needed. (Luuk van Baal, closes #12376)

Patch 9.0.1543

Problem: Display errors when making topline shorter and `'smoothscroll'` is set.

Solution: Reset `w_skipcol` when the topline becomes shorter than its current value. (Luuk van Baal, closes #12367)

Patch 9.0.1544

Problem: Recent `glibc` marks `sigset()` as a deprecated.

Solution: Use `sigaction()` in `mch_signal()` if possible. (Ozaki Kiichi, closes #12373)

Patch 9.0.1545

Problem: Text not scrolled when cursor moved with `"g0"` and `"h"`.

Solution: Adjust `w_skipcol` when needed. (Luuk van Baal, closes #12387)

Patch 9.0.1546

Problem: Some commands for opening a file don't use `'switchbuf'`.

Solution: Use `'switchbuf'` for more commands. (Yegappan Lakshmanan, closes #12383, closes #12381)

Patch 9.0.1547

Problem: Coveralls workflow on CI is commented out.

Solution: Remove the Coveralls workflow. (closes #12389)

Patch 9.0.1548

Problem: CI: check in `sound-dummy` module may throw an error.

Solution: Check whether `apt-cache` can show the package description. (Christian Brabandt, closes #12390)

Patch 9.0.1549

Problem: `USD` filetype is not recognized.

Solution: Add patterns for `USD` filetype. (Colin Kennedy, closes #12370)

Patch 9.0.1550

Problem: In `cmdline` window `S-Tab` does not select previous completion. (Maxim Kim)

Solution: Add a mapping for `S-Tab`. (closes #12116)

Patch 9.0.1551

Problem: Position of marker for `'smoothscroll'` not computed correctly.

Solution: Take `'list'` and other options into account. (Luuk van Baal, closes #12393)

Patch 9.0.1552

Problem: CI: `sound-dummy` module is not installed.

Solution: Invert using the result of the condition. (closes #12394)

Patch 9.0.1553

Problem: CI: using slightly outdated gcc version.
Solution: Use "brew" to get a more recent gcc version. (closes #12391)

Patch 9.0.1554

Problem: Code for handling '**switchbuf**' is repeated.
Solution: Add a function to handle '**switchbuf**'. (Yegappan Lakshmanan, closes #12397)

Patch 9.0.1555

Problem: setchsearch() does not clear last searched char properly.
Solution: Do not accept lastc_bytelen smaller than one. (closes #12398)

Patch 9.0.1556

Problem: Vim9: error for missing "return" after "throw".
Solution: Set had_return flag for "throw". (closes #12262)

Patch 9.0.1557

Problem: Test failures for unreachable code.
Solution: Add a test override to ignore unreachable code.

Patch 9.0.1558

Problem: Wrong error for unreachable code after :throw.
Solution: Adjust the error message.

Patch 9.0.1559

Problem: Function argument types not always checked and using v:none may cause an error.
Solution: Check argument types once the function type is known. Do not give an error for using v:none as an argument. (closes #12200)

Patch 9.0.1560

Problem: Win32: When '**encoding**' is set \$PATH has duplicate entries.
Solution: Only append the directory if it is not there yet. (Ken Takata, closes #12400, closes #12372)

Patch 9.0.1561

Problem: Display wrong when moving cursor to above the top line and '**smoothscroll**' is set.
Solution: Call adjust_skipcol() in more places and make it work better. (Luuk van Baal, closes #12395)

Patch 9.0.1562

Problem: Mixing package managers is not a good idea.
Solution: Install gcc 13 with apt-get. (closes #12405)

Patch 9.0.1563

Problem: GTK3: window manager resize hints are incomplete.
Solution: Use NULL for second argument of gtk_window_set_geometry_hints(). (Kenny Stauffer closes #11055)

Patch 9.0.1564

Problem: Display moves up and down with '**incsearch**' and '**smoothscroll**'.
Solution: Do not check if w_skipcol changed. (Luuk van Baal, closes #12410, closes #12409)

Patch 9.0.1565

Problem: Json lines files are not recognized.

Solution: Add a pattern to detect "jsonl" files. (issue #7520)

Patch 9.0.1566

Problem: Motif: GUI scrollbar test fails in 24 lines terminal.

Solution: Skip the part of the test that fails for now.

Patch 9.0.1567

Problem: Profiler calculation may be wrong on 32 bit builds.

Solution: Use 64 bit variable if possible. (Isao Sato, closes #12412)

Patch 9.0.1568

Problem: With '**smoothscroll**' cursor may move below botline.

Solution: Call redraw_later() if needed, Compute cursor row with adjusted condition. (Luuk van Baal, closes #12415)

Patch 9.0.1569

Problem: Cannot use "this.member" in lambda in class method.

Solution: Adjust check for reserved keyword. (Hirohito Higashi, closes #12416, closes #12076, closes #12336)

Patch 9.0.1570

Problem: Some tests are slow.

Solution: Make a few test cases faster.

Patch 9.0.1571

Problem: RedrawingDisabled not used consistently.

Solution: Avoid RedrawingDisabled going negative. Set RedrawingDisabled in win_split_ins(). (closes #11961)

Patch 9.0.1572

Problem: Error messages are not translated.

Solution: Add _().

Patch 9.0.1573

Problem: Error for function name has wrong line number.

Solution: Set the line number before giving the error.

Patch 9.0.1574

Problem: MS-Windows: list of translation input files incomplete.

Solution: Move the list of files to a common file. (closes #12426)

Patch 9.0.1575

Problem: "file N of M" message is not translated.

Solution: Make argument count message translatable. (close #12429)

Patch 9.0.1576

Problem: Users may not know what to do with an internal error.

Solution: Add a translated message with instructions.

Patch 9.0.1577

Problem: MS-Windows: context menu translations may be wrong.

Solution: Set the encoding before using `gettext()`. (Ken Takata, closes #12441, closes #12431)

Patch 9.0.1578

Problem: SpellCap highlight not always updated when needed.

Solution: Handle updating line below closed fold and other situations where only part of the window is redrawn. (Luuk van Baal, closes #12428, closes #12420)

Patch 9.0.1579

Problem: Some error messages are not translated.

Solution: Add the `N_()` marker on messages. (closes #12427)

Patch 9.0.1580

Problem: CI: indent test hangs on FreeBSD.

Solution: Set `'nomore'` when running the indent tests. (Ozaki Kiichi, closes #12446)

Patch 9.0.1581

Problem: Translation does not work for plural argument.

Solution: Use `PLURAL_MSG()` for errors and with `xgettext`. (closes #12443)

Patch 9.0.1582

Problem: `:stopinsert` may not work in a popup close handler. (Ben Jackson)

Solution: Restore `stop_insert_mode` when appropriate. (closes #12452, closes #12434)

Patch 9.0.1583

Problem: Get E304 when using `'cryptmethod'` "xchacha20v2". (Steve Mynott)

Solution: Add 4th crypt method to block zero ID check. Avoid syncing a swap file before reading the file. (closes #12433)

Patch 9.0.1584

Problem: Not all meson files are recognized.

Solution: Add `"meson.options"`. (Liam Beguin, closes #12444)

Patch 9.0.1585

Problem: Weird use of static variables for spell checking.

Solution: Move the variables to a structure and pass them from `win_update()` to `win_line()`. (Luuk van Baal, closes #12448)

Patch 9.0.1586

Problem: Checking translations gives an error for using two messages with `ngettext()` that differ in `"%"` items.

Solution: Adjust the check script to tolerate omitting one `"%"` item.

Patch 9.0.1587

Problem: Corn config files are not recognized.

Solution: Add a pattern for Corn config files. (Jake Stanger, closes #12449)

Patch 9.0.1588

Problem: Incsearch not triggered when pasting clipboard register on the command line.

Solution: Also set `"literally"` when using a clipboard register. (Ken Takata,

closes #12460)

Patch 9.0.1589

Problem: Filetype test contains too many special characters.

Solution: Use Vim9 syntax for a few things.

Patch 9.0.1590

Problem: Filetype test has trailing white space.

Solution: Remove trailing white space.

Patch 9.0.1591

Problem: Some "gomod" files are not recognized.

Solution: Check for "go.mod" file name before checking out the contents.
(Omar El Halabi, closes #12462)

Patch 9.0.1592

Problem: Not all timer tests are marked as flaky.

Solution: Set the flaky flag for all timer tests. (closes #12355)

Patch 9.0.1593

Problem: MS-Windows: assert error when compiled with debug mode.

Solution: Adjust arguments to setvbuf(). (Ken Takata, closes #12467)

Patch 9.0.1594

Problem: Some internal error messages are translated.

Solution: Consistently do not translate internal error messages.
(closes #12459)

Patch 9.0.1595

Problem: Line pointer becomes invalid when using spell checking.

Solution: Call ml_get() at the right places. (Luuk van Baal, closes #12456)

Patch 9.0.1596

Problem: :registers command does not work in sandbox.

Solution: Add flag to the command. (closes #12473)

Patch 9.0.1597

Problem: Cursor ends up below the window after a put.

Solution: Mark w_crow and w_botline invalid when changing the cursor line.
(closes #12465)

Patch 9.0.1598

Problem: screenchar(), screenchars() and screenstring() do not work properly when **'encoding'** is set to a double-byte encoding.

Solution: Fix the way the bytes of the characters are obtained.
(issue #12469)

Patch 9.0.1599

Problem: Cursor not adjusted when near top or bottom of window and **'splitkeep'** is not "cursor".

Solution: Move boundary checks to outer cursor move functions, inner functions should only return valid cursor positions. (Luuk van Baal, closes #12480)

Patch 9.0.1600

Problem: screenpos() does not take w_skipcol into account.

Solution: Subtract w_skipcol from column. (closes #12486, closes #12476)

Patch 9.0.1601

Problem: Filetype detection fails for *.conf file without comments.
(Dmitrii Tcyganok)

Solution: Use "conf" filetype as a fallback for an empty .conf file.
(closes #12487, closes #12483)

Patch 9.0.1602

Problem: Stray character is visible if 'smoothscroll' marker is displayed
on top of a double-wide character.

Solution: When overwriting a double-width character with the 'smoothscroll'
marker clear the second half. (closes #12469)

Patch 9.0.1603

Problem: Display wrong when scrolling multiple lines with 'smoothscroll'
set.

Solution: Redraw when w_skipcol changed. (closes #12477, closes #12468)

Patch 9.0.1604

Problem: Errors from the codestyle test are a bit confusing.

Solution: Use assert_report() with a clearer message. Avoid a warning for
an existing swap file.

Patch 9.0.1605

Problem: Crash when calling method on super in child constructor. (Israel
Chauca Fuentes)

Solution: Clear the type list. (Ernie Rael, closes #12489, closes #12471)

Patch 9.0.1606

Problem: Using freed memory when 'foldcolumn' is set.

Solution: Save extra pointer to free it later. (closes #12492)

Patch 9.0.1607

Problem: screenpos() returns wrong row with diff filler lines.

Solution: Only add filler lines when appropriate. Also don't add the
'smoothscroll' marker when w_skipcol is zero. (closes #12485,
closes #12484)

Patch 9.0.1608

Problem: update_topleft() is called twice.

Solution: Do not call update_topleft() before curs_columns(). (Luuk van
Baal, closes #12495)

Patch 9.0.1609

Problem: Crash when an object indirectly references itself.

Solution: Avoid clearing an object while it is already being cleared.
(closes #12494)

Patch 9.0.1610

Problem: Display is wrong when 'smoothscroll' is set and scrolling multiple
lines.

Solution: Redraw with UPD_NOT_VALID when "skipcol" is or was set.
(closes #12490, closes #12468)

Patch 9.0.1611

Problem: v:maxcol can be changed in a :for loop.
Solution: Check for read-only loop variable. (closes #12470)

Patch 9.0.1612

Problem: "skipcol" not reset when using multi-byte characters.
Solution: Compare with w_virtcol instead of w_cursor.col. (closes #12457)

Patch 9.0.1613

Problem: Some make output gets picked up by **'errorformat'**.
Solution: Ignore make output by default. (Gregory Anders, closes #12481)

Patch 9.0.1614

Problem: strlen() called too often for :spellrepl.
Solution: Store the result in a variable. (closes #12497)

Patch 9.0.1615

Problem: URL shortcut files are not recognized.
Solution: Add a pattern for URL shortcut files. (closes #12474)

Patch 9.0.1616

Problem: Quickfix text field is truncated.
Solution: Fix output of text field after pattern field in quickfix buffer.
(Shane Harper, closes #12498)

Patch 9.0.1617

Problem: charidx() and utf16idx() result is not consistent with byteidx().
Solution: When the index is equal to the length of the text return the
length of the text instead of -1. (Yegappan Lakshmanan,
closes #12503)

Patch 9.0.1618

Problem: Trace32 files are not recognized.
Solution: Add patterns for the t32 filetype. (Christoph Sax, closes #12505)

Patch 9.0.1619

Problem: The focus gained/lost escape sequences cause trouble for a
terminal where Vim does not expect them.
Solution: Always recognize the codes for focus gained/lost. (closes #12499)

Patch 9.0.1620

Problem: Nix files are not recognized from the hashbang line.
Solution: Add a hashbang check. (issue #12507)

Patch 9.0.1621

Problem: FILETYPE_FILE is defined to the same value multiple times. Same
for a few similar macros.
Solution: Define FILETYPE_FILE and others in feature.h only

Patch 9.0.1622

Problem: Filetype name t32 is a bit obscure.

Solution: Rename t32 to trace32. (Christoph Sax, closes #12512)

Patch 9.0.1623

Problem: The program to filetype translation is not exported.

Solution: Export Exe2filetype().

Patch 9.0.1624

Problem: Crash when calling object constructor from legacy script. (Israel Chauca Fuentes)

Solution: Pass a pointer for "ufunc". (closes #12502)

Patch 9.0.1625

Problem: "super" is not considered a reserved name.

Solution: Add "super" to the list of reserved names. (closes #12515)

Patch 9.0.1626

Problem: Visual area not shown when using '**showbreak**' and start of line is not visible. (Jaehwang Jung)

Solution: Adjust "fromcol" for the space taken by '**showbreak**'. (closes #12514)

Patch 9.0.1627

Problem: No generic mechanism to test syntax plugins.

Solution: Add a syntax plugin test mechanism, using screendumps. Add a simple test for "c".

Patch 9.0.1628

Problem: Syntax tests fail on FreeBSD.

Solution: Pass the Vim executable path with VIMPROG. (Ken Takata, closes #12535) Adjust the paths.

Patch 9.0.1629

Problem: Having utf16idx() rounding up is inconvenient.

Solution: Make utf16idx() round down. (Yegappan Lakshmanan, closes #12523)

Patch 9.0.1630

Problem: "make clean" at the toplevel fails.

Solution: Clean the indent and syntax directories in a sub-shell. (Ben Jackson, closes #12536, closes #12526)

Patch 9.0.1631

Problem: Passing a wrong variable type to an option gives multiple errors.

Solution: Bail out early on failure. (closes #12504)

Patch 9.0.1632

Problem: Not all cabal config files are recognized.

Solution: Add a couple of patterns. (Marcin Szamotulski, closes #12463)

Patch 9.0.1633

Problem: Duplicate code for converting float to string.

Solution: Use tv_get_string(). (closes #12521)

Patch 9.0.1634

Problem: Message is cleared when removing mode message (Gary Johnson).

Solution: Do not clear the command line after displaying a message.

Patch 9.0.1635

Problem: Error message is cleared when removing mode message.

Solution: Also reset flags when the message is further down.

Patch 9.0.1636

Problem: Expanding a pattern interferes with command line completion.

Solution: Set the file index only when appropriate. (closes #12519)

Patch 9.0.1637

Problem: Compiler warning for uninitialized variable.

Solution: Move the variable to an inner block and initialize it. (Christian Brabandt, closes #12549)

Patch 9.0.1638

Problem: crypt tests hang and cause memory errors

Solution: Move variable to start of function.

Patch 9.0.1639

Problem: Build failure without the crypt feature.

Solution: Adjust #ifdefs

Patch 9.0.1640

Problem: Compiler warning for unused variables without the crypt feature.

Solution: Adjust #ifdefs

Patch 9.0.1641

Problem: The log file does not give information about window sizes.

Solution: Add a few log messages about obtaining the window size.

Patch 9.0.1642

Problem: Build failure with tiny features.

Solution: Add #ifdef's.

Patch 9.0.1643

Problem: Filetype detection fails if file name ends in many '~'.

Solution: Strip multiple '~' at the same time. (closes #12553)

Patch 9.0.1644

Problem: Not all filetype file name matches are tested.

Solution: Add more file names to test with. (Jonas Strittmatter, closes #12569)

Patch 9.0.1645

Problem: zserio files are not recognized.

Solution: Add a pattern for zserio files. (Dominique Pellé, closes #12544)

Patch 9.0.1646

Problem: CI: codecov may take a very long time to run.

Solution: Add a timeout. (Philip Heiduck, closes #12559)

Patch 9.0.1647

Problem: Insufficient testing for syntax plugins.
Solution: Add shell file examples. (Charles Campbell) Create a messages file for easier debugging and reporting the test results.

Patch 9.0.1648

Problem: Result of syntax tests is hard to see.
Solution: List the failed tests.

Patch 9.0.1649

Problem: Syntax test failure causes script to abort.
Solution: Fix appending string to list.

Patch 9.0.1650

Problem: MS-Windows: default **'viewdir'** may include read-only directory.
Solution: Use \$HOME instead of \$VIM for **'viewdir'** default. (closes #12119)

Patch 9.0.1651

Problem: Unclear why syntax test fails on Mac.
Solution: Temporarily show the whole "messages" file.

Patch 9.0.1652

Problem: Unclear why syntax test fails on Mac.
Solution: Echo v:errors when it's not empty.

Patch 9.0.1653

Problem: Amiga: default **'viewdir'** may not work.
Solution: Use "home:" instead of "\$VIM". Add a test. (Christian Brabandt, closes #12576)

Patch 9.0.1654

Problem: MS-Windows: test for default **'viewdir'** fails.
Solution: Escape the pattern.

Patch 9.0.1655

Problem: Syntax test fails when Vim window is not tall enough.
Solution: Make sure each terminal window is closed.

Patch 9.0.1656

Problem: Syntax test fails when detected shell type differs.
Solution: Avoid using "/bin/sh", it depends on the system. Add a check that the shell type detection is correct.

Patch 9.0.1657

Problem: One more syntax test depends on the system.
Solution: Use "dash" instead of "sh".

Patch 9.0.1658

Problem: Autoload files for "zig" are not installed.
Solution: Add install and uninstall rules in the makefile. (Christian Brabandt, closes #12577, closes #12567)

Patch 9.0.1659

Problem: Termdebug: default highlight cleared when changing colorscheme.
Solution: Use a ColorScheme autocommand. (Christian Brabandt, closes #12566,

closes #12555)

Patch 9.0.1660

Problem: Error for using matchfuzzy() in Vim9 script returning a list of dicts.

Solution: Make return type of matchfuzzy() list<any>. (Yegappan Lakshmanan, closes #12574)

Patch 9.0.1661

Problem: BUCK files are not recognized.

Solution: Recognize BUCK files as "bzl". (Son Luong Ngoc, closes #12564)

Patch 9.0.1662

Problem: Crash when using a class member twice. (Christian J. Robinson)

Solution: Make a copy of the value.

Patch 9.0.1663

Problem: Termdebug on MS-Windows: some file names are not recognized.

Solution: Do not always change \t and \n. (Christian Brabandt, closes #12565, closes #12560, closes #12550)

Patch 9.0.1664

Problem: Divide by zero when scrolling with '**smoothscroll**' set.

Solution: Avoid using a negative width. (closes #12540, closes #12528)

Patch 9.0.1665

Problem: Empty CmdlineEnter autocommand causes errors in Ex mode.

Solution: Save and restore ex_pressedreturn. (Christian Brabandt, closes # 12581, closes #12578)

Patch 9.0.1666

Problem: Compiler may warn for uninitialized variable.

Solution: Initialize this_props_len. (Christian Brabandt, closes #12599)

Patch 9.0.1667

Problem: Regression test doesn't fail when fix is reverted.

Solution: Add "n" to '**cpoptions**' instead of using :winsize. (closes #12587, issue #12528)

Patch 9.0.1668

Problem: PEM files are not recognized.

Solution: Add patterns to match PEM files. (closes #12582)

Patch 9.0.1669

Problem: Crash syncing swapfile in new buffer when using sodium crypt. (James McCoy)

Solution: Add checks for sodium encryption. (Christian Brabandt, closes #12591, closes #12585)

Patch 9.0.1670

Problem: Resetting local option to global value is inconsistent.

Solution: Handle "<" specifically for '**scrolloff**' and '**sidescrolloff**'. (closes #12594)

Patch 9.0.1671

Problem: Termdebug: error with more than 99 breakpoints.

Solution: Use a different sign for breakpoint 100 and over. (closes #12589, closes #12588)

Patch 9.0.1672

Problem: Tabline highlight wrong after truncated double width label.

Solution: Fill up half a double width character later. (closes #12614)

Patch 9.0.1673

Problem: Cannot produce a status 418 or 503 message.

Solution: Add err_teapot().

Patch 9.0.1674

Problem: Help for builtin functions is not sorted properly.

Solution: Put err_teapot() help in the right position.

Patch 9.0.1675

Problem: Test may run into timeout when using valgrind.

Solution: Use a longer timeout when using valgrind.

Patch 9.0.1676

Problem: Warning for buffer in use when exiting early.

Solution: Change file names to be able to see what buffer is in use when exiting.

Patch 9.0.1677

Problem: Typo in syntax test input file.

Solution: Fix the typo and the expected dump files. (THARAK HEGDE, closes #12635)

Patch 9.0.1678

Problem: Blade files are not recognized.

Solution: Add a pattern for Blade files. (closes #12650)

Patch 9.0.1679

Problem: Tests may leave leftover files around

Solution: Clean up tests and remove files

Patch 9.0.1680

Problem: sodium test fails in Github CI

Solution: Catch sodium_mlock() errors and do not error out

Patch 9.0.1681

Problem: Build Failure with Perl 5.38

Solution: Fix Build Failure

Patch 9.0.1682

Problem: crypt: sodium encryption is not portable

Solution: use little-endian byte order for sodium encrypted files

Patch 9.0.1683

Problem: need runtime files updated

Solution: merge various github PRs

Patch 9.0.1684

Problem: libvterm slightly outdated

Solution: Update libvterm from rev 818 to rev 839

Patch 9.0.1685

Problem: Python 3.11 interface throws deprecation warnings

Solution: ignore those warnings for gcc and clang

Patch 9.0.1686

Problem: undotree() only works for the current buffer

Solution: Add an optional "buffer number" parameter to undotree(). If omitted, use the current buffer for backwards compatibility.

Patch 9.0.1687

Problem: mapset() not properly handling script ID

Solution: replace_termcodes() may accept a script ID

Patch 9.0.1688

Problem: cannot store custom data in quickfix list

Solution: add ``user_data`` field for the quickfix list

Patch 9.0.1689

Problem: python 3.12 produces warnings and fails test

Solution: Make use of raw strings in python3 tests

Patch 9.0.1690

Problem: popup_create() not aborting on errors

Solution: check for errors in arguments given and abort if an error occurred

Patch 9.0.1691

Problem: wrong viewport restored for incsearch and smoothscroll

Solution: Save and restore skipcol as well

Patch 9.0.1692

Problem: Android not handling AI_V4MAPPED ai_flag

Solution: don't set AI_V4MAPPED flag when on Android, since Android's getaddrinfo returns EAI_BADFLAGS if ai_flags contains it

Patch 9.0.1693

Problem: Ctrl-Q not handled like Ctrl-V in replace mode

Solution: Handle Ctrl-Q like Ctrl-V

Patch 9.0.1694

Problem: wrong mapping applied when replaying a char search

Solution: Store a NOP after the ESC

Patch 9.0.1695

Problem: Crash with overlong textprop above

Solution: Consider only positive padding

Patch 9.0.1696

Problem: sodium_mlock may still fail in CI
Solution: Catch E1230 in testscript and skip test

Patch 9.0.1697

Problem: incsearch test not sufficient (after 9.0.1691)
Solution: add an additional test

Patch 9.0.1698

Problem: Test_map_restore_sid fails in GUI
Solution: Feed an unsimplified Ctrl-B

Patch 9.0.1699

Problem: compile warning for xdiff/xutils on MS-Windows
Solution: add explicit type cast from size_t to long

Patch 9.0.1700

Problem: Cannot compile with dynamic perl < 5.38 (after 9.0.1681)
Solution: Fix if_perl/dyn from perl 5.32 to 5.38

Patch 9.0.1701

Problem: vim9 crash when class member overridden
Solution: Use method_count field instead

Patch 9.0.1702

Problem: Undo test is flaky.
Solution: Apply filter and change time to "1 second ago" in both dumps.

Patch 9.0.1703

Problem: Vim9 Calling a method in an extended class fails
Solution: use method index directly

Patch 9.0.1704

Problem: Cannot use positional arguments for printf()
Solution: Support positional arguments in string formatting

Patch 9.0.1705

Problem: cursor position wrong when clicking on an unprintable char
Solution: Don't update prev_ptr when wlv.n_extra is not zero.

Patch 9.0.1706

Problem: typos in the xxd manpage
Solution: Fix typos and formatting

Patch 9.0.1707

Problem: Cannot wrap around in popup_filter_menu()
Solution: Allow to wrap around by default

Patch 9.0.1708

Problem: getcompletion() fails for user-defined commands
Solution: set context for completion function

Patch 9.0.1709

Problem: dynamic build with python 3.12 breaks
Solution: if_python3.c: Fix building dynamic Python3 interpreter

Patch 9.0.1710

Problem: sidescolloff and scrolloff options work slightly different than other global-local options

Solution: Make it behave consistent for all global-local options

Patch 9.0.1711

Problem: dead code in charset.c

Solution: remove it

Patch 9.0.1712

Problem: missing null check in object_clear()

Solution: Add null check of cl

Patch 9.0.1713

Problem: Github CI fails to load snd-dummy kernel module

Solution: Make installation of linux-modules-extra optional

Patch 9.0.1714

Problem: getcompletion() "cmdline" fails after :autocmd

Solution: Use set_cmd_context() instead of set_one_cmd_context().

Patch 9.0.1715

Problem: duplicate test in message_test.c

Solution: Remove duplicate test and make functions static

Patch 9.0.1716

Problem: Windows build with python 3.12 and clang fails

Solution: Remove the PyBool_Type function pointer for python 3.12

Patch 9.0.1717

Problem: virtcol2col returns last byte of a multi-byte char

Solution: Make it return the first byte for a multi-byte char

Patch 9.0.1718

Problem: dict-completion does not respect region

Solution: respect selected region in dict completion

Patch 9.0.1719

Problem: if_lua: crash for Lua functions invoked via Vim callbacks

Solution: Use Lua registry rather than upvalues for udata cache

Patch 9.0.1720

Problem: Vim9 class using wrong index for overridden method

Solution: Use correct index for overridden method

Patch 9.0.1721

Problem: Build failure on Windows with dynamic lua (after 9.0.1719)

Solution: move definition further down in if_lua

Patch 9.0.1722

Problem: wrong error messages when passing wrong types to count()

Solution: fix it

Patch 9.0.1723

Problem: Fix regression in `{func}` argument of `reduce()`
Solution: pass function name as string again

Patch 9.0.1724

Problem: vim9class constructor argument type checking bug
Solution: fix it

Patch 9.0.1725

Problem: Wrong cursor position when clicking after concealed text with `'virtualedit'`.
Solution: Store virtual columns in `ScreenCols[]` instead of text columns, and always use `coladvance()` when clicking.

Patch 9.0.1726

Problem: incorrect heights in `win_size_restore()`
Solution: avoid restoring incorrect heights in `win_size_restore()`

Patch 9.0.1727

Problem: minor problems with the teapot()
Solution: remove the null check, update documentation

Patch 9.0.1728

Problem: missing `winid` argument for `virtcol()`
Solution: Add a `{winid}` argument to `virtcol()`

Patch 9.0.1729

Problem: `screenpos()` wrong result with `w_skipcol` and `cptions+=n`
Solution: Use `adjust_plines_for_skipcol()` instead of subtracting `w_skipcol`.

Patch 9.0.1730

Problem: passing multiple patterns to `runtime` not working
Solution: prepend prefix to each argument separately

Patch 9.0.1731

Problem: blockwise Visual highlight not working with virtual text
Solution: Reset the correct variable at the end of virtual selection and Check for double-width char inside virtual text.

Patch 9.0.1732

Problem: vimexpr: shadowing variable
Solution: Rename local variable

Patch 9.0.1733

Problem: CI: cannot cache linux-modules-extra
Solution: Enable caching and reduce failed downloads

Patch 9.0.1734

Problem: runtime completion fails for multiple args
Solution: Make it work

Patch 9.0.1735

Problem: Rename completion specific `findex` var

Solution: Move "findex" static variable to xp_selected in expand_T

Patch 9.0.1736

Problem: Github Actions times out after 20 minutes

Solution: Increase the timeout to 25 minutes

Patch 9.0.1737

Problem: Calling a base class method through an extended class fails

Solution: Create lookup table for member index in the interface to
to the member class implementing the interface

Patch 9.0.1738

Problem: Duplicate code to reverse a string

Solution: Move reverse_text() to strings.c and remove string_reverse().

Patch 9.0.1739

Problem: leftover files in libvterm

Solution: Fix cleaning of libvterm directory

Patch 9.0.1740

Problem: segfault when reading invalid viminfo file

Solution: Check the expected type in the viminfo file

Patch 9.0.1741

Problem: No type checking in interfaces

Solution: Implement member type check in vim9 interfaces

Patch 9.0.1742

Problem: Wrong curswant when clicking and the second cell of a
double-width char.

Solution: Don't copy virtcol of the first char to the second one.

Patch 9.0.1743

Problem: Parameter of gui_gtk:gui_mch_browse incorrectly marked as
UNUSED.

Solution: Remove UNUSED flag.

Patch 9.0.1744

Problem: Dead code in open_cmdwin()

Solution: Remove it

Patch 9.0.1745

Problem: Missing test coverage for blockwise Visual highlight with
virtual that starts with a double-width char.

Solution: Add a new virtual text to the test. Some other small fixes.

Patch 9.0.1746

Problem: vim9class compile error for char/char_u conversion

Solution: Correctly cast to (char *)

Patch 9.0.1747

Problem: screenpos() may cause unnecessary redraw.

Solution: Don't unnecessarily reset VALID_WROW flag.

Patch 9.0.1748

Problem: CI: cannot label issues automatically
Solution: Create CI labeler

Patch 9.0.1749

Problem: Text property highlight doesn't override a sign highlight over a tab character
Solution: Let text_property override tab highlighting

Patch 9.0.1750

Problem: CI: fails because of changed error messages (after: 9.0.1741)
Solution: Adjust expected error messages

Patch 9.0.1751

Problem: CI: labeler configuration not found (after 9.0.1748)
Solution: set configuration path

Patch 9.0.1752

Problem: CI: Coveralls is no longer used
Solution: Remove .coveralls.yml

Patch 9.0.1753

Problem: can't move to last non-blank char
Solution: Make g<end> behave like that

Patch 9.0.1754

Problem: still ci breakage (after 9.0.1741)
Solution: fix remaining issue

Patch 9.0.1755

Problem: CI still fails with sodium mlock error
Solution: catch mlock failure

Patch 9.0.1756

Problem: failing cursorline sign test
Solution: only reset char attr, if cursorline option is not set

Patch 9.0.1757

Problem: ex_class() function is too long
Solution: refactor it

Patch 9.0.1758

Problem: vim9 no class identifiers in stack dumps
Solution: Prefix class members in stack traces with the class name followed by a dot.

Patch 9.0.1759

Problem: Visual highlight not working with cursor at end of screen line and **'showbreak'**.
Solution: Only update "vcol_prev" when drawing buffer text.

Patch 9.0.1760

Problem: vim9 class problem with new() constructor

Solution: Don't allow a return type for the new() class constructor.

Patch 9.0.1761

Problem: g<kEnd> behaves different from g<end>

Solution: Make g<kEnd> behave like g<End>

Patch 9.0.1762

Problem: Not able to get the virtual text property

Solution: Make prop_list() return virtual text and alignment

Patch 9.0.1763

Problem: crash when passing invalid buffer to undotree()

Solution: Use get_buf_arg() instead of tv_get_buf_from_arg().

Patch 9.0.1764

Problem: CI: label should not be set on all yml files

Solution: only set it for specific yml files in .github

Patch 9.0.1765

Problem: Error when cross-compiling Vim

Solution: use AC_CHECK_SIZEOF to find sizeof(wchar_t)

Patch 9.0.1766

Problem: Runtime: Missing QML support

Solution: Add QML support to Vim

Patch 9.0.1767

Problem: '.' no allowed in highlight group names

Solution: Allow dot and hyphen characters in highlight group names

Patch 9.0.1768

Problem: Runtime: no support for bicep files

Solution: Add filetype support for bicepparam

Patch 9.0.1769

Problem: executable() ignoring symlinks on Windows

Solution: resolve reparse points

Patch 9.0.1770

Problem: lines disappear when modifying chars before virt text

Solution: take virtual text property length into account

Patch 9.0.1771

Problem: regex: combining chars in collections not handled

Solution: Check for following combining characters for NFA and BT engine

Patch 9.0.1772

Problem: Cursor is adjusted in window that did not change in size by **'splitkeep'**.

Solution: Only check that cursor position is valid in a window that has changed in size.

Patch 9.0.1773

Problem: cannot distinguish Forth and Fortran *.f files
Solution: Add Filetype detection Code

Patch 9.0.1774

Problem: no support for custom cmdline completion
Solution: Add new vimscript functions

Patch 9.0.1775

Problem: Wrong comparison in vim9type.c
Solution: Change condition to false

Patch 9.0.1776

Problem: No support for stable Python 3 ABI
Solution: Support Python 3 stable ABI

Patch 9.0.1777

Problem: patch 9.0.1771 causes problems
Solution: revert it

Patch 9.0.1778

Problem: if_py_both: code-style issue
Solution: add space

Patch 9.0.1779

Problem: Need more state() tests
Solution: Add a few more tests for operator pending mode and register yank command

Patch 9.0.1780

Problem: Vim9 type not defined during object creation
Solution: Define type during object creation and not during class definition, parse multi-line member initializers, fix lock initialization

Patch 9.0.1781

Problem: Problems when setting bin/paste option
Solution: When setting binary/paste, remember that this also affects depending options, so that :verbose set returns the right location.

Patch 9.0.1782

Problem: prop_list() does not return text_padding_left
Solution: Store and return the text_padding_left value for text properties

Patch 9.0.1783

Problem: Wrong display with wrapping virtual text or unprintable chars, **'showbreak'** and **'smoothscroll'**.
Solution: Don't skip cells taken by **'showbreak'** in screen lines before "w_skipcol". Combined "n_skip" and "skip_cells".

Patch 9.0.1784

Problem: redundant else in pum_set_selected()

Solution: Remove it

Patch 9.0.1785

Problem: wrong cursor position with **'showbreak'** and lcs-eol

Solution: Add size of **'showbreak'** before when **'listchars'** "eol" is used.
Also fix wrong cursor position with wrapping virtual text on empty line and **'showbreak'**.

Patch 9.0.1786

Problem: Vim9: need instanceof() function

Solution: Implement instanceof() builtin

Patch 9.0.1787

Problem: Cannot build with latest luajit

Solution: adjust sed regexp and don't expect '-' in version output

Patch 9.0.1788

Problem: C4090 warnings in strings.c

Solution: Add type casts

Patch 9.0.1789

Problem: too early declaration of variable in pum_set_selected()

Solution: Move declaration to where it is actually used

Patch 9.0.1790

Problem: The Content-Type header is an optional header that some LSP servers struggle with and may crash when encountering it.

Solution: Drop the Content-Type header from all messages, because we use the default value anyway.

Patch 9.0.1791

Problem: No tests for the Termdebug plugin

Solution: Add some simple tests for the Termdebug plugin

Patch 9.0.1792

Problem: Normal mode "gM", "gj", "gk" commands behave incorrectly with virtual text.

Solution: Use linetabsize() instead of linetabsize_str().

Patch 9.0.1793

Problem: obsolete macros in configure script

Solution: Remove those and start moving to autoconf 2.71

Patch 9.0.1794

Problem: autoconf: not correctly detecting include dirs

Solution: make use of python3 to generate includedirs

Patch 9.0.1795

Problem: Indentation issues

Solution: Fix code indentation issues.

Patch 9.0.1796

Problem: Vim9 problems with null_objects

Solution: Vim9 improve null_object usage

Patch 9.0.1797

Problem: Vimball/Visual Basic filetype detection conflict

Solution: runtime(vb): Improve Vimball and Visual Basic detection logic

Patch 9.0.1798

Problem: The **'syntax'** option has no completion.

Solution: Add syntax option completion.

Patch 9.0.1799

Problem: Russian menu translation can be improved

Solution: update the Russian menu files

Patch 9.0.1800

Problem: Cursor position still wrong with **'showbreak'** and virtual text after last character or **'listchars'** "eol".

Solution: Remove unnecessary w_wcol adjustment in curs_columns(). Also fix first char of virtual text not shown at the start of a screen line.

Patch 9.0.1801

Problem: Vim9 instanceof() fails in a def func

Solution: allow Objects in compile time check

Patch 9.0.1802

Problem: Multiline regex with Visual selection fails when Visual selection contains virtual text after last char.

Solution: Only include virtual text after last char when getting full line length.

Patch 9.0.1803

Problem: Cannot detect norg markup files

Solution: Add norg markup language detection

Patch 9.0.1804

Problem: Vim9: no support for private object methods

Solution: Add support for private object/class methods

Patch 9.0.1805

Problem: Vim9: problem compiling object method as function call arg

Solution: After a object/class method call, remove the object/class from the stack.

Patch 9.0.1806

Problem: Vim9: bogus error on export

Solution: Don't error out when the export command is not executed

Patch 9.0.1807

Problem: runtime: crystal scripts not recognised

Solution: Filetype detect Crystal scripts by shebang line

Patch 9.0.1808

Problem: Termdebug: Typo in Termdebug test

Solution: fix the typos

Patch 9.0.1809

Problem: Termdebug test flaky

Solution: wait slightly longer

Patch 9.0.1810

Problem: camel-case spelling has issues with digits

Solution: Improve the camCase spell checking by taking digits and caps into account

Patch 9.0.1811

Problem: still some issues with term_debug test

Solution: Use WaitForAssert()

Patch 9.0.1812

Problem: CI still fails with sodium_mlock error()

Solution: Catch and ignore E1230 error in test_crypt

Patch 9.0.1813

Problem: **'linebreak'** is incorrectly drawn after **'breakindent'**.

Solution: Don't include **'breakindent'** size when already after it.

Patch 9.0.1814

Problem: Vim9 no error on duplicate object member var

Solution: detect duplicate members and error out

Patch 9.0.1815

Problem: pango_coverage_unref() deprecated in pango > 1.51

Solution: use g_object_unref() instead

Patch 9.0.1816

Problem: configure: sed uses non-portable regex

Solution: use '*' modifier instead of '\?' in regex for luajit version detection

Patch 9.0.1817

Problem: configure: using obsolete AC_HEADER_STDC

Solution: Remove it and re-create configure

Patch 9.0.1818

Problem: dynamically linking perl is broken

Solution: Fix all issues

Patch 9.0.1819

Problem: Github CI too complex

Solution: CI: Tidy up matrix

Patch 9.0.1820

Problem: REXX files may not be recognised

Solution: Add shebang detection and improve disambiguation of *.cls files

Patch 9.0.1821

Problem: Vim9 constructors are always static

Solution: make the "static" keyword an error

Patch 9.0.1822

Problem: Vim9: no check for duplicate members in extended classes

Solution: Check for duplicate members in extended classes.
Fix memory leak.

Patch 9.0.1823

Problem: Autoconf 2.69 too old

Solution: Migrate to Autoconf 2.71

Patch 9.0.1824

Problem: Vim9: private members may be modifiable

Solution: prevent modification for def function

Patch 9.0.1825

Problem: Wrong cursor position with virtual text before a whitespace character and **'linebreak'**.

Solution: Always set "col_adj" to "size - 1" and apply **'linebreak'** after adding the size of **'breakindent'** and **'showbreak'**.

Patch 9.0.1826

Problem: keytrans() doesn't translate recorded key typed in a GUI

Solution: Handle CSI like K_SPECIAL, like in mb_unescape()

Patch 9.0.1827

Problem: xxd: no color support

Solution: Add color support using xxd -R

Patch 9.0.1828

Problem: Wrong cursor position with virtual text before double-width char at window edge.

Solution: Check for double-width char before adding virtual text size.

Patch 9.0.1829

Problem: Vim9 missing access-checks for private vars

Solution: Use the proper check for private/readonly variable. Access level for a member cannot be changed in a class implementing an interface. Update the code indentation

Patch 9.0.1830

Problem: Vim9: crash when accessing a null object

Solution: Check accessing a NULL object in def function

Patch 9.0.1831

Problem: Vim9: failing null test

Solution: Use required public keyword

Patch 9.0.1832

Problem: xxd: reporting wrong version (after 9.0.1827)

Solution: Update version string

Patch 9.0.1833

Problem: runtime files may execute code in current dir

Solution: only execute, if not run from current directory

Patch 9.0.1834

Problem: Some problems with xxd coloring

Solution: Fix the following problems:

Patch 9.0.1835

Problem: Perl interface has problems with load PL_current_context

Solution: Fix Perl interface to load PL_current_context from library

Patch 9.0.1836

Problem: Wrong display with "above" virtual text and '**linebreak**' or '**breakindent**' and '**showbreak**'.

Solution: Exclude size of "above" virtual text when calculating them.

Patch 9.0.1837

Problem: Vim9: class_member_type() can be optimized

Solution: class_member_type() provides more information;
safe an additional alloc()/free()

Patch 9.0.1838

Problem: Vim9: Cannot modify class member vars from def function

Solution: Add support for modifying class member variables from a def function

Patch 9.0.1839

Problem: No Makefile rule to build cscope database

Solution: Add rule

Patch 9.0.1840

Problem: use-after-free in do_ecmd

Solution: Verify oldwin pointer after reset_VIsual()

Patch 9.0.1841

Problem: style: trailing whitespace in ex_cmds.c

Solution: remove it

Patch 9.0.1842

Problem: Need more accurate profiling

Solution: Improve profiling results

Patch 9.0.1843

Problem: xxd color test flaky

Solution: Filter unneeded lines

Patch 9.0.1844

Problem: doc helptags may not be up to date

Solution: Add CI jobs to verify helptags are updated

Patch 9.0.1845

Problem: xxd: Test_xxd_color start failing

Solution: Revert changes to dump file

Patch 9.0.1846

Problem: crash in fullcommand
Solution: Check for typeval correctly

Patch 9.0.1847

Problem: potential oob write in do_addsub()
Solution: don't overflow buf2, check size in for loop()

Patch 9.0.1848

Problem: buffer-overflow in vim_regsub_both()
Solution: Check remaining space

Patch 9.0.1849

Problem: CI error on different signedness
Solution: cast unsigned to int

Patch 9.0.1850

Problem: Vim9: wrong line number where options set
Solution: Set source line number earlier

Patch 9.0.1851

Problem: Virtual text at a column causes **'breakindent'** and **'showbreak'** to be missing (after patch 9.0.1124).
Solution: Add check for "tp_col" in another place where TP_FLAG_WRAP is checked.

Patch 9.0.1852

Problem: i_CTRL-O does not reset Select Mode
Solution: Reset select mode on **CTRL-O** in insert mode

Patch 9.0.1853

Problem: CI error on different signedness in regexp.c (after patch 9.0.1848)
Solution: Cast strlen() call to int

Patch 9.0.1854

Problem: test_crash1() fails on CI
Solution: don't run Screendump test, verify that it doesn't crash by running it through a shell command line, testing the exit value and concatenating success cmd using '&&'

Patch 9.0.1855

Problem: mode() doesn't indicate command line for terminal
Solution: make it return **'ct'** for command-line from Terminal mode

Patch 9.0.1856

Problem: issues with formatting positional arguments
Solution: fix them, add tests and documentation

Patch 9.0.1857

Problem: heap-use-after-free in is_qf_win()
Solution: Check buffer is valid before accessing it

Patch 9.0.1858

Problem: heap use after free in ins_compl_get_exp()

Solution: validate buffer before accessing it

Patch 9.0.1859

Problem: heap-use-after-free in bt_normal()

Solution: check that buffer is still valid

Patch 9.0.1860

Problem: CI: test_crash1() is flaky

Solution: Wait a bit longer

Patch 9.0.1861

Problem: xxd: issue when -R is specified several times

Solution: Fix command line parsing

Patch 9.0.1862

Problem: Vim9 Garbage Collection issues

Solution: Class members are garbage collected early leading to use-after-free problems. Handle the garbage collection of classes properly.

Patch 9.0.1863

Problem: wrong format specifiers in e_aotypes_is_null_str_nr

Solution: Fix the wrong format specifier

Patch 9.0.1864

Problem: crash with bt_quickfix1_poc when cleaning up and EXITFREE is defined

Solution: Test if buffer is valid in a window, else close window directly, don't try to access buffer properties

Patch 9.0.1865

Problem: Vim9: garbage collection may cause crash

Solution: validate that class members typeval is not null

Patch 9.0.1866

Problem: Undo is synced after character find.

Solution: Set no_u_sync when calling gotchars_nop().

Patch 9.0.1867

Problem: Vim9: access to interface statics possible

Solution: Prevent direct access to interface statics

Patch 9.0.1868

Problem: test_crash still fails for circle ci

Solution: give even more time to complete

Patch 9.0.1869

Problem: Coverity warns about uninitialized var

Solution: initialize it

Patch 9.0.1870

Problem: Vim9: disassemble does not show static

Solution: Show static flag

Patch 9.0.1871

Problem: Github CI does not run i386 job
Solution: Add a i386 architecture

Patch 9.0.1872

Problem: CI: test_crash() fails on CI
Solution: Skip test on BSD

Patch 9.0.1873

Problem: heap-buffer-overflow in vim_regsub_both
Solution: Disallow exchanging windows when textlock is active

Patch 9.0.1874

Problem: CI may fail in test_recover_empty_swap
Solution: Set directory option

Patch 9.0.1875

Problem: Vim9: improve test for disassemble + static
Solution: Add a Vim9 script disassemble test for an interface with static members

Patch 9.0.1876

Problem: Vim9: parsing commands with newlines wrong
Solution: Accept a '\n' for parsing lists and command arguments

Patch 9.0.1877

Problem: missing test for patch 9.0.1873
Solution: add a test trying to exchange windows

Patch 9.0.1878

Problem: tests running sh have problems
Solution: Check that dash is installed

Patch 9.0.1879

Problem: Vim9: incorrect duplicate class member detection
Solution: Incorrect duplicate class member detection when variable names have the same prefix. Not able to access class member variables using an object. Fix coding style issues

Patch 9.0.1880

Problem: Vim9: Need more tests for inheritance
Solution: Add access tests and fixes.

Patch 9.0.1881

Problem: Test_crash fails on Mac
Solution: Skip test on Mac

Patch 9.0.1882

Problem: Trailing white space in tests
Solution: Delete it

Patch 9.0.1883

Problem: Vim9: Calling an interface method using a child object fails
Solution: Search methods of parent class

Patch 9.0.1884

Problem: Wrong order of arguments for error messages

Solution: Reverse order of arguments for e_atypes_is_null_nr_str

Patch 9.0.1885

Problem: Vim9: no support for abstract methods

Solution: Add support for defining abstract methods in an abstract class

Patch 9.0.1886

Problem: Various Typos

Solution: Fix Typos

Patch 9.0.1887

Problem: Vim9: class members are accessible via object

Solution: Disable class member variable access using an object

Patch 9.0.1888

Problem: Vim9: Problem trying to invoke class method

Solution: Lookup the class method inside other classes

Patch 9.0.1889

Problem: Vim9 static tests fail

Solution: Fix tests, make CI happy ;)

Patch 9.0.1890

Problem: Vim9: lookup code for class/object repeated

Solution: Refactor and make use of lookup functions

Patch 9.0.1891

Problem: No runtime support for Mojo

Solution: Add basic filetype and syntax plugins

Patch 9.0.1892

Problem: CI: no FreeBSD 14 support

Solution: Drop support for FreeBSD 12, add FreeBSD 14

Patch 9.0.1893

Problem: CI: strtptime test fails on BSD14

Solution: Skip the test

Patch 9.0.1894

Problem: CI: trailing white space in tests

Solution: clean up the trailing white space

Patch 9.0.1895

Problem: Vim9: finding method/member is inefficient

Solution: Use lookups

Patch 9.0.1896

Problem: "below" virtual text doesn't work with **'rightleft'**.

Solution: Use column from right border with **'rightleft'**.

Patch 9.0.1897

Problem: Vim9: confusing error with .= in compiled functions
Solution: Check in error condition, if .= was attempted and in that case give a different error message.

Patch 9.0.1898

Problem: Vim9: restrict access to static vars and methods
Solution: Class members are accessible only from the class where they are defined.

Patch 9.0.1899

Problem: potential buffer overflow in PBYTE macro
Solution: Check returned memline length

Patch 9.0.1900

Problem: Configure script uses non-portable == comparison
Solution: Use the standard and portable "=" instead

Patch 9.0.1901

Problem: win32: not correctly freeing environment
Solution: After we call GetEnvironmentStringsW, we should call FreeEnvironmentStringsW

Patch 9.0.1902

Problem: Vim9: Coverity complains about dead code
Solution: Copy only object methods from the super class to a subclass when extending a class. Fix Coverity warning.

Patch 9.0.1903

Problem: Github Actions fails because snd-dummy modules missing in current runner images
Solution: ignore modprobe error

Patch 9.0.1904

Problem: Cirrus-CI fails because we have used all credits
Solution: Remove FreeBSD 13.1 and MacOS M1

Patch 9.0.1905

Problem: FEAT_FLOAT no longer defined
Solution: Remove last existing FEAT_FLOAT ifdefs in message_test

Patch 9.0.1906

Problem: Vim9: Interfaces should not support class methods and variables
Solution: Make sure interface follow the interface specification

Patch 9.0.1907

Problem: No support for liquidsoap filetypes
Solution: Add liquidsoap filetype detection code

Patch 9.0.1908

Problem: undefined behaviour upper/lower function ptrs
Solution: Fix UBSAN error in regexp and simplify upper/lowercase

modifier code

Patch 9.0.1909

Problem: Vim9: problem calling class method from other class
Solution: Fix this problem, fix readonly object access, update error messages.

Patch 9.0.1910

Problem: Mac OS X: missing sound support on older versions
Solution: Check Macro MAC_OS_X_VERSION_MIN_REQUIRED

Patch 9.0.1911

Problem: Vim9: segfault with null object and instanceof()
Solution: return early

Patch 9.0.1912

Problem: Cirrus-CI running out of credits
Solution: disable Cirrus-CI for now

Patch 9.0.1913

Problem: if_python: undefined behaviour for function pointers
Solution: Fix if_python undefined behavior for function pointer casts

Patch 9.0.1914

Problem: Vim9: few issues when accessing object members
Solution: When calling an object method, check for null object.
Accessing a Dict object member doesn't work.

Patch 9.0.1915

Problem: r_CTRL-C works differently in visual mode
Solution: Make r_CTRL-C behave consistent in visual mode
in terminal and Windows GUI

Patch 9.0.1916

Problem: Crash when allocating large terminal screen
Solution: Don't allow values > 1000 for terminal
screen columns and rows

Patch 9.0.1917

Problem: undefined behaviour with python function pointer
Solution: correctly cast function pointers from void

Patch 9.0.1918

Problem: No filetype detection for Authzed filetypes
Solution: Detect the *.zed file extension as authzed filetype

Patch 9.0.1919

Problem: Wrong curswant when clicking on empty line or with vsplits.
Solution: Don't check for ScreenCols[] before the start of the window
and handle empty line properly.

Patch 9.0.1920

Problem: Vim9: cannot write public var in nested object
Solution: Write variable in nested read-only object reference.

Also test write fails.

Patch 9.0.1921

Problem: not possible to use the jumplist like a stack

Solution: Add the '**jumpoptions**' setting to make the jumplist a stack.

Patch 9.0.1922

Problem: LSP server request message is misinterpreted as a response message

Solution: Check that the message does not have the "message" field

Patch 9.0.1923

Problem: curswant wrong on click with '**ve**' and '**wrap**' set

Solution: Add w_leftcol to mouse click column.

Patch 9.0.1924

Problem: LSP server message still wrongly handled (after 9.0.1922)

Solution: Handle '**method**' messages properly, don't discard them, add tests.

Patch 9.0.1925

Problem: if_python: still undefined behaviour with function pointer

Solution: fix remaining problems

Patch 9.0.1926

Problem: Vim9: not enough info in error message

Solution: Add class name, change member to variable, quote names

Patch 9.0.1927

Problem: patch 1916 (fixed terminal size) not optimal

Solution: Add defines to make it easier changeable later

Patch 9.0.1928

Problem: Vim9: constructor type checking bug

Solution: Fix class constructor regression

Patch 9.0.1929

Problem: runtime tests fail with tiny vim

Solution: check for tiny vim, run runtime tests in CI even for tiny version

Patch 9.0.1930

Problem: compiler warnings with clang-17

Solution: Fix function prototypes and function pointer

Patch 9.0.1931

Problem: make test_compilers fails on ubuntu

Solution: set LC_ALL=C

Patch 9.0.1932

Problem: Vim9: error when using null object constructor

Solution: Check for a null object only when calling an object method

Patch 9.0.1933

Problem: Can change the type of a v: variable using if_lua.
Solution: Add additional handling of v: variables like :let.

Patch 9.0.1934

Problem: bwipe fails after switching window from aucmd_win.
Solution: Decrement b_nwindows after switching back to aucmd_win.

Patch 9.0.1935

Problem: Vim9: not consistent error messages
Solution: Make error messages more consistent. Use "variable" for (object/class) member

Patch 9.0.1936

Problem: test: using wrong expected message in test_crypt
Solution: make use of single quotes

Patch 9.0.1937

Problem: missing test for mouse click + 'virtedit'
Solution: Add test for clicking after eol with 'virtualedit' and wrapped line

Patch 9.0.1938

Problem: multispace wrong when scrolling horizontally
Solution: Update position in "multispace" or "leadmultispace" also in skipped chars. Reorder conditions to be more consistent.

Patch 9.0.1939

Problem: still a problem when processing LSP RPC requests
Solution: When processing async LSP RPC requests, compare sequence numbers only in response messages

Patch 9.0.1940

Problem: wrong upstream version in libvterm README
Solution: correct version to 839

Patch 9.0.1941

Problem: Memory leak detected (after 9.0.1928)
Solution: Free arg_objm in get_lambda_tv()

Patch 9.0.1942

Problem: Vim9: execution stack invalidated with null object
Solution: Check for a null object before adjusting the execution stack

Patch 9.0.1943

Problem: CI not run with clang-17
Solution: Update CI to use clang-17

Patch 9.0.1944

Problem: Vim9: function instruction pointer invalidated
Solution: Use the funcref index instead of the instruction pointer

Patch 9.0.1945

Problem: Vim9: missing support for ro-vars in interface
Solution: Support only read-only object variables in an interface,

add additional checks when parsing class definitions.

Patch 9.0.1946

Problem: filename expansion using ** in bash may fail

Solution: Try to enable the globstar setting

Patch 9.0.1947

Problem: Bash Expansion test fails on Windows/MacOS

Solution: Disable Test_glob_extended_bash for now

Patch 9.0.1948

Problem: Vim9: object variable "this." should only be used in constructor

Solution: Disallow to this in normal object methods (other than constructors)

Patch 9.0.1949

Problem: Vim9: allows reserved keywords as members

Solution: Disallow reserved keywords, disallow duplicate object and class variables

Patch 9.0.1950

Problem: Vim9: error codes spread out

Solution: group them together and reserve 100 more for future use

Patch 9.0.1951

Problem: Vim9: hard to debug vim9_class errors from CI

Solution: Include the line number in assert_xxx() calls. Include the entire error message in the tests. Fix the indentation in the test file. Add tags for new error codes.

Patch 9.0.1952

Problem: Vim9: unused static field

Solution: remove it and simplify code

Patch 9.0.1953

Problem: Misplaced comment in errors.h

Solution: Move it up

Patch 9.0.1954

Problem: CI: change netrw label in labeller bot

Solution: Rename it to 'plugin-netrw'

Patch 9.0.1955

Problem: Vim9: lockvar issues with objects/classes

Solution: fix `get_lhs()` object/class access and avoid `SEGV`, make error messages more accurate.

Patch 9.0.1956

Problem: Custom cmdline completion skips original cmdline when pressing Ctrl-P at first match if completion function invokes glob().

Solution: Move orig_save into struct expand_T.

Patch 9.0.1957

Problem: termcap options should change on keyprotocol setting
Solution: Apply termcap entries when **'keyprotocol'** changes

Patch 9.0.1958

Problem: cannot complete option values
Solution: Add completion functions for several options

Patch 9.0.1959

Problem: Vim9: methods parameters and types are covariant
Solution: Support contra-variant type check for object method arguments (similar to Dart).

Patch 9.0.1960

Problem: Make CI checks more strict
Solution: Add `-Wstrict-prototypes` `-Wmissing-prototypes` to CI, fix uncovered problems

Patch 9.0.1961

Problem: Cmdline completion for **'listchars'** fields doesn't include "multispace" and "leadmultispace" (after 9.0.1958).
Solution: Include "multispace" and "leadmultispace" in `lcstab`.

Patch 9.0.1962

Problem: No support for writing extended attributes
Solution: Add extended attribute support for linux

Patch 9.0.1963

Problem: Configure script may not detect `xattr` correctly
Solution: include `sys/xattr` instead of `attr/xattr`, make `Test_write_with_xattr_support()` test `xattr` feature correctly

Patch 9.0.1964

Problem: `xattr` support fails to build on MacOS X
Solution: Disable `xattr` support for MacOS X

Patch 9.0.1965

Problem: wrong `auto/configure` script
Solution: regenerate with `autoconf 2.71`

Patch 9.0.1966

Problem: `configure` prints stray 6 when checking `libruby`
Solution: redirect `stdout` to `dev/null`

Patch 9.0.1967

Problem: `xattr` errors not translated
Solution: mark for translation, consistently capitalize first letter.

Patch 9.0.1968

Problem: cmdline completion should consider key option
Solution: Disable cmdline completion for key option, slightly refactor how `P_NO_CMD_EXPAND` is handled

Patch 9.0.1969

Problem: buffer-overflow in trunc_string()

Solution: Add NULL at end of buffer

Patch 9.0.1970

Problem: win32: high-dpi support can be improved

Solution: implement WM_GETDPISCALEDSIZE

Patch 9.0.1971

Problem: macOS: FEAT_SOUND guard too restrictive

Solution: check for older macOS support properly

Patch 9.0.1972

Problem: win32: missing '**' expansion test (after v9.0.1947)

Solution: Add test for MS-Windows

Patch 9.0.1973

Problem: Clean up cmdline option completion code

Solution: Fix various minor problems

Patch 9.0.1974

Problem: vim9: using contra-variant type-checks (after v9.0.1959)

Solution: Use invariant type checking instead

Patch 9.0.1975

Problem: xattr: permission-denied errors on write

Solution: ignore those errors

Patch 9.0.1976

Problem: style: space before tab in optionstr.c

Solution: remove the space

Patch 9.0.1977

Problem: Vim9: object members can change type

Solution: Check type during assignment to object/class var

Patch 9.0.1978

Problem: No filetype detection for just files

Solution: Detect just files (*.just, justfile, etc)

Patch 9.0.1979

Problem: Cirrus CI disabled

Solution: re-enable Cirrus CI

Patch 9.0.1980

Problem: win32: issues with stable python ABI

Solution: if_python3,win32: Fix Python3 stable ABI

Patch 9.0.1981

Problem: Cannot scroll up in diff mode with many filler lines and zero
'scrolloff'.

Solution: Invalidate w_cline_row before calling comp_botline().

Patch 9.0.1982

Problem: vim9: clean up from v9.0.1955

Solution: Fix a few remaining issues, improve error message

Patch 9.0.1983

Problem: Scrolling non-current window using mouse is inconsistent depending on '**scrollbind**'/'**scrolloff**' and different from GUI vertical scrollbar when '**cursorbind**' is set.

Solution: Don't move cursor in non-current windows for '**cursorbind**' if cursor in the current window didn't move.

Patch 9.0.1984

Problem: CI: Test_open_delay*() fails on FreeBSD 14

Solution: Skip it on BSD

Patch 9.0.1985

Problem: CI: codecov is intrusive

Solution: disable codecov comments

Patch 9.0.1986

Problem: Vim9: accepting type-annotations

Solution: Reject type annotations outside of declarations.

Patch 9.0.1987

Problem: win32: font-size calculation can be improved

Solution: calculate font size before the window size

Patch 9.0.1988

Problem: Vim9: potential use-after-free for class members

Solution: Use the class-related grow array for storing the member type instead of using a temporary type list grow array

Patch 9.0.1989

Problem: Vim9: double error message given

Solution: Only give second error message, if there wasn't one given before

Patch 9.0.1990

Problem: strange error number

Solution: change error number, add doc tag for E1507

Patch 9.0.1991

Problem: no cmdline completion for setting the font

Solution: enable it on Win32 and GTK builds

Patch 9.0.1992

Problem: segfault in exmode when redrawing

Solution: skip gui_scroll when exmode_active

Patch 9.0.1993

Problem: warning about unused function definition

Solution: add ifdefs

Patch 9.0.1994

Problem: inconsistent feature description

Solution: delete old mentioned feature sets small and big

Patch 9.0.1995

Problem: Invalid memory access when **'foldexpr'** returns empty string.

Solution: Check for NUL.

Patch 9.0.1996

Problem: Cannot build with python312

Solution: Define wrapper types and functions for python 3.12

Patch 9.0.1997

Problem: Some unused code in move.c and string.c

Solution: Remove it

Patch 9.0.1998

Problem: xxd: cannot reverse a bit dump

Solution: implement reversing the bit dump using `-b -r`

Patch 9.0.1999

Problem: Vim9: some error messages can be improved

Solution: Mention the defining class for variable access error message

Patch 9.0.2000

Problem: Vim9: use-after-free in deep call stack

Solution: Get the object pointer from execution stack

Patch 9.0.2001

Problem: Vim9: segfault with `islocked()`

Solution: Check that the lval pointer is not null for objects and class variables

Patch 9.0.2002

Problem: Vim9: need cleanup of class related interface code

Solution: Remove the unused class variable and class method related code for interfaces.

Patch 9.0.2003

Problem: xxd: compilation warning

Solution: initialize variables

Patch 9.0.2004

Problem: Missing test file

Solution: git-add the file to the repo

Patch 9.0.2005

Problem: partially revert patch v9.0.1997

Solution: add a comment, to make clear it's not used

Patch 9.0.2006

Problem: Vim9: need more tests

Solution: add additional disassembly tests

Patch 9.0.2007

Problem: Vim9: covariant parameter types allowed when assigning functions

Solution: Enforce invariant type check for arguments and return value when assigning a funcref

Patch 9.0.2008

Problem: test: undofile left behind

Solution: cleanup undofile

Patch 9.0.2009

Problem: cmdline-completion for comma-separated options wrong

Solution: Fix command-line expansions for options with filenames with commas

Patch 9.0.2010

Problem: [security] use-after-free from buf_contents_changed()

Solution: block autocommands

Patch 9.0.2011

Problem: INI files not detected

Solution: detect uppercase .INI as dosini files

Patch 9.0.2012

Problem: Vim9: error message can be more accurate

Solution: Fix the error messages

Patch 9.0.2013

Problem: Unicode tables outdated

Solution: Update Unicode tables to v15.1 (released 23.09.2023)

Patch 9.0.2014

Problem: confusing ifdefs in if_<lang>.c

Solution: refactor ifndefs to #ifdefs

Patch 9.0.2015

Problem: Vim9: does not handle islocked() from a method correctly

Solution: Handle islocked() builtin from a method.

Patch 9.0.2016

Problem: Vim9: assignment operators don't work for class vars

Solution: implement it

Patch 9.0.2017

Problem: linebreak applies for leading whitespace

Solution: only apply linebreak, once we have found non-breakat chars in the line

Patch 9.0.2018

Problem: complete_info() returns wrong index

Solution: Make order of 'info' in completion_info consistent

Patch 9.0.2019

Problem: Vim9: no support for funcrefs
Solution: Add support for object/class funcref members

Patch 9.0.2020

Problem: Vim9: islocked() needs more work
Solution: rework islocked() and remove sync_root
from get_lval()

Patch 9.0.2021

Problem: Coverity complains about change in charset (after v9.0.2017)
Solution: check pointer t at index 0

Patch 9.0.2022

Problem: When clicking in the middle of a TAB, getmousepos() returns
the column of the next char instead of the TAB.
Solution: Break out of the loop when the vcol to find is inside current
char. Fix invalid memory access when calling virtcol2col() on
an empty line.

Patch 9.0.2023

Problem: need more tests for :cq
Solution: Add more tests, including wraparound on linux

Patch 9.0.2024

Problem: no filetype detection for Debian sources
Solution: Add new deb822sources filetype

Patch 9.0.2025

Problem: no cmdline completion for ++opt args
Solution: Add cmdline completion for :e ++opt=arg and :terminal
[++options]

Patch 9.0.2026

Problem: win32: python3 dll loading can be improved
Solution: Load DLL from registry path

Patch 9.0.2027

Problem: Vim9: no support for bitwise operators in lambda funcs
Solution: move "evaluate" assignment a bit up in order to decide
to perform bitwise operations

Patch 9.0.2028

Problem: confusing build dependencies
Solution: clean them up, make them parallelizable

Patch 9.0.2029

Problem: Vim9: no support for partials using call()
Solution: Add support

Patch 9.0.2030

Problem: no max callback recursion limit
Solution: bail out, if max call recursion for callback functions
has been reached.

Patch 9.0.2031

Problem: ``TextChangedI`` can trigger on entering Insert mode if there was previously a change not in Insert mode.

Solution: Make it trigger only when text is actually changed in Insert mode.

Patch 9.0.2032

Problem: Cannot accurately get mouse clicking position when clicking on a TAB or with virtual text.

Solution: Add a "coladd" field to getmousepos() result.

Patch 9.0.2033

Problem: gcc overflow-warning for f_resolve

Solution: use pointer p instead of pointer q[-1]

Patch 9.0.2034

Problem: don't try to copy SMACK attribute, when none exist

Solution: return early if SMACK extended attributes do not exist or if they are not supported

Patch 9.0.2035

Problem: [security] use-after-free with wildmenu

Solution: properly clean up the wildmenu when exiting

Patch 9.0.2036

Problem: if_python: rework python3.12 build dependency (after 9.0.1996)

Solution: use PyTuple_Size instead of inlining the Py_SIZE into the Vim code base

Patch 9.0.2037

Problem: A few remaining cmdline completion issues with C-E/Y

Solution: Fix cmdline completion fuzzy/Ctrl-E/Ctrl-Y/options when not used at the end

Patch 9.0.2038

Problem: Vim9: object method funcref not cleaned up after use

Solution: Clean up type stack after using object method funcref, remove now longer used ISN_DEFEROBJ instruction

Patch 9.0.2039

Problem: completion shows current word after completion restart

Solution: remove the word being completed after completion restart

Patch 9.0.2040

Problem: trim(): hard to use default mask

Solution: Use default 'mask' when it is v:none

Patch 9.0.2041

Problem: trim(): hard to use default mask (partly revert v9.0.2040)

Solution: use default mask when it is empty

Patch 9.0.2042

Problem: Test_cq_zero_exmode fails without channel feature

Solution: Make the test check the channel feature

Patch 9.0.2043

Problem: Vim9: issue with funcref assignment and varargs

Solution: Fix funcref type checking

Patch 9.0.2044

Problem: Vim9: exceptions confuse deferred functions

Solution: save and restore exception state when calling deferred functions

Patch 9.0.2045

Problem: tests: checking for swap files takes time

Solution: don't check for swap files when test has been skipped

Patch 9.0.2046

Problem: win32,python: warning that MS_WIN64 got re-defined

Solution: Do not define MS_WIN64, as it is no longer used

Patch 9.0.2047

Problem: perl: warning about inconsistent dll linkage

Solution: suppress warning

Patch 9.0.2048

Problem: python: uninitialized warning

Solution: initialize **'minor'** always

Patch 9.0.2049

Problem: Vim9: not recognizing qualified class vars for infix ops

Solution: Drop the class type from the stack before generating the CLASSMEMBER instruction

Patch 9.0.2050

Problem: Vim9: crash with deferred function call and exception

Solution: Save and restore exception state

Patch 9.0.2051

Problem: Vim9: wrong error for non-existing object var

Solution: mention object or class depending on whether the var is an object or class variable.

Patch 9.0.2052

Problem: win32: using deprecated wsock32 api

Solution: Use winsock2 (ws2_32) consistently

Patch 9.0.2053

Problem: zig filetype detection test wrong

Solution: Remove .zir pattern, add new test for .zon pattern

Patch 9.0.2054

Problem: win32: iscygpty needs update

Solution: Update iscygpty to the latest version, make use iswascii() API function

Patch 9.0.2055

Problem: Vim9: non-consistent error messages

Solution: make error messages more consistent with common structure

Patch 9.0.2056

Problem: no digraph for quadruple prime

Solution: add quadruple prime digraph using 4'

Patch 9.0.2057

Problem: Vim9: no strict type checks for funcrefs varargs

Solution: Perform strict type checking when declaring funcrefs with vararg declaration, add tests

Patch 9.0.2058

Problem: tests: avoid error when no swap files exist

Solution: use unlet! so that no error message is reported in case the variable does not exists

Patch 9.0.2059

Problem: outstanding exceptions may be skipped

Solution: When restoring exception state, process remaining outstanding exceptions

Patch 9.0.2060

Problem: *.{gn,gni} files are not recognized

Solution: Detect some as gn filetype (without adding an extra filetype)

Patch 9.0.2061

Problem: not able to detect xkb filetypes

Solution: Detect files below /u/s/X11/xkb as xkb files (without adding an extra filetype)

Patch 9.0.2062

Problem: Janet files are not recognised

Solution: Add filename and shebang detection (without adding an extra filetype plugin)

Patch 9.0.2063

Problem: pacman hooks are detected as conf filetype

Solution: make it consistent to pacman.conf and detect those hooks as confini

Patch 9.0.2064

Problem: cannot use buffer-number for errorformat

Solution: add support for parsing a buffer number using '%b' in **'errorformat'**

Patch 9.0.2065

Problem: EXPAND flag set for filetype option

Solution: Remove P_EXPAND flag from the **'filetype'** option

Patch 9.0.2066

Problem: xxd: corrupting files when reversing bit dumps

Solution: handle reversing bit dump slightly differently

Patch 9.0.2067

Problem: xxd: coloring was disabled on Cygwin
Solution: don't include WIN32

Patch 9.0.2068

Problem: [security] overflow in :history
Solution: Check that value fits into int

Patch 9.0.2069

Problem: possible to escape bracketed paste mode with Ctrl-C
Solution: Do not handle Ctrl-C specially when key_protocol
is in use, makes bracketed paste mode more robust

Patch 9.0.2070

Problem: [security] disallow setting env in restricted mode
Solution: Setting environment variables in restricted mode could
potentially be used to execute shell commands. Disallow this.

Patch 9.0.2071

Problem: objdump files not recognized
Solution: detect *.objdump files, add a filetype plugin

Patch 9.0.2072

Problem: Vim9: no nr2str conversion in list-unpack
Solution: Generate 2STRING instruction to convert dict index to string

Patch 9.0.2073

Problem: typo in quickfix.c comments
Solution: fix them

Patch 9.0.2074

Problem: Completion menu may be wrong
Solution: Check for the original direction of the completion menu,
add more tests, make it work with 'noselect'

Patch 9.0.2075

Problem: TextChangedI may not always trigger
Solution: trigger it in more cases: for insert/
append/change operations, and when
opening a new line,

Patch 9.0.2076

Problem: Vim9: No support for type aliases
Solution: Implement :type command

Patch 9.0.2077

Problem: CI fails because of trailing whitespace in test
Solution: Remove it

Patch 9.0.2078

Problem: several problems with type aliases
Solution: Check for more error conditions, add tests,
fix issues

Patch 9.0.2079

Problem: Not all Dart files detected
Solution: Add shebang filetype detection for Dart

Patch 9.0.2080

Problem: vim9_script test too large
Solution: split vim9 type alias test into separate test file

Patch 9.0.2081

Problem: With **'smoothscroll'** set, "w_skipcol" is not reset when unsetting **'wrap'**. Resulting in incorrect calculation of the cursor position.
Solution: Reset "w_skipcol" when unsetting **'wrap'**.

Patch 9.0.2082

Problem: test_channel may fail because of IPv6 config issues
Solution: Catch and skip the test, if getaddrinfo() fails with 'Address family not supported'

Patch 9.0.2083

Problem: Perl: xsubpp may be in non-standard location
Solution: Add --with-subpp configure option

Patch 9.0.2084

Problem: Vim9: abstract static methods are possible
Solution: Disallow abstract static methods

Patch 9.0.2085

Problem: Vim9: abstract can be used in interface
Solution: Disallow the use of abstract in an interface

Patch 9.0.2086

Problem: code cleanup for option callbacks needed
Solution: remove flag os_doskip, it's not necessary, as we can check, whether an error message was returned

Patch 9.0.2087

Problem: build-failure in vim9class
Solution: reference correct error message, disable non-failing test

Patch 9.0.2088

Problem: Vim9: still allows abstract static methods (after v9.0.2084, v9.0.2085 and v9.0.2087)
Solution: Disallow abstract static methods

Patch 9.0.2089

Problem: sound_playfile() fails when using powershell
Solution: quote filename using doublequotes, don't escape filename, because it doesn't use the shell

Patch 9.0.2090

Problem: complete_info() skips entries with **'noselect'**

Solution: Check, if first entry is at original text state

Patch 9.0.2091

Problem: Vim9: cannot convert list to string using +=
(after 9.0.2072)

Solution: convert dict index to string later in compile_member()

Patch 9.0.2092

Problem: tests: failure in test_arabic

Solution: adjust the test for the changed arabic keymap

Patch 9.0.2093

Problem: Unsupported option causes rest of modeline test to be skipped.

Solution: Revert the change from patch 8.2.1432.

Patch 9.0.2094

Problem: Vim9: need more assignment tests

Solution: Add test for using different types in assignment, function arguments and return values

Patch 9.0.2095

Problem: statusline may look different than expected

Solution: do not check for highlighting of stl and stlnc characters

Patch 9.0.2096

Problem: Vim9: confusing usage of private

Solution: clarify and use protected keyword instead

Patch 9.0.2097

Problem: No support for cypher files

Solution: Add cypher filetype detection

Patch 9.0.2098

Problem: No filetype support for xcompose files

Solution: Add filetype detection

Patch 9.0.2099

Problem: Terminal control codes' are sent even when silent mode is on, causing the terminal to clear up

Solution: Block any terminal codes when silent mode is on

Patch 9.0.2100

Problem: CI: test_Termdebug fails

Solution: only test for a changed winlayout, if the window width actually changed

Patch 9.0.2101

Problem: CI: test_termdebug may still fail

Solution: use term_wait() to make it more robust

Patch 9.0.2102

Problem: matchparen highlight not cleared in completion mode

Solution: Clear matchparen highlighting in completion mode

Patch 9.0.2103

Problem: recursive callback may cause issues on some archs
Solution: Decrease the limit drastically to 20

Patch 9.0.2104

Problem: wast filetype should be replaced by wat filetype
Solution: start using the official wat filetype name

Patch 9.0.2105

Problem: Skipcol is not reset when topline changed scrolling cursor to top
Solution: reset skipcol

Patch 9.0.2106

Problem: [security]: Use-after-free in win_close()
Solution: Check window is valid, before accessing it

Patch 9.0.2107

Problem: [security]: FPE in adjust_plines_for_skipcol
Solution: don't divide by zero, return zero

Patch 9.0.2108

Problem: [security]: overflow with count for :s command
Solution: Abort the :s command if the count is too large

Patch 9.0.2109

Problem: [security]: overflow in nv_z_get_count
Solution: break out, if count is too large

Patch 9.0.2110

Problem: [security]: overflow in ex address parsing
Solution: Verify that lnum is positive, before subtracting from
LONG_MAX

Patch 9.0.2111

Problem: [security]: overflow in get_number
Solution: Return 0 when the count gets too large

Patch 9.0.2112

Problem: [security]: overflow in shift_line
Solution: allow a max indent of INT_MAX

Patch 9.0.2113

Problem: Coverity warns for another overflow in shift_line()
Solution: Test for INT_MAX after the if condition, cast integer values
to (long long) before multiplying.

Patch 9.0.2114

Problem: overflow detection not accurate when adding digits
Solution: Use a helper function

Patch 9.0.2115

Problem: crash when callback function aborts because of recursiveness
Solution: correctly initialize rettv

Patch 9.0.2116

Problem: No test for defining sign without attribute
Solution: Add test for defining sign without attributes

Patch 9.0.2117

Problem: [security] use-after-free in qf_free_items
Solution: only access qfpnext, if it hasn't been freed

Patch 9.0.2118

Problem: [security]: avoid double-free
Solution: Only free plain_font, when it is not the same as bold_font

Patch 9.0.2119

Problem: remove dead-condition in ex_class()
Solution: remove the extra condition

Patch 9.0.2120

Problem: un-used assignment in do_source_buffer_init
Solution: Remove it

Patch 9.0.2121

Problem: [security]: use-after-free in ex_substitute
Solution: always allocate memory

Patch 9.0.2122

Problem: [security]: prevent overflow in indenting
Solution: use long long and remove cast to (int)

Patch 9.0.2123

Problem: Problem with initializing the length of range() lists
Solution: Set length explicitly when it shouldn't contain any items

Patch 9.0.2124

Problem: INT overflow logic can be simplified
Solution: introduce trim_to_int() function

Patch 9.0.2125

Problem: File info disappears immediately when 'cmdheight' has just decreased due to switching tabpage and 'shortmess' doesn't contain 'o' or 'O'.
Solution: Make sure msg_row isn't smaller than cmdline_row.

Patch 9.0.2126

Problem: Unused assignments when checking the value of 'listchars'.
Solution: Loop only once when just checking the value. Add a test to check that this change doesn't cause double-free.

Patch 9.0.2127

Problem: translation Makefiles can be improved
Solution: Modified and extended po-related Makefiles and related files

Patch 9.0.2128

Problem: No runtime files for SWIG filetypes

Solution: Add syntax and filetype plugins for SWIG (Simplified Wrapper Interface Generator) description files.

Patch 9.0.2129

Problem: [\[security\]](#): use-after-free in call_dfunc()

Solution: Refresh dfunc pointer

Patch 9.0.2130

Problem: some errors with translation Makefiles

Solution: fix issues

Patch 9.0.2131

Problem: not all nushell files detected

Solution: use *.nu to detect nushell files

Patch 9.0.2132

Problem: Duplicate Netbeans Error Message

Solution: Remove duplicate message

Patch 9.0.2133

Problem: Cannot detect overstrike mode in Cmdline mode

Solution: Make mode() return "cr" for overstrike

Patch 9.0.2134

Problem: ml_get error when scrolling after delete

Solution: mark topline to be validated in main_loop
if it is larger than current buffers line
count

Patch 9.0.2135

Problem: No test for mode() when executing Ex commands

Solution: Add some test cases and simplify several other test cases.
Also add a few more test cases for ModeChanged.

Patch 9.0.2136

Problem: MSVC errorformat can be improved

Solution: parse error type and column number in MSVC errorformat

Patch 9.0.2137

Problem: Can't detect angular & mustache filetypes

Solution: Detect *.mustache as Mustache filetype;
detect *.component.html as html.angular filetype

Patch 9.0.2138

Problem: Overflow logic requires long long

Solution: Define vimlong_T data type to make life easier
for porters

Patch 9.0.2139

Problem: html.angular ft is problematic

Solution: partly revert v9.0.2137

Patch 9.0.2140

Problem: [\[security\]](#): use-after-free in win-enter

Solution: validate window pointer before calling win_enter()

Patch 9.0.2141

Problem: [security]: buffer-overflow in suggest_trie_walk

Solution: Check n before using it as index into bytes array

Patch 9.0.2142

Problem: [security]: stack-buffer-overflow in option callback functions

Solution: pass size of errbuf down the call stack, use snprintf()
instead of sprintf()

Patch 9.0.2143

Problem: [security]: buffer-overflow in ex_substitute

Solution: clear memory after allocating

Patch 9.0.2144

Problem: Text properties causes wrong line wrapping to be drawn.

Solution: Find the index of the last text property that inserts text.

Patch 9.0.2145

Problem: Wrong scrolling in Insert mode with 'smoothscroll' at the
bottom of the window.

Solution: Don't use set_topleft() when 'smoothscroll' is set.

Patch 9.0.2146

Problem: text-property without type errors when joining

Solution: count all text-properties, with or without type
before joining lines

Patch 9.0.2147

Problem: Type check tests fail without the channel feature

Solution: only run tests, when Vim was build with +channel

Patch 9.0.2148

Problem: Vim does not detect pacman.log file

Solution: Detect pacmanlogs and add syntax highlighting

Patch 9.0.2149

Problem: [security]: use-after-free in exec_instructions()

Solution: get tv pointer again

Patch 9.0.2150

Problem: Using int for errbuflen in option funcs

Solution: Use size_t for errbuflen in string option functions

Patch 9.0.2151

Problem: 'breakindent' is not drawn after diff filler lines.

Solution: Correct check for whether 'breakindent' should be drawn.

Patch 9.0.2152

Problem: Using type unknown for List/Dict containers

Solution: Use 'any' instead

Patch 9.0.2153

Problem: no support to build on OpenVMS
Solution: Add OpenVMS X86_64 platform port

Patch 9.0.2154

Problem: The options[] array is not sorted alphabetically.
Solution: Sort it alphabetically. Add a test. Avoid unnecessary loop iterations in findoption().

Patch 9.0.2155

Problem: Vim9: type not kept when assigning vars
Solution: When assigning a List or a Dict value to a variable of type **'any'**, keep the type

Patch 9.0.2156

Problem: Vim9: can use typealias in an assignment
Solution: Generate errors when class/typealias involved in the rhs of an assignment

Patch 9.0.2157

Problem: Vim9: incorrectly parses :def func definitions
Solution: check for more context when parsing function args

Patch 9.0.2158

Problem: **[security]**: use-after-free in check_argument_type
Solution: Reset function type pointer when freeing the function type list

Patch 9.0.2159

Problem: screenpos() may crash with neg. column
Solution: validate and correct column

Patch 9.0.2160

Problem: instanceof() should use varargs as second arg
Solution: Modify **instanceof()** to use varargs instead of list

Patch 9.0.2161

Problem: Vim9: not able to use imported interfaces and classes
Solution: Detect imported class/interfaces names correctly

Patch 9.0.2162

Problem: Vim9: type documentation out-dated
Solution: Update documentation, fix typo in type alias definition

Patch 9.0.2163

Problem: Vim9: type can be assigned to list/dict
Solution: Prevent assigning a **'type'** to a **'list'** or **'dict'**

Patch 9.0.2164

Problem: Vim9: can use type a func arg/return value
Solution: Check if using type as function argument or return value

Patch 9.0.2165

Problem: Vim9: can simplify arg type checking code

Solution: In ``f_argcheck`` array use ``arg_any``, instead of NULL

Patch 9.0.2166

Problem: Memory leak in Configure Script when checking GTK

Solution: Free the allocated memory

Patch 9.0.2167

Problem: Vim9-script object/class variable declarations use syntax that is inconsistent with the rest of the language.

Solution: Use `:var` to declare object and class variables.

Patch 9.0.2168

Problem: Moving tabpages on `:drop` may cause an endless loop

Solution: Disallow moving tabpages on `:drop` when cleaning up the arglist first

Patch 9.0.2169

Problem: Vim9: builtin funcs may accept a non-value

Solution: Restrict builtin functions that accept ``type``

Patch 9.0.2170

Problem: Vim9: no support for const/final class/objects vars

Solution: Support final and const object and class variables

Patch 9.0.2171

Problem: The `options[]` array is still not sorted alphabetically (after: v9.0.2154), causing test failures

Solution: Sort the remaining items

Patch 9.0.2172

Problem: Vim9: compiling `:defer` may fail

Solution: compile defer, when `ctx_skip` is not `SKIP_YES`

Patch 9.0.2173

Problem: Vim9: Vim crashes when compiling a for statement with a non-existing type

Solution: Error out when `lhs_type` is not null

Patch 9.0.2174

Problem: Vim9: segfault when assigning to type

Solution: do not clear `typeval`, add missing patch number

Patch 9.0.2175

Problem: Compiler warning for uninitialized var

Solution: initialize variable to NULL

Patch 9.0.2176

Problem: Compile error with Motif UI + mouse support (after v9.0.1262)

Solution: Use correct `oldval` option pointer

Patch 9.0.2177

Problem: Wrong cursor position when dragging out of window.

Solution: Don't use `ScreenCols[]` when mouse is not in current window.

Patch 9.0.2178

Problem: reg_executing() returns wrong result in :normal with range when '**showcmd**' is set (after 8.2.4705).

Solution: Reset "pending_end_reg_executing" when executing a register.

Patch 9.0.2179

Problem: no filetype detection for execline scripts

Solution: Add filetype detection for execline

Patch 9.0.2180

Problem: POSIX function name in exarg struct causes issues on OpenVMS

Solution: Rename getline member in exarg struct to ea_getline, remove isinf() workaround for VMS

Patch 9.0.2181

Problem: Vim9: missing error messages

Solution: Add one more error message

Patch 9.0.2182

Problem: Vim9: need a way to reserve future extension

Solution: reserve double underscore prefix for future use (Yegappan Lakshmanan)

Patch 9.0.2183

Problem: Maximum callback depth is not configurable.

Solution: Revert patch 9.0.2103. Set '**maxfuncdepth**' in test. (zeertzjq)

Patch 9.0.2184

Problem: Vim9: inconsistent :type/:class messages

Solution: Update the Messages (Ernie Rael)

Patch 9.0.2185

Problem: Coverity complains about not checking return value in compare_isn_not_values (after 9.0.2184)

Solution: cast return value to "(void)" to make intention clear

Patch 9.0.2186

Problem: LTCG compile error on Win/ARM64 for ``write_chars()``

Solution: Explicitly initialise the storage to use data rather than BSS (Saleem Abdulrasool)

Patch 9.0.2187

Problem: Visual selection isn't drawn with '**breakindent**' when the line doesn't fit in the window (Jaehwang Jung)

Solution: Adjust wlv->fromcol also for '**breakindent**' (zeertzjq)

Patch 9.0.2188

Problem: cursor wrong after { in single line buffer (Edwin Chan)

Solution: do not place the cursor at the end for a single line buffer when moving backwards (Gary Johnson)

Patch 9.0.2189

Problem: Wrong display when **'breakindentopt'** contains "sbr" and **'showbreak'** and **'nobreakindent'** are set.

Solution: Always reset wlv->need_showbreak regardless of the values of **'breakindent'** and **'showbreak'**, as they aren't checked when setting wlv->need_showbreak (zeertzjq)

Patch 9.0.2190

Problem: proto files need update

Solution: re-generate them

VERSION 9.2

[version-9.2](#) [version9.2](#) [vim-9.2](#)

This section is about improvements made between version 9.1 and 9.2 and is a work in progress.

Support for Wayland UI.

Vim9 script

Add support for internal builtin functions with vim9 objects, see [builtin-object-methods](#)

Other improvements

[new-other-9.2](#)

Changed

[changed-9.2](#)

Added

[added-9.2](#)

Various syntax, indent and other plugins were added.

Functions:

<code>diff()</code>	diff two Lists of strings
<code>foreach()</code>	apply function to List items
<code>matchbuflines()</code>	all the matches of a pattern in a buffer
<code>matchstrlist()</code>	all the matches of a pattern in a List of strings
<code>getregion()</code>	get a region of text from a buffer

Autocommands:

<code>TermResponseAll</code>	after the terminal response to <code>t_RV</code> and others is received
<code>WinNewPre</code>	before creating a new window

Commands:

Options:

'winfixbuf' Keep buffer focused in a window

INCOMPATIBLE CHANGES

[incompatible-9.2](#)

Improved/Different MS-Windows mapping support
[w32-experimental-keycode-trans-strategy](#)

IMPROVEMENTS

[improvements-9.2](#)

Support for command-line completion of **'keymap'** option values.

Support for compiling all the methods in a Vim9 class using `:defcompile` .

Support for alternate font highlighting using `t_CF` terminal code.

Support for Super key mappings in GTK using `<D-Key>`.

Improved visual highlighting.

Python3 support in OpenVMS.

COMPILE TIME CHANGES

[compile-changes-9.2](#)

Support for building with Ruby 3.3.

Support for building Vim 9 in z/OS (MVS).

PATCHES

[patches-9.2](#) [bug-fixes-9.2](#)
[patches-after-9.1](#)

The list of patches that got included since 9.1.0. This includes all the new features, but does not include runtime file changes (syntax, indent, ftplugin, documentation, etc.)

VIM REFERENCE MANUAL by Ralf Schandl

zOS z/OS OS390 os390 MVS

This file contains the particulars for the z/OS UNIX version of Vim.

- | | |
|-----------------------------------|-----------------|
| 1. ASCII/EBCDIC dependent scripts | zOS-has-ebcdic |
| 2. Putty and Colors | zOS-PuTTY |
| 3. Motif Problems | zOS-Motif |
| 4. Bugs | zOS-Bugs |
| 5. Limitations | zOS-limitations |
| 6. Open source on z/OS UNIX | zOS-open-source |

Contributors:

The port to z/OS UNIX was done by Ralf Schandl for the Redbook mentioned below.

Changes, bug-reports, or both by:

David Moore
Anthony Giorgio
and others

-
1. ASCII/EBCDIC dependent scripts OS390-has-ebcdic zOS-has-ebcdic

For the internal script language the feature "ebcdic" was added. With this you can fix ASCII dependent scripts like this:

```
if has("ebcdic")
    let space = 64
else
    let space = 32
endif
```

-
2. PuTTY and Colors OS390-PuTTY zOS-PuTTY

If you see problems with syntax highlighting or screen corruptions when you connect to z/OS using Putty, try the following:

- Configure Putty as "vt220" terminal (Connection->Data)
- Add the following 3 lines to your vimrc:

```
set t_AB=?[4%p1%dm
set t_AF=?[3%p1%dm
set t_C0=8
```

Note: ? is one character use <C-V><Esc> to enter it.

3. Motif Problems

[OS390-Motif](#) [zOS-Motif](#)

Note: Seen with Vim 6.*, never tested since.

It seems that in porting the Motif library to z/OS, a translation from EBCDIC to ASCII for the accelerator characters of the pull-down menus was forgotten. Even after I tried to hand convert the menus, the accelerator keys continued to only work for the opening of menus (like `<Alt-F>` to open the file menu). They still do not work for the menu items themselves (like `<Alt-F>0` to open the file browser).

There is no solution for this yet.

4. Bugs

[OS390-bugs](#) [zOS-Bugs](#)

- Vim will consistently hang when a large amount of text is selected in visual block mode. This may be due to a memory corruption issue. **Note** that this occurs in both the terminal and gui versions.

5. Limitations

[OS390-limitations](#) [zOS-limitations](#)

- No binary search in tag files.
The program `/bin/sort` sorts by ASCII value by default. This program is normally used by `ctags` to sort the tags. There might be a version of `ctags` out there, that does it right, but we can't be sure. So this seems to be a permanent restriction.
- The `cscope` interface ([cscope](#)) doesn't work for the version of `cscope` that we use on our mainframe. We have a copy of version 15.0b12, and it causes Vim to hang when using the `"cscope add"` command. I'm guessing that the binary format of the `cscope` database isn't quite what Vim is expecting. I've tried to port the current version of `cscope` (15.3) to z/OS, without much success. If anyone is interested in trying, drop me a line if you make any progress.
- No `glib/gtk` support. I have not been able to successfully compile `glib` on z/OS UNIX. This means you'll have to live without the pretty `gtk` toolbar.

Disabled at compile time:

- Multibyte support ([multibyte](#))
- Right-to-left mode ([rileft](#))
- Farsi key map ([Farsi](#))
- Arabic language support ([Arabic](#))
- Spell checking ([spell](#))

Never tested:

- Perl interface ([perl](#))
- Hangul input ([hangul](#))
- Encryption support ([encryption](#))
- Langmap (['langmap'](#))

- Python support (Python)
- Right-to-left mode ('rightleft')
- TCL interface (tcl)
- ...

6. Open source on z/OS UNIX [OS390-open-source](#) [zOS-open-source](#)

If you are interested in other Open Source Software on z/OS UNIX, have a look at the following Redbook:

Mike MacIsaac et al

"Open Source Software for z/OS and OS/390 UNIX"

IBM Form Number: SG24-5944-01

ISBN: 0738424633

http://www-03.ibm.com/systems/resources/servers_eserver_zseries_zos_unix_redbook_sg245

Also look at:

<http://www.redbooks.ibm.com>

<http://www-03.ibm.com/systems/z/os/zos/features/unix/>

<http://www-03.ibm.com/systems/z/os/zos/features/unix/library/IBM+Redbooks/index.html>

Amiga

This file contains the particularities for the Amiga version of Vim. There is also a section specifically for [MorphOS](#) below.

NOTE: The Amiga code is still included, but has not been maintained or tested.

Installation on the Amiga:

- Assign "VIM:" to the directory where the Vim "doc" directory is. Vim will look for the file "VIM:doc/help.txt" (for the help command). Setting the environment variable \$VIM also works. And the other way around: when \$VIM used and it is not defined, "VIM:" is used.
- With DOS 1.3 or earlier: Put "arp.library" in "libs:". Vim must have been compiled with the [+ARP](#) feature enabled. Make sure that newcli and run are in "C:" (for executing external commands).
- Put a shell that accepts a command with "-c" (e.g. "Csh" from Fish disk 624) in "c:" or in any other directory that is in your search path (for executing external commands).

If you have sufficient memory you can avoid startup delays by making Vim and csh resident with the command "rez csh vim". You will have to put "rezlib.library" in your "libs:" directory. Under 2.0 you will need rez version 0.5.

If you do not use digraphs, you can save some memory by recompiling without the [+digraphs](#) feature. If you want to use Vim with other terminals you can recompile with the TERMCAP option. Vim compiles with Manx 5.x and SAS 6.x. See the makefiles and feature.h.

If you notice Vim crashes on some files when syntax highlighting is on, or when using a search pattern with nested wildcards, it might be that the stack is too small. Try increasing the stack size. In a shell use the Stack command before launching Vim. On the Workbench, select the Vim icon, use the workbench "Info" menu and change the Stack field in the form.

If you want to use different colors set the termcap codes:

- t_mr (for inverted text)
- t_md (for bold text)
- t_me (for normal text after t_mr and t_md)
- t_so (for standout mode)
- t_se (for normal text after t_so)
- t_us (for underlined text)
- t_ue (for normal text after t_us)
- t_ZH (for italic text)
- t_ZR (for normal text after t_ZH)

Standard ANSI escape sequences are used. The codes are:

30 grey char	40 grey cell	>0 grey background	0 all attributes off
31 black char	41 black cell	>1 black background	1 boldface

32 white char	42 white cell	>2 white background	2 faint
33 blue char	43 blue cell	>3 blue background	3 italic
34 grey char	44 grey cell	>4 grey background	4 underscore
35 black char	45 black cell	>5 black background	7 reverse video
36 white char	46 white cell	>6 white background	8 invisible
37 blue char	47 blue cell	>7 blue background	

The codes with '>' must be the last. The cell and background color should be the same. The codes can be combined by separating them with a semicolon. For example to get white text on a blue background:

```
:set t_me=^V<Esc>[0;32;43;>3m
:set t_se=^V<Esc>[0;32;43;>3m
:set t_ue=^V<Esc>[0;32;43;>3m
:set t_ZR=^V<Esc>[0;32;43;>3m
:set t_md=^V<Esc>[1;32;43;>3m
:set t_mr=^V<Esc>[7;32;43;>3m
:set t_so=^V<Esc>[0;31;43;>3m
:set t_us=^V<Esc>[4;32;43;>3m
:set t_ZH=^V<Esc>[3;32;43;>3m
```

When using multiple commands with a filter command, e.g.

```
:r! echo this; echo that
```

Only the output of the last command is used. To fix this you have to group the commands. This depends on the shell you use (that is why it is not done automatically in Vim). Examples:

```
:r! (echo this; echo that)
:r! {echo this; echo that}
```

Commands that accept a single file name allow for embedded spaces in the file name. However, when using commands that accept several file names, embedded spaces need to be escaped with a backslash.

Vim for MorphOS

MorphOS

[this section mostly by Ali Akcaagac]

For the latest info about the MorphOS version:

http://www.akcaagac.com/index_vim.html

Problems

There are a couple of problems which are not MorphOS related but more Vim and UN*X related. When starting up Vim in ram: it complains with a nag requester from MorphOS please simply ignore it. Another problem is when running Vim as is some plugins will cause a few problems which you can ignore as well. Hopefully someone will be fixing it over the time.

To pass all these problems for now you can either run:

```
vim <file to be edited>
```

or if you want to run Vim plain and enjoy the motion of Helpfiles etc. it then

would be better to enter:

```
vim --noplugins <of course you can add a file>
```

Installation

- 1) Please copy the binary 'VIM' file to c:
- 2) Get the Vim runtime package from:

```
ftp://ftp.vim.org/pub/vim/amiga/vim62rt.tgz
```

and unpack it in your 'Apps' directory of the MorphOS installation. For me this would create following directory hierarchy:

```
MorphOS:Apps/Vim/Vim62/...
```

- 3) Add the following lines to your s:shell-startup (Important!).

```
;Begin VIM
Set VIM=MorphOS:Apps/Vim/Vim62
Assign HOME: ""
;End VIM
```

- 4) Copy the '.vimrc' file to s:

- 5) There is also a file named 'color-sequence' included in this archive. This will set the MorphOS Shell to show ANSI colors. Please copy the file to s: and change the s:shell-startup to:

```
;Begin VIM
Set VIM=MorphOS:Apps/Vim/Vim62
Assign HOME: ""
Execute S:Color-Sequence
Cls
;End VIM
```


os_beos.txt For Vim version 9.1. Last change: 2020 Jun 07

VIM REFERENCE MANUAL by Bram Moolenaar

This file used to contain particularities for the [beos](#) [BeOS](#) [BeBox](#) port of Vim.

The BeOS support was removed in patch 8.2.0849.

dos DOS

This file documents the common particularities of the MS-DOS and Win32 versions of Vim. Also see [os_win32.txt](#) and [os_msdos.txt](#).

- | | |
|-----------------------------|---------------------------------------|
| 1. File locations | dos-locations |
| 2. Using backslashes | dos-backslash |
| 3. Standard mappings | dos-standard-mappings |
| 4. Screen output and colors | dos-colors |
| 5. File formats | dos-file-formats |
| 6. :cd command | dos-:cd |
| 7. Interrupting | dos-CTRL-Break |
| 8. Temp files | dos-temp-files |
| 9. Shell option default | dos-shell |
| 10. PowerShell | dos-powershell |

=====

1. File locations[dos-locations](#)

If you keep the Vim executable in the directory that contains the help and syntax subdirectories, there is no need to do anything special for Vim to work. No registry entries or environment variables need to be set. Just make sure that the directory is in your search path, or use a shortcut on the desktop.

Your vimrc files ("[_vimrc](#)" and "[_gvimrc](#)") are normally located one directory up from the runtime files. If you want to put them somewhere else, set the environment variable \$VIM to the directory where you keep them. Example:

```
set VIM=C:\user\piet
```

Will find "c:\user\piet_vimrc".

Note: This would only be needed when the computer is used by several people. Otherwise it's simpler to keep your [_vimrc](#) file in the default place.

If you move the executable to another location, you also need to set the \$VIM environment variable. The runtime files will be found in "\$VIM/vim{version}". Example:

```
set VIM=E:\vim
```

Will find the version 8.2 runtime files in "e:\vim\vim82".

Note: This is not recommended. The preferred way is to keep the executable in the runtime directory.

If you move your executable AND want to put your "[_vimrc](#)" and "[_gvimrc](#)" files somewhere else, you must set \$VIM to where you vimrc files are, and set \$VIMRUNTIME to the runtime files. Example:

```
set VIM=C:\usr\piet
```

```
set VIMRUNTIME=E:\vim\vim82
```

Will find "c:\user\piet_vimrc" and the runtime files in "e:\vim\vim82".

See [\\$VIM](#) and [\\$VIMRUNTIME](#) for more information.

You can set environment variables for each user separately through the System Properties dialog box. The steps to do that:

1. Type Windows Key + R to open the "Run" dialog box.
2. Enter "sysdm.cpl" and press the "OK" button. The "System Properties" dialog box will open.
3. Select the "Advanced" tab and press the "Environment Variables..." button. The "Environment Variables" dialog box will open.
4. Select an existing variable in the "User variables" list and press the "Edit..." button to edit it. Or press the "New..." button to add a new variable.
5. After you finished editing variables, press the "OK" button to save the changes.

2. Using backslashes

dos-backslash

Using backslashes in file names can be a problem. Vi halves the number of backslashes for some commands. Vim is a bit more tolerant and does not remove backslashes from a file name, so ":e c:\foo\bar" works as expected. But when a backslash occurs before a special character (space, comma, backslash, etc.), Vim removes the backslash. Use slashes to avoid problems: ":e c:/foo/bar" works fine. Vim replaces the slashes with backslashes internally to avoid problems with some MS-DOS programs and Win32 programs.

When you prefer to use forward slashes, set the '**shellslash**' option. Vim will then replace backslashes with forward slashes when expanding file names. This is especially useful when using a Unix-like '**shell**'.

3. Standard mappings

dos-standard-mappings

The mappings for **CTRL-PageUp** and **CTRL-PageDown** have been removed, they now jump to the next or previous tab page `<C-PageUp>` `<C-PageDown>`

If you want them to move to the first and last screen line you can use these mappings:

key	key code	Normal/Visual mode	Insert mode
CTRL-PageUp	<code><M-N><M-C-D></code>	H	<code><C-O>H</code>
CTRL-PageDown	<code><M-N>v</code>	L\$	<code><C-O>L<C-O>\$</code>

Additionally, these keys are available for copy/cut/paste. In the Win32 and DJGPP versions, they also use the clipboard.

Shift-Insert	paste text (from clipboard)	<code><S-Insert></code>
CTRL-Insert	copy Visual text (to clipboard)	<code><C-Insert></code>
CTRL-Del	cut Visual text (to clipboard)	<code><C-Del></code>
Shift-Del	cut Visual text (to clipboard)	<code><S-Del></code>
CTRL-X	cut Visual text (to clipboard)	

These mappings accomplish this (Win32 and DJGPP versions of Vim):

key	key code	Normal	Visual	Insert
-----	----------	--------	--------	--------

Shift-Insert	<M-N><M-T>	"*P	"-d"*P	<C-R><C-O>*
CTRL-Insert	<M-N><M-U>		"*y	
Shift-Del	<M-N><M-W>		"*d	
CTRL-Del	<M-N><M-X>		"*d	
CTRL-X	<C-X>		"*d	

Or these mappings (non-Win32 version of Vim):

key	key code	Normal	Visual	Insert
Shift-Insert	<M-N><M-T>	P	"-dP	<C-R><C-O>"
CTRL-Insert	<M-N><M-U>		y	
Shift-Del	<M-N><M-W>		d	
CTRL-Del	<M-N><M-X>		d	

When the clipboard is supported, the "*" register is used.

4. Screen output and colors dos-colors

The default output method for the screen is to use bios calls. This works right away on most systems. You do not need ansi.sys. You can use ":mode" to set the current screen mode. See [:mode](#) .

To change the screen colors that Vim uses, you can use the [:highlight](#) command. The Normal highlight group specifies the colors Vim uses for normal text. For example, to get grey text on a blue background:

```
:hi Normal ctermbg=Blue ctermfg=grey
```

See [highlight-groups](#) for other groups that are available.

A DOS console does not support attributes like bold and underlining. You can set the color used in five modes with nine terminal options. **Note** that this is not necessary since you can set the color directly with the ":highlight" command; these options are for backward compatibility with older Vim versions. The '[highlight](#)' option specifies which of the five modes is used for which action.

```
:set t\_mr=^V^\[\\\]xxm      start of invert mode
:set t\_md=^V^\[\\\]xxm      start of bold mode
:set t\_me=^V^\[\\\]xxm      back to normal text

:set t\_so=^V^\[\\\]xxm      start of standout mode
:set t\_se=^V^\[\\\]xxm      back to normal text

:set t\_us=^V^\[\\\]xxm      start of underline mode
:set t\_ue=^V^\[\\\]xxm      back to normal text

:set t\_ZH=^V^\[\\\]xxm      start of italics mode
:set t\_ZR=^V^\[\\\]xxm      back to normal text
```

^V is **CTRL-V**

^[is <Esc>

You must replace xx with a decimal code, which is the foreground color number and background color number added together:

COLOR	FOREGROUND	BACKGROUND
Black	0	0
DarkBlue	1	16
DarkGreen	2	32
DarkCyan	3	48
DarkRed	4	64
DarkMagenta	5	80
Brown, DarkYellow	6	96
LightGray	7	112
DarkGray	8	128 *
Blue, LightBlue	9	144 *
Green, LightGreen	10	160 *
Cyan, LightCyan	11	176 *
Red, LightRed	12	192 *
Magenta, LightMagenta	13	208 *
Yellow, LightYellow	14	224 *
White	15	240 *

* Depending on the display mode, the color codes above 128 may not be available, and code 128 will make the text blink.

When you use 0, the color is reset to the one used when you started Vim (usually 7, lightgray on black, but you can override this. If you have overridden the default colors in a command prompt, you may need to adjust some of the highlight colors in your vimrc---see below). This is the default for t_me.

The defaults for the various highlight modes are:

t_mr	112	reverse mode: Black text (0) on LightGray (112)
t_md	15	bold mode: White text (15) on Black (0)
t_me	0	normal mode (revert to default)
t_so	31	standout mode: White (15) text on DarkBlue (16)
t_se	0	standout mode end (revert to default)
t_czh	225	italic mode: DarkBlue text (1) on Yellow (224)
t_czr	0	italic mode end (revert to default)
t_us	67	underline mode: DarkCyan text (3) on DarkRed (64)
t_ue	0	underline mode end (revert to default)

These colors were chosen because they also look good when using an inverted display, but you can change them to your liking.

Example:

```

:set t_mr=^V^[|97m " start of invert mode: DarkBlue (1) on Brown (96)
:set t_md=^V^[|67m " start of bold mode: DarkCyan (3) on DarkRed (64)
:set t_me=^V^[|112m " back to normal mode: Black (0) on LightGray (112)

:set t_so=^V^[|37m " start of standout mode: DarkMagenta (5) on DarkGreen
(32)
:set t_se=^V^[|112m " back to normal mode: Black (0) on LightGray (112)

```

=====

5. File formats

dos-file-formats

If the **'fileformat'** option is set to "dos" (which is the default), Vim accepts a single `<NL>` or a `<CR><NL>` pair for end-of-line (`<EOL>`). When writing a file, Vim uses `<CR><NL>`. Thus, if you edit a file and write it, Vim replaces `<NL>` with `<CR><NL>`.

If the **'fileformat'** option is set to "unix", Vim uses a single `<NL>` for `<EOL>` and shows `<CR>` as `^M`.

You can use Vim to replace `<NL>` with `<CR><NL>` by reading in any mode and writing in Dos mode (`":se ff=dos"`).

You can use Vim to replace `<CR><NL>` with `<NL>` by reading in Dos mode and writing in Unix mode (`":se ff=unix"`).

Vim sets **'fileformat'** automatically when **'fileformats'** is not empty (which is the default), so you don't really have to worry about what you are doing.

'fileformat' **'fileformats'**

If you want to edit a script file or a binary file, you should set the **'binary'** option before loading the file. Script files and binary files may contain single `<NL>` characters which Vim would replace with `<CR><NL>`. You can set **'binary'** automatically by starting Vim with the "-b" (binary) option.

6. :cd command

dos-:cd

The `":cd"` command recognizes the drive specifier and changes the current drive. Use `":cd c:"` to make drive C the active drive. Use `":cd d:\foo"` to go to the directory "foo" in the root of drive D. Vim also recognizes UNC names if the system supports them; e.g., `":cd \\server\share\dir"`. `:cd`

7. Interrupting

dos-CTRL-Break

Use **CTRL-Break** instead of **CTRL-C** to interrupt searches. Vim does not detect the **CTRL-C** until it tries to read a key.

8. Temp files

dos-temp-files

Only for the 16 bit and 32 bit DOS version:

Vim puts temporary files (for filtering) in the first of these directories that exists and in which Vim can create a file:

- \$TMP
- \$TEMP
- C:\TMP
- C:\TEMP
- current directory

For the Win32 version (both console and GUI):

Vim uses standard Windows functions to obtain a temporary file name (for filtering). The first of these directories that exists and in which Vim can create a file is used:

\$TMP
\$TEMP
current directory

9. Shell option default

dos-shell

The default for the **'sh'** ('shell') option is "command.com" on Windows 95 and "cmd.exe" on Windows NT. If SHELL is defined, Vim uses SHELL instead, and if SHELL is not defined but COMSPEC is, Vim uses COMSPEC. Vim starts external commands with "<shell> /c <command_name>". Typing **CTRL-Z** starts a new command subshell. Return to Vim with "exit". **'shell'** CTRL-Z

If you are running a third-party shell, you may need to set the **'shellcmdflag'** ('shcf') and **'shellquote'** ('shq') or **'shellxquote'** ('sxq') options. Unfortunately, this also depends on the version of Vim used. For example, with the MKS Korn shell or with bash, the values of the options should be:

	DOS 16 bit	DOS 32 bit	Win32
'shellcmdflag'	-c	-c	-c
'shellquote'	"		
'shellxquote'			"

For Dos 16 bit this starts the shell as:

```
<shell> -c "command name" >file
```

For Win32 as:

```
<shell> -c "command name >file"
```

For DOS 32 bit, DJGPP does this internally somehow.

When starting up, if Vim does not recognise a standard Windows shell it checks for the presence of "sh" anywhere in the **'shell'** option. If it is present, Vim sets the **'shellcmdflag'** and **'shellquote'** or **'shellxquote'** options will be set as described above.

10. PowerShell

dos-powershell dos-pwsh

Vim supports PowerShell Desktop and PowerShell Core. PowerShell Desktop is the version of PowerShell that is installed with Windows, while PowerShell Core is a separate downloadable version that works cross-platform. To see which version you are using then enter the following in a PowerShell prompt - \$PSVersionTable.PSEdition

If **'shell'** includes "powershell" in the filename at startup then VIM sets **'shellcmdflag'**, **'shellxquote'**, **'shellpipe'**, and **'shellredir'** options to the following values:

'shellcmdflag'	-Command
'shellxquote'	"
'shellpipe'	2>&1 Out-File -Encoding default
'shellredir'	2>&1 Out-File -Encoding default

If **'shell'** includes "pwsh" in the filename at startup then VIM sets

'shellcmdflag', 'shellquote', 'shellpipe', and 'shellredir' options to the following values:

```
'shellcmdflag' -c
'shellquote'   "
'shellpipe'    >%s 2>&1
'shellredir'   >%s 2>&1
```

If you find that PowerShell commands are taking a long time to run then try with "-NoProfile" at the beginning of the 'shellcmdflag'. Note this will prevent any PowerShell environment setup by the profile from taking place.

If you have problems running PowerShell scripts through the 'shell' then try with "-ExecutionPolicy RemoteSigned -Command" at the beginning of 'shellcmdflag'. See online Windows documentation for more information on PowerShell Execution Policy settings.

See [option-backslash](#) about including spaces in 'shellcmdflag' when using multiple flags.

The 'shellpipe' and 'shellredir' option values re-encode the UTF-16LE output from PowerShell Desktop to your currently configured console codepage. The output can be forced into a different encoding by changing "default" to one of the following:

```
unicode          - UTF-16LE (default output from PowerShell 5.1)
bigendianunicode - UTF-16
utf8             - UTF-8
utf7            - UTF-7 (no BOM)
utf32           - UTF-32
ascii           - 7-bit ASCII character set
default         - System's active code page (typically ANSI)
oem            - System's current OEM code page
```

Note The above multi-byte Unicode encodings include a leading BOM unless otherwise indicated.

By default PowerShell Core's output is UTF-8 encoded without a BOM. If you want to force the output of PowerShell Core into a different encoding then set 'shellredir' and 'shellpipe' to "2>&1 | Out-File -Encoding encoding" where encoding is one of the following:

```
ascii           - 7-bit ASCII character set
bigendianunicode - UTF-16BE
bigendianutf32  - UTF-32BE
oem            - System's current OEM code page
unicode        - UTF-16LE
utf7           - UTF-7
utf8           - UTF-8
utf8BOM        - UTF-8, with BOM
utf8NoBOM      - UTF-8, no BOM (default output from PowerShell Core)
utf32         - UTF-32
```

Since PowerShell Core 6.2, the Encoding parameter also supports specifying a

numeric ID of a registered code page (-Encoding 1251) or string names of registered code pages (-Encoding "windows-1251"). The .NET documentation for Encoding.CodePage has more information

Haiku

This file contains the particularities for the Haiku version of Vim. For matters not discussed in this file, Vim behaves very much like the Unix `os_unix.txt` version.

Haiku is an open-source operating system inspired by BeOS, that specifically targets personal computing.

- | | |
|---|--------------------------------------|
| 1. General | <code>haiku-general</code> |
| 2. Compiling Vim | <code>haiku-compiling</code> |
| 3. The Haiku GUI | <code>haiku-gui</code> |
| 4. The <code>\$VIM</code> directory | <code>haiku-vimdir</code> |
| 5. The <code>\$USER_SETTINGS_DIR</code> directory | <code>haiku-user-settings-dir</code> |
| 6. Drag & Drop | <code>haiku-dragndrop</code> |
| 7. Single Launch vs. Multiple Launch | <code>haiku-launch</code> |
| 8. Fonts | <code>haiku-fonts</code> |
| 9. The meta key modifier | <code>haiku-meta</code> |
| 10. Mouse key mappings | <code>haiku-mouse</code> |
| 11. Color names | <code>haiku-colors</code> |
| 12. GUI Toolbar Images | <code>haiku-toolbar-images</code> |
| 13. Credits | <code>haiku-support-credits</code> |
| 14. Bugs & to-do | <code>haiku-bugs</code> |

1. General `haiku-general`

The default syntax highlighting mostly works with different foreground colors to highlight items. This works best if you set your Terminal window to a darkish background and light letters. Some middle-grey background (for instance $(r,g,b)=(168,168,168)$) with black letters also works nicely.

2. Compiling Vim `haiku-compiling`

Vim can be compiled using the standard `configure/make` approach. Running `./configure` without any arguments or passing `--enable-gui=haiku`, will compile vim with the Haiku GUI support. Run `./configure --help`, to find out other features you can enable/disable.

Haiku uses "ncurses6" as its terminal library, therefore you need to have "ncurses6_devel" package installed from HaikuDepot in order to configure the Haiku build. Just append `--with-tlib=ncurses` to `./configure` command.

Now you should use "make" to compile Vim, then "make install" to install it. For seamless integration into Haiku, the GUI-less vim binary should be additionally installed over the GUI version. Typical build commands are:

```
./configure --prefix=`finddir B_SYSTEM_NONPACKAGED_DIRECTORY` \
  --datarootdir=`finddir B_SYSTEM_NONPACKAGED_DATA_DIRECTORY` \
  --mandir=`finddir B_SYSTEM_NONPACKAGED_DIRECTORY`/documentation/man \
  --with-tlib=ncurses
make clean
make install
```

```
./configure --prefix=`finddir B_SYSTEM_NONPACKAGED_DIRECTORY` \
  --datarootdir=`finddir B_SYSTEM_NONPACKAGED_DATA_DIRECTORY` \
  --mandir=`finddir B_SYSTEM_NONPACKAGED_DIRECTORY`/documentation/man \
  --with-tlib=ncurses \
  --disable-gui
make clean
make install
```

3. The Haiku GUI

haiku-gui

Normally Vim starts with the GUI if you start it as `gvim` or `vim -g`. The vim version with GUI tries to determine if it was started from the Tracker instead of the Terminal, and if so, uses the GUI anyway. However, the current detection scheme is fooled if you use the command `"vim - </dev/null"`.

Stuff that does not work yet:

- Mouse up events are not generated when outside the window. You can notice this when selecting text and moving the cursor outside the window, then letting go of the mouse button. Another way is when you drag the scrollbar and do the same thing. Because Vim still thinks you are still playing with the scrollbar it won't change it itself. There is a workaround which kicks in when the window is activated or deactivated (so it works best with focus-follows-mouse turned on).
- The cursor does not flash.

4. The \$VIM directory

haiku-vimdir

\$VIM is the symbolic name for the place where Vim's support files are stored. The default value for \$VIM is set at compile time and can be determined with:

```
:version
```

The normal value is `/boot/system/data/vim` for Haikuports version, `/boot/system/non-packaged/data/vim` for manual builds. If you don't like it you can set the VIM environment variable to override this, or set **'helpfile'** in your `.vimrc`:

```
:if version >= 500
:  set helpfile=~/.vim/runtime/doc/help.txt
:  syntax on
:endif
```

5. The \$USER_SETTINGS_DIR directory

[haiku-user-settings-dir](#)

\$USER_SETTINGS_DIR is the symbolic name for the place where Haiku configuration and settings files are stored.

The normal value is /boot/home/config/settings.

6. Drag & Drop

[haiku-dragdrop](#)

You can drop files and directories on either the Vim icon (starts a new Vim session, unless you use the File Types application to set Vim to be "Single Launch") or on the Vim window (starts editing the files). Dropping a folder sets Vim's current working directory `:cd :pwd`. If you drop files or folders with either SHIFT key pressed, Vim changes directory to the folder that contains the first item dropped. When starting Vim, there is no need to press shift: Vim behaves as if you do.

Files dropped set the current argument list. [argument-list](#)

7. Single Launch vs. Multiple Launch

[haiku-launch](#)

As distributed Vim's Application Flags (as seen in the FileTypes preference) are set to Multiple Launch. If you prefer, you can set them to Single Launch instead. Attempts to start a second copy of Vim will cause the first Vim to open the files instead. This works from the Tracker but also from the command line. In the latter case, non-file (option) arguments are not supported. Another drawback of the Single Launch is silent ignore of "Open With ..." requests by vim instance that running as non-GUI application even GUI support was compiled in. Vim instance running with GUI has no such problems.

NB: Only the GUI version has a BApplication (and hence Application Flags). This section does not apply to the GUI-less version, should you compile one.

8. Fonts

[haiku-fonts](#)

Set fonts with

```
:set guifont=DejaVu_Sans_Mono/Book/12
```

where the first part is the font family, the second part the style, and the third part the size. You can use underscores instead of spaces in family and style.

Best results are obtained with monospaced fonts. Vim attempts to use all fonts in B_FIXED_SPACING mode but apparently this does not work for proportional fonts (despite what the BeBook says).

To verify which encodings are supported by the current font give the

```
:digraphs
```

command, which lists a bunch of characters with their ISO Latin 1 encoding. If, for instance, there are "box" characters among them, or the last character isn't a dotted-y, then for this font the encoding does not work.

If the font you specify is unavailable, you get the system fixed font.

GUI Font Selection Dialog is available at giving the:

```
:set guifont=*
```

command.

9. The meta key modifier

[haiku-meta](#)

The META key modifier is obtained by the left or right OPTION keys. This is because the ALT (aka COMMAND) keys are not passed to applications.

10. Mouse key mappings

[haiku-mouse](#)

Vim calls the various mouse buttons LeftMouse, MiddleMouse and RightMouse. If you use the default Mouse preference settings these names indeed correspond to reality. Vim uses this mapping:

```
Button 1 -> LeftMouse,  
Button 2 -> RightMouse,  
Button 3 -> MiddleMouse.
```

If your mouse has fewer than 3 buttons you can provide your own mapping from mouse clicks with modifier(s) to other mouse buttons. See the file `$VIM/macros/swapmous.vim` for an example.

[gui-mouse-mapping](#)

11. Color names

[haiku-colors](#)

Vim has a number of color names built-in. Additional names can be defined in `v:colornames`. See `:colorscheme` for details.

12. GUI Toolbar Images

[haiku-toolbar-images](#)

Alternative set of toolbar images should be the PNG image of any height you like. Image width is calculated to contain at least 32 buttons in one-row cells.

The image should be stored under the name `$VIRUNTIME/bitmaps/builtin-tools.png`. More info about the buttons assignment are at [builtin-tools](#).

13. Credits

[haiku-support-credits](#)

Haiku port is based on work done for BeOS version by many people

- BeBox GUI support Copyright 1998 by Olaf Seibert;
- Ported to R4 by Richard Offer richard@whitequeen.com Jul 99;

- Those who contributed, not listed above but not forgotten;
- Haiku support by Siarzhuk Zharski <imker@gmx.li> Apr-Mai 2009.

All the changes and patches released under vim-license.

Thank you, all!

14. Bugs & to-do

[haiku-bugs](#)

The port is under development now and far away from the perfect state. For bug reports, patches and wishes, please use the Vim mailing list or Vim Github repository.

Mailing list: <https://www.vim.org/maillist.php>

Vim Github repository: <https://github.com/vim/vim>

This file documents the particularities of the Macintosh version of Vim.

NOTE: This file is a bit outdated. You might find more useful info here:
<http://macvim.org/>

- | | |
|--------------------------|---------------------------------------|
| 1. Filename Convention | mac-filename |
| 2. .vimrc and .vim files | mac-vimfile |
| 3. Standard mappings | mac-standard-mappings |
| 4. FAQ | mac-faq |
| 5. Known Lack | mac-lack |
| 6. Mac Bug Report | mac-bug |
| 7. Compiling Vim | mac-compile |
| 8. The darwin feature | mac-darwin-feature |

There was a Mac port for version 3.0 of Vim. Here are the first few lines from the old file:

VIM Release [Notes](#)

Initial Macintosh release, VIM version 3.0
 19 October 1994

Eric Fischer

enf1@midway.uchicago.edu, eric@jcp.uchicago.edu, etaoin@uchicago.edu
 5759 N. Guilford Ave
 Indianapolis IN 46220 USA

1. Filename Convention

[mac-filename](#)

Starting with [Vim version 7](#) you can just use the unix path separators with Vim. In order to determine if the specified filename is relative to the current folder or absolute (i.e. relative to the "Desktop"), the following algorithm is used:

```
If the path start by a "/", the path is absolute
If the path start by a ":", the path is relative
If the path doesn't start by neither a "/" nor ":",
and a ":" is found before a "/" then the path is absolute
```

```
:e /HD/text
:e HD:text
Edit the file "text" of the disk "HD"
:e :src:main.c
:e src/main.c
Edit the file "main.c" in the folder "src" in the current folder
:e os_mac.c
```

Edit the file "os_mac.c" in the current folder.

You can use the `$VIM` and `$VIMRUNTIME` variable.

```
:so $VIMRUNTIME:syntax:syntax.vim
```

2. .vimrc and .vim files

mac-vimfile

It is recommended to use Unix style line separators for Vim scripts, thus a single newline character.

When starting up Vim will load the `$VIMRUNTIME/macmap.vim` script to define default command-key mappings.

On older systems files starting with a dot "." are discouraged, thus the rc files are named "vimrc" or "_vimrc" and "gvimrc" or "_gvimrc". These files can be in any format (mac, dos or unix). Vim can handle any file format when the '`nocompatible`' option is set, otherwise it will only handle mac format files.

3. Standard mappings

mac-standard-mappings

The following mappings are available for cut/copy/paste from/to clipboard.

key	Normal	Visual	Insert	Description
Command-v	"*P	"-d"*P	<C-R>* <D-v>	paste text
Command-c		"*y		copy Visual text <D-c>
Command-x		"*d		cut Visual text <D-x>
Backspace		"*d		cut Visual text

4. Mac FAQ

mac-faq

On the internet: <http://macvim.org/OSX/index.php#FAQ>

Q: I can't enter non-ASCII character in Apple Terminal.

A: Under Window Settings, Emulation, make sure that "Escape non-ASCII characters" is not checked.

Q: How do I start the GUI from the command line?

A: Assuming that Vim.app is located in /Applications:
open /Applications/Vim.app

Or:

```
/Applications/Vim.app/Contents/MacOS/Vim -g {arguments}
```

Q: How can I set \$PATH to something reasonable when I start Vim.app from the GUI or with open?

A: The following trick works with most shells. Put it in your vimrc file. This is included in the system vimrc file included with the binaries distributed at macvim.org .

```
let s:path = system("echo echo VIMPATH'${PATH}' | $SHELL -l")  
let $PATH = matchstr(s:path, 'VIMPATH\zs.\{-}\ze\n')
```

5. Mac Lack

[mac-lack](#)

In a terminal **CTRL-^** needs to be entered as Shift-Control-6. **CTRL-@** as Shift-Control-2.

6. Mac Bug Report

[mac-bug](#)

When reporting any Mac specific bug or feature change, please use the vim-mac maillist [vim-mac](#). However, you need to be subscribed. An alternative is to send a message to the current MacVim maintainers:

`mac@vim.org`

7. Compiling Vim

[mac-compile](#)

See the file "src/INSTALLmac.txt" that comes with the source files.

8. The Darwin Feature

[mac-darwin-feature](#)

If you have a Mac that isn't very old, you will be running OS X, also called Darwin. The last pre-Darwin OS was Mac OS 9. The darwin feature makes Vim use Darwin-specific properties.

What is accomplished with this feature is two-fold:

- Make Vim interoperable with the system clipboard.
- Incorporate into Vim a converter module that bridges the gap between some character encodings specific to the platform and those known to Vim.

Needless to say, both are not to be missed for any decent text editor to work nicely with other applications running on the same desktop environment.

As Vim is not an application dedicated only to macOS, we need an extra feature to add in order for it to offer the same user experience that our users on other platforms enjoy to people on macOS.

For brevity, the feature is referred to as "darwin" to signify it one of the Vim features that are specific to that particular platform.

The feature is a configuration option. Accordingly, whether it is enabled or not is determined at build time; once it is selected to be enabled, it is compiled in and hence cannot be disabled at runtime.

The feature is enabled by default. For most macOS users, that should be sufficient unless they have specific needs mentioned briefly below.

If you want to disable it, pass `--disable-darwin` to the configure script:

```
./configure --disable-darwin <other options>
```

and then run `make` to build Vim. The order of the options doesn't matter.

To make sure at runtime whether or not the darwin feature is compiled in, you can use `has('osxdarwin')` which returns 1 if the feature is compiled in; 0 otherwise. For backward compatibility, you can still use `macunix` instead of `osxdarwin`.

Notable use cases where `--disable-darwin` is turned out to be useful are:

- When you want to use `x11-selection` instead of the system clipboard.
- When you want to use `x11-clientserver` .

Since both have to make use of X11 inter-client communication for them to work properly, and since the communication mechanism can come into conflict with the system clipboard, the darwin feature should be disabled to prevent Vim from hanging at runtime.

os_mint.txt For Vim version 9.1. Last change: 2020 Jul 14

VIM REFERENCE MANUAL by Jens M. Felderhoff

MiNT Atari

The Atari MiNT support was removed with patch 8.2.1215. It probably didn't work at that time, since the code was old and not maintained.

Originally added by: Jens M. Felderhoff, e-mail: <jmf@infko.uni-koblenz.de>

os_msdos.txt For Vim version 9.1. Last change: 2016 Feb 26

VIM REFERENCE MANUAL by Bram Moolenaar

msdos ms-dos MSDOS MS-DOS

This file used to contain the particularities for the MS-DOS version of Vim. MS-DOS support was removed in patch 7.4.1399. If you want to use it you will need to get a version older than that. **Note** that the MS-DOS version doesn't work, there is not enough memory. The DOS32 version (using DJGPP) might still work on older systems.

os_os2.txt For Vim version 9.1. Last change: 2015 Dec 31

VIM REFERENCE MANUAL by Paul Sloodman

This file used to contain the particularities for the `os2` `OS2` `OS/2` version of Vim.
The OS/2 support was removed in patch 7.4.1008.

VIM REFERENCE MANUAL by Julian Kinraid

QNX qnx

1. General	qnx-general
2. Compiling Vim	qnx-compiling
3. Terminal support	qnx-terminal
4. Photon GUI	photon-gui
5. Photon fonts	photon-fonts
6. Bugs & things To Do	

1. General

qnx-general

Vim on QNX behaves much like other unix versions. [os_unix.txt](#)

2. Compiling Vim

qnx-compiling

Vim can be compiled using the standard configure/make approach. If you want to compile for X11, pass the `--with-x` option to configure. Otherwise, running `./configure` without any arguments or passing `--enable-gui=photon`, will compile vim with the Photon gui support. Run `./configure --help`, to find out other features you can enable/disable.

3. Terminal support

qnx-terminal

Vim has support for the mouse and clipboard in a pterm, if those options are compiled in, which they are normally.

The options that affect mouse support are `'mouse'` and `'ttymouse'`. When using the mouse, only simple left and right mouse clicking/dragging is supported. If you hold down shift, ctrl, or alt while using the mouse, pterm will handle the mouse itself. It will make a selection, separate from what vim's doing.

When the mouse is in use, you can press Alt-RightMouse to open the pterm menu. To turn the mouse off in vim, set the mouse option to nothing, set `mouse=`

4. Photon GUI

photon-gui

To start the gui for vim, you need to run either `gvim` or `vim -g`, otherwise the terminal version will run. For more info - [gui-x11-start](#)

Supported features:

<code>:browse</code> command	<code>:browse</code>
<code>:confirm</code> command	<code>:confirm</code>

Cursor blinking	'guicursor'
Menus, popup menus and menu priorities	:menu popup-menu menu-priority
Toolbar	gui-toolbar 'toolbar'
Font selector (:set guifont=*)	photon-fonts
Mouse focus	'mousefocus'
Mouse hide	'mousehide'
Mouse cursor shapes	'mousethshape'
Clipboard	gui-clipboard

Unfinished features:

Various international support, such as Farsi & Hebrew support, different encodings, etc.

This help file

Unsupported features:

Find & Replace window	:promptfind
Tearoff menus	

Other things which I can't think of so I can't list them

5. Fonts

photon-fonts

You set fonts in the gui with the guifont option

```
:set guifont=Lucida\ Terminal
```

The font must be a monospace font, and any spaces in the font name must be escaped with a '\'. The default font used is PC Terminal, size 8. Using '*' as the font name will open a standard Photon font selector where you can select a font.

Following the name, you can include optional settings to control the size and style of the font, each setting separated by a ':'. Not all fonts support the various styles.

The options are,

s{size}	Set the size of the font to {size}
b	Bold style
a	Use antialiasing
i	Italic style

Examples:

Set the font to monospace size 10 with antialiasing

```
:set guifont=monospace:s10:a
```

Set the font to Courier size 12, with bold and italics

```
:set guifont=Courier:s12:b:i
```

Select a font with the requester

```
:set guifont=*
```

6. Bugs & things To Do

Known problems:

- Vim hangs sometimes when running an external program. Workaround:
put this line in your `vimrc` file:
`set noguipity`

Bugs:

- Still a slight problem with menu highlighting.
- When using phditto/phinfo/etc., if you are using a font that doesn't support the bold attribute, when vim attempts to draw bold text it will be all messed up.
- The cursor can sometimes be hard to see.
- A number of minor problems that can be fixed. :)

Todo:

- Improve multi-language support.
- Options for setting the fonts used in the menu and toolbar.
- Find & Replace dialog.
- The clientserver features.
- Maybe tearoff menus.

- Replace usage of `fork()` with `spawn()` when launching external programs.

os_risc.txt For Vim version 9.1. Last change: 2011 May 10

VIM REFERENCE MANUAL by Thomas Leonard

riscos RISCOS RISC-OS

The RISC OS support has been removed from Vim with patch 7.3.187.
If you would like to use Vim on RISC OS get the files from before that patch.

unix Unix

This file contains the particularities for the Unix version of Vim.

For compiling Vim on Unix see "INSTALL" and "Makefile" in the src directory.

The default help file name is "/usr/local/lib/vim/help.txt"

The files "\$HOME/.vimrc" and "\$HOME/.exrc" are used instead of "s:.vimrc" and "s:.exrc". Additionally "/usr/local/etc/vimrc" is used first.

If "/usr/local/share" exists it is used instead of "/usr/local/lib".

Temporary files (for filtering) are put in "/tmp". If you want to place them somewhere else, set the environment variable \$TMPDIR to the directory you prefer.

With wildcard expansion you can use '~' (home directory) and '\$' (environment variable).

fork spoon

For executing external commands fork()/exec() is used when possible, otherwise system() is used, which is a bit slower. The output of ":version" includes +fork when fork()/exec() is used, +system() when system() is used. This can be changed at compile time.

(For forking of the GUI version see [gui-fork](#) .)

For historic reasons terminal updating under Unix is expected to be slow (e.g. serial line terminal, shell window in suntools), the 'showcmd' and 'ruler' options are off by default. If you have a fast terminal, try setting them on:

```
set showcmd ruler
```

When using Vim in an xterm the mouse clicks can be used by Vim by setting 'mouse' to "a". If there is access to an X-server gui style copy/paste will be used and visual feedback will be provided while dragging with the mouse. If you then still want the xterm copy/paste with the mouse, press the shift key when using the mouse. See [mouse-using](#) . Visual feedback while dragging can also be achieved via the 'ttymouse' option if your xterm is new enough.

terminal-colors

To use colors in Vim you can use the following example (if your terminal supports colors, but "T_Co" is empty or zero):

```
:set t_me=[0;1;36m " normal mode (undoes t_mr and t_md)
:set t_mr=[0;1;33;44m " reverse (invert) mode
:set t_md=[1;33;41m " bold mode
:set t_se=[1;36;40m " standout end
:set t_so=[1;32;45m " standout mode
:set t_ue=[0;1;36m " underline end
:set t_us=[1;32m " underline mode start
```

[the ^[is an <Esc>, type **CTRL-V** <Esc> to enter it]

For real color terminals the `":highlight"` command can be used.

The file `"tools/vim132"` is a shell script that can be used to put Vim in 132 column mode on a vt100 and lookalikes.

VIM REFERENCE MANUAL

VMS vms

This file contains the particularities for the VMS version of Vim. You can reach this information file by typing :help VMS in Vim command prompt.

- | | |
|------------------------|-------------------------------|
| 1. Getting started | vms-started |
| 2. Download files | vms-download |
| 3. Compiling | vms-compiling |
| 4. Problems | vms-problems |
| 5. Deploy | vms-deploy |
| 6. Practical usage | vms-usage |
| 7. GUI mode questions | vms-gui |
| 8. Useful notes | vms-notes |
| 9. VMS related changes | vms-changes |
| 10. Authors | vms-authors |

1. Getting started

[vms-started](#)

Vim (Vi IMproved) is a Vi-compatible text editor that runs on nearly every operating system known to humanity. Now use Vim on OpenVMS too, in character or X/Motif environment. It is fully featured and absolutely compatible with Vim on other operating systems.

2. Download files

[vms-download](#)

You can download the Vim source code by ftp from the official Vim site:

<ftp://ftp.vim.org/pub/vim/>

Or use one of the mirrors:

<ftp://ftp.vim.org/pub/vim/MIRRORS>

You can download precompiled executables from:

<http://www.polarhome.com/vim/>

<ftp://ftp.polarhome.com/pub/vim/>

To use the precompiled binary version, you need one of these archives:

vim-XX-exe-x86-gui.zip	X86_64 GUI/Motif executables
vim-XX-exe-x86-term.zip	X86_64 console executables
vim-XX-exe-ia64-gui.zip	IA64 GUI/Motif executables
vim-XX-exe-ia64-gtk.zip	IA64 GUI/GTK executables
vim-XX-exe-ia64-term.zip	IA64 console executables
vim-XX-exe-axp-gui.zip	Alpha GUI/Motif executables
vim-XX-exe-axp-gtk.zip	Alpha GUI/GTK executables
vim-XX-exe-axp-term.zip	Alpha console executables

vim-XX-exe-vax-gui.zip VAX GUI executables
vim-XX-exe-vax-term.zip VAX console executables

and of course (optional)
vim-XX-runtime.zip runtime files

The binary archives contain: vim.exe, ctags.exe, xxd.exe files.

For GTK executables you will need GTKLIB that is available for Alpha and IA64 platforms.

3. Compiling

vms-compiling

See the file [.SRC]INSTALLVMS.TXT.

4. Problems

vms-problems

The code has been tested under Open VMS 6.2 - 9.2 on Alpha, VAX, IA64 and X86_64 platforms with the DEC C compiler. It should work without major problems. If your system does not have some include libraries you can tune in the OS_VMS_CONF.H file.

If you decided to build Vim with +perl, +python, etc. options, first you need to download OpenVMS distributions of Perl and Python. Build and deploy the libraries and change adequate lines in MAKE_VMS.MMS file. There should not be a problem from Vim side.

Also GTK, XPM library paths should be configured in MAKE_VMS.MMS

Note: Under VAX it should work with the DEC C compiler without problems. The VAX C compiler is not fully ANSI C compatible in pre-processor directives semantics, therefore you have to use a converter program that will do the lion part of the job. For detailed instructions read file INSTALLvms.txt

To build XXD.EXE, you should change to the subdirectory and build it separately.

CTAGS is not part of the Vim source distribution anymore, however the OpenVMS specific source might contain CTAGS source files as described above.

You can find more information about CTAGS on VMS at <http://www.polarhome.com/ctags/>

Advanced users may try some acrobatics in FEATURE.H file as well.

It is possible to compile with +xfontset +xim options too, but then you have to set up GUI fonts etc. correctly. See :help xim from Vim command prompt.

You may want to use GUI with GTK icons, then you have to download and install GTK for OpenVMS or at least runtime shareable images - LIBGTK from polarhome.com

Post 7.2 Vim uses GTK2+ while the last GTK on OpenVMS is 1.2.10, therefore

the GTK build is no longer available.

For more advanced questions, please send your problem to Vim on VMS mailing list <vim-vms@polarhome.com>

More about the vim-vms list can be found at:

<http://www.polarhome.com/mailman/listinfo/vim-vms>

5. Deploy

vms-deploy

Vim uses a special directory structure to hold the document and runtime files:

```
vim (or wherever)
|- tmp
|- vim57
|----- doc
|----- syntax
|- vim62
|----- doc
|----- syntax
|- vim64
|----- doc
|----- syntax
vimrc      (system rc files)
gvimrc
```

Use:

```
define/nolog VIM          device:[path.vim]
define/nolog VIMRUNTIME  device:[path.vim.vim60]
define/nolog TMP         device:[path.tmp]
```

To get vim.exe to find its document, filetype, and syntax files, and to specify a directory where temporary files will be located. Copy the "runtime" subdirectory of the Vim distribution to vimruntime.

Logicals \$VIMRUNTIME and \$TMP are optional.

If \$VIMRUNTIME is not set, Vim will guess and try to set up automatically. Read more about it at :help runtime

If \$TMP is not set, you will not be able to use some functions as CTAGS, XXD, printing etc. that use temporary directory for normal operation. The \$TMP directory should be readable and writable by the user(s). The easiest way to set up \$TMP is to define a logical:

```
define/nolog TMP SYS$SCRATCH
or as:
define/nolog TMP SYS$LOGIN
```

6. Practical usage

vms-usage

Usually, you want to run just one version of Vim on your system, therefore it is enough to dedicate one directory for Vim. Copy the whole Vim runtime directory structure to the deployment position. Add the following lines to your LOGIN.COM (in SYS\$LOGIN directory). Set up the logical \$VIM as:

```
$ define VIM device:<path>
```

Set up some symbols:

```
$ ! vi starts Vim in chr. mode.
$ vi*m  ::= mcr VIM:VIM.EXE

$ !gvi starts Vim in GUI mode.
$ gv*im ::= spawn/nowait mcr VIM:VIM.EXE -g
```

Please, check the notes for customization and configuration of symbols.

You may want to create .vimrc and .gvimrc files in your home directory (SYS\$LOGIN) to overwrite default settings.

The easiest way is just rename example files. You may leave the menu file (MENU.VIM) and files vimrc and gvimrc in the original \$VIM directory. It will be the default setup for all users, and for users it is enough to just have their own additions or resetting in their home directory in files .vimrc and .gvimrc. It should work without problems.

Note: Remember, system rc files (default for all users) don't have a leading ".". So, system rc files are:

```
$VIM:vimrc
$VIM:gvimrc
$VIM:menu.vim
```

and user customized rc files are:

```
sys$login:.vimrc
sys$login:.gvimrc
```

You can check that everything is at the right place with the :version command.

Example LOGIN.COM:

```
$ define/nolog VIM DKA0:[UTIL.VIM81]
$ vi*m  ::= mcr VIM:VIM.EXE
$ gv*im::= spawn/nowait/input=NLA0 mcr VIM:VIM.EXE -g -GEOMETRY 80x40
$ set term/inq/ins ! inquire the terminal capabilities
$ set disp/create/node=192.168.10.202/trans=tcpip
```

Note: This set-up should be enough, if you are working on a standalone server or clustered environment, but if you want to use Vim as an internode editor in DECNET environment, it will satisfy as well.

You just have to define the "whole" path:

```
$ define VIM "<server_name>[\"user password\"]::device:<path>"
$ vi*m ::= "mcr VIM:VIM.EXE"
```

For example:

```
$ define VIM "PLUTO::RF10:[UTIL.VIM]"
$ define VIM "PLUTO\"ZAY mypass\"::RF10:[UTIL.VIM]" ! if passwd required
```

You can also use the \$VIMRUNTIME logical to point to the proper version of Vim if you have installed more versions at the same time. If \$VIMRUNTIME is not defined Vim will borrow its value from the \$VIM logical. You can find more information about the \$VIMRUNTIME logical by typing :help runtime as a Vim command.

System administrators might want to set up a system wide Vim installation, then add to the SYS\$STARTUP:SYLOGICALS.COM

```
$ define/nolog/sys VIM device:<path>
$ define/nolog/sys TMP SYS$SCRATCH
```

And to the SYS\$STARTUP:SYLOGIN.COM

```
$ vi*m ::= mcr VIM:VIM.EXE
$ gv*im:: spawn/nowait/input=NLA0 mcr VIM:VIM.EXE -g -GEOMETRY 80x40
```

It will set up a normal Vim work environment for every user on the system.

IMPORTANT: Vim on OpenVMS (and on other case insensitive system) command line parameters are assumed to be lowercase. In order to indicate that a command line parameter is uppercase "/" sign must be used.

Examples:

```
vim -R filename ! means: -r List swap files and exit
vim -/r filename ! means: -R Readonly mode (like "view")
vim -u <vimrc> ! means: -u Use <vimrc> instead of any .vimrc
vim -/u <gvimrc> ! means: -U Use <gvimrc> instead of any .gvimrc
```

7. GUI mode questions

vms-gui

OpenVMS is a real mainframe OS, therefore even if it has a GUI console, most of the users do not use a native X/Window environment during normal operation. It is not possible to start Vim in GUI mode "just like that". But anyhow it is not too complicated either.

First of all: you will need an executable that is built with the GUI enabled.

Second: you need to have installed DECW/Motif on your VMS server, otherwise you will get errors that some shareable libraries are missing.

Third: If you choose to run Vim with extra features such as GUI/GTK then you need a GTK installation too or at least a GTK runtime environment (LIBGTK can be downloaded from <http://www.polarhome.com/vim/>).

- 1) If you are working on the VMS X/Motif console:
Start Vim with the command:

```
$ mc device:<path>VIM.EXE -g
```

or type :gui as a command to the Vim command prompt. For more info :help gui

- 2) If you are working on some other X/Window environment like Unix or a remote X VMS console. Set up display to your host with:

```
$ set disp/create/node=<your IP address>/trans=<transport-name>
```

and start Vim as in point 1. You can find more help in VMS documentation or type: help set disp in VMS prompt.

Examples:

```
$ set disp/create/node=192.168.5.159           ! default trans is DECnet
$ set disp/create/node=192.168.5.159/trans=tcpip ! TCP/IP network
$ set disp/create/node=192.168.5.159/trans=local ! display on the same node
```

Note: you should define just one of these.

For more information type \$help set disp in VMS prompt.

- 3) Another elegant solution is XDM if you have installed on OpenVMS box.
It is possible to work from XDM client as from GUI console.
- 4) If you are working on MS-Windows or some other non X/Window environment you need to set up one X server and run Vim as in point 2.
For MS-Windows there are available free X servers as MIX, Omni X etc., as well as excellent commercial products as eXcursion or ReflectionX with built-in DEC support.

Please note, that executables without GUI are slightly faster during startup than with enabled GUI in character mode. Therefore, if you do not use GUI features, it is worth to choose non GUI executables.

=====

8. Useful notes

vms-notes

- 8.1 Backspace/delete
- 8.2 Filters
- 8.3 VMS file version numbers
- 8.4 Directory conversion
- 8.5 Remote host invocation
- 8.6 Terminal problems
- 8.7 Hex-editing and other external tools
- 8.8 Sourcing vimrc and gvimrc
- 8.9 Printing from Vim

- 8.10 Setting up the symbols
- 8.11 diff and other GNU programs
- 8.12 diff-mode
- 8.13 Allow '\$' in C keywords
- 8.14 VIMTUTOR for beginners
- 8.15 Slow start in console mode issue
- 8.16 Common VIM directory - different architectures

8.1 Backspace/delete

There are backspace/delete key inconsistencies with VMS.

:fixdel doesn't do the trick, but the solution is (without "<" in 'cpo'):

```
:inoremap <C-?> <C-H>    " for terminal mode
:inoremap <Del> <C-H>    " for gui mode
```

Read more in ch: 8.6 (Terminal problems).

(Bruce Hunsaker <BNHunsaker@chq.byu.edu> Vim 5.3)

8.2 Filters

Vim supports filters, i.e., if you have a sort program that can handle input/output redirection like Unix (<infile >outfile), you could use

```
:map \s @!'aqsort<CR>
```

(Charles E. Campbell, Jr. <cec@gryphon.gsfc.nasa.gov> Vim 5.4)

8.3 VMS file version numbers

Vim is saving files into a new file with the next higher file version number, try these settings.

```
:set nobackup      " does not create *.*_ backup files
:set nowritebackup " does not have any purpose on VMS. It's the
                  " default.
```

Recovery is working perfectly as well from the default swap file.

Read more with :help swapfile

(Claude Marinier <ClaudeMarinier@xwavesolutions.com> Vim 5.5, Zoltan Arpadffy Vim 5.6)

8.4 Directory conversion

Vim will internally convert any unix-style paths and even mixed unix/VMS paths into VMS style paths. Some typical conversions resemble:

```
/abc/def/ghi      -> abc:[def]ghi.
/abc/def/ghi.j    -> abc:[def]ghi.j
/abc/def/ghi.j;2  -> abc:[def]ghi.j;2
```

/abc/def/ghi/jkl/mno	-> abc:[def.ghi.jkl]mno.
abc:[def.ghi]jkl/mno	-> abc:[def.ghi.jkl]mno.
./	-> current directory
../	-> relative parent directory
[.def.ghi]	-> relative child directory
./def/ghi	-> relative child directory

Note: You may use <, > brackets as well (device:<path>file.ext;version) as
rf10:<user.zay.work>test.c;1

(David Elins <delins@foliage.com>, Jerome Lauret
<JLAURET@mail.chem.sunysb.edu> Vim 5.6)

8.5 Remote host invocation

It is possible to use Vim as an internode editor.

1. Edit some file from remote node:

```
vi "<server>""username passwd""::<device>:<path><filename>;<version>"
```

Example:

```
vi "pluto""zay passwd""::RF10:<USER.ZAY.WORK>TEST.C;1"
```

Note: syntax is very important, otherwise VMS will recognize more parameters instead of one (resulting with: file not found)

2. Set up Vim as your internode editor. If Vim is not installed on your host, just set up your IP address, the full Vim path including the server name and run the command procedure below:

```
$ if (p1 .eqs. "") .OR. (p2 .eqs. "") then goto usage
$ set disp/create/node=<your_IP_here>/trans=tcpip
$ define "VIM "<vim_server>""'p1' 'p2'""::<device>:<vim_path>"
$ vi*m := "mcr VIM:VIM.EXE"
$ gv*im := "spawn/nowait mcr VIM:VIM.EXE -g"
$ goto end
$ usage:
$ write sys$output " Please enter username and password as a parameter."
$ write sys$output " Example: @SETVIM.COM username passwd"
$ end:
```

Note: Never use it in a clustered environment (you do not need it), loading could be very-very slow, but even faster than a local Emacs. :-)

(Zoltan Arpadffy, Vim 5.6)

8.6 Terminal problems

If your terminal name is not known to Vim and it is trying to find the default one you will get the following message during start-up:

```
---
Terminal entry not found in termcap
```

```
'unknown-terminal' not known. Available built-in terminals are:
  builtin_gui
  builtin_riscos
  builtin_amiga
  builtin_ansi
  builtin_vt320
  builtin_vt52
  builtin_pcansi
  builtin_win32
  builtin_xterm
  builtin_iris-ansi
  builtin_debug
  builtin_dumb
defaulting to 'vt320'
---
```

Try to force to inquire the terminal capabilities with:

```
$ set term/inquire
```

If the inquire did not help, the solutions is to define the default terminal name:

```
$ ! unknown terminal name. Let us use vt320 or ansi instead.
$ ! Note: it's case sensitive
$ define term "vt320"
```

Terminals from VT100 to VT320 (as V300, VT220, VT200) do not need any extra keyboard mappings. They should work perfectly as they are, including arrows, Ins, Del buttons etc., except Backspace in GUI mode. To solve it, add to .gvimrc:

```
inoremap <Del> <BS>
```

Vim will also recognize that they are fast terminals.

If you're using Vim on remote host or through a very slow connection, you might want to reset fast terminal option with:

```
set nottyfast " set terminal to slow mode
```

8.7 Hex-editing and other external tools

A very important difference between OpenVMS and other systems is that VMS uses special commands to execute executables:

```
RUN <path>filename
MCR <path>filename <parameters>
```

OpenVMS users always have to be aware that the Vim command :! "just" drop them to DCL prompt. This feature is possible to use without any problem with all DCL commands, but if we want to execute some programs such as XXD, CTAGS, JTAGS, etc. we're running into trouble if we follow the Vim documentation (see: help xxd).

Solution: Execute with the MC command and add the full path to the executable.
Example: Instead of :%!xxd command use:

```
:%!mc vim:xxd
```

... or in general:

```
:!mc <path>filename <parameters>
```

Note: You can use XXD and CTAGS from GUI menu.

To customize ctags it is possible to define the logical \$CTAGS with standard parameters as:

```
define/nolog CTAGS "--totals -o sys$login:tags"
```

For additional information, please read :help tagsearch and CTAGS documentation at <http://ctags.sourceforge.net/ctags.html>.

(Zoltan Arpadffy, Vim 5.6-70)

8.8 Sourcing vimrc and gvimrc

If you want to use your .vimrc and .gvimrc from other platforms (e.g. Windows) you can get in trouble if you ftp that file(s): VMS has different end-of-line indication.

The symptom is that Vim is not sourcing your .vimrc/.gvimrc, even if you say:

```
:so sys$login:.vimrc
```

One trick is to compress (e.g. zip) the files on the other platform and uncompress it on VMS; if you have the same symptom, try to create the files with copy-paste (for this you need both op. systems reachable from one machine, e.g. an Xterm on Windows or telnet to Windows from VMS).

(Sandor Kopanyi, <sandor.kopanyi@mailbox.hu> Vim 6.0a)

8.9 Printing from Vim

To be able to print from Vim (running in GUI mode) under VMS you have to set up \$TMP logical which should point to some temporary directory and logical SYS\$PRINT to your default print queue.

Example:

```
$define SYS$PRINT HP5ANSI
```

You can print out the whole buffer or just the marked area.
More info under :help hardcopy

(Zoltan Arpadffy, Vim 6.0c)

8.10 Setting up the symbols

When I use gvim this way and press **CTRL-Y** in the parent terminal, gvim exits. I now use a different symbol that seems to work OK and fixes the problem. I suggest this instead:

```
$ GV*IM:==SPAWN/NOWAIT/INPUT=NLA0: MCR VIM:VIM.EXE -G -GEOMETRY 80X40
```

The /INPUT=NLA0: separates the standard input of the gvim process from the parent terminal, to block signals from the parent window. Without the -GEOMETRY, the gvim window size will be minimal and the menu will be confused after a window-resize.

(Carlo Mekenkamp, Coen Engelbarts, Vim 6.0ac)

8.11 diff and other GNU programs

From 6.0 diff functionality has been implemented, but OpenVMS does not use GNU/Unix like diff therefore built in diff does not work. There is a simple solution to solve this anomaly. Install a Unix like diff and Vim will work perfectly in diff mode too. You just have to redefine your diff program as:

```
define /nolog diff <GNU_PATH>diff.exe
```

Another, more sophisticated solution is described below (8.12 diff-mode) There are other programs such as patch, make etc that may cause the same problems. At www.polarhome.com is possible to download an GNU package for Alpha and VAX boxes that is meant to solve GNU problems on OpenVMS. (Zoltan Arpadffy, Vim 6.1)

8.12 diff-mode

Vim 6.0 and higher supports Vim diff-mode (See [new-diff-mode](#) , [diff-mode](#) and [08.7](#)). This uses the external program '**diff**' and expects a Unix-like output format from diff. The standard VMS diff has a different output format. To use Vim on VMS in diff-mode, you need to:

- 1 Install a Unix-like diff program, e.g. GNU diff
- 2 Tell Vim to use the Unix-like diff for diff-mode.

You can download GNU diff from the VIM-VMS website, it is one of the GNU tools in http://www.polarhome.com/vim/files/gnu_tools.zip. I suggest to unpack it in a separate directory "GNU" and create a logical GNU: that points to that directory, e.g:

```
DEFINE GNU <DISK>:[<DIRECTORY>.BIN.GNU]
```

You may also want to define a symbol GDIFF, to use the GNU diff from the DCL prompt:

```
GDIFF := $GNU:DIFF.EXE
```

Now you need to tell Vim to use the new diff program. Take the example settings from `diff-diffexpr` and change the call to the external diff program to the new diff on VMS. Add this to your `.vimrc` file:

```
" Set up vimdiff options
if v:version >= 600
  " Use GNU diff on VMS
  set diffexpr=MyDiff()
  function MyDiff()
    let opt = ""
    if &diffopt =~ "icase"
      let opt = opt .. "-i "
    endif
    if &diffopt =~ "iwhite"
      let opt = opt .. "-b "
    endif
    silent execute "!mc GNU:diff.exe -a " .. opt .. v:fname_in .. " " .. v:fname_out
      \ " > " .. v:fname_out
  endfunction
endif
```

You can now use Vim in diff-mode, e.g. to compare two files in read-only mode:

```
$ VIM -D/R <FILE1> <FILE2>
```

You can also define new symbols for vimdiff, e.g.:

```
$ VIMDIFF      := 'VIM' -D/R
$ GVIMDIFF     := 'GVIM' -D/R
```

You can now compare files in 4 ways:

1. VMS diff: \$ DIFF <FILE1> <FILE2>
2. GNU diff: \$ GDIFF <FILE1> <FILE2>
3. VIM diff: \$ VIMDIFF <FILE1> <FILE2>
4. GVIM diff: \$ GVIMDIFF <FILE1> <FILE2>

(Coen Engelbarts, Vim 6.1)

8.13 Allow '\$' in C keywords

DEC C uses many identifiers with '\$' in them. This is not allowed in ANSI C, and Vim recognises the '\$' as the end of the identifier. You can change this with the `'iskeyword'` option.

Add this command to your `.vimrc` file:

```
autocmd FileType c,cpp,cs set iskeyword+=
```

You can also create the file(s) `$VIM/FTPLUGIN/C.VIM` (and/or `CPP.VIM` and `CS.VIM`) and add this command:

```
set iskeyword+=
```

Now word-based commands, e.g. the '*'-search-command and the **CTRL-]** tag-lookup, work on the whole identifier. (Ctags on VMS also supports '\$' in C keywords since ctags version 5.1.)

(Coen Engelbarts, Vim 6.1)

8.14 VIMTUTOR for beginners

The VIMTUTOR.COM DCL script can help Vim beginners to learn/make their first steps with Vim on OpenVMS. Depending of binary distribution you may start it with:

```
@vim:vimtutor
```

(Thomas.R.Wyant III, Vim 6.1)

8.16 Slow start in console mode issue

As GUI/GTK Vim works equally well in console mode, many administrators deploy those executables system wide. Unfortunately, on a remote slow connections GUI/GTK executables behave rather slow when user wants to run Vim just in the console mode - because of X environment detection timeout.

Luckily, there is a simple solution for that. Administrators need to deploy both GUI/GTK build and just console build executables, like below:

```
| - vim73
| ----- doc
| ----- syntax
  vimrc      (system rc files)
  gvimrc
  gvim.exe  (the renamed GUI or GTK built vim.exe)
  vim.exe   (the console only executable)
```

Define system symbols like below in for ex in LOGIN.COM or SYLOGIN.COM:

```
$ define/nolog VIM RF10:[UTIL.VIM73] ! where you VIM directory is
$ vi*m  := mcr VIM:VIM.EXE
$ gvi*m := mcr VIM:GVIM.EXE
$ ! or you can try to spawn with
$ gv*im := spawn/nowait/input=NLA0 mcr VIM:GVIM.EXE -g -GEOMETRY 80x40
```

Like this, users that do not have X environment and want to use Vim just in console mode can avoid performance problems.

(Zoltan Arpadffy, Vim 7.2)

8.15 Common VIM directory - different architectures

In a cluster that contains nodes with different architectures like below:

```
$show cluster
```

```
View of Cluster from system ID 11655  node: TOR
```

SYSTEMS		MEMBERS
NODE	SOFTWARE	STATUS
TOR	VMS V7.3-2	MEMBER
TITAN2	VMS V8.3	MEMBER
ODIN	VMS V7.3-2	MEMBER

It is convenient to have a common VIM directory but execute different executables.

There are several solutions for this problem:

Solution 1. All executables in the same directory with different names
This is easily done with the following script that can be added
to the login.com or sylogin.com:

```
$ if f$getsyi("NODE_HWTYPE") .eqs. "VAX"
$ then
$     say "VAX platform"
$     vi*m:= mcr vim:VIM.EXE_VAX
$ endif
$ if f$getsyi("NODE_HWTYPE") .eqs. "ALPH"
$ then
$     say "ALPHA platform"
$     vi*m := mcr vim:VIM.EXE_AXP
$ endif
$ if f$getsyi("ARCH_NAME") .eqs. "IA64"
$ then
$     say "IA64 platform"
$     vi*m := mcr vim:VIM.EXE_IA64
$ endif
```

Solution 2. Different directories:

```
$ if f$getsyi("NODE_HWTYPE") .eqs. "VAX"
$ then
$     say "VAX platform"
$     define/nolog VIM RF10:[UTIL.VAX_EXE] ! VAX executables
$ endif
$ if f$getsyi("NODE_HWTYPE") .eqs. "ALPH"
$ then
$     say "ALPHA platform"
$     define/nolog VIM RF10:[UTIL.AXP_EXE] ! AXP executables
$ endif
$ if f$getsyi("ARCH_NAME") .eqs. "IA64"
$ then
$     say "IA64 platform"
$     define/nolog VIM RF10:[UTIL.IA64_EXE] ! IA64 executables
$ endif
$! VIMRUNTIME must be defined in order to find runtime files
```


`$ define/nolog VIMRUNTIME RF10:[UTIL.VIM73]`

A good example for this approach is the [\[GNU\]gnu_tools.com](http://www.gnu.org/software/gnu_tools/) script from GNU_TOOLS.ZIP package downloadable from <http://www.polarhome.com/vim/>

(Zoltan Arpadffy, Vim 7.2)

9. VMS related changes

vms-changes

Version 9.0 (2023 Nov 27)

- Vim is ported to the X86_64 architecture
 - IMPORTANT: because of the getline function name used in structs like in ex_cmds.h on X86_64 the CRTL_VER is kept under 80500000 level. The proper solution would be to rename the getline function to something else in the struct (and in all places it is used) - and avoiding to use POSIX functions in structs, but this change would impact on all other operating systems. (added to the VMS TODO list)
 - Read more about at <https://forum.vmssoftware.com/viewtopic.php?f=38&t=8914&p=20049>
- os_vms_conf.h includes have been reviewed for all architectures
- added support for the MODIFIED_BY define

Version 8.2 (2020 Feb 6)

- make all changes needed for clean compile build of v8.2 on VMS on all platforms
- fix the call mkdir bug (vicente_polo@yahoo.es)
- test on VSI OpenVMS Alpha and Itanium platforms
- added LUA support
- added XPM support - Motif GUI with toolbar on all platforms
- XPM v3.4.11 libraries for IA64, AXP and VAX are added
- start integrating the new test scripts

Version 8.1 (2019 Jan 9)

- make necessary changes to build v8.1 on VMS
- GTK1.2.10 on VAX

Version 8.0 (2016 Nov 21)

- solve the 100% cpu usage issue while waiting for a keystroke
- correct the VMS warnings and errors around handling the INFINITY (used in json.c)
- minor VMS port related changes
- correct the make_vms.mms file for 8.0
- fix [.TESTDIR]make_vms.mms for 8.0

Version 7.4 (2013 Aug 10)

- Undo: VMS can not handle more than one dot in the filenames use "dir/name" -> "dir/_un_name" add _un_ at the beginning to keep the extension
- correct swap file name wildcard handling
- handle iconv usage correctly
- do not optimize on vax - otherwise it hangs compiling crypto files
- fileio.c fix the comment
- correct RealWaitForChar
- after 7.4-119 use different functions lib\$cvtf_to_internal_time because Alpha and VAX have G_FLOAT but IA64 uses IEEE float otherwise Vim crashes
- guard against crashes that are caused by mixed filenames
- [TESTDIR]make_vms.mms changed to see the output files

- Improve tests, update known issues
- minor compiler warnings fixed
- CTAGS 5.8 +regex included

Version 7.3 (2010 Aug 15)

- CTAGS 5.8 included
- VMS compile warnings fixed - floating-point overflow warning corrected on VAX
- filepath completion corrected - too many chars were escaped in filename and shell commands
- the following plugins are included into VMS runtime:
 - genutils 2.4, multiselect 2.2, multvals 3.1, selectbuf 4.3,
 - bufexplorer 7.1.7, taglist 4.5
- minor changes in vimrc (just in VMS runtime)
- make_vms.mms - HUGE model is the default
- [TESTDIR]make_vms.mms include as many tests possible
- modify test30 and test54 for VMS
- enable FLOAT feature in VMS port
- os_vms.txt updated

Version 7.2 (2008 Aug 9)

- VCF files write corrected
- CTAGS 5.7 included
- corrected make_vms.mms (on VAX gave syntax error)

Version 7.1 (2007 Jun 15)

- create TAGS file from menu

Version 7 (2006 May 8)

- Improved low level char input (affects just console mode)
- Fixed plugin bug
- CTAGS 5.6 included

Version 6.4 (2005 Oct 15)

- GTKLIB and Vim build on IA64
- colors in terminal mode
- syntax highlighting in terminal mode
- write problem fixed (extra CR)
- ESC and ESC sequence recognition in terminal mode
- make file changed to support new MMS version
- env variable expansion in path corrected
- printing problems corrected
- help text added for case insensitive arguments

Version 6.3 (2004 May 10)

- Improved vms_read function
- CTAGS v5.5.4 included
- Documentation corrected and updated

Version 6.2 (2003 May 7)

- Corrected VMS system call results
- Low level character input is rewritten
- Correction in tag and quickfix handling
- First GTK build
- Make file changes

- GTK feature added
- Define for OLD_VMS
- OpenVMS version 6.2 or older
- Documentation updated with GTK features
- CTAGS v5.5 included
- VMS VIM tutor created

Version 6.1 (2002 Mar 25)

- TCL init_tcl() problem fixed
- CTAGS v5.4 included
- GNU tools binaries for OpenVMS
- Make file changes
 - PERL, PYTHON and TCL support improved
 - InstallVMS.txt has a detailed description HOWTO build
- VMS/Unix file handling rewritten
- Minor casting and bug fixes

Version 6.0 (2001 Sep 28)

- Unix and VMS code has been merged
 - separated "really" VMS related code
 - included all possible Unix functionality
 - simplified or deleted the configuration files
 - makefile MAKE_VMS.MMS reviewed
- menu changes (fixed printing, CTAGS and XXD usage)
- fixed variable RMS record format handling anomaly
- corrected syntax, ftplugin etc files load
- changed expand_wildcards and expandpath functions to work more general
- created OS_VMS_FILTER.COM - DECC->VAXC pre-processor directive convert script.
- Improved code's VAXC and new DECC compilers compatibility
- changed quickfix parameters:
 - errormessage format to suite DECC
 - search, make and other commands to suite VMS system
- updated and renamed MMS make files for Vim and CTAGS.
- CTAGS has been removed from source distribution of Vim but it will remain in OpenVMS binary distributions.
- simplified build/configuration procedure
- created INSTALLvms.txt - detailed compiling instructions under VMS.
- updated test scripts.

Version 5.8 (2001 Jun 1)

- OS_VMS.TXT updated with new features.
- other minor fixes.
- documentation updated
- this version had been tested much more than any other OpenVMS version earlier

Version 5.7 (2000 Jun 24)

- New CTAGS v5.0 in distribution
- Documentation updated

Version 5.6 (2000 Jan 17)

- VMS filename related changes:
 - version handling (open everything, save to new version)

- correct file extension matching for syntax (version problem)
- handle <, > characters and passwords in directory definition
- handle internode/remote invocation and editing with passwords
- OpenVMS files will be treated case insensitive from now
- corrected response of expand("%:.") etc path related functions (in one word: VMS directory handling internally)
- version command
 - corrected (+,-) information data
 - added compiler and OS version
 - added user and host information
 - resolving \$VIM and \$VIMRUNTIME logicals
- VMS port is in MAX_FEAT (maximum features) club with Unix, Win32 and OS/2.
 - enabled farsi, rightleft etc. features
 - undo level raised up to 1000
- Updated OS_VMS.MMS file.
 - maximum features ON is default
 - Vim is compilable with +perl, +python and +tcl features.
 - improved MMK compatibility
- Created MAKEFILE_VMS.MMS, makefile for testing Vim during development.
- Defined DEC terminal VT320
 - compatibility for VT3*0, VT2*0 and VT1*0 - ANSI terminals backwards, but not VT340 and newer with colour capability.
 - VT320 is default terminal for OpenVMS
 - these new terminals are also fast ttys (default for OpenVMS).
 - allowed dec_mouse tty
- Updated files vimrc and gvimrc with VMS specific suggestions.
- OS_VMS.TXT updated with new features.

Version 5.5 (1999 Dec 3)

- Popup menu line crash corrected.
- Handle full file names with version numbers.
- Directory handling (CD command etc.)
- Corrected file name conversion VMS to Unix and v.v.
- Correct response of expand wildcards
- Recovery is working from this version under VMS as well.
- Improved terminal and signal handing.
- Improved OS_VMS.TXT

Version 5.4 (1999 Sep 9)

- Cut and paste mismatch corrected.
- Motif directories during open and save are corrected.

Version 5.3 (1998 Oct 12)

- Minor changes in the code
- Standard distribution with +GUI option

Version 5.1 (1998 Apr 21)

- Syntax and DEC C changes in the code
- Fixing problems with the /doc subdirectory
- Improve OS_VMS.MMS

Version 4.5 (1996 Dec 16)

- First VMS port by Henk Elbers <henk@xs4all.nl>

10. Authors

vms-authors

OpenVMS documentation and executables are maintained by:

Zoltan Arpadffy <zoltan.arpadffy@gmail.com>

OpenVMS Vim page: <http://www.polarhome.com/vim/>

This document uses parts and remarks from earlier authors and contributors of OS_VMS.TXT:

Charles E. Campbell, Jr. <cec@gryphon.gsfc.nasa.gov>

Bruce Hunsaker <BNHunsaker@chq.byu.edu>

Sandor Kopanyi <sandor.kopanyi@mailbox.hu>

VIM REFERENCE MANUAL by George Reilly

win32 Win32 MS-Windows

This file documents the idiosyncrasies of the Win32 version of Vim.

The Win32 version of Vim works on Windows 7, 8, 10 and 11. There are both console and GUI versions.

If you have Windows XP or Vista then Vim 9.0 up to patch level 495 can be used.

The 32 bit version also runs on 64 bit MS-Windows systems.

- 1. Known problems [win32-problems](#)
- 2. Startup [win32-startup](#)
- 3. Restore screen contents [win32-restore](#)
- 4. Using the mouse [win32-mouse](#)
- 5. Running under Windows 95 [win32-win95](#)
- 6. Running under Windows 3.1 [win32-win3.1](#)
- 7. Installation package [win32-installer](#)
- 8. Win32 mini FAQ [win32-faq](#)

Additionally, there are a number of common Win32 and DOS items:

- File locations [dos-locations](#)
- Using backslashes [dos-backslash](#)
- Standard mappings [dos-standard-mappings](#)
- Screen output and colors [dos-colors](#)
- File formats [dos-file-formats](#)
- :cd command [dos-:cd](#)
- Interrupting [dos-CTRL-Break](#)
- Temp files [dos-temp-files](#)
- Shell option default [dos-shell](#)
- PowerShell defaults [dos-powershell](#)

Win32 GUI [gui-w32](#)

Credits:

The Win32 version was written by George V. Reilly <george@reilly.org>. The original Windows NT port was done by Roger Knobbe <RogerK@wonderware.com>. The GUI version was made by George V. Reilly and Robert Webb.

For compiling see "src/INSTALLpc.txt". [win32-compiling](#)

WSL

When using Vim on WSL (Windows Subsystem for Linux) the remarks here do not apply, `has('win32')` will return false then. In case you need to know whether Vim is running on WSL you can use `exists('\$WSLENV')`.

=====

- 1. Known problems [win32-problems](#)

When doing file name completion, Vim also finds matches for the short file name. But Vim will still find and use the corresponding long file name. For example, if you have the long file name "this_is_a_test" with the short file name "this_i~1", the command ":e *1" will start editing "this_is_a_test".

2. Startup

win32-startup

Current directory

win32-curdir

If Vim is started with a single file name argument, and it has a full path (starts with "x:\"), Vim assumes it was started from the file explorer and will set the current directory to where that file is. To avoid this when typing a command to start Vim, use a forward slash instead of a backslash. Example:

```
vim c:\text\files\foo.txt
```

Will change to the "C:\text\files" directory.

```
vim c:/text/files/foo.txt
```

Will use the current directory.

Term option

win32-term

The only kind of terminal type that the Win32 version of Vim understands is "win32", which is built-in. If you set '**term**' to anything else, you will probably get very strange behavior from Vim. Therefore Vim does not obtain the default value of '**term**' from the environment variable "TERM".

\$PATH

win32-PATH

The directory of the Vim executable is appended to \$PATH. This is mostly to make "!xd" work, as it is in the Tools menu. And it also means that when executable() returns 1 the executable can actually be executed.

Command line arguments

win32-cmdargs

Analysis of a command line into parameters is not standardised in MS-Windows. Vim and gvim used to use different logic to parse it (before 7.4.432), and the logic was also depended on what it was compiled with. Now Vim and gvim both use the CommandLineToArgvW() Win32 API, so they behave in the same way.

The basic rules are:

win32-backslashes

- a) A parameter is a sequence of graphic characters.
- b) Parameters are separated by white space.
- c) A parameter can be enclosed in double quotes to include white space.
- d) A sequence of zero or more backslashes (\) and a double quote (") is special. The effective number of backslashes is halved, rounded down. An even number of backslashes reverses the acceptability of spaces and tabs, an odd number of backslashes produces a literal

double quote.

So:

```
"      is a special double quote
\"     is a literal double quote
\\"    is a literal backslash and a special double quote
\\\\"  is a literal backslash and a literal double quote
\\\\"  is 2 literal backslashes and a special double quote
\\\\"  is 2 literal backslashes and a literal double quote
etc.
```

Example:

```
vim "C:\My Music\freude" +"set ignorecase" +/"\"foo\\" +\"bar\\\\"
```

opens "C:\My Music\freude" and executes the line mode commands:

```
set ignorecase; /"foo\ and /bar\"
```

These rules are also described in the reference of the CommandLineToArgvW API:

<https://msdn.microsoft.com/en-us/library/windows/desktop/bb776391.aspx>

[win32-quotes](#)

There are additional rules for quotes (which are not well documented). As described above, quotes inside a file name (or any other command line argument) can be escaped with a backslash. E.g.

```
vim -c "echo 'foo\"bar'"
```

Alternatively use three quotes to get one:

```
vim -c "echo 'foo\"\"\"bar'"
```

The quotation rules are:

1. A `` starts quotation.
2. Another `` or `\"` ends quotation. If the quotation ends with `\"`, a `` is produced at the end of the quoted string.

Examples, with [] around an argument:

```
"foo"      -> [foo]
"foo\"     -> [foo"]
"foo\"bar  -> [foobar]
"foo\" bar -> [foo], [bar]
"foo\"\"bar -> [foo\"bar]
"foo\"\" bar -> [foo\"], [bar]
"foo\"\"\"bar -> [foo\"bar]
```

=====
3. Restore screen contents

[win32-restore](#)

When **'restorescreen'** is set (which is the default), Vim will restore the original contents of the console when exiting or when executing external commands. If you don't want this, use `":set nors"`. ['restorescreen'](#)

=====
4. Using the mouse

[win32-mouse](#)

The Win32 version of Vim supports using the mouse. If you have a two-button mouse, the middle button can be emulated by pressing both left and right buttons simultaneously - but **note** that in the Win32 GUI, if you have the right mouse button pop-up menu enabled (see '**mouse**'), you should err on the side of pressing the left button first. [mouse-using](#)

When the mouse doesn't work, try disabling the "Quick Edit Mode" feature of the console.

5. Running under Windows 95

[win32-win95](#)

[windows95](#) [windows98](#) [windowsme](#)

Windows 95/98/ME support was removed in patch 8.0.0029. If you want to use it you will need to get a version older than that.

6. Running under Windows 3.1

[win32-win3.1](#)

[win32s](#) [windows-3.1](#) [gui-w32s](#) [win16](#)

There was a special version of gvim that runs under Windows 3.1 and 3.11. Support was removed in patch 7.4.1364.

7. Installation package

[win32-installer](#)

A simple installer for windows is available at <http://www.vim.org/download.php> (stable version) and nightly builds are also available at <https://github.com/vim/vim-win32-installer/releases/>

The nightly builds include 32bit and 64bit builds, have most features enabled and usually also contain an extra cryptographic signed installer, so Windows will not complain.

To use the installer, simply run the exe file. The following switches are also supported:

```
gvim_<version>.exe /S          -> silent install without any dialogues
gvim_<version>.exe /D=C:\vim    -> Install into directory c:\vim
                                -> /D must be the last argument
gvim_<version>.exe /S /D=c:\vim -> silent install into c:\vim
```

The default installation directory can alternatively be given by setting the \$VIM environment variable.

8. Win32 mini FAQ

[win32-faq](#)

Q. How do I change the font?

A. In the GUI version, you can use the '**guifont**' option. Example:

```
:set guifont=Lucida_Console:h15:cDEFAULT
```

In the console version, you need to set the font of the console itself.

You cannot do this from within Vim.

Q. How do I type dead keys on Windows NT?

A. Dead keys work on NT 3.51. Just type them as you would in any other application.

On NT 4.0, you need to make sure that the default locale (set in the Keyboard part of the Control Panel) is the same as the currently active locale. Otherwise the NT code will get confused and crash! This is a NT 4.0 problem, not really a Vim problem.

Q. I'm using Vim to edit a symbolically linked file on a Unix NFS file server. When I write the file, Vim does not "write through" the symlink. Instead, it deletes the symbolic link and creates a new file in its place. Why?

A. On Unix, Vim is prepared for links (symbolic or hard). A backup copy of the original file is made and then the original file is overwritten. This assures that all properties of the file remain the same. On non-Unix systems, the original file is renamed and a new file is written. Only the protection bits are set like the original file. However, this doesn't work properly when working on an NFS-mounted file system where links and other things exist. The only way to fix this in the current version is not making a backup file, by `":set nobackup nowritebackup" 'writebackup'`

Q. I'm using Vim to edit a file on a Unix file server through Samba. When I write the file, the owner of the file is changed. Why?

A. When writing a file Vim renames the original file, this is a backup (in case writing the file fails halfway). Then the file is written as a new file. Samba then gives it the default owner for the file system, which may differ from the original owner.

To avoid this set the `'backupcopy'` option to "yes". Vim will then make a copy of the file for the backup, and overwrite the original file. The owner isn't changed then.

Q. How do I get to see the output of `":make"` while it's running?

A. Basically what you need is to put a tee program that will copy its input (the output from make) to both stdout and to the errorfile. You can find a copy of tee (and a number of other GNU tools) at

<http://gnuwin32.sourceforge.net> or <http://unxutils.sourceforge.net>

Alternatively, try the more recent Cygnus version of the GNU tools at

<http://www.cygwin.com> Other Unix-style tools for Win32 are listed at

http://directory.google.com/Top/Computers/Software/Operating_Systems/Unix/Win32/

When you do get a copy of tee, you'll need to add

```
:set shellpipe=\\| tee
```

to your `_vimrc`.

Q. I'm storing files on a remote machine that works with VisionFS, and files disappear!

A. VisionFS can't handle certain dot (.) three letter extension file names. SCO declares this behavior required for backwards compatibility with 16bit DOS/Windows environments. The two commands below demonstrate the behavior:

```
echo Hello > file.bat~  
dir > file.bat
```

The result is that the "dir" command updates the "file.bat~" file, instead of creating a new "file.bat" file. This same behavior is exhibited in Vim when editing an existing file named "foo.bat" because the default behavior

of Vim is to create a temporary file with a '~' character appended to the name. When the file is written, it winds up being deleted.

Solution: Add this command to your `_vimrc` file:

```
:set backupext=.temporary
```

Q. How do I change the blink rate of the cursor?

A. You can't! This is a limitation of the NT console. NT 5.0 is reported to be able to set the blink rate for all console windows at the same time.

`:!start`

Q. How can I asynchronously run an external command or program, or open a document or URL with its default program?

A. When using `!` to run an external command, you can run it with "start". For example, to run notepad:

```
:!start notepad
```

To open "image.jpg" with the default image viewer:

```
:!start image.jpg
```

To open the folder of the current file in Windows Explorer:

```
:!start %:h
```

To open the Vim home page with the default browser:

```
:!start http://www.vim.org/
```

Using "start" stops Vim switching to another screen, opening a new console, or waiting for the program to complete; it indicates that you are running a program that does not affect the files you are editing. Programs begun with `:!start` do not get passed Vim's open file handles, which means they do not have to be closed before Vim.

To avoid this special treatment, use `":! start"`.

There are two optional arguments (see the next Q):

```
/min the window will be minimized
```

```
/b no console window will be opened
```

You can use only one of these flags at a time. A second one will be treated as the start of the command.

`windows-asynchronously`

Q. How do I avoid getting a window for programs that I run asynchronously?

A. You have three possible solutions depending on what you want:

1) You may use the `/min` flag in order to run program in a minimized state with no other changes. It will work equally for console and GUI applications.

2) You can use the `/b` flag to run console applications without creating a console window for them (GUI applications are not affected). But you should use this flag only if the application you run doesn't require any input. Otherwise it will get an EOF error because its input stream (stdin) would be redirected to `\\.\NUL` (stdout and stderr too).

3) Set the `!!` flag in the `'guioptions'` option `'go-!'`. This will make Vim run the "start" command inside Vims terminal window and not open a console window.

Example for a console application, run Exuberant ctags:

```
:!start /min ctags -R .
```

When it has finished you should see file named "tags" in your current directory. You should notice the window title blinking on your taskbar. This is more noticeable for commands that take longer.

Now delete the "tags" file and run this command:

```
:!start /b ctags -R .
```

You should have the same "tags" file, but this time there will be no blinking on the taskbar.

Example for a GUI application:

```
:!start /min notepad
```

```
:!start /b notepad
```

The first command runs notepad minimized and the second one runs it normally.

windows-icon

- Q. I don't like the Vim icon, can I change it?
- A. Yes, place your favorite icon in `bitmaps/vim.ico` in a directory of `'runtimepath'`. For example `~/vimfiles/bitmaps/vim.ico`.

GETSCRIPT REFERENCE MANUAL by Charles E. Campbell

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
(remove NOSPAM from the email address)

Copyright: (c) 2004-2012 by Charles E. Campbell [GetLatestVimScripts-copyright](#)
[glvs-copyright](#)

The VIM LICENSE (see [copyright](#)) applies to the files in this package, including getscripPlugin.vim, getscrip.vim, GetLatestVimScripts.dist, and pi_getscript.txt, except use "getscrip" instead of "Vim". Like anything else that's free, getscrip and its associated files are provided *as is* and comes with no warranty of any kind, either expressed or implied. No guarantees of merchantability. No guarantees of suitability for any purpose. By using this plugin, you agree that in no event will the copyright holder be liable for any damages resulting from the use of this software. Use at your own risk!

Getscrip is a plugin that simplifies retrieval of the latest versions of the scripts that you yourself use! Typing `:GLVS` will invoke getscrip; it will then use the `<GetLatestVimScripts.dat>` (see [GetLatestVimScripts_dat](#)) file to get the latest versions of scripts listed therein from <http://vim.sf.net/>.

=====

1. Contents	glvs-contents	glvs	getscrip
		GetLatestVimScripts	
1. Contents.....	glvs-contents		
2. GetLatestVimScripts -- Getting Started.....	glvs-install		
3. GetLatestVimScripts Usage.....	glvs-usage		
4. GetLatestVimScripts Data File.....	glvs-data		
5. GetLatestVimScripts Friendly Plugins.....	glvs-plugins		
6. GetLatestVimScripts AutoInstall.....	glvs-autoinstall		
7. GetLatestViMScripts Options.....	glvs-options		
8. GetLatestVimScripts Algorithm.....	glvs-alg		
9. GetLatestVimScripts History.....	glvs-hist		

=====

2. GetLatestVimScripts -- Getting Started	getscrip-start
	getlatestvimscripts-install

VERSION FROM VIM DISTRIBUTION [glvs-dist-install](#)

Vim 7.0 does not include the GetLatestVimScripts.dist file which serves as an example and a template. So, you'll need to create your own! See [GetLatestVimScripts_dat](#) .

VERSION FROM VIM SF NET [glvs-install](#)

NOTE: The last step, that of renaming/moving the GetLatestVimScripts.dist file, is for those who have just downloaded GetLatestVimScripts.tar.bz2 for the first time.

The GetLatestVimScripts.dist file serves as an example and a template for your own personal list. Feel free to remove all the scripts mentioned within it; the "important" part of it is the first two lines.

Your computer needs to have wget or curl for GetLatestVimScripts to do its work.

1. if compressed: `gunzip getscript.vba.gz`
2. Unix:

```
vim getscript.vba
:so %
:q
cd ~/.vim/GetLatest
mv GetLatestVimScripts.dist GetLatestVimScripts.dat
(edit GetLatestVimScripts.dat to install your own personal
list of desired plugins -- see GetLatestVimScripts\_dat )
```
3. Windows:

```
vim getscript.vba
:so %
:q
cd **path-to-vimfiles**/GetLatest
mv GetLatestVimScripts.dist GetLatestVimScripts.dat
(edit GetLatestVimScripts.dat to install your own personal
list of desired plugins -- see GetLatestVimScripts\_dat )
```

3. GetLatestVimScripts Usage glvs-usage :GLVS

Unless it has been defined elsewhere,

```
:GLVS
```

will invoke GetLatestVimScripts(). If some other plugin has defined that command, then you may type

```
:GetLatestVimScripts
```

The script will attempt to update and, if permitted, will automatically install scripts from <http://vim.sourceforge.net/>. To do so it will peruse a file,

```
~/.vim/GetLatest/GetLatestVimScripts.dat (unix)
```

or

```
..wherever..\vimfiles\GetLatest\GetLatestVimScripts.dat (windows)
(see glvs-data ), and examine plugins in your [.\vim|vimfiles]/plugin
directory (see glvs-plugins ).
```

Scripts which have been downloaded will appear in the
~/.vim/GetLatest (unix) or ..wherever..\vimfiles\GetLatest (windows)
subdirectory. GetLatestVimScripts will attempt to automatically
install them if you have the following line in your `<.vimrc>`:


```
let g:GetLatestVimScripts_allowautoinstall=1
```

The `<GetLatestVimScripts.dat>` file will be automatically be updated to reflect the latest version of script(s) so downloaded.
(also see [glvs-options](#))

4. GetLatestVimScripts Data File [getscript-data](#) [glvs-data](#) [:GetLatestVimScripts_dat](#)

The data file `<GetLatestVimScripts.dat>` must have for its first two lines the following text:

```
-----  
ScriptID SourceID Filename  
-----
```

Following those two lines are three columns; the first two are numeric followed by a text column. The `GetLatest/GetLatestVimScripts.dist` file contains an example of such a data file. Anything following a `#...` is ignored, so you may embed comments in the file.

The first number on each line gives the script's ScriptID. When you're about to use a web browser to look at scripts on <http://vim.sf.net/>, just before you click on the script's link, you'll see a line resembling

```
http://vim.sourceforge.net/scripts/script.php?script\_id=40
```

The "40" happens to be a ScriptID that `GetLatestVimScripts` needs to download the associated page, and is assigned by `vim.sf.net` itself during initial uploading of the plugin.

The second number on each line gives the script's SourceID. The SourceID records the count of uploaded scripts as determined by `vim.sf.net`; hence it serves to indicate "when" a script was uploaded. Setting the SourceID to 1 insures that `GetLatestVimScripts` will assume that the script it has is out-of-date.

The SourceID is extracted by `GetLatestVimScripts` from the script's page on `vim.sf.net`; whenever it is greater than the one stored in the `GetLatestVimScripts.dat` file, the script will be downloaded (see [GetLatestVimScripts_dat](#)).

If your script's author has included a special comment line in his/her plugin, the plugin itself will be used by `GetLatestVimScripts` to build your `<GetLatestVimScripts.dat>` file, including any dependencies on other scripts it may have. As an example, consider:

```
" GetLatestVimScripts: 884 1 :AutoInstall: AutoAlign.vim
```

This comment line tells `getscript.vim` to check `vimscrip #884` and that the script is automatically installable. `Getscript` will also use this line to help build the `GetLatestVimScripts.dat` file, by including a line such as:

```
884 1 :AutoInstall: AutoAlign.vim
```

assuming that such a line isn't already in GetLatestVimScripts.dat file. See [glvs-plugins](#) for more. Thus, GetLatestVimScripts thus provides a comprehensive ability to keep your plugins up-to-date!

In summary:

- * Optionally tell getscrip that it is allowed to build/append a GetLatestVimScripts.dat file based upon already installed plugins:

```
let g:GetLatestVimScripts_allowautoinstall=1
```

- * A line such as

```
" GetLatestVimScripts: 884 1 :AutoInstall: AutoAlign.vim
```

in an already-downloaded plugin constitutes the concurrence of the plugin author that getscrip may do AutoInstall. Not all plugins may be AutoInstall-able, and the plugin's author is best situated to know whether or not his/her plugin will AutoInstall properly.

- * A line such as

```
884 1 :AutoInstall: AutoAlign.vim
```

in your GetLatestVimScripts.dat file constitutes your permission to getscrip to do AutoInstall. AutoInstall requires both your and the plugin author's permission. See [GetLatestVimScripts_dat](#) .

[GetLatestVimScripts_dat](#)

As an example of a `<GetLatestVimScripts.dat>` file:

```
ScriptID SourceID Filename
-----
294 1 :AutoInstall: Align.vim
120 2 Decho.vim
 40 3 DrawIt.tar.gz
451 4 EasyAccents.vim
195 5 engspchk.vim
642 6 GetLatestVimScripts.vim
489 7 Manpageview.vim
```

Note: the first two lines are required, but essentially act as comments.

```
=====
5. GetLatestVimScripts Friendly Plugins getscrip-plugins glvs-plugins
      (this section is for plugin authors)
```

If a plugin author includes the following comment anywhere in their plugin, GetLatestVimScripts will find it and use it to automatically build the user's GetLatestVimScripts.dat files:

```
src_id
v
" GetLatestVimScripts: ### ## yourscriptname
```

^
scriptid

As an author, you should include such a line in to refer to your own script plus any additional lines describing any plugin dependencies it may have. Same format, of course!

If your command is auto-installable (see [glvs-autoinstall](#)), and most scripts are, then you may include `:AutoInstall:` just before "yourscriptname":

```
src_id  
  v  
" GetLatestVimScripts: ### ### :AutoInstall: yourscriptname  
  ^  
scriptid
```

NOTE: The `:AutoInstall:` feature requires both the plugin author's and the user's permission to operate!

GetLatestVimScripts commands for those scripts are then appended, if not already present, to the user's GetLatest/GetLatestVimScripts.dat file. It is a relatively painless way to automate the acquisition of any scripts your plugins depend upon.

Now, as an author, you probably don't want GetLatestVimScripts to download your own scripts atop your own copy, thereby overwriting your not-yet-released hard work. GetLatestVimScripts provides a solution for this: `put`

```
@ @ yourscripname
```

into your `<GetLatestVimScripts.dat>` file and GetLatestVimScripts will skip examining the "yourscripname" scripts for those GetLatestVimScripts comment lines. As a result, those lines won't be inadvertently installed into your `<GetLatestVimScripts.dat>` file and subsequently used to download your own scripts. This is especially important to do if you've included the `:AutoInstall:` option.

Be certain to use the same "yourscripname" in the "@ @ yourscripname" line as you've used in your GetLatestVimScripts comment!

```
=====
6. GetLatestVimScripts AutoInstall                                getscrip-autoinstall
                                                                    glvs-autoinstall
```

GetLatestVimScripts now supports "AutoInstall". Not all scripts are supportive of auto-install, as they may have special things you need to do to install them (please refer to the script's "install" directions). On the other hand, most scripts will be auto-installable.

To let GetLatestVimScripts do an autoinstall, the data file's comment field should begin with (surrounding blanks are ignored):

```
:AutoInstall:
```

Both colons are needed, and it should begin the comment (yourscriptname) field.

One may prevent any autoinstalling by putting the following line in your `<.vimrc>`:

```
let g:GetLatestVimScripts_allowautoinstall= 0
```

With `:AutoInstall:` enabled, as it is by default, files which end with

```
---.tar.bz2 : decompressed & untarred in .vim/ directory
---.vba.bz2 : decompressed in .vim/ directory, then vimball handles it
---.vim.bz2 : decompressed & moved into .vim/plugin directory
---.tar.gz  : decompressed & untarred in .vim/ directory
---.vba.gz  : decompressed in .vim/ directory, then vimball handles it
---.vim.gz  : decompressed & moved into .vim/plugin directory
---.vba     : unzipped in .vim/ directory
---.vim     : moved to .vim/plugin directory
---.zip     : unzipped in .vim/ directory
```

and which merely need to have their components placed by the `untar/gunzip` or `move-to-plugin-directory` process should be auto-installable. Vimballs, of course, should always be auto-installable.

When is a script not auto-installable? Let me give an example:

```
.vim/after/syntax/blockhl.vim
```

The `<blockhl.vim>` script provides block highlighting for C/C++ programs; it is available at:

http://vim.sourceforge.net/scripts/script.php?script_id=104

Currently, vim's `after/syntax` only supports by-filetype scripts (in `blockhl.vim`'s case, that's `after/syntax/c.vim`). Hence, auto-install would possibly overwrite the current user's `after/syntax/c.vim` file.

In my own case, I use `<aftersyntax.vim>` (renamed to `after/syntax/c.vim`) to allow a `after/syntax/c/` directory:

http://vim.sourceforge.net/scripts/script.php?script_id=1023

The script allows multiple syntax files to exist separately in the `after/syntax/c` subdirectory. I can't bundle `aftersyntax.vim` in and build an appropriate tarball for auto-install because of the potential for the `after/syntax/c.vim` contained in it to overwrite a user's `c.vim`.

```
=====
7. GetLatestVimScripts Options glvs-options
```

```
g:GetLatestVimScripts_wget
default= "wget"
```

This variable holds the name of the command for obtaining scripts.

`g:GetLatestVimScripts_options`

default= "-q -0"

This variable holds the options to be used with the `g:GetLatestVimScripts_wget` command.

`g:GetLatestVimScripts_allowautoinstall`

default= 1

This variable indicates whether `GetLatestVimScripts` is allowed to attempt to automatically install scripts. Furthermore, the plugin author has to have explicitly indicated that his/her plugin is automatically installable (via the `:AutoInstall:` keyword in the `GetLatestVimScripts` comment line).

`g:GetLatestVimScripts_autoinstalldir`

default= \$HOME/.vim (linux)

default= \$HOME/vimfiles (windows)

Override where `:AutoInstall:` scripts will be installed. Doesn't override vimball installation.

`g:GetLatestVimScripts_scriptaddr`

default='http://vim.sourceforge.net/script.php?script_id='

Override this if your system needs

... ='http://vim.sourceforge.net/script/script.php?script_id='

8. GetLatestVimScripts Algorithm

`glvs-algorithm` `glvs-alg`

The Vim sourceforge page dynamically creates a page by keying off of the so-called script-id. Within the webpage of

http://vim.sourceforge.net/scripts/script.php?script_id=40

is a line specifying the latest source-id (`src_id`). The source identifier numbers are always increasing, hence if the `src_id` is greater than the one recorded for the script in `GetLatestVimScripts` then it's time to download a newer copy of that script.

`GetLatestVimScripts` will then download the script and update its internal database of script ids, source ids, and scriptnames.

The `AutoInstall` process will:

Move the file from `GetLatest/` to the following directory

Unix : \$HOME/.vim

Windows: \$HOME\vimfiles

if the downloaded file ends with ".bz2"

`bunzip2 it`

else if the downloaded file ends with ".gz"

`gunzip it`

if the resulting file ends with ".zip"

`unzip it`

```

else if the resulting file ends with ".tar"
    tar -oxvf it
else if the resulting file ends with ".vim"
    move it to the plugin subdirectory

```

9. GetLatestVimScripts History getscript-history glvs-hist {{{1

- v36 Apr 22, 2013 : * (glts) suggested use of plugin/**/*.vim instead of plugin/*.vim in globpath() call.
 - * (Andy Wokula) got warning message when setting g:loaded_getscriptPlugin
- v35 Apr 07, 2012 : * (MengHuan Yu) pointed out that the script URL has changed (somewhat). However, it doesn't work, and the original one does (under Linux). I'll make it yet-another-option.
- v34 Jun 23, 2011 : * handles additional decompression options for tarballs (tgz taz tbz txz)
- v33 May 31, 2011 : * using fnameescape() instead of escape()
 - * *.xz support
- v32 Jun 19, 2010 : * (Jan Steffens) added support for xz compression
- v31 Jun 29, 2008 : * (Bill McCarthy) fixed having hls enabled with getscript
 - * (David Schaefer) the acd option interferes with vimballs
 - Solution: bypass the acd option
- v30 Jun 13, 2008 : * GLVS now checks for existence of fnameescape() and will issue an error message if it is not supported
- v29 Jan 07, 2008 : * Bram M pointed out that cpo is a global option and that getscriptPlugin.vim was setting it but not restoring it.
- v28 Jan 02, 2008 : * improved shell quoting character handling, cygwin interface, register-a bypass
 - Oct 29, 2007 * Bill McCarthy suggested a change to getscript that avoids creating pop-up windows
- v24 Apr 16, 2007 : * removed save&restore of the fo option during script loading
- v23 Nov 03, 2006 : * ignores comments (#...)
 - * handles vimballs
- v22 Oct 13, 2006 : * supports automatic use of curl if wget is not available
- v21 May 01, 2006 : * now takes advantage of autoloading.
- v20 Dec 23, 2005 : * Eric Haarbauer found&fixed a bug with unzip use; unzip needs the -o flag to overwrite.
- v19 Nov 28, 2005 : * v18's GetLatestVimScript line accessed the wrong script! Fixed.
- v18 Mar 21, 2005 : * bugfix to automatic database construction
 - * bugfix - nowrapscan caused an error (tnx to David Green for the fix)
 - Apr 01, 2005 * if shell is bash, "mv" instead of "ren" used in :AutoInstall:s, even though its o/s is windows
 - Apr 01, 2005 * when downloading errors occurred, GLVS was terminating early. It now just goes on to trying the next script (after trying three times to download a script description page)
 - Apr 20, 2005 * bugfix - when a failure to download occurred,

GetLatestVimScripts would stop early and claim that everything was current. Fixed.

v17 Aug 25, 2004 : * g:GetLatestVimScripts_allowautoinstall, which defaults to 1, can be used to prevent all :AutoInstall:

v16 Aug 25, 2004 : * made execution of bunzip2/gunzip/tar/zip silent
* fixed bug with :AutoInstall: use of helptags

v15 Aug 24, 2004 : * bugfix: the "0 0 comment" download prevention wasn't always preventing downloads (just usually). Fixed.

v14 Aug 24, 2004 : * bugfix -- helptags was using dotvim, rather than s:dotvim. Fixed.

v13 Aug 23, 2004 : * will skip downloading a file if its scriptid or srcid is zero. Useful for script authors; that way their own GetLatestVimScripts activity won't overwrite their scripts.

v12 Aug 23, 2004 : * bugfix - a "return" got left in the distribution that was intended only for testing. Removed, now works.
* :AutoInstall: implemented

v11 Aug 20, 2004 : * GetLatestVimScripts is now a plugin:
* :GetLatestVimScripts command
* (runtimepath)/GetLatest/GetLatestVimScripts.dat now holds scripts that need updating

v10 Apr 19, 2004 : * moved history from script to doc

v9 Jan 23, 2004 : windows (win32/win16/win95) will use double quotes (") whereas other systems will use single quotes (') around the urls in calls via wget

v8 Dec 01, 2003 : makes three tries at downloading

v7 Sep 02, 2003 : added error messages if "Click on..." or "src_id=" not found in downloaded webpage
Uses t_ti, t_te, and rs to make progress visible
final status messages now display summary of work ("Downloaded someqty scripts" or "Everything was current")
Now GetLatestVimScripts is careful about downloading GetLatestVimScripts.vim itself!
(goes to [<NEW_GetLatestVimScripts.vim>](#))

v5 Aug 04, 2003 : missing an endif near bottom

v4 Jun 17, 2003 : redraw! just before each "considering" message

v3 May 27, 2003 : Protects downloaded files from errant shell expansions with single quotes: '...'

v2 May 14, 2003 : extracts name of item to be obtained from the script file. Uses it instead of comment field for output filename; comment is used in the "considering..." line and is now just a comment!
* Fixed a bug: a string-of-numbers is not the same as a number, so I added zero to them and they became numbers. Fixes comparison.

=====

Editing compressed files with Vim `gzip` `bzip2` `compress`

1. Autocommands `gzip-autocmd`

The functionality mentioned here is a `standard-plugin`. This plugin is only available if `'compatible'` is not set. You can avoid loading this plugin by setting the "loaded_gzip" variable:

```
:let loaded_gzip = 1
```

`g:gzip_exec`

For security reasons, one may prevent that Vim runs executables automatically when opening a buffer. This option (default: "1") can be used to prevent executing the executables command when set to "0":

```
:let g:gzip_exec = 0
```

1. Autocommands `gzip-autocmd`

The plugin installs autocommands to intercept reading and writing of files with these extensions:

extension	compression
*.Z	compress (Lempel-Ziv)
*.gz	gzip
*.bz2	bzip2
*.lzma	lzma
*.xz	xz
*.lz	lzip
*.zst	zstd

That's actually the only thing you need to know. There are no options.

After decompressing a file, the filetype will be detected again. This will make a file like "foo.c.gz" get the "c" filetype.

If you have `'patchmode'` set, it will be appended after the extension for compression. Thus editing the patchmode file will not give you the automatic decompression. You have to rename the file if you want this.

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
 Copyright: (c) 2004-2016 by Charles E. Campbell [logiPat-copyright](#)
 The VIM LICENSE applies to LogiPat.vim and LogiPat.txt
 (see [copyright](#)) except use "LogiPat" instead of "Vim"
 No warranty, express or implied. Use At-Your-Own-Risk.

1. Contents [logiPat](#) [logiPat-contents](#)

- 1. Contents.....: [logiPat-contents](#)
- 2. LogiPat Manual.....: [logiPat-manual](#)
- 3. LogiPat Examples.....: [logiPat-examples](#)
- 4. Caveat.....: [logiPat-caveat](#)
- 5. LogiPat History.....: [logiPat-history](#)

2. LogiPat Manual [logiPat-manual](#) [logiPat-man](#)

[logiPat-arg](#) [logiPat-input](#) [logiPat-pattern](#) [logiPat-operators](#)

Boolean logic patterns are composed of

```

operators  ! = not
           | = logical-or
           & = logical-and
grouping   ( ... )
patterns   "pattern"
```

[logiPat-cmd](#)

```

:LogiPat {boolean-logic pattern} :LogiPat
  :LogiPat is a command which takes a boolean-logic
  argument ( logiPat-arg ).
```

```

:LP {boolean-logic pattern} :LP
  :LP is a shorthand command version of :LogiPat
  ( logiPat-cmd ).
```

```

:LPE {boolean-logic pattern} :LPE
  No search is done, but the conversion from the
  boolean logic pattern to the regular expression
  is performed and echoed onto the display.
```

```

:LogiPatFlags {search flags} LogiPat-flags
  LogiPat uses the search\(\) command. The flags
  passed to that call to search() may be specified
  by the :LogiPatFlags command.
```

```

:LPF {search flags} :LPF
  :LPF is a shorthand version of :LogiPatFlags.
```

```

:let pat=LogiPat({boolean-logic pattern}) LogiPat\(\)
  If one calls LogiPat() directly, no search
```

is done, but the transformation from the boolean logic pattern into a regular expression pattern is performed and returned.

To get a " inside a pattern, as opposed to having it delimit the pattern, double it.

3. LogiPat Examples

logiPat-examples

LogiPat takes Boolean logic arguments and produces a regular expression which implements the choices. A series of examples follows:

```
:LogiPat "abc"
    will search for lines containing the string :abc:

:LogiPat "ab""cd"
    will search for lines containing the string :ab"cd:

:LogiPat !"abc"
    will search for lines which don't contain the string :abc:

:LogiPat "abc"|"def"
    will search for lines which contain either the string
    :abc: or the string :def:

:LogiPat !("abc"|"def")
    will search for lines which don't contain either
    of the strings :abc: or :def:

:LogiPat "abc"&"def"
    will search for lines which contain both of the strings
    :abc: and :def:

:let pat= LogiPat('!"abc"')
    will return the regular expression which will match
    all lines not containing :abc: . The double quotes
    are needed to pass normal patterns to LogiPat, and
    differentiate such patterns from boolean logic
    operators.
```

4. Caveat

logiPat-caveat

The "not" operator may be fragile; ie. it may not always play well with the & (logical-and) and | (logical-or) operators. Please try out your patterns, possibly with :set hls, to insure that what is matching is what you want.

5. LogiPat History

logiPat-history

- v4 Jun 22, 2015 * LogiPat has been picked up by Bram M for standard plugin distribution; hence the name change
- v3 Sep 25, 2006 * LP_Or() fixed; it now encapsulates its output in \%(...\) parentheses
- Dec 12, 2011 * :LPE added
- * "" is mapped to a single " and left inside patterns
- v2 May 31, 2005 * LPF and LogiPatFlags commands weren't working
- v1 May 23, 2005 * initial release



NETRW REFERENCE MANUAL by Charles E. Campbell

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
(remove NOSPAM from Campbell's email first)

Copyright: Copyright (C) 2017 Charles E Campbell [netrw-copyright](#)
The VIM LICENSE applies to the files in this package, including netrw.vim, pi_netrw.txt, netrwFileHandlers.vim, netrwSettings.vim, and syntax/netrw.vim. Like anything else that's free, netrw.vim and its associated files are provided *as is* and comes with no warranty of any kind, either expressed or implied. No guarantees of merchantability. No guarantees of suitability for any purpose. By using this plugin, you agree that in no event will the copyright holder be liable for any damages resulting from the use of this software. Use at your own risk!

[netrw](#)
[dav](#) [ftp](#) [netrw-file](#) [rcp](#) [scp](#)
[davs](#) [http](#) [netrw.vim](#) [rsync](#) [sftp](#)
[fetch](#) [network](#)

=====

1. Contents	netrw-contents {{{1
1. Contents.....	netrw-contents
2. Starting With Netrw.....	netrw-start
3. Netrw Reference.....	netrw-ref
EXTERNAL APPLICATIONS AND PROTOCOLS.....	netrw-externapp
READING.....	netrw-read
WRITING.....	netrw-write
SOURCING.....	netrw-source
DIRECTORY LISTING.....	netrw-dirlist
CHANGING THE USERID AND PASSWORD.....	netrw-chgup
VARIABLES AND SETTINGS.....	netrw-variables
PATHS.....	netrw-path
4. Network-Oriented File Transfer.....	netrw-xfer
NETRC.....	netrw-netrc
PASSWORD.....	netrw-passwd
5. Activation.....	netrw-activate
6. Transparent Remote File Editing.....	netrw-transparent
7. Ex Commands.....	netrw-ex
8. Variables and Options.....	netrw-variables
9. Browsing.....	netrw-browse
Introduction To Browsing.....	netrw-intro-browse
Quick Reference: Maps.....	netrw-browse-maps
Quick Reference: Commands.....	netrw-browse-cmds
Banner Display.....	netrw-I
Bookmarking A Directory.....	netrw-mb
Browsing.....	netrw-cr
Squeezing the Current Tree-Listing Directory.....	netrw-s-cr
Browsing With A Horizontally Split Window.....	netrw-o

Browsing With A New Tab..... netrw-t
 Browsing With A Vertically Split Window..... netrw-v
 Change File Permission..... netrw-gp
 Change Listing Style.(thin wide long tree)..... netrw-i
 Changing To A Bookmarked Directory..... netrw-gb
 Changing To A Predecessor Directory..... netrw-u
 Changing To A Successor Directory..... netrw-U
 Customizing Browsing With A Special Handler..... netrw-x
 Deleting Bookmarks..... netrw-mB
 Deleting Files Or Directories..... netrw-D
 Directory Exploring Commands..... netrw-explore
 Exploring With Stars and Patterns..... netrw-star
 Displaying Information About File..... netrw-qf
 Edit File Or Directory Hiding List..... netrw-ctrl-h
 Editing The Sorting Sequence..... netrw-S
 Forcing treatment as a file or directory..... netrw-gd netrw-gf
 Going Up..... netrw--
 Hiding Files Or Directories..... netrw-a
 Improving Browsing..... netrw-ssh-hack
 Listing Bookmarks And History..... netrw-qb
 Making A New Directory..... netrw-d
 Making The Browsing Directory The Current Directory. netrw-cd
 Marking Files..... netrw-mf
 Unmarking Files..... netrw-mF
 Marking Files By Location List..... netrw-qL
 Marking Files By QuickFix List..... netrw-qF
 Marking Files By Regular Expression..... netrw-mr
 Marked Files: Arbitrary Shell Command..... netrw-mx
 Marked Files: Arbitrary Shell Command, En Bloc..... netrw-mX
 Marked Files: Arbitrary Vim Command..... netrw-mv
 Marked Files: Argument List..... netrw-ma netrw-mA
 Marked Files: Buffer List..... netrw-cb netrw-cB
 Marked Files: Compression And Decompression..... netrw-mz
 Marked Files: Copying..... netrw-mc
 Marked Files: Diff..... netrw-md
 Marked Files: Editing..... netrw-me
 Marked Files: Grep..... netrw-mg
 Marked Files: Hiding and Unhiding by Suffix..... netrw-mh
 Marked Files: Moving..... netrw-mm
 Marked Files: Printing..... netrw-mp
 Marked Files: Sourcing..... netrw-ms
 Marked Files: Setting the Target Directory..... netrw-mt
 Marked Files: Tagging..... netrw-mT
 Marked Files: Target Directory Using Bookmarks..... netrw-Tb
 Marked Files: Target Directory Using History..... netrw-Th
 Marked Files: Unmarking..... netrw-mu
 Netrw Browser Variables..... netrw-browser-var
 Netrw Browsing And Option Incompatibilities..... netrw-incompatible
 Netrw Settings Window..... netrw-settings-window
 Obtaining A File..... netrw-O
 Preview Window..... netrw-p
 Previous Window..... netrw-P
 Refreshing The Listing..... netrw-ctrl-l
 Reversing Sorting Order..... netrw-r

Renaming Files Or Directories.....	netrw-R
Selecting Sorting Style.....	netrw-s
Setting Editing Window.....	netrw-C
10. Problems and Fixes.....	netrw-problems
11. Debugging Netrw Itself.....	netrw-debug
12. History.....	netrw-history
13. Todo.....	netrw-todo
14. Credits.....	netrw-credits

2. Starting With Netrw netrw-start {{{1

Netrw makes reading files, writing files, browsing over a network, and local browsing easy! First, make sure that you have plugins enabled, so you'll need to have at least the following in your `<.vimrc>`: (or see `netrw-activate`)

```
set nocp                " 'compatible' is not set
filetype plugin on      " plugins are enabled
```

(see `'cp'` and `:filetype-plugin-on`)

Netrw supports "transparent" editing of files on other machines using urls (see `netrw-transparent`). As an example of this, let's assume you have an account on some other machine; if you can use scp, try:

```
vim scp://hostname/path/to/file
```

Want to make ssh/scp easier to use? Check out `netrw-ssh-hack` !

So, what if you have ftp, not ssh/scp? That's easy, too; try

```
vim ftp://hostname/path/to/file
```

Want to make ftp simpler to use? See if your ftp supports a file called `<.netrc>` -- typically it goes in your home directory, has read/write permissions for only the user to read (ie. not group, world, other, etc), and has lines resembling

```
machine HOSTNAME login USERID password "PASSWORD"
machine HOSTNAME login USERID password "PASSWORD"
...
default          login USERID password "PASSWORD"
```

Windows' ftp doesn't support `.netrc`; however, one may have in one's `.vimrc`:

```
let g:netrw_ftp_cmd= 'c:\Windows\System32\ftp -s:C:\Users\MyUserName\MACHINE'
```

Netrw will substitute the host's machine name for "MACHINE" from the URL it is attempting to open, and so one may specify

```
userid
password
```

for each site in a separate file: `c:\Users\MyUserName\MachineName`.

Now about browsing -- when you just want to look around before editing a file. For browsing on your current host, just "edit" a directory:

```
vim .
vim /home/userid/path
```

For browsing on a remote host, "edit" a directory (but make sure that the directory name is followed by a "/"):

```
vim scp://hostname/
vim ftp://hostname/path/to/dir/
```

See [netrw-browse](#) for more!

There are more protocols supported by netrw than just scp and ftp, too: see the next section, [netrw-externapp](#), on how to use these external applications with netrw and vim.

PREVENTING LOADING

[netrw-noload](#)

If you want to use plugins, but for some reason don't wish to use netrw, then you need to avoid loading both the plugin and the autoloader portions of netrw. You may do so by placing the following two lines in your `<.vimrc>`:

```
:let g:loaded_netrw      = 1
:let g:loaded_netrwPlugin = 1
```

3. Netrw Reference

[netrw-ref](#) {{{1

Netrw supports several protocols in addition to scp and ftp as mentioned in [netrw-start](#). These include dav, fetch, http,... well, just look at the list in [netrw-externapp](#). Each protocol is associated with a variable which holds the default command supporting that protocol.

EXTERNAL APPLICATIONS AND PROTOCOLS

[netrw-externapp](#) {{{2

Protocol	Variable	Default Value	
dav:	g:netrw_dav_cmd	= "cadaver"	if cadaver is executable
dav:	<code>g:netrw_dav_cmd</code>	= "curl -o"	elseif curl is available
fetch:	g:netrw_fetch_cmd	= "fetch -o"	if fetch is available
ftp:	g:netrw_ftp_cmd	= "ftp"	
http:	g:netrw_http_cmd	= "elinks"	if elinks is available
http:	<code>g:netrw_http_cmd</code>	= "links"	elseif links is available
http:	<code>g:netrw_http_cmd</code>	= "curl"	elseif curl is available
http:	<code>g:netrw_http_cmd</code>	= "wget"	elseif wget is available
http:	<code>g:netrw_http_cmd</code>	= "fetch"	elseif fetch is available
http:	g:netrw_http_put_cmd	= "curl -T"	
rcp:	g:netrw_rcp_cmd	= "rcp"	
rsync:	g:netrw_rsync_cmd	= "rsync"	(see g:netrw_rsync_sep)
scp:	g:netrw_scp_cmd	= "scp -q"	
sftp:	g:netrw_sftp_cmd	= "sftp"	

file: `g:netrw_file_cmd` = "elinks" or "links"

`g:netrw_http_xcmd` : the option string for http://... protocols are specified via this variable and may be independently overridden. By default, the option arguments for the http-handling commands are:

```
elinks : "-source >"
links  : "-dump >"
curl   : "-L -o"
wget   : "-q -O"
fetch  : "-o"
```

For example, if your system has elinks, and you'd rather see the page using an attempt at rendering the text, you may wish to have
`let g:netrw_http_xcmd= "-dump >"`
in your .vimrc.

`g:netrw_http_put_cmd`: this option specifies both the executable and any needed options. This command does a PUT operation to the url.

READING

`netrw-read` `netrw-nread` {{{2

Generally, one may just use the URL notation with a normal editing command, such as

```
:e ftp://[user@]machine/path
```

Netrw also provides the Nread command:

<code>:Nread ?</code>	give help
<code>:Nread "machine:path"</code>	uses rcp
<code>:Nread "machine path"</code>	uses ftp w/ <code><.netrc></code>
<code>:Nread "machine id password path"</code>	uses ftp
<code>:Nread "dav://machine[:port]/path"</code>	uses cadaver
<code>:Nread "fetch://[user@]machine/path"</code>	uses fetch
<code>:Nread "ftp://[user@]machine[:port]/path"</code>	uses ftp w/ <code><.netrc></code>
<code>:Nread "http://[user@]machine/path"</code>	uses http uses wget
<code>:Nread "rcp://[user@]machine/path"</code>	uses rcp
<code>:Nread "rsync://[user@]machine[:port]/path"</code>	uses rsync
<code>:Nread "scp://[user@]machine[:port]/path"</code>	uses scp
<code>:Nread "sftp://[user@]machine/path"</code>	uses sftp

WRITING

`netrw-write` `netrw-nwrite` {{{2

One may just use the URL notation with a normal file writing command, such as

```
:w ftp://[user@]machine/path
```

Netrw also provides the Nwrite command:

<code>:Nwrite ?</code>	give help
<code>:Nwrite "machine:path"</code>	uses rcp

```

:Nwrite "machine path"                uses ftp w/ <.netrc>
:Nwrite "machine id password path"    uses ftp
:Nwrite "dav://machine[:port]/path"   uses cadaver
:Nwrite "ftp://[user@]machine[[:#]port]/path" uses ftp w/ <.netrc>
:Nwrite "rcp://[user@]machine/path"   uses rcp
:Nwrite "rsync://[user@]machine[:port]/path" uses rsync
:Nwrite "scp://[user@]machine[[:#]port]/path" uses scp
:Nwrite "sftp://[user@]machine/path"  uses sftp
http: not supported!

```

SOURCING

`netrw-source` {{{2

One may just use the URL notation with the normal file sourcing command, such as

```
:so ftp://[user@]machine/path
```

Netrw also provides the Nsource command:

```

:Nsource ?                give help
:Nsource "dav://machine[:port]/path"    uses cadaver
:Nsource "fetch://[user@]machine/path"   uses fetch
:Nsource "ftp://[user@]machine[[:#]port]/path" uses ftp w/ <.netrc>
:Nsource "http://[user@]machine/path"    uses http uses wget
:Nsource "rcp://[user@]machine/path"     uses rcp
:Nsource "rsync://[user@]machine[:port]/path" uses rsync
:Nsource "scp://[user@]machine[[:#]port]/path" uses scp
:Nsource "sftp://[user@]machine/path"    uses sftp

```

DIRECTORY LISTING

`netrw-trailingslash` `netrw-dirlist` {{{2

One may browse a directory to get a listing by simply attempting to edit the directory:

```
:e scp://[user]@hostname/path/
:e ftp://[user]@hostname/path/
```

For remote directory listings (ie. those using scp or ftp), that trailing "/" is necessary (the slash tells netrw to treat the argument as a directory to browse instead of as a file to download).

The Nread command may also be used to accomplish this (again, that trailing slash is necessary):

```
:Nread [protocol]://[user]@hostname/path/
```

CHANGING USERID AND PASSWORD

`netrw-login` `netrw-password`
`netrw-chgup` `netrw-userpass` {{{2

Attempts to use ftp will prompt you for a user-id and a password. These will be saved in global variables `g:netrw_uid` and `s:netrw_passwd`; subsequent use of ftp will re-use those two strings, thereby simplifying use of ftp. However, if you need to use a different user id and/or password, you'll want to call `NetUserPass()`

first. To work around the need to enter passwords, check if your ftp supports a `<.netrc>` file in your home directory. Also see [netrw-passwd](#) (and if you're using ssh/scp hoping to figure out how to not need to use passwords for scp, look at [netrw-ssh-hack](#)).

```
:NetUserPass [uid [password]]          -- prompts as needed
:call NetUserPass()                    -- prompts for uid and password
:call NetUserPass("uid")                -- prompts for password
:call NetUserPass("uid","password")     -- sets global uid and password
```

(Related topics: [ftp](#) [netrw-userpass](#) [netrw-start](#))

NETRW VARIABLES AND SETTINGS

[netrw-variables](#) {{{2

(Also see:

```
netrw-browser-var      : netrw browser option variables
netrw-protocol        : file transfer protocol option variables
netrw-settings        : additional file transfer options
netrw-browser-options : these options affect browsing directories
```

)

Netrw provides a lot of variables which allow you to customize netrw to your preferences. One way to look at them is via the command `:NetrwSettings` (see [netrw-settings](#)) which will display your current netrw settings. Most such settings are described below, in [netrw-browser-options](#) , and in [netrw-externapp](#) :

```
b:netrw\_lastfile      last file Network-read/written retained on a
per-buffer basis (supports plain :Nw )

g:netrw\_bufsettings  the settings that netrw buffers have
(default) noma nomod nonu nowrap ro nobl

g:netrw\_chgwin       specifies a window number where subsequent file edits
will take place. (also see netrw-C )
(default) -1

g:Netrw\_funcref     specifies a function (or functions) to be called when
netrw edits a file. The file is first edited, and
then the function reference ( Funcref ) is called.
This variable may also hold a List of Funcrefs.
(default) not defined. (the capital in g:Netrw...
is required by its holding a function reference)

Example: place in .vimrc; affects all file opening
fun! MyFuncRef()
endfun
let g:Netrw_funcref= function("MyFuncRef")

g:Netrw\_UserMaps    specifies a function or List of functions which can
be used to set up user-specified maps and functionality.
See netrw-usermaps

g:netrw\_ftp         if it doesn't exist, use default ftp
```

=0 use default ftp (uid password)
 =1 use alternate ftp method (user uid password)
 If you're having trouble with ftp, try changing the value of this variable to see if the alternate ftp method works for your setup.

g:netrw_ftp_options Chosen by default, these options are supposed to turn interactive prompting off and to restrain ftp from attempting auto-login upon initial connection. However, it appears that not all ftp implementations support this (ex. ncftp).
 ="-i -n"

g:netrw_ftpextracmd default: doesn't exist
 If this variable exists, then any string it contains will be placed into the commands set to your ftp client. As an example:
 ="passive"

g:netrw_ftpmode ="binary" (default)
 ="ascii"

g:netrw_ignorenetrc =0 (default for linux, cygwin)
 =1 If you have a `<.netrc>` file but it doesn't work and you want it ignored, then set this variable as shown. (default for Windows + cmd.exe)

g:netrw_menu =0 disable netrw's menu
 =1 (default) netrw's menu enabled

g:netrw_nogx if this variable exists, then the "gx" map will not be available (see [netrw-gx](#))

g:netrw_uid (ftp) user-id, retained on a per-vim-session basis
s:netrw_passwd (ftp) password, retained on a per-vim-session basis

g:netrw_preview =0 (default) preview window shown in a horizontally split window
 =1 preview window shown in a vertically split window. Also affects the "previous window" (see [netrw-P](#)) in the same way.
 The [g:netrw_alto](#) variable may be used to provide additional splitting control:

g:netrw_preview	g:netrw_alto	result
0	0	:aboveleft
0	1	:belowright
1	0	:topleft
1	1	:botright

To control sizing, see [g:netrw_winsize](#)

g:netrw_scpport = "-P" : option to use to set port for scp
g:netrw_sshport = "-p" : option to use to set port for ssh

g:netrw_sepchr =\0xff

=\0x01 for enc == euc-jp (and perhaps it should be for others, too, please let me know)
Separates priority codes from filenames internally.
See [netrw-p12](#) .

g:netrw_silent =0 : transfers done normally
=1 : transfers done silently

g:netrw_use_errorwindow =2: messages from netrw will use a popup window
Move the mouse and pause to remove the popup window.
(default value if popup windows are available)
=1 : messages from netrw will use a separate one line window. This window provides reliable delivery of messages.
(default value if popup windows are not available)
=0 : messages from netrw will use echoerr ;
messages don't always seem to show up this way, but one doesn't have to quit the window.

g:netrw_win95ftp =1 if using Win95, will remove four trailing blank lines that o/s's ftp "provides" on transfers
=0 force normal ftp behavior (no trailing line removal)

g:netrw_cygwin =1 assume scp under windows is from cygwin. Also permits network browsing to use ls with time and size sorting (default if windows)
=0 assume Windows' scp accepts windows-style paths
Network browsing uses dir instead of ls
This option is ignored if you're using unix

g:netrw_use_nt_rcp =0 don't use the rcp of WinNT, Win2000 and WinXP
=1 use WinNT's rcp in binary mode (default)

PATHS

netrw-path {{{2

Paths to files are generally user-directory relative for most protocols. It is possible that some protocol will make paths relative to some associated directory, however.

example: vim scp://user@host/somefile
example: vim scp://user@host/subdir1/subdir2/somefile

where "somefile" is in the "user"'s home directory. If you wish to get a file using root-relative paths, use the full path:

example: vim scp://user@host//somefile
example: vim scp://user@host//subdir1/subdir2/somefile

4. Network-Oriented File Transfer

netrw-xfer {{{1

Network-oriented file transfer under Vim is implemented by a vim script (<netrw.vim>) using plugin techniques. It currently supports both reading and

writing across networks using rcp, scp, ftp or ftp+<.netrc>, scp, fetch, dav/cadaver, rsync, or sftp.

http is currently supported read-only via use of wget or fetch.

<netrw.vim> is a standard plugin which acts as glue between Vim and the various file transfer programs. It uses autocommand events (BufReadCmd, FileReadCmd, BufWriteCmd) to intercept reads/writes with url-like filenames.

ex. vim ftp://hostname/path/to/file

The characters preceding the colon specify the protocol to use; in the example, it's ftp. The <netrw.vim> script then formulates a command or a series of commands (typically ftp) which it issues to an external program (ftp, scp, etc) which does the actual file transfer/protocol. Files are read from/written to a temporary file (under Unix/Linux, /tmp/...) which the <netrw.vim> script will clean up.

Now, a word about Jan Minář's "FTP User Name and Password Disclosure"; first, ftp is not a secure protocol. User names and passwords are transmitted "in the clear" over the internet; any snooper tool can pick these up; this is not a netrw thing, this is a ftp thing. If you're concerned about this, please try to use scp or sftp instead.

Netrw re-uses the user id and password during the same vim session and so long as the remote hostname remains the same.

Jan seems to be a bit confused about how netrw handles ftp; normally multiple commands are performed in a "ftp session", and he seems to feel that the uid/password should only be retained over one ftp session. However, netrw does every ftp operation in a separate "ftp session"; so remembering the uid/password for just one "ftp session" would be the same as not remembering the uid/password at all. IMHO this would rapidly grow tiresome as one browsed remote directories, for example.

On the other hand, thanks go to Jan M. for pointing out the many vulnerabilities that netrw (and vim itself) had had in handling "crafted" filenames. The [shellescape\(\)](#) and [fnameescape\(\)](#) functions were written in response by Bram Moolenaar to handle these sort of problems, and netrw has been modified to use them. Still, my advice is, if the "filename" looks like a vim command that you aren't comfortable with having executed, don't open it.

[netrw-putty](#) [netrw-pscp](#) [netrw-psftp](#)

One may modify any protocol's implementing external application by setting a variable (ex. scp uses the variable g:netrw_scp_cmd, which is defaulted to "scp -q"). As an example, consider using PuTTY:

```
let g:netrw_scp_cmd = '"c:\Program Files\PuTTY\pscp.exe" -q -batch'  
let g:netrw_sftp_cmd= '"c:\Program Files\PuTTY\psftp.exe"'
```

(note: it has been reported that windows 7 with putty v0.6's "--batch" option doesn't work, so its best to leave it off for that system)

See [netrw-p8](#) for more about putty, pscp, psftp, etc.

Ftp, an old protocol, seems to be blessed by numerous implementations. Unfortunately, some implementations are noisy (ie., add junk to the end of the file). Thus, concerned users may decide to write a NetReadFixup() function that will clean up after reading with their ftp. Some Unix systems (ie., FreeBSD) provide a utility called "fetch" which uses the ftp protocol but is not noisy and more convenient, actually, for <netrw.vim> to use. Consequently, if "fetch" is available (ie. executable), it may be preferable to use it for ftp://... based transfers.

For rcp, scp, sftp, and http, one may use network-oriented file transfers transparently; ie.

```
vim rcp://[user@]machine/path
vim scp://[user@]machine/path
```

If your ftp supports <.netrc>, then it too can be transparently used if the needed triad of machine name, user id, and password are present in that file. Your ftp must be able to use the <.netrc> file on its own, however.

```
vim ftp://[user@]machine[[:#]portnumber]/path
```

Windows provides an ftp (typically c:\Windows\System32\ftp.exe) which uses an option, -s:filename (filename can and probably should be a full path) which contains ftp commands which will be automatically run whenever ftp starts. You may use this feature to enter a user and password for one site:

```
userid
password
```

netrw-windows-netrc netrw-windows-s

If g:netrw_ftp_cmd contains -s:[path/]MACHINE, then (on Windows machines only) netrw will substitute the current machine name requested for ftp connections for MACHINE. Hence one can have multiple machine.ftp files containing login and password for ftp. Example:

```
let g:netrw_ftp_cmd= 'c:\Windows\System32\ftp -s:C:\Users\Myself\MACHINE'
vim ftp://myhost.somewhere.net/
```

will use a file

```
C:\Users\Myself\myhost.ftp
```

Often, ftp will need to query the user for the userid and password. The latter will be done "silently"; ie. asterisks will show up instead of the actually-typed-in password. Netrw will retain the userid and password for subsequent read/writes from the most recent transfer so subsequent transfers (read/write) to or from that machine will take place without additional prompting.

netrw-urls

Reading	Writing	Uses
DAV:		
dav://host/path		cadaver

:Nread dav://host/path	:Nwrite dav://host/path	cadaver
DAV + SSL: davs://host/path :Nread davs://host/path	:Nwrite davs://host/path	cadaver cadaver
FETCH: fetch://[user@]host/path fetch://[user@]host:http/path :Nread fetch://[user@]host/path	Not Available	fetch
FILE: file:///.* file://localhost/.*	file:///.* file://localhost/.*	
FTP: (*3) ftp://[user@]host/path :Nread ftp://host/path :Nread host path :Nread host uid pass path	(*3) ftp://[user@]host/path :Nwrite ftp://host/path :Nwrite host path :Nwrite host uid pass path	ftp (*2) ftp+.netrc ftp+.netrc ftp
HTTP: wget is executable: (*4) http://[user@]host/path	Not Available	wget
HTTP: fetch is executable (*4) http://[user@]host/path	Not Available	fetch
RCP: rcp://[user@]host/path	rcp://[user@]host/path	rcp
RSYNC: rsync://[user@]host/path :Nread rsync://host/path :Nread rcp://host/path	rsync://[user@]host/path :Nwrite rsync://host/path :Nwrite rcp://host/path	rsync rsync rcp
SCP: scp://[user@]host/path :Nread scp://host/path	scp://[user@]host/path :Nwrite scp://host/path	scp scp (*1)
SFTP: sftp://[user@]host/path :Nread sftp://host/path	sftp://[user@]host/path :Nwrite sftp://host/path	sftp sftp (*1)

(*1) For an absolute path use scp://machine//path.

(*2) if <.netrc> is present, it is assumed that it will work with your ftp client. Otherwise the script will prompt for user-id and password.

(*3) for ftp, "machine" may be machine#port or machine:port if a different port is needed than the standard ftp port

(*4) for http:..., if wget is available it will be used. Otherwise,

if fetch is available it will be used.

Both the :Nread and the :Nwrite ex-commands can accept multiple filenames.

NETRC

netrw-netrc

The `<.netrc>` file, typically located in your home directory, contains lines therein which map a hostname (machine name) to the user id and password you prefer to use with it.

The typical syntax for lines in a `<.netrc>` file is given as shown below. Ftp under Unix usually supports `<.netrc>`; ftp under Windows usually doesn't.

```
machine {full machine name} login {user-id} password "{password}"
default login {user-id} password "{password}"
```

Your ftp client must handle the use of `<.netrc>` on its own, but if the `<.netrc>` file exists, an ftp transfer will not ask for the user-id or password.

Note:

Since this file contains passwords, make very sure nobody else can read this file! Most programs will refuse to use a `.netrc` that is readable for others. Don't forget that the system administrator can still read the file! Ie. for Linux/Unix: `chmod 600 .netrc`

Even though Windows' ftp clients typically do not support `.netrc`, netrw has a work-around: see [netrw-windows-s](#) .

PASSWORD

netrw-passwd

The script attempts to get passwords for ftp invisibly using `inputsecret()` , a built-in Vim function. See [netrw-userpass](#) for how to change the password after one has set it.

Unfortunately there doesn't appear to be a way for netrw to feed a password to scp. Thus every transfer via scp will require re-entry of the password. However, [netrw-ssh-hack](#) can help with this problem.

5. Activation

netrw-activate {{{1

Network-oriented file transfers are available by default whenever Vim's `'nocompatible'` mode is enabled. Netrw's script files reside in your system's plugin, autoload, and syntax directories; just the `plugin/netrwPlugin.vim` script is sourced automatically whenever you bring up vim. The main script in `autoload/netrw.vim` is only loaded when you actually use netrw. I suggest that, at a minimum, you have at least the following in your `<.vimrc>` customization file:

```
set nocp
```

```

if version >= 600
  filetype plugin indent on
endif

```

By also including the following lines in your `.vimrc`, one may have `netrw` immediately activate when using `[g]vim` without any filenames, showing the current directory:

```

" Augroup VimStartup:
augroup VimStartup
  au!
  au VimEnter * if expand("%") == "" | e . | endif
augroup END

```

6. Transparent Remote File Editing netrw-transparent {{{1

Transparent file transfers occur whenever a regular file read or write (invoked via an `:autocmd` for `BufReadCmd`, `BufWriteCmd`, or `SourceCmd` events) is made. Thus one may read, write, or source files across networks just as easily as if they were local files!

```

vim ftp://[user@]machine/path
...
:wq

```

See [netrw-activate](#) for more on how to encourage your vim to use plugins such as `netrw`.

For password-free use of `scp:`, see [netrw-ssh-hack](#).

7. Ex Commands netrw-ex {{{1

The usual read/write commands are supported. There are also a few additional commands available. Often you won't need to use `Nwrite` or `Nread` as shown in [netrw-transparent](#) (ie. simply use

```

:e URL
:r URL
:w URL

```

instead, as appropriate) -- see [netrw-urls](#). In the explanations below, a `{netfile}` is a URL to a remote file.

```

:[range]Nw[rite]           :Nwrite   :Nw
                        Write the specified lines to the current
                        file as specified in b:netrw_lastfile.
                        (related: netrw-nwrite)

```

```

:[range]Nw[rite] {netfile} [{netfile}]...
                        Write the specified lines to the {netfile}.

```

```

:Nread   :Nr

```

```

:Nr[ead]          Read the lines from the file specified in b:netrw_lastfile
                  into the current buffer. (related: netrw-nread )

:Nr[ead] {netfile} {netfile}...
                  Read the {netfile} after the current line.

:Ns[ource] {netfile}
                  :Nsource   :Ns
                  Source the {netfile}.
                  To start up vim using a remote .vimrc, one may use
                  the following (all on one line) (tnx to Antoine Mechelynck)
                  vim -u NORC -N
                  --cmd "runtime plugin/netrwPlugin.vim"
                  --cmd "source scp://HOSTNAME/.vimrc"
                  (related: netrw-source )

:call NetUserPass()
                  NetUserPass\(\)
                  If g:netrw_uid and s:netrw_passwd don't exist,
                  this function will query the user for them.
                  (related: netrw-userpass )

:call NetUserPass("userid")
                  This call will set the g:netrw_uid and, if
                  the password doesn't exist, will query the user for it.
                  (related: netrw-userpass )

:call NetUserPass("userid","passwd")
                  This call will set both the g:netrw_uid and s:netrw_passwd.
                  The user-id and password are used by ftp transfers. One may
                  effectively remove the user-id and password by using empty
                  strings (ie. "").
                  (related: netrw-userpass )

:NetrwSettings  This command is described in netrw-settings -- used to
                  display netrw settings and change netrw behavior.

```

8. Variables and Options [netrw-var](#) [netrw-settings](#) {{{1

(also see: [netrw-options](#) [netrw-variables](#) [netrw-protocol](#)
[netrw-browser-settings](#) [netrw-browser-options](#))

The <netrw.vim> script provides several variables which act as options to affect <netrw.vim>'s file transfer behavior. These variables typically may be set in the user's <.vimrc> file: (see also [netrw-settings](#) [netrw-protocol](#))
[netrw-options](#)

Option	Meaning

```

b:netrw_col           Holds current cursor position (during NetWrite)
g:netrw_cygwin        =1 assume scp under windows is from cygwin
                        (default/windows)
                        =0 assume scp under windows accepts windows
                        style paths          (default/else)
g:netrw_ftp           =0 use default ftp          (uid password)
g:netrw_ftpmode       ="binary"          (default)
                        ="ascii"          (your choice)
g:netrw_ignorenetr    =1          (default)
                        if you have a <.netrc> file but you don't
                        want it used, then set this variable. Its
                        mere existence is enough to cause <.netrc>
                        to be ignored.

b:netrw_lastfile     Holds latest method/machine/path.
b:netrw_line         Holds current line number      (during NetWrite)
g:netrw_silent       =0 transfers done normally
                        =1 transfers done silently
g:netrw_uid          Holds current user-id for ftp.
g:netrw_use_nt_rcp   =0 don't use WinNT/2K/XP's rcp (default)
                        =1 use WinNT/2K/XP's rcp, binary mode
g:netrw_win95ftp     =0 use unix-style ftp even if win95/98/ME/etc
                        =1 use default method to do ftp

```

netrw-internal-variables

The script will also make use of the following variables internally, albeit temporarily.

Temporary Variables

Variable	Meaning
b:netrw_method	Index indicating rcp/ftp+.netrc/ftp
w:netrw_method	(same as b:netrw_method)
g:netrw_machine	Holds machine name parsed from input
b:netrw_fname	Holds filename being accessed

netrw-protocol

Netrw supports a number of protocols. These protocols are invoked using the variables listed below, and may be modified by the user.

Protocol Control Options

Option	Type	Setting	Meaning
netrw_ftp	variable	=doesn't exist =0 =1	userid set by "user userid" userid set by "user userid" userid set by "userid"
NetReadFixup	function	=doesn't exist	no change

```

                                =exists           Allows user to have files
                                                read via ftp automatically
                                                transformed however they wish
                                                by NetReadFixup()
g:netrw_dav_cmd      var    ="cadaver"           if cadaver is executable
g:netrw_dav_cmd      var    ="curl -o"           elseif curl is executable
g:netrw_fetch_cmd    var    ="fetch -o"         if fetch is available
g:netrw_ftp_cmd      var    ="ftp"
g:netrw_http_cmd     var    ="fetch -o"         if      fetch is available
g:netrw_http_cmd     var    ="wget -O"         else if wget is available
g:netrw_http_put_cmd var    ="curl -T"
  g:netrw_list_cmd    var    ="ssh USEPORT HOSTNAME ls -Fa"
g:netrw_rcp_cmd      var    ="rcp"
g:netrw_rsync_cmd    var    ="rsync"
  g:netrw_rsync_sep   var    ="/"               used to separate the hostname
                                                from the file spec
g:netrw_scp_cmd      var    ="scp -q"
g:netrw_sftp_cmd     var    ="sftp"

```

netrw-ftp

The `g:netrw_..._cmd` options (`g:netrw_ftp_cmd` and `g:netrw_sftp_cmd`) specify the external program to use handle the ftp protocol. They may include command line options (such as `-p` for passive mode). Example:

```
let g:netrw_ftp_cmd= "ftp -p"
```

Browsing is supported by using the `g:netrw_list_cmd` ; the substring "HOSTNAME" will be changed via substitution with whatever the current request is for a hostname.

Two options (`g:netrw_ftp` and `netrw-fixup`) both help with certain ftp's that give trouble . In order to best understand how to use these options if ftp is giving you troubles, a bit of discussion is provided on how netrw does ftp reads.

For ftp, netrw typically builds up lines of one of the following formats in a temporary file:

```
IF g:netrw_ftp !exists or is not 1
```

```

open machine [port]
user userid password
[g:netrw_ftpmode]
[g:netrw_ftpextracmd]
get filename tempfile

```

```
IF g:netrw_ftp exists and is 1
```

```

open machine [port]
userid password
password
[g:netrw_ftpmode]
[g:netrw_extracmd]
get filename tempfile

```

The `g:netrw_ftpmode` and `g:netrw_ftpextracmd` are optional.

Netrw then executes the lines above by use of a filter:

```
:%! {g:netrw_ftp_cmd} -i [-n]
```

where

```
g:netrw_ftp_cmd is usually "ftp",  
-i tells ftp not to be interactive  
-n means don't use netrc and is used for Method #3 (ftp w/o <.netrc>)
```

If `<.netrc>` exists it will be used to avoid having to query the user for userid and password. The transferred file is put into a temporary file. The temporary file is then read into the main editing session window that requested it and the temporary file deleted.

If your ftp doesn't accept the "user" command and immediately just demands a userid, then try putting "let netrw_ftp=1" in your `<.vimrc>`.

`netrw-cadaver`

To handle the SSL certificate dialog for untrusted servers, one may pull down the certificate and place it into `/usr/ssl/cert.pem`. This operation renders the server treatment as "trusted".

`netrw-fixup` `netreadfixup`

If your ftp for whatever reason generates unwanted lines (such as AUTH messages) you may write a `NetReadFixup()` function:

```
function! NetReadFixup(method,line1,line2)  
  " a:line1: first new line in current file  
  " a:line2: last new line in current file  
  if a:method == 1 "rcp  
  elseif a:method == 2 "ftp + <.netrc>  
  elseif a:method == 3 "ftp + machine,uid,password,filename  
  elseif a:method == 4 "scp  
  elseif a:method == 5 "http/wget  
  elseif a:method == 6 "dav/cadaver  
  elseif a:method == 7 "rsync  
  elseif a:method == 8 "fetch  
  elseif a:method == 9 "sftp  
  else "complain  
  endif  
endfunction
```

The `NetReadFixup()` function will be called if it exists and thus allows you to customize your reading process. As a further example, `<netrw.vim>` contains just such a function to handle Windows 95 ftp. For whatever reason, Windows 95's ftp dumps four blank lines at the end of a transfer, and so it is desirable to automate their removal. Here's some code taken from `<netrw.vim>` itself:

```
if has("win95") && g:netrw_win95ftp  
  fun! NetReadFixup(method, line1, line2)  
    if method == 3 " ftp (no <.netrc>)  
      let fourblanklines= line2 - 3  
      silent fourblanklines .. ", " .. line2 .. "g/^\s*/d"  
    endif  
  endfunction
```



```
    endfunction
endif
```

(Related topics: [ftp](#) [netrw-userpass](#) [netrw-start](#))

9. Browsing [netrw-browsing](#) [netrw-browse](#) [netrw-help](#) {{{1
[netrw-browser](#) [netrw-dir](#) [netrw-list](#)

INTRODUCTION TO BROWSING [netrw-intro-browse](#) {{{2
(Quick References: [netrw-quickmaps](#) [netrw-quickcoms](#))

Netrw supports the browsing of directories on your local system and on remote hosts; browsing includes listing files and directories, entering directories, editing files therein, deleting files/directories, making new directories, moving (renaming) files and directories, copying files and directories, etc. One may mark files and execute any system command on them! The Netrw browser generally implements the previous explorer's maps and commands for remote directories, although details (such as pertinent global variable names) necessarily differ. To browse a directory, simply "edit" it!

```
vim /your/directory/
vim .
vim c:\your\directory\
```

(Related topics: [netrw-cr](#) [netrw-o](#) [netrw-p](#) [netrw-P](#) [netrw-t](#)
[netrw-mf](#) [netrw-mx](#) [netrw-D](#) [netrw-R](#) [netrw-v](#))

The Netrw remote file and directory browser handles two protocols: ssh and ftp. The protocol in the url, if it is ftp, will cause netrw also to use ftp in its remote browsing. Specifying any other protocol will cause it to be used for file transfers; but the ssh protocol will be used to do remote browsing.

To use Netrw's remote directory browser, simply attempt to read a "file" with a trailing slash and it will be interpreted as a request to list a directory:

```
vim [protocol]://[user@]hostname/path/
```

where `[protocol]` is typically scp or ftp. As an example, try:

```
vim ftp://ftp.home.vim.org/pub/vim/
```

For local directories, the trailing slash is not required. Again, because it's easy to miss: to browse remote directories, the URL must terminate with a slash!

If you'd like to avoid entering the password repeatedly for remote directory listings with ssh or scp, see [netrw-ssh-hack](#) . To avoid password entry with ftp, see [netrw-netrc](#) (if your ftp supports it).

There are several things you can do to affect the browser's display of files:

- * To change the listing style, press the "i" key ([netrw-i](#)).

Currently there are four styles: thin, long, wide, and tree.
To make that change "permanent", see [g:netrw_liststyle](#) .

* To hide files (don't want to see those xyz~ files anymore?) see [netrw-ctrl-h](#) .

* Press s to sort files by name, time, or size.

See [netrw-browse-cmds](#) for all the things you can do with netrw!

The [getftype\(\)](#) function is used to append a bit of filigree to indicate filetype to locally listed files:

```
directory : /
executable : *
fifo      : |
links     : @
sockets   : =
```

The filigree also affects the [g:netrw_sort_sequence](#) .

QUICK HELP

	netrw-quickhelp {{{2
	(Use ctrl-] to select a topic)
Intro to Browsing.....	netrw-intro-browse
Quick Reference: Maps.....	netrw-quickmap
Quick Reference: Commands.....	netrw-browse-cmds
Hiding	
Edit hiding list.....	netrw-ctrl-h
Hiding Files or Directories.....	netrw-a
Hiding/Unhiding by suffix.....	netrw-mh
Hiding dot-files.....	netrw-gh
Listing Style	
Select listing style (thin/long/wide/tree)....	netrw-i
Associated setting variable.....	g:netrw_liststyle
Shell command used to perform listing.....	g:netrw_list_cmd
Quick file info.....	netrw-qf
Sorted by	
Select sorting style (name/time/size).....	netrw-s
Editing the sorting sequence.....	netrw-S
Sorting options.....	g:netrw_sort_options
Associated setting variable.....	g:netrw_sort_sequence
Reverse sorting order.....	netrw-r

QUICK REFERENCE: MAPS

	netrw-quickmap	netrw-quickmaps	{{{2
	netrw-browse-maps		
---	-----	---	
Map	Quick Explanation	Link	
---	-----	---	
<F1>	Causes Netrw to issue help		
<cr>	Netrw will enter the directory or read the file	netrw-cr	

	Netrw will attempt to remove the file/directory	netrw-del
<c-h>	Edit file hiding list	netrw-ctrl-h
<c-l>	Causes Netrw to refresh the directory listing	netrw-ctrl-l
<c-r>	Browse using a gvim server	netrw-ctrl-r
<c-tab>	Shrink/expand a netrw/explore window	netrw-c-tab
-	Makes Netrw go up one directory	netrw--
a	Cycles between normal display, hiding (suppress display of files matching g:netrw_list_hide) and showing (display only files which match g:netrw_list_hide)	netrw-a
cd	Make browsing directory the current directory	netrw-cd
C	Setting the editing window	netrw-C
d	Make a directory	netrw-d
D	Attempt to remove the file(s)/directory(ies)	netrw-D
gb	Go to previous bookmarked directory	netrw-gb
gd	Force treatment as directory	netrw-gd
gf	Force treatment as file	netrw-gf
gh	Quick hide/unhide of dot-files	netrw-gh
gn	Make top of tree the directory below the cursor	netrw-gn
gp	Change local-only file permissions	netrw-gp
i	Cycle between thin, long, wide, and tree listings	netrw-i
I	Toggle the displaying of the banner	netrw-I
mb	Bookmark current directory	netrw-mb
mc	Copy marked files to marked-file target directory	netrw-mc
md	Apply diff to marked files (up to 3)	netrw-md
me	Place marked files on arg list and edit them	netrw-me
mf	Mark a file	netrw-mf
mF	Unmark files	netrw-mF
mg	Apply vimgrep to marked files	netrw-mg
mh	Toggle marked file suffices' presence on hiding list	netrw-mh
mm	Move marked files to marked-file target directory	netrw-mm
mp	Print marked files	netrw-mp
mr	Mark files using a shell-style regexp	netrw-mr
mt	Current browsing directory becomes markfile target	netrw-mt
mT	Apply ctags to marked files	netrw-mT
mu	Unmark all marked files	netrw-mu
mv	Apply arbitrary vim command to marked files	netrw-mv
mx	Apply arbitrary shell command to marked files	netrw-mx
mX	Apply arbitrary shell command to marked files en bloc	netrw-mX
mz	Compress/decompress marked files	netrw-mz
o	Enter the file/directory under the cursor in a new browser window. A horizontal split is used.	netrw-o
O	Obtain a file specified by cursor	netrw-O
p	Preview the file	netrw-p
P	Browse in the previously used window	netrw-P
qb	List bookmarked directories and history	netrw-qb
qf	Display information on file	netrw-qf
qF	Mark files using a quickfix list	netrw-qF
qL	Mark files using a location-list	netrw-qL
r	Reverse sorting order	netrw-r
R	Rename the designated file(s)/directory(ies)	netrw-R
s	Select sorting style: by name, time, or file size	netrw-s
S	Specify suffix priority for name-sorting	netrw-S
t	Enter the file/directory under the cursor in a new tab	netrw-t
u	Change to recently-visited directory	netrw-u

U	Change to subsequently-visited directory	netrw-U
v	Enter the file/directory under the cursor in a new browser window. A vertical split is used.	netrw-v
x	View file with an associated program	netrw-x
X	Execute filename under cursor via <code>system()</code>	netrw-X
%	Open a new file in netrw's current directory	netrw-%

netrw-mouse	netrw-leftmouse	netrw-middlemouse	netrw-rightmouse
<leftmouse>	(gvim only) selects word under mouse as if a <cr> had been pressed (ie. edit file, change directory)		
<middlemouse>	(gvim only) same as P selecting word under mouse; see netrw-P		
<rightmouse>	(gvim only) delete file/directory using word under mouse		
<2-leftmouse>	(gvim only) when: * in a netrw-selected file, AND * <code>g:netrw_retmap == 1</code> AND * the user doesn't already have a <2-leftmouse> mapping defined before netrw is autoloaded, then a double clicked leftmouse button will return to the netrw browser window. See <code>g:netrw_retmap</code> .		
<s-leftmouse>	(gvim only) like mf, will mark files. Dragging the shifted leftmouse will mark multiple files. (see netrw-mf)		

(to disable mouse buttons while browsing: `g:netrw_mousemaps`)

QUICK REFERENCE: COMMANDS	netrw-quickcom	netrw-quickcoms	
	netrw-explore-cmds	netrw-browse-cmds	{{{2
:NetrwClean[!]		netrw-clean
:NetrwSettings		netrw-settings
:Ntree		netrw-ntree
:Explore[!]	[dir]	Explore directory of current file.....	netrw-explore
:Hexplore[!]	[dir]	Horizontal Split & Explore.....	netrw-explore
:Lexplore[!]	[dir]	Left Explorer Toggle.....	netrw-explore
:Nexplore[!]	[dir]	Vertical Split & Explore.....	netrw-explore
:Pexplore[!]	[dir]	Vertical Split & Explore.....	netrw-explore
:Rexplore		Return to Explorer.....	netrw-explore
:Sexplore[!]	[dir]	Split & Explore directory	netrw-explore
:Texplore[!]	[dir]	Tab & Explore.....	netrw-explore
:Vexplore[!]	[dir]	Vertical Split & Explore.....	netrw-explore

BANNER DISPLAY

[netrw-I](#)

One may toggle the displaying of the banner by pressing "I".

Also See: [g:netrw_banner](#)

BOOKMARKING A DIRECTORY

[netrw-mb](#) [netrw-bookmark](#) [netrw-bookmarks](#) {{{2

One may easily "bookmark" the currently browsed directory by using

mb

`.netrwbook`

Bookmarks are retained in between sessions of vim in a file called `.netrwbook` as a `List`, which is typically stored in the first directory on the user's `'runtimepath'`; entries are kept in sorted order.

If there are marked files and/or directories, mb will add them to the bookmark list.

`netrw-:NetrwMB`

Additionally, one may use `:NetrwMB` to bookmark files or directories.

`:NetrwMB[!] [files/directories]`

No bang: enters files/directories into Netrw's bookmark system

No argument and in netrw buffer:

if there are marked files : bookmark marked files
otherwise : bookmark file/directory under cursor

No argument and not in netrw buffer: bookmarks current open file

Has arguments : `glob()` s each arg and bookmarks them

With bang: deletes files/directories from Netrw's bookmark system

The `:NetrwMB` command is available outside of netrw buffers (once netrw has been invoked in the session).

The file `".netrwbook"` holds bookmarks when netrw (and vim) is not active. By default, its stored on the first directory on the user's `'runtimepath'`.

Related Topics:

`netrw-gb` how to return (go) to a bookmark
`netrw-mB` how to delete bookmarks
`netrw-qb` how to list bookmarks
`g:netrw_home` controls where `.netrwbook` is kept

BROWSING

`netrw-enter` `netrw-cr` {{{2

Browsing is simple: move the cursor onto a file or directory of interest. Hitting the `<cr>` (the return key) will select the file or directory. Directories will themselves be listed, and files will be opened using the protocol given in the original read request.

CAVEAT: There are four forms of listing (see `netrw-i`). Netrw assumes that two or more spaces delimit filenames and directory names for the long and wide listing formats. Thus, if your filename or directory name has two or more sequential spaces embedded in it, or any trailing spaces, then you'll need to use the "thin" format to select it.

The `g:netrw_browse_split` option, which is zero by default, may be used to cause the opening of files to be done in a new window or tab instead of the

default. When the option is one or two, the splitting will be taken horizontally or vertically, respectively. When the option is set to three, a `<cr>` will cause the file to appear in a new tab.

When using the gui (gvim), one may select a file by pressing the `<leftmouse>` button. In addition, if

- * `g:netrw_retmap == 1` AND (its default value is 0)
- * in a netrw-selected file, AND
- * the user doesn't already have a `<2-leftmouse>` mapping defined before netrw is loaded

then a doubly-clicked leftmouse button will return to the netrw browser window.

Netrw attempts to speed up browsing, especially for remote browsing where one may have to enter passwords, by keeping and re-using previously obtained directory listing buffers. The `g:netrw_fastbrowse` variable is used to control this behavior; one may have slow browsing (no buffer re-use), medium speed browsing (re-use directory buffer listings only for remote directories), and fast browsing (re-use directory buffer listings as often as possible). The price for such re-use is that when changes are made (such as new files are introduced into a directory), the listing may become out-of-date. One may always refresh directory listing buffers by pressing `ctrl-L` (see `netrw-ctrl-l`).

`netrw-s-cr`

Squeezing the Current Tree-Listing Directory

When the tree listing style is enabled (see `netrw-i`) and one is using gvim, then the `<s-cr>` mapping may be used to squeeze (close) the directory currently containing the cursor.

Otherwise, one may remap a key combination of one's own choice to get this effect:

```
nmap <buffer> <silent> <nowait> YOURKEYCOMBO <Plug>NetrwTreeSqueeze
```

Put this line in `$HOME/ftplugin/netrw/netrw.vim`; it needs to be generated for netrw buffers only.

Related topics:

<code>netrw-ctrl-r</code>	<code>netrw-o</code>	<code>netrw-p</code>
<code>netrw-P</code>	<code>netrw-t</code>	<code>netrw-v</code>

Associated setting variables:

<code>g:netrw_browse_split</code>	<code>g:netrw_fastbrowse</code>
<code>g:netrw_ftp_list_cmd</code>	<code>g:netrw_ftp_sizelist_cmd</code>
<code>g:netrw_ftp_timelist_cmd</code>	<code>g:netrw_ssh_browse_reject</code>
<code>g:netrw_ssh_cmd</code>	<code>g:netrw_use_noswf</code>

BROWSING WITH A HORIZONTALLY SPLIT WINDOW

`netrw-o` `netrw-horiz` {{{2

Normally one enters a file or directory using the `<cr>`. However, the "o" map allows one to open a new window to hold the new directory listing or file. A horizontal split is used. (for vertical splitting, see [netrw-v](#))

Normally, the o key splits the window horizontally with the new window and cursor at the top.

Associated setting variables: [g:netrw_alto](#) [g:netrw_winsize](#)

Related topics:

[netrw-ctrl-r](#) [netrw-o](#) [netrw-p](#)
[netrw-P](#) [netrw-t](#) [netrw-v](#)

Associated setting variables:

[g:netrw_alto](#) control above/below splitting
[g:netrw_winsize](#) control initial sizing

BROWSING WITH A NEW TAB

[netrw-t](#) {{{2

Normally one enters a file or directory using the `<cr>`. The "t" map allows one to open a new window holding the new directory listing or file in a new tab.

If you'd like to have the new listing in a background tab, use `gT` .

Related topics:

[netrw-ctrl-r](#) [netrw-o](#) [netrw-p](#)
[netrw-P](#) [netrw-t](#) [netrw-v](#)

Associated setting variables:

[g:netrw_winsize](#) control initial sizing

BROWSING WITH A VERTICALLY SPLIT WINDOW

[netrw-v](#) {{{2

Normally one enters a file or directory using the `<cr>`. However, the "v" map allows one to open a new window to hold the new directory listing or file. A vertical split is used. (for horizontal splitting, see [netrw-o](#))

Normally, the v key splits the window vertically with the new window and cursor at the left.

There is only one tree listing buffer; using "v" on a displayed subdirectory will split the screen, but the same buffer will be shown twice.

Related topics:

[netrw-ctrl-r](#) [netrw-o](#) [netrw-p](#)
[netrw-P](#) [netrw-t](#) [netrw-v](#)

Associated setting variables:

[g:netrw_altv](#) control right/left splitting
[g:netrw_winsize](#) control initial sizing

BROWSING USING A GVIM SERVER

[netrw-ctrl-r](#) {{{2

One may keep a browsing gvim separate from the gvim being used to edit. Use the `<c-r>` map on a file (not a directory) in the netrw browser, and it

will use a gvim server (see `g:netrw_servername`). Subsequent use of `<cr>` (see `netrw-cr`) will re-use that server for editing files.

Related topics:

`netrw-ctrl-r` `netrw-o` `netrw-p`
`netrw-P` `netrw-t` `netrw-v`

Associated setting variables:

`g:netrw_servername` : sets name of server
`g:netrw_browse_split` : controls how `<cr>` will open files

CHANGE LISTING STYLE (THIN LONG WIDE TREE) `netrw-i` {{{2

The "i" map cycles between the thin, long, wide, and tree listing formats.

The thin listing format gives just the files' and directories' names.

The long listing is either based on the "ls" command via ssh for remote directories or displays the filename, file size (in bytes), and the time and date of last modification for local directories. With the long listing format, netrw is not able to recognize filenames which have trailing spaces. Use the thin listing format for such files.

The wide listing format uses two or more contiguous spaces to delineate filenames; when using that format, netrw won't be able to recognize or use filenames which have two or more contiguous spaces embedded in the name or any trailing spaces. The thin listing format will, however, work with such files. The wide listing format is the most compact.

The tree listing format has a top directory followed by files and directories preceded by one or more "|", which indicate the directory depth. One may open and close directories by pressing the `<cr>` key while atop the directory name.

One may make a preferred listing style your default; see `g:netrw_liststyle` . As an example, by putting the following line in your `.vimrc`,

```
let g:netrw_liststyle= 3
```

the tree style will become your default listing style.

One typical way to use the netrw tree display is to:

```
vim .  
(use i until a tree display shows)  
navigate to a file  
v (edit as desired in vertically split window)  
ctrl-w h (to return to the netrw listing)  
P (edit newly selected file in the previous window)  
ctrl-w h (to return to the netrw listing)  
P (edit newly selected file in the previous window)  
...etc...
```

Associated setting variables: `g:netrw_liststyle` `g:netrw_maxfilenamelen`
`g:netrw_timefmt` `g:netrw_list_cmd`

CHANGE FILE PERMISSION

`netrw-gp` {{{2

"gp" will ask you for a new permission for the file named under the cursor. Currently, this only works for local files.

Associated setting variables: `g:netrw_chgperm`

CHANGING TO A BOOKMARKED DIRECTORY

`netrw-gb` {{{2

To change directory back to a bookmarked directory, use

```
{cnt}gb
```

Any count may be used to reference any of the bookmarks.

Note that `netrw-qb` shows both bookmarks and history; to go to a location stored in the history see `netrw-u` and `netrw-U` .

Related Topics:

- `netrw-mB` how to delete bookmarks
- `netrw-mb` how to make a bookmark
- `netrw-qb` how to list bookmarks

CHANGING TO A PREDECESSOR DIRECTORY

`netrw-u` `netrw-updir` {{{2

Every time you change to a new directory (new for the current session), netrw will save the directory in a recently-visited directory history list (unless `g:netrw_dirhistmax` is zero; by default, it holds ten entries). With the "u" map, one can change to an earlier directory (predecessor). To do the opposite, see `netrw-U` .

The "u" map also accepts counts to go back in the history several slots. For your convenience, qb (see `netrw-qb`) lists the history number which may be used in that count.

`.netrwhist`

See `g:netrw_dirhistmax` for how to control the quantity of history stack slots. The file ".netrwhist" holds history when netrw (and vim) is not active. By default, its stored on the first directory on the user's `'runtimepath'` .

Related Topics:

- `netrw-U` changing to a successor directory
- `g:netrw_home` controls where .netrwhist is kept

CHANGING TO A SUCCESSOR DIRECTORY

`netrw-U` `netrw-downdir` {{{2

With the "U" map, one can change to a later directory (successor). This map is the opposite of the "u" map. (see `netrw-u`) Use the qb map to list both the bookmarks and history. (see `netrw-qb`)

The "U" map also accepts counts to go forward in the history several slots.

See `g:netrw_dirhistmax` for how to control the quantity of history stack slots.

CHANGING TREE TOP `netrw-ntree` `:Ntree` `netrw-gn` {{{2

One may specify a new tree top for tree listings using

```
:Ntree [dirname]
```

Without a "dirname", the current line is used (and any leading depth information is elided).

With a "dirname", the specified directory name is used.

The "gn" map will take the word below the cursor and use that for changing the top of the tree listing.

NETRW CLEAN `netrw-clean` `:NetrwClean` {{{2

With `:NetrwClean` one may easily remove netrw from one's home directory; more precisely, from the first directory on your `'runtimepath'`.

With `:NetrwClean!`, netrw will attempt to remove netrw from all directories on your `'runtimepath'`. Of course, you have to have write/delete permissions correct to do this.

With either form of the command, netrw will first ask for confirmation that the removal is in fact what you want to do. If netrw doesn't have permission to remove a file, it will issue an error message.

CUSTOMIZING BROWSING WITH A SPECIAL HANDLER `netrw-gx` `netrw-x` `netrw-handler` {{{2 (also see `netrw_filehandler`)

Certain files, such as html, gif, jpeg, (word/office) doc, etc, files, are best seen with a special handler (ie. a tool provided with your computer's operating system). Netrw allows one to invoke such special handlers by:

- * when Exploring, hit the "x" key
- * when editing, hit `gx` with the cursor atop the special filename (latter not available if the `g:netrw_nogx` variable exists)

Netrw determines which special handler by the following method:

- * if `g:netrw_browsex_viewer` exists, then it will be used to attempt to view files. Examples of useful settings (place into your `<.vimrc>`):

```
:let g:netrw_browsex_viewer= "kfmclient exec"
```

or

```
:let g:netrw_browsex_viewer= "xdg-open"
```

If `g:netrw_browsex_viewer == '-'`, then `netrwFileHandlers#Invoke()` will be

used instead (see [netrw_filehandler](#)).

If the viewer you wish to use does not support handling of a remote URL directory, set `g:netrw_browseex_support_remote` to 0.

- * for Windows 32 or 64, the URL and FileProtocolHandler dlls are used.
- * for Gnome (with `gnome-open`): `gnome-open` is used.
- * for KDE (with `kfmclient`) : `kfmclient` is used
- * for Mac OS X : `open` is used.
- * otherwise the `netrwFileHandler` plugin is used.

The file's suffix is used by these various approaches to determine an appropriate application to use to "handle" these files. Such things as OpenOffice (*.sfx), visualization (*.jpg, *.gif, etc), and PostScript (*.ps, *.eps) can be handled.

The `gx` mapping extends to all buffers; apply "`gx`" while atop a word and `netrw` will apply a special handler to it (like "`x`" works when in a `netrw` buffer). One may also use visual mode (see [visual-start](#)) to select the text that the special handler will use. Normally `gx` uses `expand("<cfile>")` to pick up the text under the cursor; one may change what `expand()` uses via the `g:netrw_gx` variable (options include "`<word>`", "`<WORD>`"). Note that `expand("<cfile>")` depends on the '`isfname`' setting. Alternatively, one may select the text to be used by `gx` by making a visual selection (see [visual-block](#)) and then pressing `gx`.

Associated setting variables:

```
g:netrw_gx      control how gx picks up the text under the cursor
g:netrw_nogx    prevent gx map while editing
g:netrw_suppress_gx_mesg  controls gx's suppression of browser messages
```

[netrw_filehandler](#)

When `g:netrw_browseex_viewer` exists and is "-", then `netrw` will attempt to handle the special file with a vim function. The "`x`" map applies a function to a file, based on its extension. Of course, the handler function must exist for it to be called!

```
Ex. mypgm.html  x -> NFH_html("scp://user@host/some/path/mypgm.html")
```

Users may write their own `netrw` File Handler functions to support more suffixes with special handling. See [<autoload/netrwFileHandlers.vim>](#) for examples on how to make file handler functions. As an example:

```
" NFH_suffix(filename)
fun! NFH_suffix(filename)
..do something special with filename..
endfun
```

These functions need to be defined in some file in your `.vim/plugin` (`vimfiles\plugin`) directory. Vim's function names may not have punctuation characters (except for the underscore) in them. To support suffixes that contain such characters, `netrw` will first convert the suffix using the following table:

@ -> AT	! -> EXCLAMATION	% -> PERCENT
: -> COLON	= -> EQUAL	? -> QUESTION
, -> COMMA	- -> MINUS	; -> SEMICOLON
\$ -> DOLLAR	+ -> PLUS	~ -> TILDE

So, for example:

```
file.rcs,v -> NFH_rcsCOMMAv()
```

If more such translations are necessary, please send me email:

NcampObell@SdrPchip.AorgM-NOSPAM

with a request. (remove the embedded NOSPAM first)

Associated setting variable: [g:netrw_browsex_viewer](#)

DELETING BOOKMARKS

[netrw-curdir](#)
[netrw-mB](#) {{{2

To delete a bookmark, use

```
{cnt}mB
```

If there are marked files, then mB will remove them from the bookmark list.

Alternatively, one may use :NetrwMB! (see [netrw-:NetrwMB](#)).

```
:NetrwMB! [files/directories]
```

Related Topics:

[netrw-gb](#) how to return (go) to a bookmark
[netrw-mb](#) how to make a bookmark
[netrw-qb](#) how to list bookmarks

DELETING FILES OR DIRECTORIES [netrw-delete](#) [netrw-D](#) [netrw-del](#) {{{2

If files have not been marked with [netrw-mf](#) : (local marked file list)

Deleting/removing files and directories involves moving the cursor to the file/directory to be deleted and pressing "D". Directories must be empty first before they can be successfully removed. If the directory is a softlink to a directory, then netrw will make two requests to remove the directory before succeeding. Netrw will ask for confirmation before doing the removal(s). You may select a range of lines with the "V" command (visual selection), and then pressing "D".

If files have been marked with [netrw-mf](#) : (local marked file list)

Marked files (and empty directories) will be deleted; again, you'll be asked to confirm the deletion before it actually takes place.

A further approach is to delete files which match a pattern.

* use `:MF` pattern (see `netrw-MF`); then press "D".

* use `mr` (see `netrw-mr`) which will prompt you for pattern.
This will cause the matching files to be marked. Then,
press "D".

Please **note** that only empty directories may be deleted with the "D" mapping.
Regular files are deleted with `delete()` , too.

The `g:netrw_rm_cmd` , `g:netrw_rmf_cmd` , and `g:netrw_rmdir_cmd` variables are used to control the attempts to remove remote files and directories. The `g:netrw_rm_cmd` is used with files, and its default value is:

```
g:netrw_rm_cmd: ssh HOSTNAME rm
```

The `g:netrw_rmdir_cmd` variable is used to support the removal of directories.
Its default value is:

```
g:netrw_rmdir_cmd : ssh HOSTNAME rmdir
```

If removing a directory fails with `g:netrw_rmdir_cmd`, `netrw` then will attempt to remove it again using the `g:netrw_rmf_cmd` variable. Its default value is:

```
g:netrw_rmf_cmd : ssh HOSTNAME rm -f
```

Related topics: `netrw-d`

Associated setting variable: `g:netrw_rm_cmd` `g:netrw_ssh_cmd`

```
netrw-explore  netrw-hexplore  netrw-nexplore  netrw-pexplore  
netrw-rexplore netrw-sexplore  netrw-texplore  netrw-vexplore  netrw-lexplore  
DIRECTORY EXPLORATION COMMANDS  {{{2
```

```
: [N]Explore[!] [dir]... Explore directory of current file           :Explore  
: [N]Hexplore[!] [dir]... Horizontal Split & Explore                :Hexplore  
: [N]Lexplore[!] [dir]... Left Explorer Toggle                      :Lexplore  
: [N]Sexplore[!] [dir]... Split&Explore current file's directory   :Sexplore  
: [N]Vexplore[!] [dir]... Vertical Split & Explore                  :Vexplore  
: Texplore      [dir]... Tab & Explore                              :Texplore  
: Rexplore      ... Return to/from Explorer                        :Rexplore
```

```
Used with :Explore **/pattern : (also see netrw-starstar )  
:Nexplore..... go to next matching file           :Nexplore  
:Pexplore..... go to previous matching file       :Pexplore
```

`netrw-:Explore`

`:Explore` will open the local-directory browser on the current file's directory (or on directory `[dir]` if specified). The window will be split only if the file has been modified and `'hidden'` is not set, otherwise the browsing window will take over that window. Normally the splitting is taken horizontally.
Also see: `netrw-:Rexplore`

`:Explore!` is like `:Explore`, but will use vertical splitting.

netrw-:Hexplore

:Hexplore [dir] does an :Explore with :belowright horizontal splitting.
:Hexplore! [dir] does an :Explore with :aboveleft horizontal splitting.

netrw-:Lexplore

:[N]Lexplore [dir] toggles a full height Explorer window on the left hand side of the current tab. It will open a netrw window on the current directory if [dir] is omitted; a :Lexplore [dir] will show the specified directory in the left-hand side browser display no matter from which window the command is issued.

By default, :Lexplore will change an uninitialized g:netrw_chgwin to 2; edits will thus preferentially be made in window#2.

The [N] specifies a g:netrw_winsize just for the new :Lexplore window. That means that

- if [N] < 0 : use N columns for the Lexplore window
- if [N] = 0 : a normal split is made
- if [N] > 0 : use N% of the current window will be used for the new window

Those who like this method often also like tree style displays; see g:netrw_liststyle .

:[N]Lexplore! [dir] is similar to :Lexplore, except that the full-height Explorer window will open on the right hand side and an uninitialized g:netrw_chgwin will be set to 1 (eg. edits will preferentially occur in the leftmost window).

Also see: netrw-C g:netrw_browse_split g:netrw_wiw
 netrw-p netrw-P g:netrw_chgwin
 netrw-c-tab g:netrw_winsize

netrw-:Sexplore

:[N]Sexplore will always split the window before invoking the local-directory browser. As with Explore, the splitting is normally done horizontally.

:[N]Sexplore! [dir] is like :Sexplore, but the splitting will be done vertically.

netrw-:Texplore

:Texplore [dir] does a :tabnew before generating the browser window

netrw-:Vexplore

:[N]Vexplore [dir] does an :Explore with :leftabove vertical splitting.
:[N]Vexplore! [dir] does an :Explore with :rightbelow vertical splitting.

The optional parameters are:

[N]: This parameter will override g:netrw_winsize to specify the quantity of rows and/or columns the new explorer window should have. Otherwise, the g:netrw_winsize variable, if it has been specified by the user, is used to control the quantity of rows and/or columns new explorer windows should have.

[dir]: By default, these explorer commands use the current file's directory. However, one may explicitly provide a directory (path) to use instead; ie.

`:Explore /some/path`

`netrw-:Rexplore`

`:Rexplore` This command is a little different from the other Explore commands as it doesn't necessarily open an Explorer window.

`Return to Explorer`

When one edits a file using netrw which can occur, for example, when pressing `<cr>` while the cursor is atop a filename in a netrw browser window, a `:Rexplore` issued while editing that file will return the display to that of the last netrw browser display in that window.

`Return from Explorer`

Conversely, when one is editing a directory, issuing a `:Rexplore` will return to editing the file that was last edited in that window.

The `<2-leftmouse>` map (which is only available under gvim and cooperative terms) does the same as `:Rexplore`.

Also see: `g:netrw_alto` `g:netrw_altv` `g:netrw_winsize`

`netrw-star` `netrw-starpat` `netrw-starstar` `netrw-starstarpat` `netrw-grep`
EXPLORING WITH STARS AND PATTERNS {{{2

When Explore, Sexplore, Hexplore, or Vexplore are used with one of the following four patterns Explore generates a list of files which satisfy the request for the local file system. These exploration patterns will not work with remote file browsing.

```
*/filepat  files in current directory which satisfy filepat
**/filepat files in current directory or below which satisfy the
           file pattern
*//pattern files in the current directory which contain the
           pattern (vimgrep is used)
**//pattern files in the current directory or below which contain
           the pattern (vimgrep is used)
```

<

The cursor will be placed on the first file in the list. One may then continue to go to subsequent files on that list via `:Nexplore` or to preceding files on that list with `:Pexplore`. Explore will update the directory and place the cursor appropriately.

A plain

`:Explore`

will clear the explore list.

If your console or gui produces recognizable shift-up or shift-down sequences, then you'll likely find using shift-downarrow and shift-uparrow convenient. They're mapped by netrw as follows:

```
<s-down> == Nexplore, and
<s-up>    == Pexplore.
```

As an example, consider

```
:Explore */*.c
:Nexplore
:Nexplore
:Pexplore
```

The status line will show, on the right hand side of the status line, a message like "Match 3 of 20".

Associated setting variables:

```
g:netrw_keepdir          g:netrw_browse_split
g:netrw_fastbrowse      g:netrw_ftp_browse_reject
g:netrw_ftp_list_cmd    g:netrw_ftp_sizelist_cmd
g:netrw_ftp_timelist_cmd g:netrw_list_cmd
g:netrw_liststyle
```

DISPLAYING INFORMATION ABOUT FILE

`netrw-qf` {{{2

With the cursor atop a filename, pressing "qf" will reveal the file's size and last modification timestamp. Currently this capability is only available for local files.

EDIT FILE OR DIRECTORY HIDING LIST

`netrw-ctrl-h` `netrw-edithide` {{{2

The "`<ctrl-h>`" map brings up a requestor allowing the user to change the file/directory hiding list contained in `g:netrw_list_hide`. The hiding list consists of one or more patterns delimited by commas. Files and/or directories satisfying these patterns will either be hidden (ie. not shown) or be the only ones displayed (see `netrw-a`).

The "gh" mapping (see `netrw-gh`) quickly alternates between the usual hiding list and the hiding of files or directories that begin with ".".

As an example,

```
let g:netrw_list_hide= '\(^|\s\s)\zs\.\S\+'
```

Effectively, this makes the effect of a `netrw-gh` command the initial setting. What it means:

```
\(^|\s\s)  : if the line begins with the following, -or-
             two consecutive spaces are encountered
\s        : start the hiding match now
\.\       : if it now begins with a dot
\S\+     : and is followed by one or more non-whitespace
           characters
```


Associated setting variables: `g:netrw_hide` `g:netrw_list_hide`
Associated topics: `netrw-a` `netrw-gh` `netrw-mh`

EDITING THE SORTING SEQUENCE `netrw-sort-sequence`
`netrw-S` `netrw-sortsequence` {{{2

When "Sorted by" is name, one may specify priority via the sorting sequence (`g:netrw_sort_sequence`). The sorting sequence typically prioritizes the name-listing by suffix, although any pattern will do. Patterns are delimited by commas. The default sorting sequence is (all one line):

For Unix:

```
'[\\/]$, \<core\%(\\.d\+)\|=, \.[a-np-z]$, \.h$, \.c$, \.cpp$, *, \.o$, \.obj$,  
\.info$, \.swp$, \.bak$, \~$'
```

Otherwise:

```
'[\\/]$, \.[a-np-z]$, \.h$, \.c$, \.cpp$, *, \.o$, \.obj$, \.info$,  
\.swp$, \.bak$, \~$'
```

The lone `*` is where all filenames not covered by one of the other patterns will end up. One may change the sorting sequence by modifying the `g:netrw_sort_sequence` variable (either manually or in your `<.vimrc>`) or by using the "S" map.

Related topics: `netrw-s` `netrw-S`
Associated setting variables: `g:netrw_sort_sequence` `g:netrw_sort_options`

EXECUTING FILE UNDER CURSOR VIA SYSTEM() `netrw-X` {{{2

Pressing `X` while the cursor is atop an executable file will yield a prompt using the filename asking for any arguments. Upon pressing a `[return]`, `netrw` will then call `system()` with that command and arguments. The result will be displayed by `:echomsg`, and so `:messages` will repeat display of the result. Ansi escape sequences will be stripped out.

See `cmdline-window` for directions for more on how to edit the arguments.

FORCING TREATMENT AS A FILE OR DIRECTORY `netrw-gd` `netrw-gf` {{{2

Remote symbolic links (ie. those listed via `ssh` or `ftp`) are problematic in that it is difficult to tell whether they link to a file or to a directory.

To force treatment as a file: use
`gf`

To force treatment as a directory: use
`gd`

GOING UP `netrw--` {{{2

To go up a directory, press "-" or press the <cr> when atop the ../ directory entry in the listing.

Netrw will use the command in `g:netrw_list_cmd` to perform the directory listing operation after changing HOSTNAME to the host specified by the user-prpvided url. By default netrw provides the command as:

```
ssh HOSTNAME ls -FLa
```

where the HOSTNAME becomes the [user@]hostname as requested by the attempt to read. Naturally, the user may override this command with whatever is preferred. The NetList function which implements remote browsing expects that directories will be flagged by a trailing slash.

HIDING FILES OR DIRECTORIES

```
netrw-a netrw-hiding {{{2
```

Netrw's browsing facility allows one to use the hiding list in one of three ways: ignore it, hide files which match, and show only those files which match.

If no files have been marked via `netrw-mf` :

The "a" map allows the user to cycle through the three hiding modes.

The `g:netrw_list_hide` variable holds a comma delimited list of patterns based on regular expressions (ex. `^.*\obj$,^\.`) which specify the hiding list. (also see `netrw-ctrl-h`) To set the hiding list, use the <c-h> map. As an example, to hide files which begin with a ".", one may use the <c-h> map to set the hiding list to `^\..*` (or one may put `let g:netrw_list_hide= '^\..*'` in one's `<.vimrc>`). One may then use the "a" key to show all files, hide matching files, or to show only the matching files.

```
Example: \.[ch]$
This hiding list command will hide/show all *.c and *.h files.
```

```
Example: \.c$,\.h$
This hiding list command will also hide/show all *.c and *.h files.
```

Don't forget to use the "a" map to select the mode (normal/hiding/show) you want!

If files have been marked using `netrw-mf` , then this command will:

```
if showing all files or non-hidden files:
  modify the g:netrw_list_hide list by appending the marked files to it
  and showing only non-hidden files.
```

```
else if showing hidden files only:
  modify the g:netrw_list_hide list by removing the marked files from it
  and showing only non-hidden files.
endif
```

`netrw-gh` `netrw-hide`

As a quick shortcut, one may press

`gh`

to toggle between hiding files which begin with a period (dot) and not hiding them.

Associated setting variables: `g:netrw_list_hide` `g:netrw_hide`

Associated topics: `netrw-a` `netrw-ctrl-h` `netrw-mh`

`netrw-gitignore`

Netrw provides a helper function '`netrw_gitignore#Hide()`' that, when used with `g:netrw_list_hide` automatically hides all git-ignored files.

'`netrw_gitignore#Hide`' searches for patterns in the following files:

```
 './.gitignore'  
 './.git/info/exclude'  
 global gitignore file: `git config --global core.excludesfile`  
 system gitignore file: `git config --system core.excludesfile`
```

Files that do not exist, are ignored.

Git-ignore patterns are taken from existing files, and converted to patterns for hiding files. For example, if you had '*.log' in your '.gitignore' file, it would be converted to '*.\\log'.

To use this function, simply assign its output to `g:netrw_list_hide` option.

```
Example: let g:netrw_list_hide= netrw_gitignore#Hide()  
Git-ignored files are hidden in Netrw.
```

```
Example: let g:netrw_list_hide= netrw_gitignore#Hide('my_gitignore_file')  
Function can take additional files with git-ignore patterns.
```

```
Example: let g:netrw_list_hide= netrw_gitignore#Hide() .. '*.\\swp$'  
Combining 'netrw_gitignore#Hide' with custom patterns.
```

IMPROVING BROWSING

`netrw-listhack` `netrw-ssh-hack` {{{2

Especially with the remote directory browser, constantly entering the password is tedious.

For Linux/Unix systems, the book "Linux Server Hacks - 100 industrial strength tips & tools" by Rob Flickenger (O'Reilly, ISBN 0-596-00461-3) gives a tip for setting up no-password ssh and scp and discusses associated security issues. It used to be available at <http://hacks.oreilly.com/pub/h/66>, but apparently that address is now being redirected to some "hackzine". I'll attempt a summary based on that article and on a communication from Ben Schmidt:

1. Generate a public/private key pair on the local machine (ssh client):
`ssh-keygen -t rsa`

(saving the file in ~/.ssh/id_rsa as prompted)

2. Just hit the <CR> when asked for passphrase (twice) for no passphrase. If you do use a passphrase, you will also need to use ssh-agent so you only have to type the passphrase once per session. If you don't use a passphrase, simply logging onto your local computer or getting access to the keyfile in any way will suffice to access any ssh servers which have that key authorized for login.

3. This creates two files:

```
~/.ssh/id_rsa  
~/.ssh/id_rsa.pub
```

4. On the target machine (ssh server):

```
cd  
mkdir -p .ssh  
chmod 0700 .ssh
```

5. On your local machine (ssh client): (one line)

```
ssh {serverhostname}  
cat '>>' '~/.ssh/authorized_keys2' < ~/.ssh/id_rsa.pub
```

or, for OpenSSH, (one line)

```
ssh {serverhostname}  
cat '>>' '~/.ssh/authorized_keys' < ~/.ssh/id_rsa.pub
```

You can test it out with

```
ssh {serverhostname}
```

and you should be log onto the server machine without further need to type anything.

If you decided to use a passphrase, do:

```
ssh-agent $SHELL  
ssh-add  
ssh {serverhostname}
```

You will be prompted for your key passphrase when you use ssh-add, but not subsequently when you use ssh. For use with vim, you can use

```
ssh-agent vim
```

and, when next within vim, use

```
:!ssh-add
```

Alternatively, you can apply ssh-agent to the terminal you're planning on running vim in:

```
ssh-agent xterm &
```

and do ssh-add whenever you need.

For Windows, folks on the vim mailing list have mentioned that Pageant helps with avoiding the constant need to enter the password.

Kingston Fung wrote about another way to avoid constantly needing to enter passwords:

In order to avoid the need to type in the password for scp each time, you provide a hack in the docs to set up a non password ssh account. I found a better way to do that: I can use a regular ssh account which uses a

password to access the material without the need to key-in the password each time. It's good for security and convenience. I tried ssh public key authorization + ssh-agent, implementing this, and it works! Here are two links with instructions:

<http://www.ibm.com/developerworks/library/l-keyc2/>
<http://sial.org/howto/openssh/publickey-auth/>

Ssh hints:

Thomer Gil has provided a hint on how to speed up netrw+ssh:
http://thomer.com/howtos/netrw_ssh.html

Alex Young has several hints on speeding ssh up:
<http://usevim.com/2012/03/16/editing-remote-files/>

LISTING BOOKMARKS AND HISTORY `netrw-qb` `netrw-listbookmark` {{{2

Pressing "qb" (query bookmarks) will list both the bookmarked directories and directory traversal history.

Related Topics:

- `netrw-gb` how to return (go) to a bookmark
- `netrw-mb` how to make a bookmark
- `netrw-mB` how to delete bookmarks
- `netrw-u` change to a predecessor directory via the history stack
- `netrw-U` change to a successor directory via the history stack

MAKING A NEW DIRECTORY `netrw-d` {{{2

With the "d" map one may make a new directory either remotely (which depends on the global variable `g:netrw_mkdir_cmd`) or locally (which depends on the global variable `g:netrw_localmkdir`). Netrw will issue a request for the new directory's name. A bare `<CR>` at that point will abort the making of the directory. Attempts to make a local directory that already exists (as either a file or a directory) will be detected, reported on, and ignored.

Related topics: `netrw-D`

Associated setting variables: `g:netrw_localmkdir` `g:netrw_mkdir_cmd`
`g:netrw_remote_mkdir` `netrw-%`

MAKING THE BROWSING DIRECTORY THE CURRENT DIRECTORY `netrw-cd` {{{2

By default, `g:netrw_keepdir` is 1. This setting means that the current directory will not track the browsing directory. (done for backwards compatibility with v6's file explorer).

Setting `g:netrw_keepdir` to 0 tells netrw to make vim's current directory track netrw's browsing directory.

However, given the default setting for `g:netrw_keepdir` of 1 where netrw

maintains its own separate notion of the current directory, in order to make the two directories the same, use the "cd" map (type cd). That map will set Vim's notion of the current directory to netrw's current browsing directory.

`netrw-cd` : This map's name was changed from "c" to cd (see `netrw-cd`).
This change was done to allow for `netrw-cb` and `netrw-cB` maps.

Associated setting variable: `g:netrw_keepdir`

MARKING FILES

`netrw-:MF` `netrw-mf` {{{2

(also see `netrw-mr`)

Netrw provides several ways to mark files:

- * One may mark files with the cursor atop a filename and then pressing "mf".
- * With gvim, in addition one may mark files with `<s-leftmouse>`. (see `netrw-mouse`)
- * One may use the `:MF` command, which takes a list of files (for local directories, the list may include wildcards -- see `glob()`)

```
:MF *.c
```

(Note that `:MF` uses `<f-args>` to break the line at spaces)

- * Mark files using the `argument-list` (`netrw-mA`)
- * Mark files based upon a `location-list` (`netrw-qL`)
- * Mark files based upon the quickfix list (`netrw-qF`) (`quickfix-error-lists`)

The following netrw maps make use of marked files:

```
netrw-a      Hide marked files/directories  
netrw-D      Delete marked files/directories  
netrw-ma     Move marked files' names to arglist  
netrw-mA     Move arglist filenames to marked file list  
netrw-mb     Append marked files to bookmarks  
netrw-mB     Delete marked files from bookmarks  
netrw-mc     Copy marked files to target  
netrw-md     Apply vimdiff to marked files  
netrw-me     Edit marked files  
netrw-mF     Unmark marked files  
netrw-mg     Apply vimgrep to marked files  
netrw-mm     Move marked files to target  
netrw-mp     Print marked files  
netrw-ms     Netrw will source marked files  
netrw-mt     Set target for netrw-mm and netrw-mc
```

```

netrw-mT  Generate tags using marked files
netrw-mv  Apply vim command to marked files
netrw-mx  Apply shell command to marked files
netrw-mX  Apply shell command to marked files, en bloc
netrw-mz  Compress/Decompress marked files
netrw-O   Obtain marked files
netrw-R   Rename marked files

```

One may unmark files one at a time the same way one marks them; ie. place the cursor atop a marked file and press "mf". This process also works with <s-leftmouse> using gvim. One may unmark all files by pressing "mu" (see [netrw-mu](#)).

Marked files are highlighted using the "netrwMarkFile" highlighting group, which by default is linked to "Identifier" (see Identifier under [group-name](#)). You may change the highlighting group by putting something like

```

highlight clear netrwMarkFile
hi link netrwMarkFile ..whatever..

```

into \$HOME/.vim/after/syntax/netrw.vim .

If the mouse is enabled and works with your vim, you may use <s-leftmouse> to mark one or more files. You may mark multiple files by dragging the shifted leftmouse. (see [netrw-mouse](#))

[markfilelist](#) [global_markfilelist](#) [local_markfilelist](#)

All marked files are entered onto the global marked file list; there is only one such list. In addition, every netrw buffer also has its own buffer-local marked file list; since netrw buffers are associated with specific directories, this means that each directory has its own local marked file list. The various commands which operate on marked files use one or the other of the marked file lists.

Known Problem: if one is using tree mode ([g:netrw_liststyle](#)) and several directories have files with the same name, then marking such a file will result in all such files being highlighted as if they were all marked. The [markfilelist](#) , however, will only have the selected file in it. This problem is unlikely to be fixed.

UNMARKING FILES

[netrw-mF](#) {{{2

(also see [netrw-mf](#) , [netrw-mu](#))

The "mF" command will unmark all files in the current buffer. One may also use mf ([netrw-mf](#)) on a specific, already marked, file to unmark just that file.

MARKING FILES BY LOCATION LIST

[netrw-ql](#) {{{2

(also see [netrw-mf](#))

One may convert [location-list](#) s into a marked file list using "qL". You may then proceed with commands such as me ([netrw-me](#)) to edit them.

MARKING FILES BY QUICKFIX LIST
(also see [netrw-mf](#))

[netrw-qF](#) {{{2

One may convert [quickfix-error-lists](#) into a marked file list using "qF". You may then proceed with commands such as [me](#) ([netrw-me](#)) to edit them. Quickfix error lists are generated, for example, by calls to [:vimgrep](#) .

MARKING FILES BY REGULAR EXPRESSION
(also see [netrw-mf](#))

[netrw-mr](#) {{{2

One may also mark files by pressing "mr"; netrw will then issue a prompt, "Enter regexp: ". You may then enter a shell-style regular expression such as *.c\$ (see [glob\(\)](#)). For remote systems, [glob\(\)](#) doesn't work -- so netrw converts "*" into ".*" (see [regexp](#)) and marks files based on that. In the future I may make it possible to use [regexp](#) s instead of [glob\(\)](#)-style expressions (yet-another-option).

See [cmdline-window](#) for directions on more on how to edit the regular expression.

MARKED FILES, ARBITRARY VIM COMMAND

[netrw-mv](#) {{{2

(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the local marked-file list)

The "mv" map causes netrw to execute an arbitrary vim command on each file on the local marked file list, individually:

- * [!split](#)
- * [!keepalt e file](#)
- * [!run vim command](#)
- * [!keepalt wq!](#)

A prompt, "Enter vim command: ", will be issued to elicit the vim command you wish used. See [cmdline-window](#) for directions for more on how to edit the command.

MARKED FILES, ARBITRARY SHELL COMMAND

[netrw-mx](#) {{{2

(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the local marked-file list)

Upon activation of the "mx" map, netrw will query the user for some (external) command to be applied to all marked files. All "%s" in the command will be substituted with the name of each marked file in turn. If no "%s" are in the command, then the command will be followed by a space and a marked filename.

Example:

```
(mark files)
mx
Enter command: cat
```


The result is a series of shell commands:
cat 'file1'
cat 'file2'
...

MARKED FILES, ARBITRARY SHELL COMMAND, EN BLOC netrw-mX {{{2
(See netrw-mf and netrw-mr for how to mark files)
(uses the global marked-file list)

Upon activation of the 'mX' map, netrw will query the user for some (external) command to be applied to all marked files on the global marked file list. The "en bloc" means that one command will be executed on all the files at once:

command files

This approach is useful, for example, to select files and make a tarball:

(mark files)
mX
Enter command: tar cf mynewtarball.tar

The command that will be run with this example:

tar cf mynewtarball.tar 'file1' 'file2' ...

MARKED FILES: ARGUMENT LIST netrw-ma netrw-mA
(See netrw-mf and netrw-mr for how to mark files)
(uses the global marked-file list)

Using ma, one moves filenames from the marked file list to the argument list.
Using mA, one moves filenames from the argument list to the marked file list.

See Also: netrw-cb netrw-cB netrw-qF argument-list :args

MARKED FILES: BUFFER LIST netrw-cb netrw-cB
(See netrw-mf and netrw-mr for how to mark files)
(uses the global marked-file list)

Using cb, one moves filenames from the marked file list to the buffer list.
Using cB, one copies filenames from the buffer list to the marked file list.

See Also: netrw-ma netrw-mA netrw-qF buffer-list :buffers

MARKED FILES: COMPRESSION AND DECOMPRESSION netrw-mz {{{2
(See netrw-mf and netrw-mr for how to mark files)
(uses the local marked file list)

If any marked files are compressed, then "mz" will decompress them.
If any marked files are decompressed, then "mz" will compress them
using the command specified by g:netrw_compress ; by default,

that's "gzip".

For decompression, netrw uses a [Dictionary](#) of suffices and their associated decompressing utilities; see [g:netrw_decompress](#) .

Remember that one can mark multiple files by regular expression (see [netrw-mr](#)); this is particularly useful to facilitate compressing and decompressing a large number of files.

Associated setting variables: [g:netrw_compress](#) [g:netrw_decompress](#)

MARKED FILES: COPYING [netrw-mc](#) {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(Uses the global marked file list)

Select a target directory with mt ([netrw-mt](#)). Then change directory, select file(s) (see [netrw-mf](#)), and press "mc". The copy is done from the current window (where one does the mf) to the target.

If one does not have a target directory set with [netrw-mt](#) , then netrw will query you for a directory to copy to.

One may also copy directories and their contents (local only) to a target directory.

Associated setting variables:
[g:netrw_localcopycmd](#) [g:netrw_localcopycmdopt](#)
[g:netrw_localcopydircmd](#) [g:netrw_localcopydircmdopt](#)
[g:netrw_ssh_cmd](#)

MARKED FILES: DIFF [netrw-md](#) {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the global marked file list)

Use [vimdiff](#) to visualize difference between selected files (two or three may be selected for this). Uses the global marked file list.

MARKED FILES: EDITING [netrw-me](#) {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the global marked file list)

The "me" command will place the marked files on the [arglist](#) and commence editing them. One may return the to explorer window with [:Rexplore](#) .
(use [:n](#) and [:p](#) to edit next and previous files in the arglist)

MARKED FILES: GREP [netrw-mg](#) {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the global marked file list)

The "mg" command will apply [:vimgrep](#) to the marked files.
The command will ask for the requested pattern; one may then enter:

```
/pattern/[g][j]  
! /pattern/[g][j]
```

pattern

With `/pattern/`, editing will start with the first item on the `quickfix` list that `vimgrep` sets up (see `:copen`, `:cnext`, `:cprevious`, `:cclose`). The `:vimgrep` command is in use, so without `'g'` each line is added to quickfix list only once; with `'g'` every match is included.

With `/pattern/j`, `"mg"` will winnow the current marked file list to just those marked files also possessing the specified pattern. Thus, one may use

```
mr ...file-pattern...
mg /pattern/j
```

to have a marked file list satisfying the file-pattern but also restricted to files containing some desired pattern.

MARKED FILES: HIDING AND UNHIDING BY SUFFIX netrw-mh {{{2
(See `netrw-mf` and `netrw-mr` for how to mark files)
(uses the local marked file list)

The `"mh"` command extracts the suffices of the marked files and toggles their presence on the hiding list. Please [note](#) that marking the same suffix this way multiple times will result in the suffix's presence being toggled for each file (so an even quantity of marked files having the same suffix is the same as not having bothered to select them at all).

Related topics: `netrw-a` `g:netrw_list_hide`

MARKED FILES: MOVING netrw-mm {{{2
(See `netrw-mf` and `netrw-mr` for how to mark files)
(uses the global marked file list)

WARNING: moving files is more dangerous than copying them. A file being moved is first copied and then deleted; if the copy operation fails and the delete succeeds, you will lose the file. Either try things out with unimportant files first or do the copy and then delete yourself using `mc` and `D`. Use at your own risk!

Select a target directory with `mt` (`netrw-mt`). Then change directory, select file(s) (see `netrw-mf`), and press `"mm"`. The move is done from the current window (where one does the `mf`) to the target.

Associated setting variable: `g:netrw_localmovecmd` `g:netrw_ssh_cmd`

MARKED FILES: PRINTING netrw-mp {{{2
(See `netrw-mf` and `netrw-mr` for how to mark files)
(uses the local marked file list)

When `"mp"` is used, `netrw` will apply the `:hardcopy` command to marked files. What `netrw` does is open each file in a one-line window, execute `hardcopy`, then close the one-line window.

MARKED FILES: SOURCING netrw-ms {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the local marked file list)

With "ms", netrw will source the marked files (using vim's `:source` command)

MARKED FILES: SETTING THE TARGET DIRECTORY netrw-mt {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)

Set the marked file copy/move-to target (see [netrw-mc](#) and [netrw-mm](#)):

- * If the cursor is atop a file name, then the netrw window's currently displayed directory is used for the copy/move-to target.
- * Also, if the cursor is in the banner, then the netrw window's currently displayed directory is used for the copy/move-to target. Unless the target already is the current directory. In which case, typing "mf" clears the target.
- * However, if the cursor is atop a directory name, then that directory is used for the copy/move-to target
- * One may use the `:MT [directory]` command to set the target [netrw-:MT](#). This command uses `<q-args>`, so spaces in the directory name are permitted without escaping.
- * With mouse-enabled vim or with gvim, one may select a target by using `<c-leftmouse>`

There is only one copy/move-to target at a time in a vim session; ie. the target is a script variable (see [s:var](#)) and is shared between all netrw windows (in an instance of vim).

When using menus and gvim, netrw provides a "Targets" entry which allows one to pick a target from the list of bookmarks and history.

Related topics:

- Marking Files..... [netrw-mf](#)
- Marking Files by Regular Expression..... [netrw-mr](#)
- Marked Files: Target Directory Using Bookmarks..... [netrw-Tb](#)
- Marked Files: Target Directory Using History..... [netrw-Th](#)

MARKED FILES: TAGGING netrw-mT {{{2
(See [netrw-mf](#) and [netrw-mr](#) for how to mark files)
(uses the global marked file list)

The "mT" mapping will apply the command in [g:netrw_ctags](#) (by default, it is "ctags") to marked files. For remote browsing, in order to create a tags file netrw will use ssh (see [g:netrw_ssh_cmd](#)), and so ssh must be available for this to work on remote systems. For your local system, see [ctags](#) on how to get a version. I myself use hdrtags, currently available at

<http://www.drchip.org/astronaut/src/index.html> , and have

```
let g:netrw_ctags= "hdrtag"
```

in my `<.vimrc>`.

When a remote set of files are tagged, the resulting tags file is "obtained"; ie. a copy is transferred to the local system's directory. The now local tags file is then modified so that one may use it through the network. The modification made concerns the names of the files in the tags; each filename is preceded by the netrw-compatible URL used to obtain it. When one subsequently uses one of the go to tag actions (`tags`), the URL will be used by netrw to edit the desired file and go to the tag.

Associated setting variables: `g:netrw_ctags` `g:netrw_ssh_cmd`

MARKED FILES: TARGET DIRECTORY USING BOOKMARKS `netrw-Tb` {{{2

Sets the marked file copy/move-to target.

The `netrw-qb` map will give you a list of bookmarks (and history). One may choose one of the bookmarks to become your marked file target by using `[count]Tb` (default count: 1).

Related topics:

- Copying files to target..... `netrw-mc`
- Listing Bookmarks and History..... `netrw-qb`
- Marked Files: Setting The Target Directory..... `netrw-mt`
- Marked Files: Target Directory Using History..... `netrw-Th`
- Marking Files..... `netrw-mf`
- Marking Files by Regular Expression..... `netrw-mr`
- Moving files to target..... `netrw-mm`

MARKED FILES: TARGET DIRECTORY USING HISTORY `netrw-Th` {{{2

Sets the marked file copy/move-to target.

The `netrw-qb` map will give you a list of history (and bookmarks). One may choose one of the history entries to become your marked file target by using `[count]Th` (default count: 0; ie. the current directory).

Related topics:

- Copying files to target..... `netrw-mc`
- Listing Bookmarks and History..... `netrw-qb`
- Marked Files: Setting The Target Directory..... `netrw-mt`
- Marked Files: Target Directory Using Bookmarks..... `netrw-Tb`
- Marking Files..... `netrw-mf`
- Marking Files by Regular Expression..... `netrw-mr`
- Moving files to target..... `netrw-mm`

MARKED FILES: UNMARKING `netrw-mu` {{{2
(See `netrw-mf` , `netrw-mF`)

The "mu" mapping will unmark all currently marked files. This command differs from "mF" as the latter only unmarks files in the current directory whereas "mu" will unmark global and all buffer-local marked files. (see [netrw-mF](#))

NETRW BROWSER VARIABLES [netrw-browser-settings](#)
 [netrw-browser-options](#) [netrw-browser-var](#) {{{2

(if you're interested in the netrw file transfer settings, see [netrw-options](#) and [netrw-protocol](#))

The `<netrw.vim>` browser provides settings in the form of variables which you may modify; by placing these settings in your `<.vimrc>`, you may customize your browsing preferences. (see also: [netrw-settings](#))

--- Var ---	----- Explanation -----
g:netrw_altfile	some like <code>CTRL-^</code> to return to the last edited file. Choose that by setting this parameter to 1. Others like <code>CTRL-^</code> to return to the netrw browsing buffer. Choose that by setting this parameter to 0. default: =0
g:netrw_alto	change from above splitting to below splitting by setting this variable (see netrw-o) default: =&sb (see 'sb')
g:netrw_altv	change from left splitting to right splitting by setting this variable (see netrw-v) default: =&spr (see 'spr')
g:netrw_banner	enable/suppress the banner =0: suppress the banner =1: banner is enabled (default)
g:netrw_bannerbackslash	if this variable exists and is not zero, the banner will be displayed with backslashes rather than forward slashes.
g:netrw_browse_split	when browsing, <code><cr></code> will open the file by: =0: re-using the same window (default) =1: horizontally splitting the window first =2: vertically splitting the window first =3: open file in new tab =4: act like "P" (ie. open previous window) Note that g:netrw_preview may be used to get vertical splitting instead of horizontal splitting. =[servername,tab-number,window-number]

Given a `List` such as this, a remote server named by the "servername" will be used for editing. It will also use the specified tab and window numbers to perform editing (see `clientserver` , `netrw-ctrl-r`)
This option does not affect the production of `:Lexplore` windows.

Related topics:

`g:netrw_alto` `g:netrw_altv`
`netrw-C` `netrw-cr`
`netrw-ctrl-r`

`g:netrw_browsex_viewer`

specify user's preference for a viewer:
 `"kfmclient exec"`
 `"gnome-open"`

If

`"_"`

is used, then `netrwFileHandler()` will look for a script/function to handle the given extension. (see `netrw_filehandler`).

`g:netrw_browsex_support_remote`

specify if the specified viewer supports a remote URL. (see `netrw-handler`).

`g:netrw_chgperm`

Unix/Linux: `"chmod PERM FILENAME"`
Windows: `"cacls FILENAME /e /p PERM"`
Used to change access permission for a file.

`g:netrw_clipboard`

=1

By default, `netrw` will attempt to insure that the clipboard's values will remain unchanged. However, some users report that they have speed problems with this; consequently, this option, when set to zero, lets such users prevent `netrw` from saving and restoring the clipboard (the latter is done only as needed). That means that if the clipboard is changed (inadvertently) by normal `netrw` operation that it will not be restored to its prior state.

`g:netrw_compress`

=`"gzip"`
Will compress marked files with this command

`g:Netrw_corehandler`

Allows one to specify something additional to do when handling `<core>` files via `netrw`'s browser's "x" command (see `netrw-x`). If present, `g:Netrw_corehandler` specifies either one or more function references (see `Funcref`). (the capital `g:Netrw...` is required its holding a function reference)

`g:netrw_ctags` = "ctags"
The default external program used to create tags

`g:netrw_cursor` = 2 (default)
This option controls the use of the `'cursorline'` (cul) and `'cursorcolumn'` (cuc) settings by netrw:

Value	Thin-Long-Tree	Wide
=0	u-cul u-cuc	u-cul u-cuc
=1	u-cul u-cuc	cul u-cuc
=2	cul u-cuc	cul u-cuc
=3	cul u-cuc	cul cuc
=4	cul cuc	cul cuc
=5	U-cul U-cuc	U-cul U-cuc
=6	U-cul U-cuc	cul U-cuc
=7	cul U-cuc	cul U-cuc
=8	cul U-cuc	cul cuc

Where

u-cul : user's `'cursorline'` initial setting used
u-cuc : user's `'cursorcolumn'` initial setting used
U-cul : user's `'cursorline'` current setting used
U-cuc : user's `'cursorcolumn'` current setting used
cul : `'cursorline'` will be locally set
cuc : `'cursorcolumn'` will be locally set

The "initial setting" means the values of the `'cuc'` and `'cul'` settings in effect when netrw last saw `g:netrw_cursor` >= 5 or when netrw was initially run.

`g:netrw_decompress` = { ".gz" : "gunzip" ,
".bz2" : "bunzip2" ,
".zip" : "unzip" ,
".tar" : "tar -xf" }
A dictionary mapping suffices to decompression programs.

`g:netrw_dirhistmax` =10: controls maximum quantity of past history. May be zero to suppress history.
(related: `netrw-qb` `netrw-u` `netrw-U`)

`g:netrw_dynamic_maxfilenamen` =32: enables dynamic determination of `g:netrw_maxfilenamen` , which affects local file long listing.

`g:netrw_errorlvl` =0: error levels greater than or equal to this are permitted to be displayed
0: notes
1: warnings

2: errors

`g:netrw_fastbrowse`

=0: slow speed directory browsing;
never re-uses directory listings;
always obtains directory listings.
=1: medium speed directory browsing;
re-use directory listings only
when remote directory browsing.
(default value)
=2: fast directory browsing;
only obtains directory listings when the
directory hasn't been seen before
(or `netrw-ctrl-l` is used).

Fast browsing retains old directory listing buffers so that they don't need to be re-acquired. This feature is especially important for remote browsing. However, if a file is introduced or deleted into or from such directories, the old directory buffer becomes out-of-date. One may always refresh such a directory listing with `netrw-ctrl-l`. This option gives the user the choice of trading off accuracy (ie. up-to-date listing) versus speed.

`g:netrw_ffkeep`

(default: doesn't exist)
If this variable exists and is zero, then netrw will not do a save and restore for `'fileformat'`.

`g:netrw_fname_escape`

= ' ?&;%'
Used on filenames before remote reading/writing

`g:netrw_ftp_browse_reject`

ftp can produce a number of errors and warnings that can show up as "directories" and "files" in the listing. This pattern is used to remove such embedded messages. By default its value is:

```
'^total\s\+\d\+$\|  
^Trying\s\+\d\+.*$\|  
^KERBEROS_V\d rejected\|  
^Security extensions not\|  
No such file\|  
: connect to address [0-9a-fA-F:]*  
: No route to host$'
```

`g:netrw_ftp_list_cmd`

options for passing along to ftp for directory listing. Defaults:
unix or `g:netrw_cygwin` set: : "ls -lF"
otherwise "dir"

`g:netrw_ftp_sizelist_cmd`

options for passing along to ftp for directory

listing, sorted by size of file.
Defaults:
unix or g:netrw_cygwin set: : "ls -sLF"
otherwise "dir"

g:netrw_ftp_timelist_cmd options for passing along to ftp for directory listing, sorted by time of last modification.
Defaults:
unix or g:netrw_cygwin set: : "ls -tLF"
otherwise "dir"

g:netrw_glob_escape ='[]*?`{~\$' (unix)
='[]*?`{\$' (windows)
These characters in directory names are escaped before applying glob()

g:netrw_gx ="<cfiler>"
This option controls how gx (netrw-gx) picks up the text under the cursor. See expand() for possibilities.

g:netrw_hide Controlled by the "a" map (see netrw-a)
=0 : show all
=1 : show not-hidden files
=2 : show hidden files only
default: =1

g:netrw_home The home directory for where bookmarks and history are saved (as .netrwbook and .netrwhist).
Netrw uses expand() on the string.
default: the first directory on the 'runtimepath'

g:netrw_keeppdir =1 (default) keep current directory immune from the browsing directory.
=0 keep the current directory the same as the browsing directory.
The current browsing directory is contained in b:netrw_curdir (also see netrw-cd)

g:netrw_keepj ="keepj" (default) netrw attempts to keep the :jumps table unaffected.
="" netrw will not use :keepjumps with exceptions only for the saving/restoration of position.

g:netrw_list_cmd command for listing remote directories
default: (if ssh is executable)
"ssh HOSTNAME ls -FLa"

g:netrw_list_cmd_options If this variable exists, then its contents are appended to the g:netrw_list_cmd. For example, use "2>/dev/null" to get rid of banner

messages on unix systems.

`g:netrw_liststyle`

Set the default listing style:
= 0: thin listing (one file per line)
= 1: long listing (one file per line with time stamp information and file size)
= 2: wide listing (multiple files in columns)
= 3: tree style listing

`g:netrw_list_hide`

comma-separated pattern list for hiding files
Patterns are regular expressions (see [regexp](#))
There's some special support for git-ignore files: you may add the output from the helper function 'netrw_gitignore#Hide()' automatically hiding all gitignored files.
For more details see [netrw-gitignore](#) .

Examples:

```
let g:netrw_list_hide= '.*\swp$'  
let g:netrw_list_hide= netrw_gitignore#Hide() .. '.*\swp$'  
default: ""
```

`g:netrw_localcopycmd`

= "cp" Linux/Unix/MacOS/Cygwin
= expand("\$COMSPEC") Windows
Copies marked files ([netrw-mf](#)) to target directory ([netrw-mt](#) , [netrw-mc](#))

`g:netrw_localcopycmdopt`

= '' Linux/Unix/MacOS/Cygwin
= ' /c copy' Windows
Options for the [g:netrw_localcopycmd](#)

`g:netrw_localcopydircmd`

= "cp" Linux/Unix/MacOS/Cygwin
= expand("\$COMSPEC") Windows
Copies directories to target directory. ([netrw-mc](#) , [netrw-mt](#))

`g:netrw_localcopydircmdopt`

= " -R" Linux/Unix/MacOS/Cygwin
= " /c xcopy /e /c /h/ /i /k" Windows
Options for [g:netrw_localcopydircmd](#)

`g:netrw_localmkdir`

= "mkdir" Linux/Unix/MacOS/Cygwin
= expand("\$COMSPEC") Windows
command for making a local directory

`g:netrw_localmkdiropt`

= "" Linux/Unix/MacOS/Cygwin
= " /c mkdir" Windows
Options for [g:netrw_localmkdir](#)

`g:netrw_localmovecmd`

= "mv" Linux/Unix/MacOS/Cygwin
= expand("\$COMSPEC") Windows
Moves marked files ([netrw-mf](#)) to target directory ([netrw-mt](#) , [netrw-mm](#))

`g:netrw_localmovecmdopt` ="" Linux/Unix/MacOS/Cygwin
 ="/c move" Windows
 Options for `g:netrw_localmovecmd`

`g:netrw_maxfilenamen` =32 by default, selected so as to make long listings fit on 80 column displays. If your screen is wider, and you have file or directory names longer than 32 bytes, you may set this option to keep listings columnar.

`g:netrw_mkdir_cmd` command for making a remote directory via ssh (also see `g:netrw_remote_mkdir`)
 default: "ssh USEPORT HOSTNAME mkdir"

`g:netrw_mousemaps` =1 (default) enables mouse buttons while browsing to:
 leftmouse : open file/directory
 shift-leftmouse : mark file
 middlemouse : same as P
 rightmouse : remove file/directory
 =0: disables mouse maps

`g:netrw_nobeval` doesn't exist (default)
 If this variable exists, then balloon evaluation will be suppressed (see 'ballooneval')

`g:netrw_sizestyle` not defined: actual bytes (default)
 ="b" : actual bytes (default)
 ="h" : human-readable (ex. 5k, 4m, 3g)
 uses 1000 base
 ="H" : human-readable (ex. 5K, 4M, 3G)
 uses 1024 base
 The long listing (`netrw-i`) and query-file maps (`netrw-qf`) will display file size using the specified style.

`g:netrw_usetab` if this variable exists and is non-zero, then the <tab> map supporting shrinking/expanding a Lexplore or netrw window will be enabled. (see `netrw-c-tab`)

`g:netrw_remote_mkdir` command for making a remote directory via ftp (also see `g:netrw_mkdir_cmd`)
 default: "mkdir"

`g:netrw_retmap` if it exists and is set to one, then:
 * if in a netrw-selected file, AND
 * no normal-mode <2-leftmouse> mapping exists, then the <2-leftmouse> will be mapped for easy return to the netrw browser window.
 example: click once to select and open a file, double-click to return.

Note that one may instead choose to:
* let g:netrw_retmap= 1, AND
* nmap <silent> YourChoice <Plug>NetrwReturn
and have another mapping instead of
<2-leftmouse> to invoke the return.

You may also use the `:Rexplore` command to do the same thing.

```
default: =0
```

`g:netrw_rm_cmd`

command for removing remote files
default: "ssh USEPORT HOSTNAME rm"

`g:netrw_rmdir_cmd`

command for removing remote directories
default: "ssh USEPORT HOSTNAME rmdir"

`g:netrw_rmf_cmd`

command for removing remote softlinks
default: "ssh USEPORT HOSTNAME rm -f"

`g:netrw_servername`

use this variable to provide a name for
`netrw-ctrl-r` to use for its server.
default: "NETRWSERVER"

`g:netrw_sort_by`

sort by "name", "time", "size", or
"exten".
default: "name"

`g:netrw_sort_direction`

sorting direction: "normal" or "reverse"
default: "normal"

`g:netrw_sort_options`

sorting is done using `:sort`; this
variable's value is appended to the
sort command. Thus one may ignore case,
for example, with the following in your
.vimrc:

```
let g:netrw_sort_options="i"  
default: ""
```

`g:netrw_sort_sequence`

when sorting by name, first sort by the
comma-separated pattern sequence. **Note** that
any filigree added to indicate filetypes
should be accounted for in your pattern.

```
default: '[\/]$,*,\.bak$,\.o$,\.h$,  
\.info$,\.swp$,\.obj$'
```

`g:netrw_special_syntax`

If true, then certain files will be shown
using special syntax in the browser:

```
netrwBak      : *.bak  
netrwCompress: *.gz *.bz2 *.Z *.zip  
netrwCoreDump: core.\d\+  
netrwData     : *.dat
```

```

netrwDoc      : *.doc,*.txt,*.pdf,
               *.pdf,*.docx
netrwHdr      : *.h
netrwLex      : *.l *.lex
netrwLib      : *.a *.so *.lib *.dll
netrwMakefile: [mM]akefile *.mak
netrwObj      : *.o *.obj
netrwPix      : *.bmp,*.fit,*.fits,*.gif,
               *.jpg,*.jpeg,*.pcx,*.ppc
               *.pgm,*.png,*.psd,*.rgb
               *.tif,*.xbm,*.xcf
netrwTags     : tags ANmenu ANtags
netrwTilde    : *
netrwTmp      : tmp* *tmp
netrwYacc     : *.y

```

In addition, those groups mentioned in 'suffixes' are also added to the special file highlighting group.

These syntax highlighting groups are linked to netrwGray or Folded by default (see [hl-Folded](#)), but one may put lines like

```
hi link netrwCompress Visual
```

into one's `<.vimrc>` to use one's own preferences. Alternatively, one may put such specifications into

```
.vim/after/syntax/netrw.vim.
```

The netrwGray highlighting is set up by netrw when

```

* netrwGray has not been previously
  defined
* the gui is running

```

As an example, I myself use a dark-background colorscheme with the following in `.vim/after/syntax/netrw.vim`:

```

hi netrwCompress term=NONE cterm=NONE gui=NONE ctermfg=10 guifg=green ctermbg=0 guibg=black
hi netrwData      term=NONE cterm=NONE gui=NONE ctermfg=9 guifg=blue ctermbg=0 guibg=black
hi netrwHdr       term=NONE cterm=NONE,italic gui=NONE guifg=SeaGreen1
hi netrwLex       term=NONE cterm=NONE,italic gui=NONE guifg=SeaGreen1
hi netrwYacc      term=NONE cterm=NONE,italic gui=NONE guifg=SeaGreen1
hi netrwLib       term=NONE cterm=NONE gui=NONE ctermfg=14 guifg=yellow
hi netrwObj       term=NONE cterm=NONE gui=NONE ctermfg=12 guifg=red
hi netrwTilde     term=NONE cterm=NONE gui=NONE ctermfg=12 guifg=red
hi netrwTmp       term=NONE cterm=NONE gui=NONE ctermfg=12 guifg=red
hi netrwTags      term=NONE cterm=NONE gui=NONE ctermfg=12 guifg=red
hi netrwDoc       term=NONE cterm=NONE gui=NONE ctermfg=220 ctermbg=27 guifg=yellow2 guibg=black
hi netrwSymLink   term=NONE cterm=NONE gui=NONE ctermfg=220 ctermbg=27 guifg=grey60

```

[g:netrw_ssh_browse_reject](#)

ssh can sometimes produce unwanted lines, messages, banners, and whatnot that one doesn't want masquerading as "directories" and "files". Use this pattern to remove such embedded messages. By default its value is:

'^total\s\+\d\+\$'

`g:netrw_ssh_cmd`

One may specify an executable command to use instead of ssh for remote actions such as listing, file removal, etc.
default: ssh

`g:netrw_suppress_gx_mesg`

=1 : browsers sometimes produce messages which are normally unwanted intermixed with the page.
However, when using links, for example, those messages are what the browser produces. By setting this option to 0, netrw will not suppress browser messages.

`g:netrw_tmpfile_escape`

= ' & ; '
escape() is applied to all temporary files to escape these characters.

`g:netrw_timefmt`

specify format string to vim's strftime(). The default, "%c", is "the preferred date and time representation for the current locale" according to my manpage entry for strftime(); however, not all are satisfied with it. Some alternatives:
"%a %d %b %Y %T",
" %a %Y-%m-%d %I-%M-%S %p"
default: "%c"

`g:netrw_use_noswf`

netrw normally avoids writing swapfiles for browser buffers. However, under some systems this apparently is causing nasty ml_get errors to appear; if you're getting ml_get errors, try putting
let g:netrw_use_noswf= 0
in your .vimrc.
default: 1

`g:netrw_winsize`

specify initial size of new windows made with "o" (see [netrw-o](#)), "v" (see [netrw-v](#)), [:Hexplore](#) or [:Vexplore](#). The g:netrw_winsize is an integer describing the percentage of the current netrw buffer's window to be used for the new window.
If g:netrw_winsize is less than zero, then the absolute value of g:netrw_winsize will be used to specify the quantity of lines or columns for the new window.
If g:netrw_winsize is zero, then a normal split will be made (ie. ['equalalways'](#) will take effect, for example).
default: 50 (for 50%)

`g:netrw_wiw`

=1 specifies the minimum window width to use

when shrinking a netrw/Lexplore window
(see [netrw-c-tab](#)).

[g:netrw_xstrlen](#)

Controls how netrw computes string lengths, including multi-byte characters' string length. (thanks to N Weibull, T Mechelynck)
=0: uses Vim's built-in strlen()
=1: number of codepoints (Latin a + combining circumflex is two codepoints) (DEFAULT)
=2: number of spacing codepoints (Latin a + combining circumflex is one spacing codepoint; a hard tab is one; wide and narrow CJK are one each; etc.)
=3: virtual length (counting tabs as anything between 1 and 'tabstop' , wide CJK as 2 rather than 1, Arabic alif as zero when immediately preceded by lam, one otherwise, etc)

[g:NetrwTopLvlMenu](#)

This variable specifies the top level menu name; by default, it's "Netrw.". If you wish to change this, do so in your .vimrc.

NETRW BROWSING AND OPTION INCOMPATIBILITIES

[netrw-incompatible](#) {{{2

Netrw has been designed to handle user options by saving them, setting the options to something that's compatible with netrw's needs, and then restoring them. However, the autochdir option:

`:set acd`

is problematic. Autochdir sets the current directory to that containing the file you edit; this apparently also applies to directories. In other words, autochdir sets the current directory to that containing the "file" (even if that "file" is itself a directory).

NETRW SETTINGS WINDOW

[netrw-settings-window](#) {{{2

With the NetrwSettings.vim plugin,

`:NetrwSettings`

will bring up a window with the many variables that netrw uses for its settings. You may change any of their values; when you save the file, the settings therein will be used. One may also press "?" on any of the lines for help on what each of the variables do.

(also see: [netrw-browser-var](#) [netrw-protocol](#) [netrw-variables](#))

OBTAINING A FILE

[netrw-obtain](#) [netrw-0](#) {{{2

If there are no marked files:

When browsing a remote directory, one may obtain a file under the cursor (ie. get a copy on your local machine, but not edit it) by pressing the 0

key.

If there are marked files:

The marked files will be obtained (ie. a copy will be transferred to your local machine, but not set up for editing).

Only ftp and scp are supported for this operation (but since these two are available for browsing, that shouldn't be a problem). The status bar will then show, on its right hand side, a message like "Obtaining filename". The statusline will be restored after the transfer is complete.

Netrw can also "obtain" a file using the local browser. Netrw's display of a directory is not necessarily the same as Vim's "current directory", unless `g:netrw_keepdir` is set to 0 in the user's `<.vimrc>`. One may select a file using the local browser (by putting the cursor on it) and pressing "O" will then "obtain" the file; ie. copy it to Vim's current directory.

Related topics:

- * To see what the current directory is, use `:pwd`
- * To make the currently browsed directory the current directory, see `netrw-cd`
- * To automatically make the currently browsed directory the current directory, see `g:netrw_keepdir` .

OPEN A NEW FILE IN NETRW'S CURRENT DIRECTORY `netrw-newfile` `netrw-createfile`
`netrw-% {{{2`

To open a new file in netrw's current directory, press "%". This map will query the user for a new filename; an empty file by that name will be placed in the netrw's current directory (ie. `b:netrw_curdir`).

If Lexplore (`netrw-:Lexplore`) is in use, the new file will be generated in the `g:netrw_chgwin` window.

Related topics: `netrw-d`

PREVIEW WINDOW `netrw-p` `netrw-preview` {{{2

One may use a preview window by using the "p" key when the cursor is atop the desired filename to be previewed. The display will then split to show both the browser (where the cursor will remain) and the file (see `:pedit`). By default, the split will be taken horizontally; one may use vertical splitting if one has set `g:netrw_preview` first.

An interesting set of netrw settings is:

```
let g:netrw_preview = 1
let g:netrw_liststyle = 3
let g:netrw_winsize = 30
```

These will:

1. Make vertical splitting the default for previewing files
2. Make the default listing style "tree"
3. When a vertical preview window is opened, the directory listing will use only 30% of the columns available; the rest of the window is used for the preview window.

Related: if you like this idea, you may also find `:Lexplore` (`netrw-:Lexplore`) or `g:netrw_chgwin` of interest

Also see: `g:netrw_chgwin` `netrw-P` `'previewwindow'` `CTRL-W_z` `:pclose`

PREVIOUS WINDOW

`netrw-P` `netrw-prvwin` {{{2

To edit a file or directory under the cursor in the previously used (last accessed) window (see `:he CTRL-W_p`), press a "P". If there's only one window, then the one window will be horizontally split (by default).

If there's more than one window, the previous window will be re-used on the selected file/directory. If the previous window's associated buffer has been modified, and there's only one window with that buffer, then the user will be asked if s/he wishes to save the buffer first (yes, no, or cancel).

Related Actions `netrw-cr` `netrw-o` `netrw-t` `netrw-v`

Associated setting variables:

`g:netrw_alto` control above/below splitting
`g:netrw_altv` control right/left splitting
`g:netrw_preview` control horizontal vs vertical splitting
`g:netrw_winsize` control initial sizing

Also see: `g:netrw_chgwin` `netrw-p`

REFRESHING THE LISTING

`netrw-refresh` `netrw-ctrl-l` `netrw-ctrl_l` {{{2

To refresh either a local or remote directory listing, press `ctrl-l` (`<c-l>`) or hit the `<cr>` when atop the `./` directory entry in the listing. One may also refresh a local directory by using `":e ."`.

REVERSING SORTING ORDER

`netrw-r` `netrw-reverse` {{{2

One may toggle between normal and reverse sorting order by pressing the "r" key.

Related topics: `netrw-s`

Associated setting variable: `g:netrw_sort_direction`

RENAMING FILES OR DIRECTORIES

`netrw-move` `netrw-rename` `netrw-R` {{{2

If there are no marked files: (see `netrw-mf`)

Renaming files and directories involves moving the cursor to the file/directory to be moved (renamed) and pressing "R". You will then be queried for what you want the file/directory to be renamed to. You may select a range of lines with the "V" command (visual selection), and then press "R"; you will be queried for each file as to what you want it renamed to.

If there are marked files: (see [netrw-mf](#))

Marked files will be renamed (moved). You will be queried as above in order to specify where you want the file/directory to be moved.

If you answer a renaming query with a "s/frompattern/topattern/", then subsequent files on the marked file list will be renamed by taking each name, applying that substitute, and renaming each file to the result. As an example :

```
mr [query: reply with *.c]
R  [query: reply with s/^(.*)\.c$/\1.cpp/]
```

This example will mark all *.c files and then rename them to *.cpp files. Netrw will protect you from overwriting local files without confirmation, but not remote ones.

The ctrl-X character has special meaning for renaming files:

<c-x> : a single ctrl-x tells netrw to ignore the portion of the response lying between the last '/' and the ctrl-x.

<c-x><c-x> : a pair of contiguous ctrl-x's tells netrw to ignore any portion of the string preceding the double ctrl-x's.

WARNING:

Note that moving files is a dangerous operation; copies are safer. That's because a "move" for remote files is actually a copy + delete -- and if the copy fails and the delete succeeds you may lose the file. Use at your own risk.

The `g:netrw_rename_cmd` variable is used to implement remote renaming. By default its value is:

```
ssh HOSTNAME mv
```

One may rename a block of files and directories by selecting them with V ([linewise-visual](#)) when using thin style.

See [cmdline-editing](#) for more on how to edit the command line; in particular, you'll find <ctrl-f> (initiates cmdline window editing) and <ctrl-c> (uses the command line under the cursor) useful in conjunction with the R command.

SELECTING SORTING STYLE

`netrw-s` `netrw-sort` {{{2

One may select the sorting style by name, time, or (file) size. The "s" map allows one to circulate amongst the three choices; the directory listing will automatically be refreshed to reflect the selected style.

Related topics: `netrw-r` `netrw-S`
Associated setting variables: `g:netrw_sort_by` `g:netrw_sort_sequence`

SETTING EDITING WINDOW `netrw-editwindow` `netrw-C` `netrw-:NetrwC` {{{2

One may select a netrw window for editing with the "C" mapping, using the `:NetrwC [win#]` command, or by setting `g:netrw_chgwin` to the selected window number. Subsequent selection of a file to edit (`netrw-cr`) will use that window.

- * `C` : by itself, will select the current window holding a netrw buffer for subsequent editing via `netrw-cr` . The `C` mapping is only available while in netrw buffers.
- * `[count]C` : the count will be used as the window number to be used for subsequent editing via `netrw-cr` .
- * `:NetrwC` will set `g:netrw_chgwin` to the current window
- * `:NetrwC win#` will set `g:netrw_chgwin` to the specified window number

Using

```
let g:netrw_chgwin= -1
```

will restore the default editing behavior
(ie. subsequent editing will use the current window).

Related topics: `netrw-cr` `g:netrw_browse_split`
Associated setting variables: `g:netrw_chgwin`

SHRINKING OR EXPANDING A NETRW OR LEXPLORE WINDOW `netrw-c-tab` {{{2

The `<c-tab>` key will toggle a netrw or `:Lexplore` window's width, but only if `g:netrw_usetab` exists and is non-zero (and, of course, only if your terminal supports differentiating `<c-tab>` from a plain `<tab>`).

- * If the current window is a netrw window, toggle its width (between `g:netrw_wiw` and its original width)
- * Else if there is a `:Lexplore` window in the current tab, toggle its width
- * Else bring up a `:Lexplore` window

If `g:netrw_usetab` exists and is zero, or if there is a pre-existing mapping for `<c-tab>`, then the `<c-tab>` will not be mapped. One may map something other than a `<c-tab>`, too: (but you'll still need to have had `g:netrw_usetab` set).

```
nmap <unique> (whatever) <Plug>NetrwShrink
```

Related topics: [:Lexplore](#)
Associated setting variable: [g:netrw_usetab](#)

USER SPECIFIED MAPS

```
netrw-usermaps {{{1
```

One may make customized user maps. Specify a variable, `g:Netrw_UserMaps`, to hold a `List` of lists of keymap strings and function names:

```
[["keymap-sequence","ExampleUserMapFunc"],...]
```

When `netrw` is setting up maps for a `netrw` buffer, if `g:Netrw_UserMaps` exists, then the internal function `netrw#UserMaps(islocal)` is called. This function goes through all the entries in the `g:Netrw_UserMaps` list:

- * sets up maps:

```
nno <buffer> <silent> KEYMAP-SEQUENCE
:call s:UserMaps(islocal,"ExampleUserMapFunc")
```
- * refreshes if result from that function call is the string "refresh"
- * if the result string is not "", then that string will be executed (:exe result)
- * if the result is a `List`, then the above two actions on results will be taken for every string in the result `List`

The user function is passed one argument; it resembles

```
fun! ExampleUserMapFunc(islocal)
```

where `a:islocal` is 1 if its a local-directory system call or 0 when remote-directory system call.

```
Use netrw#Expose("varname")      netrw-call    netrw-expose  netrw-modify
                                to access netrw-internal (script-local)
                                variables.
Use netrw#Modify("varname",newvalue) to change netrw-internal variables.
Use netrw#Call("funcname"[,args])  to call a netrw-internal function with
                                specified arguments.
```

Example: Get a copy of `netrw`'s marked file list:

```
let netrwmarkfilelist= netrw#Expose("netrwmarkfilelist")
```

Example: Modify the value of `netrw`'s marked file list:

```
call netrw#Modify("netrwmarkfilelist",[])
```

Example: Clear `netrw`'s marked file list via a mapping on `gu`

```
" ExampleUserMap: {{{2
fun! ExampleUserMap(islocal)
  call netrw#Modify("netrwmarkfilelist",[])
```

```

call netrw#Modify('netrwmarkfilemtch_{bufnr("%")}' , "")
let retval= ["refresh"]
return retval
endfun
let g:Netrw_UserMaps= [{"gu","ExampleUserMap"}]

```

10. Problems and Fixes

[netrw-problems](#) {{{1

(This section is likely to grow as I get feedback)
(also see [netrw-debug](#))

- P1. I use windows 95, and my ftp dumps four blank lines at the end of every read. [netrw-p1](#) {{{2

See [netrw-fixup](#) , and put the following into your `<.vimrc>` file:

```
let g:netrw_win95ftp= 1
```

- P2. I use Windows, and my network browsing with ftp doesn't sort by time or size! -or- The remote system is a Windows server; why don't I get sorts by time or size? [netrw-p2](#) {{{2

Windows' ftp has a minimal support for ls (ie. it doesn't accept sorting options). It doesn't support the `-F` which gives an explanatory character (ABC/ for "ABC is a directory"). Netrw then uses "dir" to get both its thin and long listings. If you think your ftp does support a full-up ls, put the following into your `<.vimrc>`:

```
let g:netrw_ftp_list_cmd      = "ls -lF"
let g:netrw_ftp_timelist_cmd= "ls -tlF"
let g:netrw_ftp_sizelist_cmd= "ls -slF"

```

Alternatively, if you have cygwin on your Windows box, put into your `<.vimrc>`:

```
let g:netrw_cygwin= 1
```

This problem also occurs when the remote system is Windows. In this situation, the various `g:netrw_ftp_[time|size]list_cmds` are as shown above, but the remote system will not correctly modify its listing behavior.

- P3. I tried `rcp://user@host/` (or protocol other than ftp) and netrw used ssh! That wasn't what I asked for... [netrw-p3](#) {{{2

Netrw has two methods for browsing remote directories: ssh and ftp. Unless you specify ftp specifically, ssh is used. When it comes time to do download a file (not just a directory

listing), netrw will use the given protocol to do so.

- P4. I would like long listings to be the default. netrw-p4 {{{2

Put the following statement into your `.vimrc` :

```
let g:netrw_liststyle= 1
```

Check out `netrw-browser-var` for more customizations that you can set.

- P5. My times come up oddly in local browsing netrw-p5 {{{2

Does your system's `strftime()` accept the "%c" to yield dates such as "Sun Apr 27 11:49:23 1997"? If not, do a "man strftime" and find out what option should be used. Then put it into your `.vimrc` :

```
let g:netrw_timefmt= "%X" (where X is the option)
```

- P6. I want my current directory to track my browsing. netrw-p6 {{{2
How do I do that?

Put the following line in your `.vimrc` :

```
let g:netrw_keepdir= 0
```

- P7. I use Chinese (or other non-ascii) characters in my filenames, netrw-p7 {{{2
and netrw (Explore, Sexplore, Hexplore, etc) doesn't display them!

(taken from an answer provided by Wu Yongwei on the vim mailing list)

I now see the problem. Your code page is not 936, right? Vim seems only able to open files with names that are valid in the current code page, as are many other applications that do not use the Unicode version of Windows APIs. This is an OS-related issue. You should not have such problems when the system locale uses UTF-8, such as modern Linux distros.

(...it is one more reason to recommend that people use utf-8!)

- P8. I'm getting "ssh is not executable on your system" -- what do I netrw-p8 {{{2
do?

(Dudley Fox) Most people I know use putty for windows ssh. It is a free ssh/telnet application. You can read more about it here:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/> Also:

(Marlin Unruh) This program also works for me. It's a single executable, so he/she can copy it into the Windows\System32 folder and create a shortcut to it.

(Dudley Fox) You might also wish to consider plink, as it sounds most similar to what you are looking for. plink is an application in the putty suite.

<http://the.earth.li/~sgtatham/putty/0.58/html/doc/Chapter7.html#plink>

(Vissale Neang) Maybe you can try OpenSSH for windows, which can be obtained from:

<http://sshwndows.sourceforge.net/>

It doesn't need the full Cygwin package.

(Antoine Mechelynck) For individual Unix-like programs needed for work in a native-Windows environment, I recommend getting them from the GnuWin32 project on sourceforge if it has them:

<http://gnuwin32.sourceforge.net/>

Unlike Cygwin, which sets up a Unix-like virtual machine on top of Windows, GnuWin32 is a rewrite of Unix utilities with Windows system calls, and its programs works quite well in the cmd.exe "Dos box".

(dave) Download WinSCP and use that to connect to the server. In Preferences > Editors, set gvim as your editor:

- Click "Add..."
- Set External Editor (adjust path as needed, include the quotes and !.! at the end):
"c:\Program Files\Vim\vim82\gvim.exe" !.
- Check that the filetype in the box below is {asterisk}.{asterisk} (all files), or whatever types you want (cec: change {asterisk} to * ; I had to write it that way because otherwise the helptags system thinks it's a tag)
- Make sure it's at the top of the listbox (click it, then click "Up" if it's not)

If using the Norton Commander style, you just have to hit <F4> to edit a file in a local copy of gvim.

(Vit Gottwald) How to generate public/private key and save public key it on server:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/0.60/html/doc/Chapter8.html#pubkey-getting>
(8.3 Getting ready for public key authentication)

How to use a private key with 'pscp':

<http://www.chiark.greenend.org.uk/~sgtatham/putty/0.60/html/doc/Chapter5.html>

(5.2.4 Using public key authentication with PSCP)

(Ben Schmidt) I find the ssh included with cwRsync is brilliant, and install cwRsync or cwRsyncServer on most Windows systems I come across these days. I guess COPSSH, packed by the same person, is probably even better for use as just ssh on Windows, and probably includes sftp, etc. which I suspect the cwRsync doesn't, though it might

(cec) To make proper use of these suggestions above, you will need to modify the following user-settable variables in your .vimrc:

```
g:netrw_ssh_cmd      g:netrw_list_cmd    g:netrw_mkdir_cmd
g:netrw_rm_cmd       g:netrw_rmdir_cmd   g:netrw_rmf_cmd
```

The first one (`g:netrw_ssh_cmd`) is the most important; most of the others will use the string in `g:netrw_ssh_cmd` by default.

- `netrw-p9` `netrw-ml_get`
P9. I'm browsing, changing directory, and bang! `ml_get` errors appear and I have to kill vim. Any way around this? {{{2

Normally `netrw` attempts to avoid writing swapfiles for its temporary directory buffers. However, on some systems this attempt appears to be causing `ml_get` errors to appear. Please try setting `g:netrw_use_noswf` to 0 in your `<.vimrc>`:

```
let g:netrw_use_noswf= 0
```

- `netrw-p10`
P10. I'm being pestered with "[something] is a directory" and "Press ENTER or type command to continue" prompts... {{{2

The "[something] is a directory" prompt is issued by Vim, not by `netrw`, and there appears to be no way to work around it. Coupled with the default `cmdheight` of 1, this message causes the "Press ENTER..." prompt. So: read `hit-enter`; I also suggest that you set your `'cmdheight'` to 2 (or more) in your `<.vimrc>` file.

- `netrw-p11`
P11. I want to have two windows; a thin one on the left and my editing window on the right. How may I accomplish this? {{{2

You probably want `netrw` running as in a side window. If so, you will likely find that `:[N]Lexplore` does what you want. The optional "[N]" allows you to select the quantity of columns you wish the `:Lexplore` window to start with (see `g:netrw_winsize` for how this parameter works).

Previous solution:

- * Put the following line in your `<.vimrc>`:
`let g:netrw_altv = 1`
- * Edit the current directory: `:e .`
- * Select some file, press `v`
- * Resize the windows as you wish (see `CTRL-W_<` and `CTRL-W_>`). If you're using `gvim`, you can drag the separating bar with your mouse.
- * When you want a new file, use `ctrl-w h` to go back to the `netrw` browser, select a file, then press `P` (see `CTRL-W_h` and `netrw-P`). If you're using `gvim`, you can press `<leftmouse>` in the browser window and then press the `<middlemouse>` to select the file.

netrw-p12

- P12. My directory isn't sorting correctly, or unwanted letters are appearing in the listed filenames, or things aren't lining up properly in the wide listing, ... {{{2

This may be due to an encoding problem. I myself usually use `utf-8`, but really only use `ascii` (ie. bytes from 32-126). Multibyte encodings use two (or more) bytes per character. You may need to change `g:netrw_sepchr` and/or `g:netrw_xstrlen`.

netrw-p13

- P13. I'm a Windows + putty + ssh user, and when I attempt to browse, the directories are missing trailing `"/`'s so `netrw` treats them as file transfers instead of as attempts to browse subdirectories. How may I fix this? {{{2

(mikeyao) If you want to use `vim` via `ssh` and `putty` under Windows, try combining the use of `pscp/psftp` with `plink`. `pscp/psftp` will be used to connect and `plink` will be used to execute commands on the server, for example: list files and directory using `'ls'`.

These are the settings I use to do this:

```
" list files, it's the key setting, if you haven't set,
" you will get a blank buffer
let g:netrw_list_cmd = "plink HOSTNAME ls -Fa"
" if you haven't add putty directory in system path, you should
" specify scp/sftp command. For examples:
"let g:netrw_sftp_cmd = "d:\\dev\\putty\\PSFTP.exe"
"let g:netrw_scp_cmd = "d:\\dev\\putty\\PSCP.exe"
```

netrw-p14

- P14. I would like to speed up writes using `Nwrite` and `scp/ssh` style connections. How? (Thomer M. Gil) {{{2

Try using `ssh`'s `ControlMaster` and `ControlPath` (see the `ssh_config` man page) to share multiple `ssh` connections over a single network connection. That cuts out the cryptographic handshake on each file write, sometimes speeding it up by an order of magnitude. (see http://thomer.com/howtos/netrw_ssh.html)

(included by permission)

Add the following to your ~/.ssh/config:

```
# you change "*" to the hostname you care about
Host *
  ControlMaster auto
  ControlPath /tmp/%r@%h:%p
```

Then create an ssh connection to the host and leave it running:

```
ssh -N host.domain.com
```

Now remotely open a file with Vim's Netrw and enjoy the zippiness:

```
vim scp://host.domain.com//home/user/.bashrc
```

- P15. How may I use a double-click instead of netrw's usual single click to open a file or directory? (Ben Fritz) netrw-p15 {{{2

First, disable netrw's mapping with

```
let g:netrw_mousemaps= 0
```

and then create a netrw buffer only mapping in

```
$HOME/.vim/after/ftplugin/netrw.vim:
```

```
nmap <buffer> <2-leftmouse> <CR>
```

Note that setting g:netrw_mousemaps to zero will turn off all netrw's mouse mappings, not just the <leftmouse> one. (see g:netrw_mousemaps)

- P16. When editing remote files (ex. :e ftp://hostname/path/file), under Windows I get an E303 message complaining that its unable to open a swap file. netrw-p16 {{{2

(romainl) It looks like you are starting Vim from a protected directory. Start netrw from your \$HOME or other writable directory.

- P17. Netrw is closing buffers on its own. netrw-p17 {{{2

What steps will reproduce the problem?

1. :Explore, navigate directories, open a file
2. :Explore, open another file
3. Buffer opened in step 1 will be closed. o

What is the expected output? What do you see instead?

I expect both buffers to exist, but only the last one does.

(Lance) Problem is caused by "set autochdir" in .vimrc.

(drchip) I am able to duplicate this problem with 'acd' set.

It appears that the buffers are not exactly closed;

a ":ls!" will show them (although ":ls" does not).

netrw-P18

P18. How to locally edit a file that's only available via another server accessible via ssh? {{{2
See <http://stackoverflow.com/questions/12469645/Using Vim to Remotely Edit A File on ServerB Only Accessible From ServerA>

netrw-P19

P19. How do I get numbering on in directory listings? {{{2
With `g:netrw_bufsettings`, you can control netrw's buffer settings; try putting
`let g:netrw_bufsettings="noma nomod nu nobl nowrap ro nornu"`
in your `.vimrc`. If you'd like to have relative numbering instead, try
`let g:netrw_bufsettings="noma nomod nonu nobl nowrap ro rnu"`

netrw-P20

P20. How may I have gvim start up showing a directory listing? {{{2
Try putting the following code snippet into your `.vimrc`:
`augroup VimStartup`
`au!`
`au VimEnter * if expand("%") == "" && argc() == 0 && \ (v:servername =~ 'GVIM\d*' || v:servername == "") \ | e . | endif`
`augroup END`
You may use `Lexplore` instead of `"e"` if you're so inclined. This snippet assumes that you have client-server enabled (ie. a "huge" vim version).

netrw-P21

P21. I've made a directory (or file) with an accented character, but netrw isn't letting me enter that directory/read that file: {{{2

Its likely that the shell or o/s is using a different encoding than you have vim (netrw) using. A patch to vim supporting "systemencoding" may address this issue in the future; for now, just have netrw use the proper encoding. For example:

```
au FileType netrw set enc=latin1
```

netrw-P22

P22. I get an error message when I try to copy or move a file: {{{2

`**error** (netrw) tried using g:netrw_localcopycmd<cp>; it doesn't work!`

What's wrong?

Netrw uses several system level commands to do things (see

```
g:netrw_localcopycmd, g:netrw_localmovecmd,  
g:netrw_mkdir_cmd ).
```

You may need to adjust the default commands for one or more of these commands by setting them properly in your `.vimrc`. Another

source of difficulty is that these commands use vim's local directory, which may not be the same as the browsing directory shown by netrw (see [g:netrw_keepdir](#)).

11. Debugging Netrw Itself netrw-debug {{{1

Step 1: check that the problem you've encountered hasn't already been resolved by obtaining a copy of the latest (often developmental) netrw at:

<http://www.drchip.org/astronaut/vim/index.html#NETRW>

The `<netrw.vim>` script is typically installed on systems as something like:

```
/usr/local/share/vim/vim8x/plugin/netrwPlugin.vim
/usr/local/share/vim/vim8x/autoload/netrw.vim
    (see output of :echo &rtp)
```

which is loaded automatically at startup (assuming `:set nocp`). If you installed a new netrw, then it will be located at

```
$HOME/.vim/plugin/netrwPlugin.vim
$HOME/.vim/autoload/netrw.vim
```

Step 2: assuming that you've installed the latest version of netrw, check that your problem is really due to netrw. Create a file called `netrw.vimrc` with the following contents:

```
set nocp
so $HOME/.vim/plugin/netrwPlugin.vim
```

Then run netrw as follows:

```
vim -u netrw.vimrc --noplugins -i NONE [some path here]
```

Perform whatever netrw commands you need to, and check that the problem is still present. This procedure sidesteps any issues due to personal `.vimrc` settings, `.viminfo` file, and other plugins. If the problem does not appear, then you need to determine which setting in your `.vimrc` is causing the conflict with netrw or which plugin(s) is/are involved.

Step 3: If the problem still is present, then get a debugging trace from netrw:

1. Get the `<Decho.vim>` script, available as:

<http://www.drchip.org/astronaut/vim/index.html#DECHO>
or
http://vim.sourceforge.net/scripts/script.php?script_id=120

Decho.vim is provided as a "vimball"; see [vimball-intro](#) . You should edit the `Decho.vba.gz` file and source it in:

```
vim Decho.vba.gz
:so %
:q
```

2. To turn on debug tracing in netrw, then edit the `<netrw.vim>` file by typing:

```
vim netrw.vim
:DechoOn
:wq
```

To restore to normal non-debugging behavior, re-edit `<netrw.vim>` and type

```
vim netrw.vim
:DechoOff
:wq
```

This command, provided by `<Decho.vim>`, will comment out all Decho-debugging statements (`Dfunc()`, `Dret()`, `Decho()`, `Dredir()`).

3. Then bring up vim and attempt to evoke the problem by doing a transfer or doing some browsing. A set of messages should appear concerning the steps that `<netrw.vim>` took in attempting to read/write your file over the network in a separate tab or server vim window.

Change the `netrw.vimrc` file to include the Decho plugin:

```
set nocp
so $HOME/.vim/plugin/Decho.vim
so $HOME/.vim/plugin/netrwPlugin.vim
```

You should continue to run vim with

```
vim -u netrw.vimrc --noplugins -i NONE [some path here]
```

to avoid entanglements with options and other plugins.

To save the file: under linux, the output will be in a separate remote server window; in it, just save the file with

```
:w! DBG
```

Under a vim that doesn't support `clientserver`, your debugging output will appear in another tab:

```
:tabnext
:set bt=
:w! DBG
```

Furthermore, it'd be helpful if you would type

```
:Dsep <command>
```

where `<command>` is the command you're about to type next, thereby making it easier to associate which part of the debugging trace is due to which command.

Please send that information to `<netrw.vim>`'s maintainer along with the o/s you're using and the vim version that you're using (see `:version`) (remove the embedded NOSPAM first)

Ncamp0bell@SdrPchip.AorgM-NOSPAM

12. History

`netrw-history` {{{1

- v172: Sep 02, 2021 * (Bram Moolenaar) Changed "l:go" to "go"
- * (Bram Moolenaar) no need for "b" in netrw-safe guioptions
- Nov 15, 2021 * removed netrw_localrm and netrw_localrmdir references
- Aug 18, 2022 * (Miguel Barro) improving compatability with powershell
- v171: Oct 09, 2020 * included code in s:NetrwOptionsSafe() to allow 'bh' to be set to delete when rather than hide when g:netrw_fastbrowse was zero.
- * Installed g:netrw_clipboard setting
- * Installed option bypass for 'guioptions' a/A settings
- * Changed popup_beval() to popup_atcursor() in netrw#ErrorMsg (lacygoill). Apparently popup_beval doesn't reliably close the popup when the mouse is moved.
- * VimEnter() now using win_execute to examine buffers for an attempt to open a directory. Avoids issues with popups/terminal from command line. (lacygoill)
- Jun 28, 2021 * (zeertzjq) provided a patch for use of xmap,xno instead of vmap,vno in netrwPlugin.vim. Avoids entanglement with select mode.
- Jul 14, 2021 * Fixed problem addressed by tst976; opening a file using tree mode, going up a directory, and opening a file there was opening the file in the wrong directory.
- Jul 28, 2021 * (Ingo Karkat) provided a patch fixing an E488 error with netrwPlugin.vim (occurred for vim versions < 8.02)
- v170: Mar 11, 2020 * (reported by Reiner Herrmann) netrw+tree would not hide with the ^\..* pattern correctly.
- * (Marcin Szamotulski) NetrwOptionRestore did not restore options correctly that had a single quote in the option string.
- Apr 13, 2020 * implemented error handling via popup

Apr 30, 2020 windows (see [popup_beval\(\)](#))
 * (reported by Manatsu Takahashi) while using Lexplore, a modified file could be overwritten. Sol'n: will not overwrite, but will emit an [E37](#) (although one cannot add an ! to override)

Jun 07, 2020 * (reported by Jo Totland) repeatedly invoking :Lexplore and quitting it left unused hidden buffers. Netrw will now set netrw buffers created by :Lexplore to 'bh' =wipe.

v169: Dec 20, 2019 * (reported by amkarthik) that netrw's x ([netrw-x](#)) would throw an error when attempting to open a local directory.

v168: Dec 12, 2019 * scp timeout error message not reported, hopefully now fixed (Shane Xb Qian)

v167: Nov 29, 2019 * netrw does a save&restore on @* and @+. That causes problems with the clipboard. Now restores occurs only if @* or @+ have been changed.
 * netrw will change @* or @+ less often. Never if I happen to have caught all the operations that modify the unnamed register (which also writes @*).
 * Modified hiding behavior so that "s" will not ignore hiding.

v166: Nov 06, 2019 * Removed a space from a nmap for "-"
 * Numerous debugging statement changes

v163: Dec 05, 2017 * (Cristi Balan) reported that a setting ('sel') was left changed
 * (Holger Mitschke) reported a problem with saving and restoring history. Fixed.
 * Hopefully I fixed a nasty bug that caused a file rename to wipe out a buffer that it should not have wiped out.
 * (Holger Mitschke) amended this help file with additional [g:netrw_special_syntax](#) items
 * Prioritized wget over curl for [g:netrw_http_cmd](#)

v162: Sep 19, 2016 * (haya14busa) pointed out two syntax errors with a patch; these are now fixed.

Oct 26, 2016 * I started using mate-terminal and found that x and gx ([netrw-x](#) and [netrw-gx](#)) were no longer working. Fixed (using atril when \$DESKTOP_SESSION is "mate").

Nov 04, 2016 * (Martin Vuille) pointed out that @+ was being restored with keepregstar rather than keepregplus.

Nov 09, 2016 * Broke apart the command from the options, mostly for Windows. Introduced new netrw settings: [g:netrw_localcopycmdopt](#)
[g:netrw_localcopydircmdopt](#)
[g:netrw_localmkdiropt](#)
[g:netrw_localmovecmdopt](#)

- Nov 21, 2016 * (mattn) provided a patch for preview; swapped winwidth() with winheight()
- Nov 22, 2016 * (glacambre) reported that files containing spaces weren't being obtained properly via scp. Fix: apparently using single quotes such as with 'file name' wasn't enough; the spaces inside the quotes also had to be escaped (ie. 'file\ name').
 - * Also fixed obtain ([netrw-0](#)) to be able to obtain files with spaces in their names
- Dec 20, 2016 * (xc1427) Reported that using "I" ([netrw-I](#)) when atop "Hiding" in the banner also caused the active-banner hiding control to occur
- Jan 03, 2017 * (Enno Nagel) reported that attempting to apply netrw to a directory that was without read permission caused a syntax error.
- Jan 13, 2017 * (Ingo Karkat) provided a patch which makes using netrw#Call() better. Now returns value of internal routines return, for example.
- Jan 13, 2017 * (Ingo Karkat) changed netrw#FileUrlRead to use `:edit` instead of `:read` . I also changed the routine name to netrw#FileUrlEdit.
- Jan 16, 2017 * (Sayem) reported a problem where `:Lexplore` could generate a new listing buffer and window instead of toggling the netrw display. Unfortunately, the directions for eliciting the problem weren't complete, so I may or may not have fixed that issue.
- Feb 06, 2017 * Implemented cb and cB. Changed "c" to "cd". (see [netrw-cb](#) , [netrw-cB](#) , and [netrw-cd](#))
- Mar 21, 2017 * previously, netrw would specify (safe) settings even when the setting was already safe for netrw. Netrw now attempts to leave such already-netrw-safe settings alone. (affects s:NetrwOptionRestore() and s:NetrwSafeOptions(); also introduced s:NetrwRestoreSetting())
- Jun 26, 2017 * (Christian Brabandt) provided a patch to allow curl to follow redirects (ie. -L option)
- Jun 26, 2017 * (Callum Howard) reported a problem with `:Lexplore` not removing the Lexplore window after a change-directory
- Aug 30, 2017 * (Ingo Karkat) one cannot switch to the previously edited file (e.g. with **CTRL-^**) after editing a file:// URL. Patch to have a "keepalt" included.
- Oct 17, 2017 * (Adam Faryna) reported that gn ([netrw-gn](#)) did not work on directories in the current tree
- v157: Apr 20, 2016 * (Nicola) had set up a "nmap [<expr>](#) ..." with a function that returned a 0 while silently invoking a shell command. The shell command activated a ShellCmdPost event which in turn

called `s:LocalBrowseRefresh()`. That looks over all netrw buffers for changes needing refreshes. However, inside a `:map-<expr>`, tab and window changes are disallowed. Fixed. (affects netrw's `s:LocalBrowseRefresh()`)

- * `g:netrw_localrmdir` not used any more, but the relevant patch that causes `delete()` to take over was #1107 (not #1109).
- * `expand()` is now used on `g:netrw_home`; consequently, `g:netrw_home` may now use environment variables
- * `s:NetrwLeftmouse` and `s:NetrwCLeftmouse` will return without doing anything if invoked when inside a non-netrw window

Jun 15, 2016 * `gx` now calls `netrw#GX()` which returns the word under the cursor. The new wrinkle: if one is in a netrw buffer, then netrw's `s:NetrwGetWord()`.

Jun 22, 2016 * Netrw was executing all its associated Filetype commands silently; I'm going to try doing that "noisily" and see if folks have a problem with that.

Aug 12, 2016 * Changed order of tool selection for handling `http://... viewing`. (Nikolay Aleksandrovich Pavlov)

Aug 21, 2016 * Included hiding/showing/all for tree listings

- * Fixed refresh (^L) for tree listings

v156: Feb 18, 2016 * Changed `=~` to `=~#` where appropriate

Feb 23, 2016 * `s:ComposePath(base,subdir)` now uses `fnameescape()` on the base portion

Mar 01, 2016 * (`gt_macki`) reported where `:Explore` would make file unlisted. Fixed (tst943)

Apr 04, 2016 * (reported by John Little) netrw normally suppresses browser messages, but sometimes those "messages" are what is wanted. See `g:netrw_suppress_gx_mesg`

Apr 06, 2016 * (reported by Carlos Pita) deleting a remote file was giving an error message. Fixed.

Apr 08, 2016 * (Charles Cooper) had a problem with an undefined `b:netrw_curdir`. He also provided a fix.

Apr 20, 2016 * Changed `s:NetrwGetBuffer()`; now uses dictionaries. Also fixed the "No Name" buffer problem.

v155: Oct 29, 2015 * (Timur Fayzrakhmanov) reported that netrw's mapping of `ctrl-l` was not allowing refresh of other windows when it was done in a netrw window.

Nov 05, 2015 * Improved `s:TreeSqueezeDir()` to use `search()` instead of a loop

- * `NetrwBrowse()` will return line to `w:netrw_bannercnt` if cursor ended up in banner

Nov 16, 2015 * Added a <Plug>NetrwTreeSqueeze (`netrw-s-cr`)

Nov 17, 2015 * Commented out `imaps` -- perhaps someone can tell me how they're useful and should be retained?

Nov 20, 2015 * Added `netrw-ma` and `netrw-mA` support

Nov 20, 2015 * `gx` (`netrw-gx`) on a URL downloaded the file in addition to simply bringing up the URL in a browser. Fixed.

Nov 23, 2015 * Added `g:netrw_sizestyle` support

Nov 27, 2015 * Inserted a lot of <code><c-u>s into various netrw maps.

Jan 05, 2016 * `netrw-qL` implemented to mark files based upon `location-list` s; similar to `netrw-qF` .

Jan 19, 2016 * using `-` call `delete(directoryname,"d")` - instead of using `g:netrw_localrmdir` if `v7.4 + patch#1107` is available

Jan 28, 2016 * changed to using `winsaveview()` and `winrestview()`

Jan 28, 2016 * `s:NetrwTreePath()` now does a save and restore of view

Feb 08, 2016 * Fixed a tree-listing problem with remote directories

v154: Feb 26, 2015 * (Yuri Kanivetsky) reported a situation where a file was not treated properly as a file due to `g:netrw_keepdir == 1`

Mar 25, 2015 * (requested by Ben Friz) one may now sort by extension

Mar 28, 2015 * (requested by Matt Brooks) netrw has a lot of buffer-local mappings; however, some plugins (such as `vim-surround`) set up conflicting mappings that cause vim to wait. The "`<nowait>`" modifier has been included with most of netrw's mappings to avoid that delay.

Jun 26, 2015 * `netrw-gn` mapping implemented

* `:Ntree NotADir` resulted in having the tree listing expand in the error messages window. Fixed.

Jun 29, 2015 * Attempting to delete a file remotely caused an error with "keepsol" mentioned; fixed.

Jul 08, 2015 * Several changes to keep the `:jumps` table correct when working with `g:netrw_fastbrowse` set to 2

* wide listing with accented characters fixed (using `%-S` instead of `%-s` with a `printf()`)

Jul 13, 2015 * (Daniel Hahler) `CheckIfKde()` could be true but `kfmclient` not installed. Changed order in `netrw#BrowseX()`: checks if `kde` and `kfmclient`, then will use `xdg-open` on a unix system (if `xdg-open` is executable)

Aug 11, 2015 * (McDonnell) tree listing mode wouldn't select a file in an open subdirectory.

* (McDonnell) when multiple subdirectories were concurrently open in tree listing

mode, a ctrl-L wouldn't refresh properly.

- * The netrw:target menu showed duplicate entries

Oct 13, 2015 * (mattn) provided an exception to handle windows with shellslash set but no shell

Oct 23, 2015 * if g:netrw_usetab and <c-tab> now used to control whether NetrwShrink is used (see netrw-c-tab)

v153: May 13, 2014 * added another g:netrw_ffkeep usage {{{2

May 14, 2014 * changed s:PerformListing() so that it always sets ft=netrw for netrw buffers (ie. even when syntax highlighting is off, not available, etc)

May 16, 2014 * introduced the netrw-ctrl-r functionality

May 17, 2014 * introduced the netrw-:NetrwMB functionality

- * mb and mB (netrw-mb , netrw-mB) will add/remove marked files from bookmark list

May 20, 2014 * (Enno Nagel) reported that :Lex <dirname> wasn't working. Fixed.

May 26, 2014 * restored test to prevent leftmouse window resizing from causing refresh. (see s:NetrwLeftmouse())

- * fixed problem where a refresh caused cursor to go just under the banner instead of staying put

May 28, 2014 * (László Bimba) provided a patch for opening the :Lexplore window 100% high, optionally on the right, and will work with remote files.

May 29, 2014 * implemented :NetrwC (see netrw-:NetrwC)

Jun 01, 2014 * Removed some "silent"s from commands used to implemented scp://... and pscp://... directory listing. Permits request for password to appear.

Jun 05, 2014 * (Enno Nagel) reported that user maps "/" caused problems with "b" and "w", which are mapped (for wide listings only) to skip over files rather than just words.

Jun 10, 2014 * g:netrw_gx introduced to allow users to override default "<cfile>" with the gx (netrw-gx) map

Jun 11, 2014 * gx (netrw-gx), with 'autowrite' set, will write modified files. s:NetrwBrowseX() will now save, turn off, and restore the 'autowrite' setting.

Jun 13, 2014 * added visual map for gx use

Jun 15, 2014 * (Enno Nagel) reported that with having hls set and wide listing style in use, that the b and w maps caused unwanted highlighting.

Jul 05, 2014 * netrw-mv and netrw-mX commands included

Jul 09, 2014 * g:netrw_keepj included, allowing optional keepj

Jul 09, 2014 * fixing bugs due to previous update

Jul 21, 2014 * (Bruno Sutic) provided an updated

netrw_gitignore.vim

Jul 30, 2014 * (Yavuz Yetim) reported that editing two remote files of the same name caused the second instance to have a "temporary" name. Fixed: now they use the same buffer.

Sep 18, 2014 * (Yasuhiro Matsumoto) provided a patch which allows scp and windows local paths to work.

Oct 07, 2014 * gx (see [netrw-gx](#)) when atop a directory, will now do [gf](#) instead

Nov 06, 2014 * For cygwin: cygstart will be available for netrw#BrowseX() to use if its executable.

Nov 07, 2014 * Began support for file://... urls. Will use [g:netrw_file_cmd](#) (typically elinks or links)

Dec 02, 2014 * began work on having mc ([netrw-mc](#)) copy directories. Works for linux machines, cygwin+vim, but not for windows+gvim.

Dec 02, 2014 * in tree mode, netrw was not opening directories via symbolic links.

Dec 02, 2014 * added resolved link information to thin and tree modes

Dec 30, 2014 * (issue#231) [:ls](#) was not showing remote-file buffers reliably. Fixed.

v152: Apr 08, 2014 * uses the ['noswapfile'](#) option (requires {{{2 vim 7.4 with patch 213)

* (Enno Nagel) turn ['rnu'](#) off in netrw buffers.

* (Quinn Strahl) suggested that netrw allow regular window splitting to occur, thereby allowing ['equalalways'](#) to take effect.

* (qingtian zhao) normally, netrw will save and restore the ['fileformat'](#); however, sometimes that isn't wanted

Apr 14, 2014 * whenever netrw marks a buffer as ro, it will also mark it as nomod.

Apr 16, 2014 * sftp protocol now supported by netrw#Obtain(); this means that one may use "mc" to copy a remote file to a local file using sftp, and that the [netrw-O](#) command can obtain remote files via sftp.

* added [\[count\]C](#) support (see [netrw-C](#))

Apr 18, 2014 * when [g:netrw_chgwin](#) is one more than the last window, then vertically split the last window and use it as the chgwin window.

May 09, 2014 * SavePosn was "saving filename under cursor" from a non-netrw window when using [:Rex](#).

v151: Jan 22, 2014 * extended [:Rexplore](#) to return to buffer {{{2 prior to Explore or editing a directory

* (Ken Takata) netrw gave error when clipboard was disabled. Sol'n: Placed several [if has\("clipboard"\)](#) tests in.

* Fixed [ftp://X@Y@Z//](#) problem; X@Y now

part of user id, and only Z is part of hostname.

- * (A Loumiotis) reported that completion using a directory name containing spaces did not work. Fixed with a retry in netrw#Explore() which removes the backslashes vim inserted.

Feb 26, 2014

- * :Rexplore now records the current file using w:netrw_rexfile when returning via :Rexplore

Mar 08, 2014

- * (David Kotchan) provided some patches allowing netrw to work properly with windows shares.
- * Multiple one-liner help messages available by pressing <cr> while atop the "Quick Help" line
- * worked on ShellCmdPost, FocusGained event handling.
- * :Lexplore path: will be used to update a left-side netrw browsing directory.

Mar 12, 2014

- * netrw-s-cr : use <s-cr> to close tree directory implemented

Mar 13, 2014

- * (Tony Mechylynck) reported that using the browser with ftp on a directory, and selecting a gzipped txt file, that an E19 occurred (which was issued by gzip.vim). Fixed.

Mar 14, 2014

- * Implemented :MF and :MT (see netrw-:MF and netrw-:MT , respectively)

Mar 17, 2014

- * :Ntree [dir] wasn't working properly; fixed

Mar 18, 2014

- * Changed all uses of set to setl

Mar 18, 2014

- * Commented the netrw_btkeep line in s:NetrwOptionSave(); the effect is that netrw buffers will remain as 'bt' =nofile. This should prevent swapfiles being created for netrw buffers.

Mar 20, 2014

- * Changed all uses of lcd to use s:NetrwLcd() instead. Consistent error handling results and it also handles Window's shares
- * Fixed netrw-d command when applied with ftp
- * https: support included for netrw#NetRead()

v150: Jul 12, 2013

- * removed a "keepalt" to allow ":e #" to {{{2 return to the netrw directory listing

Jul 13, 2013

- * (Jonas Diemer) suggested changing a <cWORD> to <cfile>.

Jul 21, 2013

- * (Yuri Kanivetsky) reported that netrw's use of mkdir did not produce directories following the user's umask.

Aug 27, 2013

- * introduced g:netrw_altfile option

Sep 05, 2013

- * s:Strlen() now uses strdisplaywidth() when available, by default

Sep 12, 2013

- * (Selyano Baldo) reported that netrw wasn't opening some directories properly from the command line.

Nov 09, 2013 * `:Lexplore` introduced
 * (Ondrej Platek) reported an issue with netrw's trees (P15). Fixed.
 * (Jorge Solis) reported that "t" in tree mode caused netrw to forget its line position.

Dec 05, 2013 * Added `<s-leftmouse>` file marking (see `netrw-mf`)

Dec 05, 2013 * (Yasuhiro Matsumoto) Explore should use `strlen()` instead `s:Strlen()` when handling multibyte chars with `strpart()` (ie. `strpart()` is byte oriented, not display-width oriented).

Dec 09, 2013 * (Ken Takata) Provided a patch; File sizes and a portion of timestamps were wrongly highlighted with the directory color when setting ``:let g:netrw_liststyle=1`` on Windows.
 * (Paul Domaskis) noted that sometimes cursorline was activating in non-netrw windows. All but one setting of cursorline was done via `setl`; there was one that was overlooked. Fixed.

Dec 24, 2013 * (esquifit) asked that netrw allow the `/cygdrive` prefix be a user-alterable parameter.

Jan 02, 2014 * Fixed a problem with netrw-based balloon evaluation (ie. `netrw#NetrwBalloonHelp()` not having been loaded error messages)

Jan 03, 2014 * Fixed a problem with tree listings
 * New command installed: `:Ntree`

Jan 06, 2014 * (Ivan Brennan) reported a problem with `netrw-P`. Fixed.

Jan 06, 2014 * Fixed a problem with `netrw-P` when the modified file was to be abandoned.

Jan 15, 2014 * (Matteo Cavalleri) reported that when the banner is suppressed and tree listing is used, a blank line was left at the top of the display. Fixed.

Jan 20, 2014 * (Gideon Go) reported that, in tree listing style, with a previous window open, that the wrong directory was being used to open a file. Fixed. (P21)

v149: Apr 18, 2013 * in wide listing format, now have maps for `[[2w` and `b` to move to next/previous file

Apr 26, 2013 * one may now copy files in the same directory; netrw will issue requests for what names the files should be copied under

Apr 29, 2013 * Trying Benzinger's problem again. Seems that commenting out the `BufEnter` and installing `VimEnter` (only) works. Weird problem! (tree listing, `vim -O Dir1 Dir2`)

May 01, 2013 * `:Explore ftp://...` wasn't working. Fixed.

May 02, 2013 * introduced `g:netrw_bannerbackslash` as requested by Paul Domaskis.

Jul 03, 2013 * Explore now avoids splitting when a buffer will be hidden.
v148: Apr 16, 2013 * changed Netrw's Style menu to allow direct choice of listing style, hiding style, and sorting style

13. Todo netrw-todo {{{1

07/29/09 : banner : [g:netrw_banner](#) can be used to suppress the banner. This feature is new and experimental, so its in the process of being debugged.
09/04/09 : "gp" : See if it can be made to work for remote systems.
: See if it can be made to work with marked files.

14. Credits netrw-credits {{{1

Vim editor by Bram Moolenaar (Thanks, Bram!)
dav support by C Campbell
fetch support by Bram Moolenaar and C Campbell
ftp support by C Campbell <NcampObell@SdrPchip.AorgM-NOSPAM>
http support by Bram Moolenaar <bram@moolenaar.net>
rcp
rsync support by C Campbell (suggested by Erik Warendorph)
scp support by raf <raf@comdyn.com.au>
sftp support by C Campbell

inputsecret(), BufReadCmd, BufWriteCmd contributed by C Campbell

Jérôme Augé -- also using new buffer method with ftp+.netrc
Bram Moolenaar -- obviously vim itself, :e and v:cmdarg use, fetch,...

Yasuhiro Matsumoto -- pointing out undo+0r problem and a solution
Erik Warendorph -- for several suggestions (g:netrw_..._cmd variables, rsync etc)

Doug Claar -- modifications to test for success with ftp operation

Modelines: {{{1

Highlighting matching parens `matchparen`

The functionality mentioned here is a `standard-plugin`. This plugin is only available if `'compatible'` is not set.

You can avoid loading this plugin by setting the "loaded_matchparen" variable:
`:let loaded_matchparen = 1`

The plugin installs CursorMoved, CursorMovedI and WinEnter autocommands to redefine the match highlighting.

To disable the plugin after it was loaded use this command:
`:NoMatchParen :DoMatchParen`

`:NoMatchParen`

And to enable it again:

`:DoMatchParen`

The highlighting used is MatchParen. You can specify different colors with the `":highlight"` command. Example:

`:hi MatchParen ctermbg=blue guibg=lightblue`

The characters to be matched come from the `'matchpairs'` option. You can change the value to highlight different matches. **Note** that not everything is possible. For example, you can't highlight single or double quotes, because the start and end are equal.

The syntax highlighting attributes are used. When the cursor currently is not in a string or comment syntax item, then matches inside string and comment syntax items are ignored. Any syntax items with "string" or "comment" somewhere in their name are considered string or comment items.

The search is limited to avoid a delay when moving the cursor. The limits are:

- What is visible in the window.
- 100 lines above or below the cursor to avoid a long delay when there are closed folds.
- `'synmaxcol'` times 2 bytes before or after the cursor to avoid a delay in a long line with syntax highlighting.
- A timeout of 300 msec (60 msec in Insert mode). This can be changed with the `g:matchparen_timeout` and `g:matchparen_insert_timeout` variables and their buffer-local equivalents `b:matchparen_timeout` and `b:matchparen_insert_timeout`.

If you would like the `%` command to work better, the `matchit` plugin can be

used, see [matchit-install](#) . This plugin also helps to skip matches in comments. This is unrelated to the matchparen highlighting, they use a different mechanism.



pi_spec.txt For Vim version 9.1. Last change: 2006 Apr 24

by Gustavo Niemeyer

This is a filetype plugin to work with rpm spec files.

Currently, this Vim plugin allows you to easily update the %changelog section in RPM spec files. It will even create a section for you if it doesn't exist yet. If you've already inserted an entry today, it will give you the opportunity to just add a new item in today's entry. If you don't provide a format string ([spec_chglog_format](#)), it'll ask you an email address and build a format string by itself.

1. How to use it [spec-how-to-use-it](#)
2. Customizing [spec-customizing](#)

1. How to use it [spec-how-to-use-it](#)

The spec_chglog plugin provides a map like the following:

```
:map <buffer> <LocalLeader>c <Plug>SpecChangelog
```

It means that you may run the plugin inside a spec file by pressing your maplocalleader key (default is '\') plus 'c'. If you do not have [spec_chglog_format](#) set, the plugin will ask you for an email address to use in this edit session.

Every time you run the plugin, it will check to see if the last entry in the changelog has been written today and by you. If the entry matches, it will just insert a new changelog item, otherwise it will create a new changelog entry. If you are running with [spec_chglog_release_info](#) enabled, it will also check if the name, version and release matches. The plugin is smart enough to ask you if it should update the package release, if you have not done so.

Setting a map [spec-setting-a-map](#)

As you should know, you can easily set a map to access any Vim command (or anything, for that matter). If you don't like the default map of [<LocalLeader>c](#), you may just set up your own key. The following line shows you how you could do this in your .vimrc file, mapping the plugin to the [<F5>](#) key:

```
au FileType spec map <buffer> <F5> <Plug>SpecChangelog
```

Note: the plugin will respect your desire to change the default mapping and won't set it.

This command will add a map only in the spec file buffers.

2. Customizing

[spec-customizing](#)

The format string

[spec_chglog_format](#)

You can easily customize how your spec file entry will look like. To do this just set the variable "spec_chglog_format" in your .vimrc file like this:

```
let spec_chglog_format = "%a %b %d %Y My Name <my@email.com>"
```

Note that "%a %b %d %Y" is the most used time format. If you don't provide a format string, when you run the SpecChangelog command for the first time, it will ask you an email address and build the `spec_chglog_format` variable for you. This way, you will only need to provide your email address once.

To discover which format options you can use, take a look at the `strftime()` function man page.

Where to insert new items

[spec_chglog_prepend](#)

The plugin will usually insert new %changelog entry items (note that it's not the entry itself) after the existing ones. If you set the `spec_chglog_prepend` variable

```
let spec_chglog_prepend = 1
```

it will insert new items before the existing ones.

Inserting release info

[spec_chglog_release_info](#)

If you want, the plugin may automatically insert release information on each changelog entry. One advantage of turning this feature on is that it may control if the release has been updated after the last change in the package or not. If you have not updated the package version or release, it will ask you if it should update the package release for you. To turn this feature on, just insert the following code in your .vimrc:

```
let spec_chglog_release_info = 1
```

Then, the first item in your changelog entry will be something like:

```
+ name-1.0-1cl
```

If you don't like the release updating feature and don't want to answer "No" each time it detects an old release, you may disable it with

```
let spec_chglog_never_increase_release = 1
```

Good luck!!

+=====+
| Tar File Interface |
+=====+

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
(remove NOSPAM from Campbell's email first)

Copyright 2005-2017: tar-copyright

The VIM LICENSE (see copyright) applies to the files in this package, including tarPlugin.vim, tar.vim, and pi_tar.txt. Like anything else that's except use "tar.vim" instead of "VIM". Like anything else that's free, tar.vim and its associated files are provided *as is* and comes with no warranty of any kind, either expressed or implied. No guarantees of merchantability. No guarantees of suitability for any purpose. By using this plugin, you agree that in no event will the copyright holder be liable for any damages resulting from the use of this software. Use at your own risk!

=====

1. Contents	tar tar-contents
1. Contents.....	tar-contents
2. Usage.....	tar-usage
3. Options.....	tar-options
4. History.....	tar-history

=====

2. Usage	tar-usage tar-manual
----------	----------------------

When one edits a *.tar file, this plugin will handle displaying a contents page. Select a file to edit by moving the cursor atop the desired file, then hit the <return> key. After editing, one may also write to the file. Currently, one may not make a new file in tar archives via the plugin.

:Vimuntar

VIMUNTAR

:Vimuntar [vimhome]

This command copies, if necessary, the tarball to the .vim or vimfiles directory using the first writable directory in the 'runtimepath' when no [vimhome] is specified. Otherwise, the [vimhome] argument allows the user to specify that directory, instead.

The copy is done using the command in g:tar_copycmd , which is cp for cygwin, unix, macunix copy for windows (32, 95, 64, 16)

The extraction is done with the command specified with g:tar_extractcmd , which by default is "tar -xf"

:TarDiff

DIFFERENCING SUPPORT

:TarDiff [filename]

This command will attempt to show the differences between the tarball version of a file and the associated file on the system. In order to find that file on the system, the script uses the path associated with the file mentioned in the tarball. If the current directory is not correct for that path, :TarDiff will fail to find the associated file.

If the [filename] is given, that that filename (and path) will be used to specify the associated file.

PREVENTING LOADING

If for some reason you do not wish to use vim to examine tar'd files, you may put the following two variables into your <.vimrc> to prevent the tar plugin from loading:

```
let g:loaded_tarPlugin= 1
let g:loaded_tar      = 1
```

3. Options

tar-options

These options are variables that one may change, typically in one's <.vimrc> file.

Variable	Default Value	Explanation
<code>g:tar_browseoptions</code>	"Ptf"	used to get a list of contents
<code>g:tar_readoptions</code>	"OPxf"	used to extract a file from a tarball
<code>g:tar_cmd</code>	"tar"	the name of the tar program
<code>g:tar_nomax</code>	0	if true, file window will not be maximized
<code>g:tar_secure</code>	undef	if exists: "--"s will be used to prevent unwanted option expansion in tar commands. Please be sure that your tar command accepts "--"; Posix compliant tar utilities do accept them. if not exists: The tar plugin will reject any tar files or member files that begin with "_" Not all tar's support the "--" which is why it isn't default.
<code>g:tar_writeoptions</code>	"uf"	used to update/replace a file

4. History

tar-history

v31 Apr 02, 2017 * (klartext) reported that browsing encrypted files in a zip archive created unencrypted

swap files. I am applying a similar fix used on zip.vim to tar.vim: new buffers are opened with `:noswapfile`.

	May 16, 2017	* When the mouse option isn't empty, the leftmouse can be used to select a file in the tar-file listing.
v30	Apr 22, 2014	* .tgz files are ambiguous: they may have been compressed with either gzip or bzip2. Tar.vim disambiguates by using unix's "file" command.
	Feb 18, 2016	* Changed =~ to =~# where appropriate
	Feb 18, 2017	* Now also permits xz decompression
v28	Jun 23, 2011	* a few more decompression options (tbz tb2 txz)
v27	May 31, 2011	* moved cygwin detection before g:tar_copycmd handling
		* inserted additional <code>:keepj</code> modifiers
		* changed silent to sil! (<code>:silent</code>)
v26	Aug 09, 2010	* uses buffer-local instead of window variables to hold tarfile name
		* inserted keepj before 0d to protect jump list
v25	Jun 19, 2010	* (Jan Steffens) added support for xz compression
v24	Apr 07, 2009	* :Untarvim command implemented
	Sep 28, 2009	* Added lzma support
v22	Aug 08, 2008	* security fixes
v16	Jun 06, 2008	* tarfile:: used instead of tarfile: when editing files inside tarballs. Fixes a problem with tarballs called things like c:\abc.tar. (tnx to Bill McCarthy)
v14	May 09, 2008	* arno caught a security bug
	May 28, 2008	* various security improvements. Now requires patch 299 which provides the fnameescape() function
	May 30, 2008	* allows one to view *.gz and *.bz2 files that are in *.tar files.
v12	Sep 07, 2007	* &shq now used if not the empty string for g:tar_shq
v10	May 02, 2006	* now using "redraw then echo" to show messages, instead of "echo and prompt user"
v9	May 02, 2006	* improved detection of masquerading as tar file
v8	May 02, 2006	* allows editing of files that merely masquerade as tar files
v7	Mar 22, 2006	* work on making tar plugin work across network
	Mar 27, 2006	* g:tar_cmd now available for users to change the name of the tar program to be used. By default, of course, it's "tar".
v6	Dec 21, 2005	* writing to files not in directories caused problems - fixed (pointed out by Christian Robinson)
v5	Nov 22, 2005	* report option workaround installed
v3	Sep 16, 2005	* handles writing files in an archive back to the archive
	Oct 18, 2005	* <amatch> used instead of <afile> in autocmds
	Oct 18, 2005	* handles writing to compressed archives
	Nov 03, 2005	* handles writing tarfiles across a network

```
v2          using netrw#NetWrite()  
* converted to use Vim7's new autoload feature  
by Bram Moolenaar  
v1      (original) * Michael Toren  
(see http://michael.toren.net/code/)
```

Vimball Archiver

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
(remove NOSPAM from Campbell's email first)

Copyright: (c) 2004-2015 by Charles E. Campbell Vimball-copyright

The VIM LICENSE (see copyright) applies to the files in this package, including vimballPlugin.vim, vimball.vim, and pi_vimball.txt. except use "vimball" instead of "VIM". Like anything else that's free, vimball.vim and its associated files are provided *as is* and comes with no warranty of any kind, either expressed or implied. No guarantees of merchantability. No guarantees of suitability for any purpose. By using this plugin, you agree that in no event will the copyright holder be liable for any damages resulting from the use of this software. Use at your own risk!

=====

1. Contents

vba vimball vimball-contents

1. Contents.....: vimball-contents
2. Vimball Introduction.....: vimball-intro
3. Vimball Manual.....: vimball-manual
 MkVimball.....: :MkVimball
 UseVimball.....: :UseVimball
 RmVimball.....: :RmVimball
4. Vimball History.....: vimball-history

=====

2. Vimball Introduction

vimball-intro

Vimball is intended to make life simpler for users of plugins. All a user needs to do with a vimball is:

```
vim someplugin.vba
:so %
:q
```

and the plugin and all its components will be installed into their appropriate directories. Note that one doesn't need to be in any particular directory when one does this. Plus, any help for the plugin will also be automatically installed.

If a user has decided to use the AsNeeded plugin, vimball is smart enough to put scripts nominally intended for .vim/plugin/ into .vim/AsNeeded/ instead.

Removing a plugin that was installed with vimball is really easy:

```
vim
:RmVimball someplugin
```

This operation is not at all easy for zips and tarballs, for example.

Vimball examines the user's 'runtimepath' to determine where to put

the scripts. The first directory mentioned on the runtimepath is usually used if possible. Use
`:echo &rtp`
to see that directory.

3. Vimball Manual

`vimball-manual`

MAKING A VIMBALL

`:MkVimball`

`:[range]MkVimball[!] filename [path]`

The range is composed of lines holding paths to files to be included in your new vimball, omitting the portion of the paths that is normally specified by the runtimepath (`'rtp'`). As an example:

`plugin/something.vim`
`doc/something.txt`

using

`:[range]MkVimball filename`

on this range of lines will create a file called "filename.vba" which can be used by Vimball.vim to re-create these files. If the "filename.vba" file already exists, then MkVimball will issue a warning and not create the file. **Note** that these paths are relative to your `.vim` (vimfiles) directory, and the files should be in that directory. The vimball plugin normally uses the first `'runtimepath'` directory that exists as a prefix; don't use absolute paths, unless the user has specified such a path.

If you use the exclamation point (!), then MkVimball will create the "filename.vba" file, overwriting it if it already exists. This behavior resembles that for `:w` .

If you wish to force slashes into the filename, that can also be done by using the exclamation mark (ie. `:MkVimball! path/filename`).

The tip at http://vim.wikia.com/wiki/Using_VimBall_with_%27Make%27 has a good idea on how to automate the production of vimballs using make.

MAKING DIRECTORIES VIA VIMBALLS

`g:vimball_mkdir`

First, the `mkdir()` command is tried (not all systems support it).

If it doesn't exist, then if `g:vimball_mkdir` doesn't exist, it is set as follows:

`|g:netrw_localmkdir|, if it exists`
`"mkdir" , if it is executable`
`"makedir" , if it is executable`
`Otherwise , it is undefined.`

One may explicitly specify the directory making command using `g:vimball_mkdir`. This command is used to make directories that are needed as indicated by the vimball.

CONTROLLING THE VIMBALL EXTRACTION DIRECTORY

`g:vimball_home`

You may override the use of the `'runtimepath'` by specifying a variable, `g:vimball_home`.

```
vim filename.vba
```

`vimball-extract`

Simply editing a Vimball will cause Vimball.vim to tell the user to source the file to extract its contents.

Extraction will only proceed if the first line of a putative vimball file holds the "Vimball Archiver by Charles E. Campbell" line.

LISTING FILES IN A VIMBALL

`:VimballList`

```
:VimballList
```

This command will tell Vimball to list the files in the archive, along with their lengths in lines.

MANUALLY INVOKING VIMBALL EXTRACTION

`:UseVimball`

```
:UseVimball [path]
```

This command is contained within the vimball itself; it invokes the `vimball#Vimball()` routine which is responsible for unpacking the vimball. One may choose to execute it by hand instead of sourcing the vimball; one may also choose to specify a path for the installation, thereby overriding the automatic choice of the first existing directory on the `'runtimepath'`.

REMOVING A VIMBALL

`:RmVimball`

```
:RmVimball vimballfile [path]
```

This command removes all files generated by the specified vimball (but not any directories it may have made). One may choose a path for de-installation, too (see `'runtimepath'`); otherwise, the default is the first existing directory on the `'runtimepath'`. To implement this, a file (`.VimballRecord`) is made in that directory containing a record of what files need to be removed for all vimballs used thus far.

PREVENTING LOADING

If for some reason you don't want to be able to extract plugins using vimballs: you may prevent the loading of vimball.vim by putting the following two variables in your `<.vimrc>`:

```
let g:loaded_vimballPlugin= 1
let g:loaded_vimball      = 1
```

WINDOWS

vimball-windows

Many vimball files are compressed with gzip. Windows, unfortunately, does not come provided with a tool to decompress gzip'ped files. Fortunately, there are a number of tools available for Windows users to un-gzip files:

<u>Item</u>	<u>Tool/Suite</u>	<u>Free</u>	<u>Website</u>
7zip	tool	y	http://www.7-zip.org/
Winzip	tool	n	http://www.winzip.com/downwz.htm
unxutils	suite	y	http://unxutils.sourceforge.net/
cygwin	suite	y	http://www.cygwin.com/
GnuWin32	suite	y	http://gnuwin32.sourceforge.net/
MinGW	suite	y	http://www.mingw.org/

4. Vimball History

vimball-history {{{1

- 37 : Jul 18, 2014 * (by request of T. Miedema) added augroup around the autocmds in vimballPlugin.vim
- Jul 06, 2015 * there are two uses of tabc; changed to tabc!
- 34 : Sep 22, 2011 * "UseVimball path" now supports a non-full path by prepending the current directory to it.
- 33 : Apr 02, 2011 * Gave priority to *.vmb over *.vba
- * Changed silent! to sil! (shorter)
- * Safed 'swf' setting (during vimball extraction, its now turned off)
- 32 : May 19, 2010 * (Christian Brabrandt) :so someplugin.vba and :so someplugin.vba.gz (and the other supported compression types) now works
- * (Jan Steffens) added support for xz compression
- * fenc extraction was erroneously picking up the end of the line number when no file encoding was present. Fixed.
- * By request, beginning the switchover from the vba extension to vmb. Currently both are supported; MkVimball, however, now will create *.vmb files.
- Feb 11, 2011 * motoyakurotsu reported an error with vimball's handling of zero-length files
- Feb 18, 2016 * Changed =~ to =~# where appropriate
- 30 : Dec 08, 2008 * fnameescape() inserted to protect error messaging using corrupted filenames from causing problems
- * RmVimball supports filenames that would otherwise be considered to have "magic" characters (ie. Abc[1].vba)
- Feb 18, 2009 * s:Escape(), g:vimball_shq, and g:netrw_shq removed (shellescape() used directly)
- Oct 05, 2009 * (Nikolai Weibull) suggested that MkVimball be allowed to use slashes in the filename.
- 26 : May 27, 2008 * g:vimball_mkdir usage installed. Makes the

\$HOME/.vim (or \$HOME\vimfiles) directory if necessary.

May 30, 2008 * (tnx to Bill McCarthy) found and fixed a bug: vimball wasn't updating plugins to AsNeeded/ when it should

25 : Mar 24, 2008 * changed vimball#Vimball() to recognize doc/*.??x files as help files, too.

Apr 18, 2008 * RmVimball command is now protected by saving and restoring settings -- in particular, acd was causing problems as reported by Zhang Shuhan

24 : Nov 15, 2007 * g:vimball_path_escape used by s:Path() to prevent certain characters from causing trouble (defunct: `fnameescape()` and `shellescape()` now used instead)

22 : Mar 21, 2007 * uses setlocal instead of set during BufEnter

21 : Nov 27, 2006 * (tnx to Bill McCarthy) vimball had a header handling problem and it now changes \s to /s

20 : Nov 20, 2006 * substitute() calls have all had the 'e' flag removed.

18 : Aug 01, 2006 * vimballs now use folding to easily display their contents.
* if a user has AsNeeded/somefile, then vimball will extract plugin/somefile to the AsNeeded/ directory

17 : Jun 28, 2006 * changes all \s to /s internally for Windows

16 : Jun 15, 2006 * A. Mechelynck's idea to allow users to specify installation root paths implemented for UseVimball, Mkvimball, and RmVimball.
* RmVimball implemented

15 : Jun 13, 2006 * bugfix

14 : May 26, 2006 * bugfixes

13 : May 01, 2006 * exists("&acd") used to determine if the acd option exists

12 : May 01, 2006 * bugfix - the 'acd' option is not always defined

11 : Apr 27, 2006 * VimballList would create missing subdirectories that the vimball specified were needed. Fixed.

10 : Apr 27, 2006 * moved all setting saving/restoration to a pair of functions. Included some more settings in them which frequently cause trouble.

9 : Apr 26, 2006 * various changes to support Windows' predilection for backslashes and spaces in file and directory names.

7 : Apr 25, 2006 * bypasses foldenable
* uses more exe and less norm! (:yank :put etc)
* does better at insuring a "Press ENTER" prompt appears to keep its messages visible

4 : Mar 31, 2006 * BufReadPost seems to fire twice; BufReadEnter only fires once, so the "Source this file..." message is now issued only once.

3 : Mar 20, 2006 * removed query, now requires sourcing to be extracted (:so %). Message to that effect included.
* :VimballList now shows files that would be extracted.

2 : Mar 20, 2006 * query, :UseVimball included
1 : Mar 20, 2006 * initial release



+-----+
| Zip File Interface |
+-----+

Author: Charles E. Campbell <Ncamp0bell@SdrPchip.AorgM-NOSPAM>
(remove NOSPAM from Campbell's email first)

Copyright: Copyright (C) 2005-2015 Charles E Campbell zip-copyright
The VIM LICENSE (see copyright) applies to the files in this
package, including zipPlugin.vim, zip.vim, and pi_zip.vim. except use
"zip.vim" instead of "VIM". Like anything else that's free, zip.vim
and its associated files are provided *as is* and comes with no
warranty of any kind, either expressed or implied. No guarantees of
merchantability. No guarantees of suitability for any purpose. By
using this plugin, you agree that in no event will the copyright
holder be liable for any damages resulting from the use of this
software. Use at your own risk!

=====
1. Contents zip zip-contents
1. Contents..... zip-contents
2. Usage..... zip-usage
3. Additional Extensions..... zip-extension
4. History..... zip-history

=====
2. Usage zip-usage zip-manual

When one edits a *.zip file, this plugin will handle displaying a
contents page. Select a file to edit by moving the cursor atop
the desired file, then hit the <return> key. After editing, one may
also write to the file. Currently, one may not make a new file in
zip archives via the plugin.

COMMANDS

x : extract a listed file when the cursor is atop it zip-x

OPTIONS

g:zip_nomax
If this variable exists and is true, the file window will not be
automatically maximized when opened.

g:zip_shq
Different operating systems may use one or more shells to execute
commands. Zip will try to guess the correct quoting mechanism to
allow spaces and whatnot in filenames; however, if it is incorrectly
guessing the quote to use for your setup, you may use

g:zip_shq
which by default is a single quote under Unix (') and a double quote
under Windows ("). If you'd rather have no quotes, simply set

g:zip_shq to the empty string (let g:zip_shq= "") in your `<.vimrc>`.

`g:zip_unzipcmd`

Use this option to specify the program which does the duty of "unzip". It's used during browsing. By default:

```
let g:zip_unzipcmd= "unzip"
```

`g:zip_zipcmd`

Use this option to specify the program which does the duty of "zip". It's used during the writing (updating) of a file already in a zip file; by default:

```
let g:zip_zipcmd= "zip"
```

`g:zip_extractcmd`

This option specifies the program (and any options needed) used to extract a file from a zip archive. By default,

```
let g:zip_extractcmd= g:zip_unzipcmd
```

`g:zip_exec`

For security reasons, one may prevent that Vim runs executables automatically when opening a buffer. This option (default: "1") can be used to prevent executing the "unzip" command when set to "0":

```
let g:zip_exec=0
```

PREVENTING LOADING

If for some reason you do not wish to use vim to examine zipped files, you may put the following two variables into your `<.vimrc>` to prevent the zip plugin from loading:

```
let g:loaded_zipPlugin= 1
let g:loaded_zip      = 1
```

3. Additional Extensions

`zip-extension`

Apparently there are a number of archivers which generate zip files that don't use the .zip extension (.jar, .xpi, etc). To handle such files, place a line in your `<.vimrc>` file:

```
au BufReadCmd *.jar,*.xpi call zip#Browse(expand("<amatch>"))
```

One can simply extend this line to accommodate additional extensions that should be treated as zip files.

Alternatively, one may change `g:zipPlugin_ext` in one's `.vimrc`. Currently (11/30/15) it holds:

```
let g:zipPlugin_ext= '*.zip,*.jar,*.xpi,*.ja,*.war,*.ear,*.celzip,
\ *.oxt,*.kmz,*.wsz,*.xap,*.docx,*.docm,*.dotx,*.dotm,*.potx,*.potm,
\ *.ppsx,*.ppsm,*.pptx,*.pptm,*.ppam,*.sldx,*.thmx,*.xlam,*.xlsx,*.xlsm,
\ *.xlsb,*.xltx,*.xltm,*.xlam,*.crtx,*.vdw,*.glox,*.gcsx,*.gqsx,*.epub'
```

4. History

zip-history {{{1

- v33 Dec 07, 2021 * *.xlam mentioned twice in zipPlugin
- v32 Oct 22, 2021 * to avoid an issue with a vim 8.2 patch, zipfile: has been changed to zipfile:// . This often shows up as zipfile:/// with zipped files that are root-based.
- v29 Apr 02, 2017 * (Klartext) reported that an encrypted zip file could be opened but the swapfile held unencrypted contents. The solution is to edit the contents of a zip file using the :noswapfile modifier.
- v28 Oct 08, 2014 * changed the sanity checks for executables to reflect the command actually to be attempted in zip#Read() and zip#Write()
 - * added the extraction of a file capability
- Nov 30, 2015 * added *.epub to the g:zipPlugin_ext list
- Sep 13, 2016 * added *.apk to the g:zipPlugin_ext list and sorted the suffices.
- v27 Jul 02, 2013 * sanity check: zipfile must have "PK" as its first two bytes.
 - * modified to allow zipfile: entries in quickfix lists
- v26 Nov 15, 2012 * (Jason Spiro) provided a lot of new extensions that are synonyms for .zip
- v25 Jun 27, 2011 * using keepj with unzip -Z (consistent with the -p variant)
 - * (Ben Staniford) now uses has("win32unix") && executable("cygpath") before converting to cygwin-style paths
- v24 Jun 21, 2010 * (Cédric Bosdonnat) unzip seems to need its filenames fnameescape'd as well as shellquote'd
 - * (Motoya Kurotsu) inserted keepj before 0d to protect jump list
- v17 May 09, 2008 * arno caught a security bug
- v15 Sep 07, 2007 * &shq now used if not the empty string for g:zip_shq
- v14 May 07, 2007 * using b:zipfile instead of w:zipfile to avoid problem when editing alternate file to bring up a zipfile
- v10 May 02, 2006 * now using "redraw then echo" to show messages, instead of "echo and prompt user"
 - * g:zip_shq provided to allow for quoting control for the command being passed via :r! ... commands.
- v8 Apr 10, 2006 * Bram Moolenaar reported that he received an error message due to "Pattern not found: ^.*\%0c"; this was caused by stridx finding a Name... at the beginning of the line; zip.vim tried 4,\$s/^\.*\%0c//, but that doesn't work. Fixed.
- v7 Mar 22, 2006 * escaped some characters that can cause filename handling problems.
- v6 Dec 21, 2005 * writing to files not in directories caused problems - fixed (pointed out by Christian Robinson)
- v5 Nov 22, 2005 * report option workaround installed
- v3 Oct 18, 2005 * <amatch> used instead of <afile> in autocmds
- v2 Sep 16, 2005 * silenced some commands (avoiding hit-enter prompt)
 - * began testing under Windows; works thus far
 - * filetype detection fixed

Nov 03, 2005 * handles writing zipfiles across a network using
netrw#NetWrite()
v1 Sep 15, 2005 * Initial release, had browsing, reading, and writing

SPONSOR VIM DEVELOPMENT

sponsor

Fixing bugs and adding new features takes a lot of time and effort. To show your appreciation for the work and motivate Bram and others to continue working on Vim please send a donation.

Since Bram does not need the money it will be used to help children in Uganda, see [uganda](#) . Donations increase Bram's motivation to keep working on Vim!

For the most recent information about sponsoring look on the Vim web site:

<http://www.vim.org/sponsor/>

More explanations can be found in the [sponsor-faq](#) .

REGISTERED VIM USER

register

You can become a registered Vim user by sending at least 10 euro. This works similar to sponsoring Vim, see [sponsor](#) above. Registration was made possible for the situation where your boss or bookkeeper may be willing to register software, but does not like the terms "sponsoring" and "donation".

More explanations can be found in the [register-faq](#) .

VOTE FOR FEATURES

vote-for-features

To give registered Vim users and sponsors an advantage over lurkers they can vote for the items Bram should work on. How does this voting work?

1. You send at least 10 euro. See below for ways to transfer money [send-money](#) .
2. You will be e-mailed a registration key. Enter this key on your account page on the Vim website. You can easily create an account if you don't have one yet.
3. You can enter your votes on the voting page. There is a link to that page on your account page after entering a registration key. Your votes will be counted for two years.
4. The voting results appear on the results page, which is visible for everybody: http://www.vim.org/sponsor/vote_results.php

Additionally, once you have sent 100 euro or more in total, your name appears in the "Vim hall of honour": http://www.vim.org/sponsor/hall_of_honour.php

But only if you enable this on your account page.

HOW TO SEND MONEY

[send-money](#)

Credit card Through PayPal, see the PayPal site for information:
<https://www.paypal.com/>
The e-mail address for sending sponsorship money is:
donate@vim.org
The e-mail address for Vim registration is:
register@vim.org
Using Euro is preferred, other currencies are also accepted.
In Euro countries a bank transfer is preferred, this has lower costs.

Other methods See [iccf-donations](#) .
Include "Vim sponsor" or "Vim registration" in the comment of your money transfer. Send me an e-mail that mentions the amount you transferred if you want to vote for features and show others you are a registered Vim user or sponsor.

QUESTIONS AND ANSWERS

[sponsor-faq](#) [register-faq](#)

Why should I give money?

If you do not show your appreciation for Vim then Bram will be less motivated to fix bugs and add new features. He will do something else instead.

How much money should I send?

That is up to you. The more you give, the more children will be helped. An indication for individuals that use Vim at home: 10 Euro per year. For professional use: 30 Euro per year per person. Send at least 10 euro to be able to vote for features.

What do I get in return?

Each registered Vim user and sponsor who donates at least 10 euro will be able to vote for new features. These votes will give priority to the work on Vim. The votes are valid for two years. The more money you send the more your votes count [votes-counted](#) .

If you send 100 Euro or more in total you will be mentioned on the "Vim hall of honour" page on the Vim web site. But only if you enable this on your account page. You can also select whether the amount will be visible.

How do I become a Vim sponsor or registered Vim user?

Send money, as explained above [send-money](#) and include your e-mail address. When the money has been received you will receive a unique registration key.

This key can be used on the Vim website to activate voting on your Vim account. You will then get an extra page where you can vote for features and choose whether others will be able to see that you donated. There is a link to this page on your "My Account" page.

What is the difference between sponsoring and registering?

It has a different name. Use the term "registration" if your boss doesn't like "sponsoring" or "donation". The benefits are the same.

How can I send money?

See [send-money](#) . Check the web site for the most recent information:
<http://www.vim.org/sponsor/>

Why don't you use the SourceForge donation system?

SourceForge takes 5% of the donations for themselves. If you want to support SourceForge you can send money to them directly.

I cannot afford to send money, may I still use Vim?

Yes.

I did not register Vim, can I use all available features?

Yes.

I noticed a bug, do I need to register before I can report it?

No, suggestions for improving Vim can always be given. For improvements use the developer [maillist](#) , for reporting bugs see [bugs](#) .

How are my votes counted?

[votes-counted](#)

You may vote when you send 10 euro or more. You can enter up to ten votes. You can select the same item several times to give it more points. You can also enter three counter votes, these count as negative points.

When you send 30 euro or more the points are doubled. Above 100 euro they count four times, above 300 euro they count six times, above 1000 euro ten times.

Can I change my votes?

You can change your votes any time you like, up to two years after you

sent money. The points will be counted right away.

Can I add an item to vote on?

Not directly. You can suggest items to vote on to Bram. He will consider fitting your item into the list.

How about Charityware?

Currently the Vim donations go to [uganda](#) anyway. Thus it doesn't matter if you sponsor Vim or ICCF. Except that Vim sponsoring will allow you to vote for features.

I donated \$\$\$, now please add feature XYZ!

There is no direct relation between your donation and the work Bram does. Otherwise you would be paying for work and we would have to pay tax over the donation. If you want to hire Bram for specific work, contact him directly, don't use the donation system.

Are the donations tax deductible?

That depends on your country. The donations to help the children in [Uganda](#) are tax deductible in Holland, Germany, Canada and in the USA. See the ICCF website <http://iccf-holland.org/donate.html>. You must send an e-mail to Bram to let him know that the donation is done because of the use of Vim.

Can you send me a bill?

No, because there is no relation between the money you send and the work that is done. But a receipt is possible.

Last updated on: 12 December 2019

VIM FAQ by: Christian Brabandt <cb@256bit.org>

Frequently Asked Questions

[vim-faq](#) [Vim-FAQ](#)

This Vim FAQ is created from the questions and answers posted to the vim@vim.org user mailing list and the comp.editors newsgroup. There are several ways to solve a problem in Vim. This FAQ gives one of those several possibilities. You can explore the other ways using the information and links given in this FAQ. The credit for the answers in this FAQ goes to Peppe, Benji, Charles Campbell and numerous others. An online version of this FAQ is available at https://vimhelp.org/vim_faq.txt.html

[faq-index](#)

INDEX

[faq-general-information](#)

SECTION 1 - GENERAL INFORMATION

- faq-1.1 What is Vim?
- faq-1.2 Who wrote Vim?
- faq-1.3 Is Vim compatible with Vi?
- faq-1.4 What are some of the improvements of Vim over Vi?
- faq-1.5 Is Vim free?

[faq-resources](#)

SECTION 2 - RESOURCES

- faq-2.1 Where can I learn more about Vim?
- faq-2.2 Is there a mailing list available?
- faq-2.3 Is there an archive available for the Vim mailing lists?
- faq-2.4 Where can I get the Vim user manual in HTML/PDF/PS format?
- faq-2.5 I have a "xyz" (some) problem with Vim. How do I determine if it is a problem with my setup or with Vim?
- faq-2.6 Where can I report bugs?
- faq-2.7 Where can the FAQ be found?
- faq-2.8 What if I don't find an answer in this FAQ?
- faq-2.9 I have a patch for implementing a Vim feature. Where do I send the patch?
- faq-2.10 I have a Vim tip or developed a new Vim syntax/indent/filetype/compiler plugin or developed a new script or a colorscheme. Is there a public website where I can upload this?

[faq-availability](#)

SECTION 3 - AVAILABILITY

- faq-3.1 What is the latest version of Vim?
- faq-3.2 Where can I find the latest version of Vim?
- faq-3.3 What platforms does it run on?
- faq-3.4 Where can I download the latest version of the Vim runtime files?

[faq-help](#)

SECTION 4 - HELP

- faq-4.1 How do I use the help files?
- faq-4.2 How do I search for a keyword in the Vim help files?
- faq-4.3 I am getting an error message E123, what did I do wrong?
- faq-4.4 Where can I read about the various modes in Vim?
- faq-4.5 How do I generate the Vim help tags file after adding a new Vim

help file?

faq-4.6 Can I use compressed versions of the help files?

[faq-editing-a-file](#)

SECTION 5 - EDITING A FILE

faq-5.1 How do I load a file in Vim for editing?

faq-5.2 How do I save the current file in another name (save as) and edit a new file?

faq-5.3 How do I change the current directory to the directory of the current file?

faq-5.4 How do I write a file without the line feed (EOL) at the end of the file?

faq-5.5 How do I configure Vim to open a file at the last edited location?

faq-5.6 When editing a file in Vim, which is being changed by an external application, Vim opens a warning window (like the confirm dialog) each time a change is detected. How do I disable this warning?

faq-5.7 How do I edit a file whose name is under the cursor?

faq-5.8 How do I reload/re-edit the current file?

faq-5.9 How do I autosave a file periodically?

faq-5.10 How do I open a file in read-only mode?

faq-5.11 How do I open a file for editing without saving the modifications to the current file?

faq-5.12 How do I reduce the loading time for very large files in Vim?

[faq-editing-multiple-files](#)

SECTION 6 - EDITING MULTIPLE FILES

faq-6.1 How do I open multiple files at once from within Vim?

faq-6.2 How do I switch between multiple files/buffers in Vim?

faq-6.3 How do I open several files in Vim, with each file in a separate window/tabpage?

faq-6.4 How do I configure Vim to autoloading several files at once similar to "work-sets" or "projects"?

faq-6.5 Is it possible to open multiple top level windows in a single instance of Vim similar to Nedit or emacs?

faq-6.6 How do I browse/explore directories from within Vim?

faq-6.7 How do I edit files over a network using ftp/scp/rcp/http?

[faq-backup](#)

SECTION 7 - BACKUP

faq-7.1 When I edit and save files, Vim creates a file with the same name as the original file and a "~" character at the end. How do I stop Vim from creating this file (or) How do I disable the Vim backup file feature?

faq-7.2 When I edit and save files, Vim creates a file with the same name as the original file and a ".un~" extension at the end. How do I stop Vim from creating this file (or) How do I disable the Vim undofile feature.

faq-7.3 How do I configure Vim to store all the backup files in a particular directory?

faq-7.4 When I save a file with Vim, the file permissions are changed. How do I configure Vim to save a file without changing the file permissions?

[faq-buffers](#)

SECTION 8 - BUFFERS

faq-8.1 I have made some modifications to a buffer. How do I edit another buffer without saving the modified buffer and also without losing the modifications?

- faq-8.2 How do I configure Vim to auto-save a modified buffer when switching to another buffer?
- faq-8.3 How do I replace the buffer in the current window with a blank buffer?
- faq-8.4 Is there a keyboard shortcut to load a buffer by the buffer number?
- faq-8.5 How do I open all the current buffers in separate windows?
- faq-8.6 How do I close (delete) a buffer without exiting Vim?
- faq-8.7 When I use the command `:%bd` to delete all the buffers, not all the buffers are deleted. Why?
- faq-8.8 How do I display the buffer number of the current buffer/file?
- faq-8.9 How do I delete a buffer without closing the window in which the buffer is displayed?
- faq-8.10 How do I map the <Tab> key to cycle through and open all the buffers?

faq-windows

SECTION 9 - WINDOWS

- faq-9.1 What is the difference between a Vim window and a buffer?
- faq-9.2 How do I increase the width of a Vim window?
- faq-9.3 How do I zoom into or out of a window?
- faq-9.4 How do I execute an ex command on all the open buffers or open windows or all the files in the argument list?

faq-motion

SECTION 10 - MOTION

- faq-10.1 How do I jump to the beginning (first line) or end (last line) of a file?
- faq-10.2 In insert mode, when I press the <Esc> key to go to command mode, the cursor moves one character to the left (except when the cursor is on the first character of the line). Is it possible to change this behavior to keep the cursor at the same column?
- faq-10.3 How do I configure Vim to maintain the horizontal cursor position when scrolling with the <Page Up>, <Page Down>, etc keys?
- faq-10.4 Some lines in a file are more than the screen width and they are all wrapped. When I use the j, k keys to move from one line to the next, the cursor is moved to the next line in the file instead of the next line on the screen. How do I move from one screen line to the next?
- faq-10.5 What is the definition of a sentence, paragraph and section in Vim?
- faq-10.6 How do I jump to beginning or end of a sentence, paragraph or a section?
- faq-10.7 I have lines in a file that extends beyond the right extent of the screen. How do I move the Vim view to the right to see the text off the screen?
- faq-10.8 How do I scroll two or more buffers simultaneously?
- faq-10.9 When I use my arrow keys, Vim changes modes, inserts weird characters in my document but doesn't move the cursor properly. What's going on?
- faq-10.10 How do I configure Vim to move the cursor to the end of the previous line, when the left arrow key is pressed and the cursor is currently at the beginning of a line?
- faq-10.11 How do I configure Vim to stay only in insert mode (modeless editing)?
- faq-10.12 How do I display some context lines when scrolling text?

faq-10.13 How do I go back to previous cursor locations?

faq-searching-text

SECTION 11 - SEARCHING TEXT

faq-11.1 After I searched for a text with a pattern, all the matched text stays highlighted. How do I turn off the highlighting temporarily/permanently?

faq-11.2 How do I enter a carriage return character in a search pattern?

faq-11.3 How do I search for the character "^M"?

faq-11.4 How can I search/replace characters that display as "~R", "~S", etc.?

faq-11.5 How do I highlight all the non-printable characters in a file?

faq-11.6 How do I search for whole words in a file?

faq-11.7 How do I search for the current word under the cursor?

faq-11.8 How do I search for a word without regard to the case (uppercase or lowercase)?

faq-11.9 How do I search for words that occur twice consecutively?

faq-11.10 How do I count the number of times a particular word occurs in a buffer?

faq-11.11 How do I place the cursor at the end of the matched word when searching for a pattern?

faq-11.12 How do I search for an empty line?

faq-11.13 How do I search for a line containing only a single character?

faq-11.14 How do I search and replace a string in multiple files?

faq-11.15 I am using the `:s` substitute command in a mapping. When a search for a pattern fails, the map terminates. I would like the map to continue processing the next command, even if the substitute command fails. How do I do this?

faq-11.16 How do I search for the n-th occurrence of a character in a line?

faq-11.17 How do I replace a tab (or any other character) with a hard return (newline) character?

faq-11.18 How do I search for a character by its ASCII value?

faq-11.19 How do I search for long lines?

faq-11.20 How do I display all the lines in the current buffer that contain a specified pattern?

faq-11.21 How do I search for a text string that spans multiple lines?

faq-11.22 How do I search for a pattern only within a range of lines in a buffer?

faq-11.23 How do I clear the last searched pattern?

faq-11.24 Why does this pattern "a.\{-}p\@!" not match?

faq-11.25 How can I use "/" within a pattern, without escaping it?

faq-11.26 How can I operate on a search match?

faq-changing-text

SECTION 12 - CHANGING TEXT

faq-12.1 How do I delete all the trailing white space characters (SPACE and TAB) at the end of all the lines in a file?

faq-12.2 How do I replace all the occurrences of multiple consecutive space characters to a single space?

faq-12.3 How do I reduce a range of empty lines into one line only?

faq-12.4 How do I delete all blank lines in a file? How do I remove all the lines containing only space characters?

faq-12.5 How do I copy/yank the current word?

faq-12.6 How do I yank text from one position to another position within a line, without yanking the entire line?

- faq-12.7 When I yank some text into a register, how do I append the text to the current contents of the register?
- faq-12.8 How do I yank a complete sentence that spans over more than one line?
- faq-12.9 How do I yank all the lines containing a pattern into a buffer?
- faq-12.10 How do I delete all the lines in a file that do not contain a pattern?
- faq-12.11 How do I add a line before each line with "pattern" in it?
- faq-12.12 Is there a way to operate on a line if the previous line contains a particular pattern?
- faq-12.13 How do I execute a command on all the lines containing a pattern?
- faq-12.14 Can I copy the character above the cursor to the current cursor position?
- faq-12.15 How do I insert a blank line above/below the current line without entering insert mode?
- faq-12.16 How do I insert the name of the current file into the current buffer?
- faq-12.17 How do I insert the contents of a Vim register into the current buffer?
- faq-12.18 How do I move the cursor past the end of line and insert some characters at some columns after the end of the line?
- faq-12.19 How to replace the word under the cursor (say: junk) with "foojunkbar" in Vim?
- faq-12.20 How do I replace a particular text in all the files in a directory?
- faq-12.21 I have some numbers in a file. How do I increment or decrement the numbers in the file?
- faq-12.22 How do I reuse the last used search pattern in a `:substitute` command?
- faq-12.23 How do I change the case of a string using the `:substitute` command?
- faq-12.24 How do I enter characters that are not present in the keyboard?
- faq-12.25 Is there a command to remove any or all digraphs?
- faq-12.26 In insert mode, when I press the backspace key, it erases only the characters entered in this instance of insert mode. How do I erase previously entered characters in insert mode using the backspace key?
- faq-12.27 I have a file which has lines longer than 72 characters terminated with "+" and wrapped to the next line. How can I quickly join the lines?
- faq-12.28 How do I paste characterwise yanked text into separate lines?
- faq-12.29 How do I change the case (uppercase, lowercase) of a word or a character or a block of text?
- faq-12.30 How do I enter ASCII characters that are not present in the keyboard?
- faq-12.31 How do I replace non-printable characters in a file?
- faq-12.32 How do I remove duplicate lines from a buffer?
- faq-12.33 How do I prefix all the lines in a file with the corresponding line numbers?
- faq-12.34 How do I exchange (swap) two characters or words or lines?
- faq-12.35 How do I change the characters used as word delimiters?

[faq-completion-in-insert-mode](#)

SECTION 13 - COMPLETION IN INSERT MODE

- faq-13.1 How do I complete words or lines in insert mode?
- faq-13.2 How do I complete file names in insert mode?
- faq-13.3 I am using **CTRL-P/CTRL-N** to complete words in insert mode. How do I complete words that occur after the just completed word?

faq-text-formatting

SECTION 14 - TEXT FORMATTING

- faq-14.1 How do I format a text paragraph so that a new line is inserted at the end of each wrapped line?
- faq-14.2 How do I format long lines in a file so that each line contains less than "n" characters?
- faq-14.3 How do I join short lines to the form a paragraph?
- faq-14.4 How do I format bulleted and numbered lists?
- faq-14.5 How do I indent lines in insert mode?
- faq-14.6 How do I format/indent an entire file?
- faq-14.7 How do I increase or decrease the indentation of the current line?
- faq-14.8 How do I indent a block/group of lines?
- faq-14.9 When I indent lines using the > or < key, the standard 8-tabstops are used instead of the current '**tabstop**' setting. Why?
- faq-14.10 How do I turn off the automatic indentation of text?
- faq-14.11 How do I configure Vim to automatically set the '**textwidth**' option to a particular value when I edit mails?
- faq-14.12 Is there a way to make Vim auto-magically break lines?
- faq-14.13 I am seeing a lot of ^M symbols in my file. I tried setting the '**fileformat**' option to '**dos**' and then '**unix**' and then '**mac**'. None of these helped. How can I hide these symbols?
- faq-14.14 When I paste some text into a Vim buffer from another application, the alignment (indentation) of the new text is messed up. How do I fix this?
- faq-14.15 When there is a very long wrapped line (wrap is "on") and a line doesn't fit entirely on the screen it is not displayed at all. There are blank lines beginning with "@" symbol instead of wrapped line. If I scroll the screen to fit the line the "@" symbols disappear and the line is displayed again. What Vim setting control this behavior?
- faq-14.16 How do I convert all the tab characters in a file to space characters?
- faq-14.17 What Vim options can I use to edit text that will later go to a word processor?
- faq-14.18 How do I join lines without adding or removing any space characters?

faq-visual-mode

SECTION 15 - VISUAL MODE

- faq-15.1 How do I do rectangular block copying?
- faq-15.2 How do I delete or change a column of text in a file?
- faq-15.3 How do I apply an ex-command on a set of visually selected lines?
- faq-15.4 How do I execute an ex command on a column of text selected in Visual block mode?
- faq-15.5 How do I select the entire file in visual mode?
- faq-15.6 When I visually select a set of lines and press the > key to indent the selected lines, the visual mode ends. How can I reselect the region for further operation? (or) How do I re-select the last selected visual area again?
- faq-15.7 How do I jump to the beginning/end of a visually selected region?

faq-15.8 When I select text with mouse and then press : to enter an ex command, the selected text is replaced with the : character. How do I execute an ex command on a text selected using the mouse similar to the text selected using the visual mode?

faq-15.9 When I select a block of text using the mouse, Vim goes into selection mode instead of Visual mode. Why?

[faq-command-line-mode](#)

SECTION 16 - COMMAND-LINE MODE

faq-16.1 How do I use the name of the current file in the command mode or an ex command line?

faq-16.2 How do I edit the text in the Vim command-line effectively?

faq-16.3 How do I switch from Vi mode to Ex mode?

faq-16.4 How do I copy the output from an ex-command into a buffer?

faq-16.5 When I press the <Tab> key to complete the name of a file in the command mode, if there are more than one matching file names, then Vim completes the first matching file name and displays a list of all matching filenames. How do I configure Vim to only display the list of all the matching filenames and not complete the first one?

faq-16.6 How do I copy text from a buffer to the command line and from the command line to a buffer?

faq-16.7 How do I put a command onto the command history without executing it?

faq-16.8 How do I increase the height of the command-line?

[faq-viminfo](#)

SECTION 17 - VIMINFO

faq-17.1 When I invoke Vim, I get error messages about illegal characters in the viminfo file. What should I do to get rid of these messages?

faq-17.2 How do I disable the viminfo feature?

faq-17.3 How do I save and use Vim marks/commands across Vim sessions?

[faq-remote-editing](#)

SECTION 18 - REMOTE EDITING

faq-18.1 How do I open a file with existing instance of gvim? What happened to the Vim 5.x OpenWithVim.exe and SendToVim.exe files?

faq-18.2 How do I send a command to a Vim server to write all buffers to disk?

faq-18.3 Where can I get the documentation about the Vim remote server functionality?

[faq-options](#)

SECTION 19 - OPTIONS

faq-19.1 How do I configure Vim in a simple way?

faq-19.2 How do I toggle the value of an option?

faq-19.3 How do I set an option that affects only the current buffer/window?

faq-19.4 How do I use space characters for a Vim option value?

faq-19.5 Can I add (embed) Vim option settings to the contents of a file?

faq-19.6 How do I display the line numbers of all the lines in a file?

faq-19.7 How do I change the width of the line numbers displayed using the 'number' option?

faq-19.8 How do I display (view) all the invisible characters like space, tabs and newlines in a file?

faq-19.9 How do I configure Vim to always display the current line and column number?

- faq-19.10 How do I display the current Vim mode?
- faq-19.11 How do I configure Vim to show pending/partial commands on the status line?
- faq-19.12 How do I configure the Vim status line to display different settings/values?
- faq-19.13 How do I configure Vim to display status line always?
- faq-19.14 How do I make a Vim setting persistent across different Vim invocations/instances/sessions?
- faq-19.15 Why do I hear a beep (why does my window flash) about 1 second after I hit the Escape key?
- faq-19.16 How do I make the "c" and "s" commands display a "\$" instead of deleting the characters I'm changing?
- faq-19.17 How do I remove more than one flag using a single `:set`` command from a Vim option?

faq-mapping-keys

SECTION 20 - MAPPING KEYS

- faq-20.1 How do I know what a key is mapped to?
- faq-20.2 How do I list all the user-defined key mappings?
- faq-20.3 How do I unmap a key?
- faq-20.4 I am not able to create a mapping for the `<xxx>` key. What is wrong?
- faq-20.5 Why does mapping the `<C-...>` key not work?
- faq-20.6 How do I map the numeric keypad keys?
- faq-20.7 How do I create a mapping that works only in visual mode?
- faq-20.8 How do I create a mapping that works only in normal and operator pending mode (but not in visual mode)?
- faq-20.9 In a Vim script, how do I know which keys to use for my mappings, so that the mapped key will not collide with an already used key?
- faq-20.10 How do I map the escape key?
- faq-20.11 How do I map a key to perform nothing?
- faq-20.12 I want to use the `<Tab>` key to indent a block of text and `<Shift-Tab>` key to unindent a block of text. How do I map the keys to do this? This behavior is similar to textpad, visual studio, etc.
- faq-20.13 In my mappings the special characters like `<CR>` are not recognized. How can I configure Vim to recognize special characters?
- faq-20.14 How do I use the "|" to separate multiple commands in a map?
- faq-20.15 If I have a mapping/abbreviation whose ending is the beginning of another mapping/abbreviation, how do I keep the first from expanding into the second one?
- faq-20.16 Why does it take a second or more for Vim to process a key, sometimes when I press a key?
- faq-20.17 How do I map a key to run an external command using a visually selected text?
- faq-20.18 How do I map the **CTRL-I** key while still retaining the functionality of the `<Tab>` key?
- faq-20.19 How do I define a map to accept a count?
- faq-20.20 How can I make my normal mode mapping work from within Insert Mode?

faq-abbreviations

SECTION 21 - ABBREVIATIONS

- faq-21.1 How do I auto correct misspelled words?
- faq-21.2 How do I create multi-line abbreviations?

faq-21.3 When my abbreviations are expanded, an additional space character is added at the end of the expanded text. How do I avoid this character?

faq-21.4 How do I insert the current date/time stamp into the file?

faq-21.5 How do I prevent an abbreviation from expanding in insert mode?

[faq-record-and-playback](#)

SECTION 22 - RECORD AND PLAYBACK

faq-22.1 How do I repeat an editing operation (insertion, deletion, paste, etc)?

faq-22.2 How I record and repeat a set of key sequences?

faq-22.3 How do I edit/modify a recorded set of key sequences?

faq-22.4 How do I write recorded key sequences to a file?

faq-22.5 I am using register 0 to record my key sequences (i.e. q0 q). In the recorded key sequences, I am yanking some text. After the first replay of the recorded key sequence, I am no longer able to play it back.

[faq-autocommands](#)

SECTION 23 - AUTOCOMMANDS

faq-23.1 How do I execute a command when I try to modify a read-only file?

faq-23.2 How do I execute a command every time when entering a buffer?

faq-23.3 How do I execute a command every time when entering a window?

faq-23.4 From an autocmd, how can I determine the name of the file or the buffer number for which the autocommand is executed?

faq-23.5 How do I automatically save all the changed buffers whenever Vim loses focus?

faq-23.6 How do I execute/run a function when Vim exits to do some cleanup?

[faq-syntax-highlight](#)

SECTION 24 - SYNTAX HIGHLIGHT

faq-24.1 How do I turn off/on syntax highlighting?

faq-24.2 How do I change the background and foreground colors used by Vim?

faq-24.3 How do I change the highlight colors to suit a dark/light background?

faq-24.4 How do I change the color of the line numbers displayed when the `:set number` command is used?

faq-24.5 How do I change the background color used for a Visually selected block?

faq-24.6 How do I highlight the special characters (tabs, trailing spaces, end of line, etc) displayed by the 'list' option?

faq-24.7 How do I specify a colorscheme in my .vimrc/.gvimrc file, so that Vim uses the specified colorscheme every time?

faq-24.8 Vim syntax highlighting is broken. When I am editing a file, some parts of the file is not syntax highlighted or syntax highlighted incorrectly.

faq-24.9 Is there a built-in function to syntax-highlight the corresponding matching bracket?

faq-24.10 How do I turn off the C comment syntax highlighting?

faq-24.11 How do I add my own syntax extensions to the standard syntax files supplied with Vim?

faq-24.12 How do I replace a standard syntax file that comes with the Vim distribution with my own syntax file?

faq-24.13 How do I highlight all the characters after a particular column?

faq-24.14 How do I convert a source file (.c, .h, etc) with the Vim syntax highlighting into a HTML file?

- faq-24.15 How do I list the definition of all the current highlight groups?
- faq-24.16 How can I embed one syntax highlighting language into another one?

faq-vim-script-writing

SECTION 25 - VIM SCRIPT WRITING

- faq-25.1 How do I list the names of all the scripts sourced by Vim?
- faq-25.2 How do I debug Vim scripts?
- faq-25.3 How do I locate the script/plugin which sets a Vim option?
- faq-25.4 I am getting some error/informational messages from Vim (possibly when running a script), the messages are cleared immediately. How do I display the messages again?
- faq-25.5 How do I save and restore a plugin specific information across Vim invocations?
- faq-25.6 How do I start insert mode from a Vim function?
- faq-25.7 How do I change the cursor position from within a Vim function?
- faq-25.8 How do I check the value of an environment variable in the .vimrc file?
- faq-25.9 How do I check whether an environment variable is set or not from a Vim function?
- faq-25.10 How do I call/use the Vim built-in functions?
- faq-25.11 I am using some normal mode commands in my Vim script. How do I avoid using the user-defined mappings for these normal mode commands and use the standard Vim functionality for these normal mode commands?
- faq-25.12 How do I get a visually selected text into a Vim variable or register?
- faq-25.13 I have some text in a Vim variable "myvar". I would like to use this variable in a `:s` substitute command to replace a text "mytext". How do I do this?
- faq-25.14 A Vim variable (bno) contains a buffer number. How do I use this variable to open the corresponding buffer?
- faq-25.15 How do I store the value of a Vim option into a Vim variable?
- faq-25.16 I have copied and inserted some text into a buffer from a Vim function. How do I indent the inserted text from the Vim function?
- faq-25.17 How do I get the character under the cursor from a Vim script?
- faq-25.18 How do I get the name of the current file without the extension?
- faq-25.19 How do I get the basename of the current file?
- faq-25.20 How do I get the output from a Vim function into the current buffer?
- faq-25.21 How do I call external programs from a Vim function?
- faq-25.22 How do I get the return status of a program executed using the `:!` command?
- faq-25.23 How do I determine whether the current buffer is modified or not?
- faq-25.24 I would like to use the carriage return character in a normal command from a Vim script. How do I specify the carriage return character?
- faq-25.25 How do I split long lines in a Vim script?
- faq-25.26 When I try to "execute" my function using the `:execute Myfunc()` command, the cursor is moved to the top of the current buffer. Why?
- faq-25.27 How do I source/execute the contents of a register?

- faq-25.28 After calling a Vim function or a mapping, when I press the "u" key to undo the last change, Vim undoes all the changes made by the mapping/function. Why?
- faq-25.29 How can I call a function defined with s: (script local function) from another script/plugin?
- faq-25.30 Is it possible to un-source a sourced script? In other words, reverse all the commands executed by sourcing a script.

faq-plugins

SECTION 26 - PLUGINS

- faq-26.1 How do I set different options for different types of files?
- faq-26.2 I have downloaded a Vim plugin or a syntax file or a indent file, or a color scheme or a filetype plugin from the web. Where should I copy these files so that Vim will find them?
- faq-26.3 How do I extend an existing filetype plugin?
- faq-26.4 How do I turn off loading the Vim plugins?
- faq-26.5 How do I turn on/off loading the filetype plugins?
- faq-26.6 How do I override settings made in a file type plugin in the global ftplugin directory for all the file types?
- faq-26.7 How do I disable the Vim directory browser plugin?
- faq-26.8 How do I set the filetype option for files with names matching a particular pattern or depending on the file extension?

faq-editing-program-files

SECTION 27 - EDITING PROGRAM FILES

- faq-27.1 How do I enable automatic indentation for C/C++ files?
- faq-27.2 How do I configure the indentation used for C/C++ files?
- faq-27.3 How do I turn off the automatic indentation feature?
- faq-27.4 How do I change the number of space characters used for the automatic indentation?
- faq-27.5 I am editing a C program using Vim. How do I display the definition of a macro or a variable?
- faq-27.6 I am editing a C program using Vim. How do I jump to the beginning or end of a code block from within the block?
- faq-27.7 When editing C++ files and when inserting new lines above or below a comment (//) line, Vim automatically inserts the C++ comment character (//) at the beginning of the line. How do I disable this?
- faq-27.8 How do I add the comment character "#" to a set of lines at the beginning of each line?
- faq-27.9 How do I edit a header file with the same name as the corresponding C source file?
- faq-27.10 How do I automatically insert comment leaders while typing comments?

faq-quickfix

SECTION 28 - QUICKFIX

- faq-28.1 How do I build programs from Vim?
- faq-28.2 When I run the make command in Vim I get the errors listed as the compiler compiles the program. When it finishes this list disappears and I have to use the `:clist` command to see the error message again. Is there any other way to see these error messages?
- faq-28.3 How can I perform a command for each item in the quickfix/location list?

faq-folding

SECTION 29 - FOLDING

- faq-29.1 How do I extend the Vim folding support?
- faq-29.2 When I enable folding by setting the `'foldmethod'` option, all the folds are closed. How do I prevent this?
- faq-29.3 How do I control how many folds will be opened when I start editing a file?
- faq-29.4 How do I open and close folds using the mouse?
- faq-29.5 How do I change the text displayed for a closed fold?
- faq-29.6 How do I store and restore manually created folds across different Vim invocations?
- faq-29.7 I have enabled syntax based folding. Why is Vim so slow?

faq-vim-with-external-applications

SECTION 30 - VIM WITH EXTERNAL APPLICATIONS

- faq-30.1 Can I run a shell inside a Vim window?
- faq-30.2 How do I pass the word under the cursor to an external command?
- faq-30.3 How do I get the output of a shell command into a Vim buffer?
- faq-30.4 How do I pipe the contents of the current buffer to an external command and replace the contents of the buffer with the output from the command?
- faq-30.5 How do I sort a section of my file?
- faq-30.6 How do I use Vim as a pager?
- faq-30.7 How do I view Unix man pages from inside Vim?
- faq-30.8 How do I change the diff command used by the Vim diff support?
- faq-30.9 How do I use the Vim diff mode without folding?

faq-gui-vim

SECTION 31 - GUI VIM

- faq-31.1 How do I create buffer specific menus?
- faq-31.2 How do I change the font used by GUI Vim?
- faq-31.3 When starting GUI Vim, how do I specify the location of the GVIM window?
- faq-31.4 How do I add a horizontal scrollbar in GVim?
- faq-31.5 How do I make the scrollbar appear in the left side by default?
- faq-31.6 How do I remove the Vim menubar?
- faq-31.7 I am using GUI Vim. When I press the `<Alt>` key and a letter, the menu starting with that letter is selected. I don't want this behavior as I want to map the `<Alt>-<key>` combination. How do I do this?
- faq-31.8 Is it possible to scroll the text by dragging the scrollbar so that the cursor stays in the original location?
- faq-31.9 How do I get gvim to start browsing files in a particular directory when using the `:`:browse`` command?
- faq-31.10 For some questions, like when a file is changed outside of Vim, Vim displays a GUI dialog box. How do I replace this GUI dialog box with a console dialog box?
- faq-31.11 I am trying to use GUI Vim as the editor for my xxx application. When the xxx application launches GUI Vim to edit a file, the control immediately returns to the xxx application. How do I start GUI Vim, so that the control returns to the xxx application only after I quit Vim?
- faq-31.12 Why does the "Select Font" dialog doesn't show all the fonts installed in my system?
- faq-31.13 How do I use the mouse in Vim command-line mode?
- faq-31.14 When I use the middle mouse button to scroll text, it pastes the last copied text. How do I disable this behavior?

- faq-31.15 How do I change the location and size of a GUI Vim window?
- faq-31.16 When splitting the Vim window vertically, Vim changes the position.

[faq-vim-on-unix](#)

SECTION 32 - VIM ON UNIX

- faq-32.1 I am running Vim in a xterm. When I press the **CTRL-S** key, Vim freezes. What should I do now?
- faq-32.2 I am seeing weird screen update problems in Vim. What can I do to solve this screen/display update problems?
- faq-32.3 I am using the terminal/console version of Vim. In insertmode, When I press the backspace key, the character before the cursor is not erased. How do I configure Vim to do this?
- faq-32.4 I am using Vim in a xterm. When I quit Vim, the screen contents are restored back to the original contents. How do I disable this?
- faq-32.5 When I start Vim, it takes quite a few seconds to start. How do I minimize the startup time?
- faq-32.6 How can I make the cursor in gvim in unix stop blinking?
- faq-32.7 How do I change the menu font on GTK Vim?
- faq-32.8 How do I prevent **CTRL-Z** from suspending Vim?
- faq-32.9 When I kill the xterm running Vim, the Vim process continues to run and takes up a lot of CPU (99%) time. Why is this happening?
- faq-32.10 How do I get the Vim syntax highlighting to work in a Unix terminal?

[faq-vim-on-ms-windows](#)

SECTION 33 - VIM ON MS-WINDOWS

- faq-33.1 In MS-Windows, **CTRL-V** doesn't start the blockwise visual mode. What happened?
- faq-33.2 When I press the **CTRL-Y** key, it acts like the **CTRL-R** key. How do I configure Vim to treat **CTRL-Y** as **CTRL-Y**?
- faq-33.3 How do I start GUI Vim in a maximized window always?
- faq-33.4 After doing some editing operations, Vim freezes. The cursor becomes an empty rectangle. I am not able enter any characters. What is happening?
- faq-33.5 I am using Windows XP, the display speed of maximized GVim is very slow. What can I do to speed the display updates?
- faq-33.6 What are the recommended settings for using Vim with cygwin?
- faq-33.7 I am trying to use GNU diff with Vim diff mode. When I run the diff from command line, it works. When I try to use the diff with Vim it doesn't work. What should I do now?
- faq-33.8 Is it possible to use Vim as an external editor for MS-Windows Outlook email client?
- faq-33.9 I am using Vim to edit HTML files. How do I start internet explorer with the current file to preview the HTML file?
- faq-33.10 I would like to use Vim with Microsoft Visual Studio. How do I do this?
- faq-33.11 Where do I place the `_vimrc` and `_gvimrc` files?
- faq-33.12 Every time I save a file, Vim warns about the file being changed outside of Vim. Why?

[faq-printing](#)

SECTION 34 - PRINTING

- faq-34.1 How do I print a file along with line numbers for all the lines?
- faq-34.2 How do I print a file with the Vim syntax highlighting colors?

[faq-building-vim-from-source](#)

SECTION 35 - BUILDING VIM FROM SOURCE

- faq-35.1 How do I build Vim from the sources on a Unix system?
- faq-35.2 How do I install Vim in my home directory or a directory other than the default installation directory in Unix?
- faq-35.3 How do I build Vim from the sources on a MS-Windows system?
- faq-35.4 The Vim help, syntax, indent files are missing from my Vim installation. How do I install these files?
- faq-35.5 I have built Vim from the source and installed the Vim package using "make install". Do I need to keep the Vim source directory?
- faq-35.6 How do I determine the Vim features which are enabled at compile time?
- faq-35.7 Can I build Vim without the GUI support?
- faq-35.8 When building Vim on a Unix system, I am getting "undefined reference to term_set_winsize" error. How do I resolve this error?
- faq-35.9 Vim configure keeps complaining about the lack of gtk-config while trying to use GTK 2.03. This is correct, since in GTK 2 they moved to using the generic pkg-config. I can get pkg-config to list the various includes and libs for gtk, but for some reason the configure script still isn't picking this up.
- faq-35.10 I did successfully download the sources and compiled Vim on Unix. But feature ... still does not work. What is wrong and how can I fix it?

faq-various

SECTION 36 - VARIOUS

- faq-36.1 How do I edit binary files with Vim?
- faq-36.2 How do I disable the visual error flash and the error beep?
- faq-36.3 How do I display the ascii value of a character displayed in a buffer?
- faq-36.4 Can I use zero as a count for a Vim command?
- faq-36.5 How do I disable the Vim welcome screen?
- faq-36.6 How do I avoid the "hit enter to continue" prompt?
- faq-36.7 How do I invoke Vim from command line to run a group of commands on a group of files?
- faq-36.8 How do I use a normal mode command from insert mode without leaving the insert mode?
- faq-36.9 How do I start Vim in insert mode?
- faq-36.10 How do I use Copy and Paste with Vim?
- faq-36.11 Why shouldn't I modify the files in the system runtime directory?

faq-unicode

SECTION 37 - UNICODE

- faq-37.1 Is it possible to create Unicode files using Vim?
- faq-37.2 Which Vim settings are particularly important for editing Unicode files?
- faq-37.3 What is the **'encoding'** option?
- faq-37.4 How does Vim name the various Unicode encodings?
- faq-37.5 How does Vim specify the presence or absence of a byte-order mark?
- faq-37.6 What is the **'fileencoding'** option?
- faq-37.7 What is the **'fileencodings'** option?
- faq-37.8 What is the **'termencoding'** option?
- faq-37.9 What is the **'bomb'** option?
- faq-37.10 Where can I find an example of a typical use of all these

options?

[faq-37.11](#) How can I insert Unicode characters into a file using Vim?

[faq-37.12](#) How can I know which digraphs are defined and for which characters?

[faq-1](#)

SECTION 1 - GENERAL INFORMATION

[faq-1.1](#)

1.1. What is Vim?

Vim stands for Vi IMproved. It used to be Vi IMitation, but there are so many improvements that a name change was appropriate. Vim is a text editor which includes almost all the commands from the Unix program "Vi" and a lot of new ones. All commands can be given with the keyboard. This has the advantage that you can keep your fingers on the keyboard and your eyes on the screen. For those who want it, there is mouse support and a GUI version with scrollbars and menus.

Vim is an editor, not a word processor. A word processor is used mainly to do layout of text. This means positioning it, changing the way it appears on output. More often than not, the final document is meant to be printed or typeset or what have you, in order to present it in a pleasing manner to others. Examples of word processors are Microsoft Word, FrameMaker, and OpenOffice Writer.

An editor is simply for entering text. Any typesetting or laying out of the document is secondary. With an editor, one's main concern is entering text, not making the text look good. Examples of editors other than Vim and Vi are Emacs, TextMate, Ultraedit and gedit. And Notepad.

For more information, read:

[intro](#)

[faq-1.2](#)

1.2. Who wrote Vim?

Most of Vim is based on Stevie and was written by Bram Moolenaar, with contributions from too many people to mention here.

For more information, read:

[author](#)
[credits](#)

[faq-1.3](#)

1.3. Is Vim compatible with Vi?

Yes. Vim is very much compatible with Vi. You can use the "-C" command-line flag to start Vim in Vi compatible mode:

```
$ vim -C
```


You can also use:

```
$ vim -u NONE
```

You can also set the **'compatible'** option to enable Vi compatibility:

```
:set compatible
```

If you want to make sure, to start Vim in a **'nocompatible'** mode to original Vi, supply the `-N` command line argument:

```
$ vim -N
```

For more information, read:

```
-C  
-N  
'compatible'  
compatible-default
```

faq-1.4

1.4. What are some of the improvements of Vim over Vi?

A short summary of the improvements of Vim over vi is listed below. The list shows that Vim is a thoroughly modern and feature-packed editor. Standard features of modern editors are implemented, and there is an equal emphasis on general power-user features and features for programmers.

Features to modernise Vi:

Multi-level undo

Allows you to set the number of times you can undo your changes in a file buffer. You can also redo an undone change. Also starting with version 7.3 Vim can permanently store your undo information, so that you can undo your changes which you have done in a previous editing session.

Tabs, Multiple windows and buffers

Each file can be displayed in its own window. You can move easily from one window to another. Each file opened during a Vim session also has an associated buffer and you can easily jump from one to the other. Also like any modern GUI, Vim supports opening several files in tabs. You can do batch processing for tabs, buffers, windows and the argumentlist.

Flexible insert mode

Vim allows you to use the arrow keys while in insert mode to move around in the file. No more hitting `<Esc>`, moving around, then hitting ``i'` or ``a'`.

Macros

Vim has a facility which allows you to record a sequence of typed characters and repeat them any number of times.

Visual mode

You can highlight sections of text and execute operations on this section of text only.

Block operators

Allow selection and highlighting of rectangular blocks of text in order to execute specific operations on them.

Online help system

You can easily find help on any aspect of using Vim. Help is displayed in its own window.

Command-line editing and history

History allows you to use the arrow keys to repeat or search for a command that has already been typed. Allows you to match the beginning of a command with the beginning of another similar command in the history buffer. You can also edit a command to correct typos or change a few values.

Command line completion.

Using the <Tab> key, you can complete commands, options, filenames, etc. as needed.

Horizontal scrolling.

Long lines can be scrolled horizontally (with or without the GUI).

Unicode and internationalization improvements.

Vim is able to edit files in unicode encoding and uses internally an utf-8 encoding. Additionally Vim can display text right to left oriented.

Advanced user features:

Text formatting

With two keystrokes, you can format large sections of text, without the use of external programs.

Completion in Insert mode

Vim provides several different possibilities to complete your text. For example Vim can complete words while you are typing, by matching

the current word with other similar words in the file.

Jump tags

Just like in an internet browser, you can jump back to previous parts of the text you were editing, and then forward again. Your brain is thus free to edit instead of navigate.

Automatic commands

Commands automatically executed when reading or writing a file, jumping to another buffer, etc.

Viminfo

Allows storing of the command line history, marks and registers in a file to be read on startup. Therefore, you can recall old search patterns, macros, etc., in a new Vim session.

Mouse support

The mouse is supported in an xterm and for MS-DOS. It can be used to position the cursor, select the visual area, paste a register, etc.

Graphical User Interface (GUI)

There are several different graphical user interfaces available. Also, it's very easy to add your own menus. Of course, console vim is still supported, and very widely used.

Scripting language

Vim has a powerful scripting language so new commands can be created. You can also use Perl, Python, TCL, Lua and Ruby to achieve the same thing!

Plugins

Extra functionality implemented via vim commands (regular commands or the scripting language) that is automatically loaded on startup. Examples: file explorer, network editing, enhanced autocompletion, syntax checks. More are being developed and shared on VimOnline all the time.

Syntax highlighting for many programming languages

Syntax highlighting (including concealing items) for hundreds of programming languages is supported. Support for others can be added.

Extended regular expressions

Vim supports extended regular expressions which are similar in functionality to that of Perl regular expressions.

Integrated Spell checking

Spell checking has been integrated into Vim.

Diff mode

Vim can highlight the differences between two, three or four files. Identical lines will be folded away and hidden.

Encryption using the blowfish algorithm

Vim allows to encrypt your files using the symmetric block cipher blowfish as well as the swap file.

Extensive customizable

Vim can be tuned and customized to work like you want by setting options. You can define your own commands, macros and even plugins to extend its capabilities

Packages

Packages have been added to keep the installation of the growing number of plugins manageable. This is a convenient way to get one or more plugins, drop them in a directory and keep them updated. Vim will load them automatically, or only when desired.

Programming performance features:

Edit-compile-edit speedup

You can compile within Vim and automatically jump to the location of errors in the source code.

Indenting for many programming languages

C, C++, Java, Perl, XML and many other languages can be automatically indented by vim while you type. Support for others can be added.

Searching for words in include files

Vim allows you to search for a match of the word under the cursor in the current and included files.

Advanced text objects

Instantly select, delete, copy, indent, format, change case, or ... to all the text between (and), or { and }, or < and >, or [and]. Or a word, sentence, or paragraph. Very powerful.

Folding

Certain parts of the text can be "folded" away. The best example is

the body of a function. You can get an overview of the code, and then open the fold of the function whose detail you need to see.

[ctags and cscope integration](#)

Using these two powerful programs, you can jump to a definition of a function from a calling instance of it, and use other tricks to navigate source code.

[Integration of several programming languages](#)

If you find the internal scripting language not powerful enough, you can extend Vim using Lua, Ruby, Tcl, Perl and Python 2 and 3.

[Asynchronous I/O support](#)

Vim uses jobs and channels to talk to other programs asynchronously. This allows to have e.g. a compiler run in the background and open the quickfix list as soon as it is finished to fix warnings and errors.

[Timers](#)

Timers are asynchronous and can fire once or repeatedly to invoke a function to do any work.

For more information, read:

[vi-differences](#)

[faq-1.5](#)

1.5. Is Vim free?

Vim is Charityware. There are no restrictions on using or copying Vim, but the author encourages you to make a donation to charity. A document explaining how to do so is included in the distribution.

For more information, read:

[copyright](#)

===== [faq-2](#)

[SECTION 2 - RESOURCES](#)

[faq-2.1](#)

2.1. Where can I learn more about Vim?

You can post your Vim questions to the vim@vim.org mailing list. You can post your Vim development related questions to the vim-dev@vim.org mailing list. Vim does not have a newsgroup of its own. But the appropriate newsgroup to post to is [comp.editors](#).

"VimOnline" is a web page that serves as a de facto homepage for vim,

although the main purpose of it is to gather tips and scripts from everywhere. Get involved! The URL is <https://www.vim.org>

Finally, read the Vi FAQ:

<http://www.faqs.org/faqs/editor-faq/vi/part1/index.html>

Finally, there are also some communities, where you can discuss features or ask questions:

<https://vi.stackexchange.com>
<https://vim.reddit.com>

For more information, read:

`mail-list`
`internet`

faq-2.2

2.2. Is there a mailing list available?

There are several:

NAME	DESCRIPTION
vim-announce	Announcements of new releases
vim	General discussion
vim-dev	Patches, bug reports, development discussions
vim-mac	Macintosh discussion
vim-fr	General discussion in French

Of these, only vim and vim-dev are of general interest. vim-announce is read-only to most people, and its messages are sent to the other lists as well. The remaining four are very low volume.

ACTION	EMAIL SEND TO
To subscribe:	<NAME>-subscribe@vim.org
To unsubscribe:	<NAME>-unsubscribe@vim.org
To get help:	<NAME>-help@vim.org

The available mailing lists are also mentioned here:

<https://www.vim.org/maillist.php>

faq-2.3

2.3. Is there an archive available for the Vim mailing lists?

Yes. Visit <https://groups.yahoo.com/>, where name is one of: vimannounce, vim, vimdev, vim-fr, vim-mac, vim-vms.

Alternatively, visit www.gmane.org to find out about GMANE, which allows you to access the mailing lists as though they were newsgroups. This offers some convenience to those who wish to browse the history or casually

observe the current threads.

faq-2.4

2.4. Where can I get the Vim user manual in HTML/PDF/PS format?

You can download the HTML/PDF/PS format of the Vim user manual from:

<https://vimdoc.sourceforge.net/>

Note, the user manual from that page is currently pretty outdated. It's best to either use the documentation that comes with vim or use the online version at <https://vimhelp.org>

You can find a pdf version of the full English help, including this faq (in letter, A4 and Ipad format) at:

<https://nathangrigg.com/vimhelp/>

This document is cross-referenced, so you can use the hyperlink functionality.

faq-2.5

2.5. I have a "xyz" (some) problem with Vim. How do I determine if it is a problem with my setup or with Vim? / Have I found a bug in Vim?

First, you need to find out, whether the error is in the actual runtime files or any plugin that is distributed with Vim or whether it is a simple side effect of any configuration option from your `.vimrc` or `.gvimrc`. So first, start vim like this:

```
vim -u NONE -U NONE -N -i NONE
```

This starts Vim in nocompatible mode (-N), without reading your viminfo file (-i NONE), without reading any configuration file (-u NONE for not reading `.vimrc` file and -U NONE for not reading a `.gvimrc` file) or even plugin.

In this invocation, try to reproduce your problem. If the error persists, the chance is good you've found a bug in Vim (see also Question 2.6. [faq-2.6](#))

If the error does not occur when starting Vim this way, then the problem is either related to some plugin of yours or some setting in one of your local setup files. You need to find out, what triggers the error, you try starting Vim this way:

```
vim -u NONE -U NONE -N
```

If the error occurs, the problem is your `.viminfo` file. Simply delete the viminfo file then. If the error does not occur, try:

```
vim -u ~/.vimrc --noplugin -N -i NONE
```

This will simply use your `.vimrc` as configuration file, but not load any

plugins. If the error occurs this time, the error is possibly caused by some configuration option inside your `.vimrc` file. Depending on the length of your `vimrc` file, it can be quite hard to trace the origin within that file.

The best way is to add `:finish` command in the middle of your `.vimrc`. Then restart again using the same command line. If the error still occurs, the bug must be caused because of a setting in the first half of your `.vimrc`. If it doesn't happen, the problematic setting must be in the second half of your `.vimrc`. So move the `:finish` command to the middle of that half, of which you know that triggers the error and move your way along, until you find the problematic option. If your `.vimrc` is 350 lines long, you need at a maximum 9 tries to find the offending line (in practise, this can often be further reduced, since often lines depend on each other).

If the problem does not occur, when only loading your `.vimrc` file, the error must be caused by a plugin or another runtime file (indent autoload or syntax script). Check the output of the `:scriptnames` command to see what files have been loaded and for each one try to disable each one by one and see which one triggers the bug. Often files that are loaded by vim, have a simple configuration variable to disable them, but you need to check inside each file separately.

You can also use the `-V` command line argument to get more debug information to analyze the problem:

```
$ vim -V2logfile
```

You can increase the value passed to the `-V` argument to get more debug information. By also specifying a logfile name, this makes sure, the debug messages don't appear on the screen and won't disturb you when trying to reproduce the problem.

For more information, read:

```
-u  
-U  
-N  
-V  
'verbose'  
:verbose  
:set-verbose
```

faq-2.6

2.6. Where can I report bugs?

First collect the required information using the following command:

```
:source $VIMRUNTIME/bugreport.vim
```

Now send the resulting text from the above command to the `bugs@vim.org` e-mail address. There is also a public bug tracker available at

<https://github.com/vim/vim/issues>. A copy of each message there will be forwarded to the Vim Development list.

The Vim Development mailing list (see Question 2.2 [faq-2.2](#)) is a good place to discuss general bugs. If the bug you find is with syntax highlighting, a runtime file, or some other "added feature" (i.e. not directly programmed into vim), attempt to inform the maintainer of that feature. His e-mail address will be mentioned at the top of the corresponding runtime file.

For more information, read:

[bug-reports](#)

[faq-2.7](#)

2.7. Where can the FAQ be found?

The FAQ can be found at https://vimhelp.org/vim_faq.txt.html. It will be auto-generated from the source that is managed in the github repository https://www.github.com/chrisbra/vim_faq (Patches are welcome).

The repository also includes the faq in different formats, e.g. manpage, pdf file, html file, plain text version and a version in vim help format.

A slightly older version (which doesn't seem to get updated anymore) can still be found at VimOnline (<https://www.vim.org/>).

[faq-2.8](#)

2.8. What if I don't find an answer in this FAQ?

This FAQ covers mainly Vim-specific questions. You may find more information suitable for most Vi clones by reading the Vi FAQ. It is posted regularly on comp.editors. You can also find a copy at

<http://www.faqs.org/faqs/editor-faq/vi/part1/index.html>

Also, since Vim has gathered so many features in the last few years, successfully documenting the frequently asked questions here is a near-impossible task. To make it possible, please email the maintainer if you have a good question. A good question is one that you've tried to answer yourself (remember, Vim has great documentation) but struggled.

[faq-2.9](#)

2.9. I have a patch for implementing a Vim feature. Where can I send this patch?

You can send your patches to the Vim developer mailing list vim-dev@vim.org.

For more information, read:

[vim-dev](#)

[faq-2.10](#)

2.10. I have a Vim tip or developed a new Vim syntax/indent/filetype/compiler plugin or developed a new script or a colorscheme. Is there a public website where I can upload this?

Yes. You can use the Vim Online website to upload your plugins/scripts, colorschemes, etc. The site is at <https://www.vim.org>
Nowadays people also seem to share their plugins/runtime files at github.

Tips can also be shared in the Wiki which you can find at

<http://vim.wikia.com>

faq-3

SECTION 3 - AVAILABILITY

faq-3.1

3.1. What is the latest version of Vim?

The latest version of Vim is 8.2 released on 12th December 2019.

The release history of different versions of Vim is below:

VERSION	RELEASE DATE
Version 8.2	12th December, 2019
Version 8.1	17th May, 2018
Version 8.0	12th September, 2016
Version 7.4	10th August, 2013
Version 7.3	15th August, 2010
Version 7.2	9th August, 2008
Version 7.1	12th May, 2007
Version 7.0	8th May, 2006
Version 6.4	15th October, 2005
Version 6.3	8th June, 2004
Version 6.2	1st June, 2003
Version 6.1	24th March, 2002
Version 6.0	27th September, 2001
Version 5.8	31st May, 2001
Version 5.7	24th June, 2000
Version 5.6	16th January, 2000
Version 5.5	21st September, 1999
Version 5.4	26th July, 1999
Version 5.3	31st August, 1998
Version 5.2	24th August, 1998
Version 5.1	7th April, 1998
Version 5.0	19th February, 1998
Version 4.6	13th March, 1997
Version 4.5	17th October, 1996
Version 4.2	5th July, 1996
Version 4.0	29th May, 1996
Version 3.0	12th August, 1994
Version 2.0	21st December, 1993

Version 1.27 23rd April, 1993
Version 1.17 21st April, 1992
Version 1.14 2nd November, 1991

If you are interested in the old release history, check out the vim-history git repository:

<https://github.com/vim/vim-history>

and especially for the release history:

<https://github.com/vim/vim-history#release-history>

For more information, read:

[new-8](#)
[new-7](#)
[new-6](#)
[new-5](#)
[chnaged-8.2](#)
[changed-8.1](#)
[changed-7.4](#)
[changed-7.3](#)
[changed-7.2](#)
[changed-7.1](#)

faq-3.2

3.2. Where can I find the latest version of Vim?

You can download the sources for the latest version of Vim from the Github repository. The URL for this site is

<https://github.com/vim/vim>

A mercurial mirror is also available:

<https://bitbucket.org/vim-mirror/vim>
<http://hg.256bit.org/vim/>
<https://hg.osdn.net/view/vim/vim>

Some users keep updated repositories for distributing latest binary versions of Vim. You can find those repositories here:

http://vim.wikia.com/wiki/Where_to_download_Vim

faq-3.3

3.3. What platforms does it run on?

All Unix platforms.

All Windows platforms (XP and later).

Amiga, Atari, BeOS, Macintosh, MachTen, OS/2, RiscOS, VMS, IBM z/OS.

For MS-DOS support has been removed with the latest releases of Vim.

16-bit DOS: latest supported version 7.1

32-bit DOS: latest supported version 7.3

faq-3.4

3.4. Where can I download the latest version of the Vim runtime files?

You can download the latest version of the Vim runtime files (syntax files, filetype plugins, compiler files, color schemes, documentation, indentation files and keymaps) from the Vim github repository

<https://github.com/vim/vim/tree/master/runtime>

Another way of downloading the runtime files is this:

```
wget https://github.com/vim/vim/archive/master.tar.gz -O- |
tar zfx - vim-master/runtime/ --strip-components=1
```

See also:

<https://www.vim.org/runtime.php>

SECTION 4 - HELP

faq-4

4.1. How do I use the help files?

faq-4.1

Help can be found for all functions of Vim. In order to use it, use the `:help` command. This will bring you to the main help page. On that first page, you will find explanations on how to move around. Basically, you move around in the help pages the same way you would in a read-only document. You can jump to specific subjects by using tags. This can be done in two ways:

- * Use the **CTRL-]** command while standing on the name of a command or option. This only works when the tag is a keyword. `<Ctrl-LeftMouse>` and `g<LeftMouse>` work just like **CTRL-]**.
- * use the `:tag <subject>` command. This works with all characters.

Use **CTRL-T** to jump back to previous positions in the help files. Use `:q` to close the help window.

If you want to jump to a specific subject on the help pages, use `:help {subject}`. If you don't know what to look for, try `:help index` to get a list of all available subjects. Use the standard search keys to locate the information you want.

You can abbreviate the `:help` command as `:h`.

For searching the help, see the next Question 4.2. [faq-4.2](#)

For more information, read:

[online-help](#)

faq-4.2

4.2. How do I search for a keyword in the Vim help files?

- a) You can press the **CTRL-D** key after typing the help keyword to get a list of all the help keywords containing the supplied pattern. You can also use the meta characters like *, \+, etc to specify the help search pattern:

```
:help init<C-D>
:help str*(<C-D>
:help '*indent<C-D>
```

- b) You can press the <Tab> key after typing a partial help keyword to expand to the matching keyword. You can continue to press the <Tab> key to see other keyword matches.
- c) From the help window, you can use the `:tag` command to search for keywords. For example,

```
:tselect /window
```

This command will list all the help keywords containing the text "window". You can select one from the list and jump to it.

- d) You can use the `:helpgrep` command to search for the given text in all the help files. The quickfix window will be opened with all the matching lines.

For more information, read:

```
help-summary
c_CTRL-D
c_<Tab>
:tselect
:help
:helpgrep
```

[faq-4.3](#)

- 4.3. I am getting an error message E123, what did I do wrong?

You can get more information about the error and the error message using:

```
E123
```

For more information, read:

```
error-messages
```

[faq-4.4](#)

- 4.4. Where can I read about the various modes in Vim?

You can get information about the different modes in Vim by reading

```
vim-modes
```

[faq-4.5](#)

4.5. How do I generate the Vim help tags file after adding a new Vim help file?

You can use the `:helptags` command to regenerate the Vim help tag file from within Vim. For example:

```
:cd $VIMRUNTIME/doc
:helptags .
```

To update all "doc" directories in your `'runtimepath'`, you can use

```
:helptags ALL
```

For more information, read:

```
:helptags
add-local-help
```

faq-4.6

4.6. Can I use compressed versions of the help files?

Yes. You can compress the help files and still be able to view them with Vim. This makes accessing the help files a bit slower and requires the "gzip" utility. Follow these steps to compress and use the Vim help files:

- Compress all the help files using "gzip doc/*.txt".
- Edit the "doc/tags" file and change the ".txt" to ".txt.gz" using

```
:%s=\(t.*\.txt\)t=\1.gz\1t=
```
- Add the following line to your vimrc:

```
set helpfile={dirname}/help.txt.gz
```

Where `{dirname}` is the directory where the help files are. The `gzip.vim` plugin supplied with the standard Vim distribution will take care of decompressing the files. You must make sure that `$VIMRUNTIME` is set to where the other Vim files are, when they are not in the same location as the compressed "doc" directory.

Note, that the `:helpgrep` command does not work with compressed help pages.

For more information, read:

```
gzip-helpfile
'helpfile'
gzip
$VIMRUNTIME
```

=====

SECTION 5 - EDITING A FILE

faq-5

5.1. How do I load a file in Vim for editing?

faq-5.1

There are several ways to load a file for editing. The simplest is to use the `:e` (:edit) command:

```
:e <filename>
```

You can also use the `:n` (:next) command to load files into Vim:

```
:n <filename(s)>
```

You can also use the `:args` command to load files into Vim:

```
:args <filename(s)>
```

For more information, read:

```
usr_07.txt  
edit-files  
:edit  
:next_f  
:args_f
```

[faq-5.2](#)

5.2. How do I save the current file in another name (save as) and edit a new file?

You can use the `:saveas` command to save the current file in another name:

```
:saveas <newfilename>
```

Alternatively, you can also use the following commands:

```
:w <newfilename>  
:edit #
```

You can also use the `:file` command, followed by `:w` command:

```
:file <newfilename>  
:w
```

For more information, read:

```
07.7  
:saveas  
:file_f  
:w
```

[faq-5.3](#)

5.3. How do I change the current directory to the directory of the current file?

You can use the following command to change the current directory to the directory of the current file:

```
:cd %:p:h
```

To automatically change the current directory to the directory of the current file, simply set the option **'autochdir'**.

```
:set autochdir
```

For more information, read:

```
:cd
:lcd
filename-modifiers
autocommand
'acd'
getcwd()
```

[faq-5.4](#)

5.4. How do I write a file without the line feed (EOL) at the end of the file?

You can turn off the **'eol'** option and turn on the **'binary'** option to write a file without the EOL at the end of the file:

```
:set binary
:set noeol
:w
```

Alternatively, you can use:

```
:set noeol
:w ++bin
```

If you rather only like Vim not to write missing EOLs, you can reset the **'fixeol'** option. This needs a Vim newer 7.4.785, so you should wrap this in an if condition in your `.vimrc` like this:

```
if exists('+fixeol')
    set nofixeol
endif
```

This has the advantage of avoiding the many side effects that the **'binary'** option has.

For more information, read:

```
'endofline'
'fixeol'
'binary'
23.4
```

[faq-5.5](#)

5.5. How do I configure Vim to open a file at the last edited location?

Vim stores the cursor position of the last edited location for each buffer

in the `''` register. You can use the following autocmd in your `.vimrc` or `.gvimrc` file to open a file at the last edited location:

```
au BufReadPost * if line("\'") > 0 && line("\'") <= line("$") |
    \ exe "normal! g\'" | endif
```

Alternatively, you can simply source the `vimrc_example.vim` file, which is distributed with Vim.

For more information, read:

```
'quote
last-position-jump
vimrc_example.vim
```

[faq-5.6](#)

5.6. When editing a file in Vim, which is being changed by an external application, Vim opens a warning window (like the confirm dialog) each time a change is detected. How do I disable this warning?

You can set the Vim **'autoread'** option to automatically read the file again when it is changed outside of Vim:

```
:set autoread
```

You can also use the following autocommand:

```
autocmd FileChangedShell *
    \ echohl WarningMsg |
    \ echo "File has been changed outside of vim." |
    \ echohl None
```

For more information, read:

```
'autoread'
FileChangedShell
timestamp
:checktime
```

[faq-5.7](#)

5.7. How do I edit a file whose name is under the cursor?

You can use the `gf` command to edit a file whose name is under the cursor. You can use the **CTRL-W** `f` command to edit the file in a new window and finally you can use **CTRL-W** `gf` to open a new tab page that contains the file name under the cursor.

For more information, read:

```
gf
CTRL-W_f
CTRL-W_gf
'isfname'
'path'
```



```
'suffixesadd'  
'includeexpr'
```

faq-5.8

5.8. How do I reload/re-edit the current file?

You can use the `:edit` command, without specifying a file name, to reload the current file. If you have made modifications to the file, you can use `:edit!` to force the reload of the current file (you will lose your modifications, but depending on your `'undoreload'` settings, those changes might be saved into the undo history).

For more information, read:

```
:edit  
:edit!  
'confirm'  
'undoreload'
```

faq-5.9

5.9. How do I autosave a file periodically?

Vim doesn't support auto-saving a file periodically.

For more information, read:

```
'updatetime'  
CursorHold  
swap-file
```

faq-5.10

5.10. How do I open a file in read-only mode?

You can open a file in read-only mode using the `:view` command:

```
:view <filename>
```

This command sets the `'readonly'` option for the opened buffer. You can also use the `"-R"` command-line option to open a file in read-only mode:

```
$ vim -R <filename>
```

You can also use the symbolic link executable `"view"` to open a file in read-only mode from the command-line:

```
$ view <filename>
```

For more information, read:

```
07.6  
'readonly'  
'modifiable'  
:view  
:sview
```

```
view
-R
-M
```

faq-5.11

5.11. How do I open a file for editing without saving the modifications to the current file?

You can open a file for editing without saving the modifications to the current file and without losing the changes using one of the following methods:

```
:split <new_filename>
:new <new_filename>
```

You can also set the **'hidden'** option and edit a new file:

```
:set hidden
:e <new_filename>
```

If you want to discard the changes made to the current file and load another file for editing, then you can use the following command:

```
:e! <new_filename>
```

For more information, read:

```
:edit!_f
'hidden'
:split
:new
```

faq-5.12

5.12. How do I reduce the loading time for very large files in Vim?

You can use the following settings to reduce the loading time for very large files in Vim:

```
:set lazyredraw
:set noswapfile
:set undolevels=-1
:set eventignore=all
:set nohidden
:set syntax=off
```

Note that the above settings will disable many Vim features including the following: Swap files support for crash recovery, undo support, syntax highlighting, filetype detection and other autocommand based features.

There is also the LargeFile plugin available at

https://www.vim.org/scripts/script.php?script_id=1506

which automatically sets these options, when working with large files

(it is configurable, what is considered to be a large file, by default, it is 100MB).

SECTION 6 - EDITING MULTIPLE FILES

faq-6

faq-6.1

6.1. How do I open multiple files at once from within Vim?

Make a difference between args, buffers, tabs and windows. They are all different things in VIM.

args is a list of arguments. Buffers are place to edit text, almost always attached to a file but not necessarily. Window is a place for a buffer and tab is set of windows, better name would be "layout".

There are several ways to open multiple files at once from within Vim. You can use the `:next` command to specify a group of files:

```
:next f1.txt f2.txt
:next *.c
```

You can use the `:args` command to specify a group of files as arguments:

```
:args f1.txt f2.txt
:args *.c
```

After loading the files, you can use the `:next` and `:prev` command to switch between the files.

To execute command for all files in argument-list use `:argdo`

For more information, read:

```
07.2
:next
:args_f
argument-list
```

faq-6.2

6.2. How do I switch between multiple files/buffers in Vim?

To list all buffers use `:ls`, to list buffers without file attached to (also known as unlisted buffers, ex. scratch buffer and help-window) use `:ls!`

There are several ways to switch between multiple files. You can use the `:buffer` command to switch between multiple files. You can also shorten command as `:b` and use only part of the filename. For example,

```
:buffer file1
:buffer file2
:b e2
```

You can also use `<TAB>` after ``:b`` for autocompletion. Try also ``:b`` followed by **CTRL-D** to see all available buffers. This works also for ``:e``.

You can also use the **CTRL-^** key to switch between buffers. By specifying a count before pressing the key, you can edit the buffer with that number. Without the count, you can edit the alternate buffer by pressing **CTRL-^**

You can also use the ``:e #`` command to edit a particular buffer:

```
:e #5
```

To close a buffer use ``:bd`` -command.

To execute command for all files in buffer-list use ``:bufdo``

For more information, read:

```
edit-files
:buffer
CTRL-^
alternate-file
22.4
07.3
```

faq-6.3

6.3. How do I open several files in Vim, with each file in a separate window/tab?

You can use the `-o` and `-O` Vim command line arguments to open multiple files in separate horizontally or vertically split Vim windows. For example:

```
$ vim -o3 f1.txt f2.txt f3.txt
```

The above command will open the files `f1.txt`, `f2.txt` and `f3.txt` in three separate horizontally split Vim windows.

```
$ vim -O3 f1.txt f2.txt f3.txt
```

The above command will open the files `f1.txt`, `f2.txt` and `f3.txt` in three separate vertically split Vim windows.

```
$ vim -p f1.txt f2.txt f3.txt
```

The above command will open the files `f1.txt`, `f2.txt` and `f3.txt` in three separate tab windows. The option `'tabpagemax'` defines, how many tabpages can be opened at the same time, by default it is set to 10.

For more information, read:

```
-o
-O
-p
```

startup-options
'tabpagemax'

faq-6.4

6.4. How do I configure Vim to autoload several files at once similar to "work-sets" or "projects"?

You can use the `:mksession` and the `:mkview` commands to autoload several files in Vim.

The `:mksession` command creates a Vim script that restores the current editing session. You can use the `:source` command to source the file produced by the `mksession` command.

The `:mkview` command creates a Vim script that restores the contents of the current window. You can use the `:loadview` command to load the view for the current file.

For more information, read:

21.4
21.5
views-sessions
'sessionoptions'
:mksession
:source
v:this_session
:mkview
:loadview
'viewdir'
buffers

faq-6.5

6.5. Is it possible to open multiple top level windows in a single instance of Vim similar to Nedit or Emacs?

No. It is currently not possible to open multiple top-level windows in a single instance of Vim. This feature is in the todo list.

faq-6.6

6.6. How do I browse/explore directories from within Vim?

You can use the `netrw.vim` plugin, supplied with the standard Vim installation, to browse/explore directories from within Vim. You can start the file explorer using one of the following commands:

```
:e <directory>  
:Explore  
:Sexplore  
:Vexplore  
:Texplore
```

From the file explorer, you can browse through directories, rename, delete and edit files.

For more information, read:

[netrw.vim](#)
22.1

[faq-6.7](#)

6.7. How do I edit files over a network using ftp/scp/rcp/http?

You can use the netrw.vim plugin, supplied with the standard Vim package, to edit files over a network using ftp/scp/rcp/http. Using this plugin, Vim will transparently load and save the files over ftp/scp/rcp/http. For example, to edit a file over ftp, you can use the following command:

```
$ vim ftp://machine/path
```

For more information, read:

[netrw.vim](#)

=====

[faq-7](#)

SECTION 7 - BACKUP

[faq-7.1](#)

7.1. When I edit and save files, Vim creates a file with the same name as the original file and a "~" character at the end. How do I stop Vim from creating this file? (or) How do I disable the Vim backup file feature?

You have set the '**backup**' option, so Vim creates a backup file when saving the original file. You can stop Vim from creating the backup file, by clearing the option:

```
:set nobackup
```

Note that, by default this option is turned off. You have explicitly enabled the '**backup**' option in one of the initialization files. You may also have to turn off the '**writebackup**' option:

```
:set nowritebackup
```

For more information, read:

07.4
[backup-table](#)
['backup'](#)
['writebackup'](#)
['backupskip'](#)
['backupdir'](#)
['backupext'](#)
['backupcopy'](#)
[backup](#)

faq-7.2

7.2. When I edit and save files, Vim creates a file with the same name as the original file and a "un~" extension at the end. How do I stop Vim from creating this file (or) How do I disable the Vim undofile feature?

Vim 7.3 contains as new feature persistent undo, that is, undo information won't be lost when quitting Vim and be stored in a file that ends with ".un~" You have set the **'undofile'** option, so Vim creates an undo file when saving the original file. You can stop Vim from creating the backup file, by clearing the option:

```
:set noundofile
```

Note that, by default this option is turned off. You have explicitly enabled the **'undofile'** option in one of the initialization files. If you want your undofiles to be stored only in a particular directory, you can point the **'undodir'** option to a directory that will contain all your aggregated undofiles.

For more information, read:

```
'undodir'  
'undofile'  
undo-persistence
```

faq-7.3

7.3. How do I configure Vim to store all the backup files in a particular directory?

You can configure Vim to store all the backup files in a particular directory using the **'backupdir'** option. For example, to store all the backup files in the ~/backup directory, you can use the following command:

```
:set backupdir=~/.backup
```

For more information, read:

```
07.4  
'backupdir'  
backup
```

faq-7.4

7.4. When I save a file with Vim, the file permissions are changed. How do I configure Vim to save a file without changing the file permissions?

This may happen, if the **'backupcopy'** option is set to "no" or "auto". **Note** that the default value for this option is set in such a way that this will correctly work in most of the cases. If the default doesn't work for you, try setting the **'backupcopy'** option to "yes" to keep the file permission when saving a file:

```
:set backupcopy=yes
```

This applies, only if you have configured Vim to make a backup whenever overwriting a file. By default, Vim will not backup files.

For more information, read:

```
'backupcopy'  
backup  
'backup'  
'writebackup'
```

faq-8

SECTION 8 - BUFFERS

faq-8.1

8.1. I have made some modifications to a buffer. How do I edit another buffer without saving the modified buffer and also without losing the modifications?

You can set the **'hidden'** option to edit a file without losing modifications to the current file:

```
:set hidden
```

By setting the **'hidden'** option, you can also save the modification history (undo-history) for the buffer. Otherwise, as you switch between files, the undo-history will be lost (unless you use persistent undofiles).

For more information, read:

```
'hidden'  
hidden-quit  
:hide
```

faq-8.2

8.2. How do I configure Vim to auto-save a modified buffer when switching to another buffer?

You can set the **'autowrite'** option to auto-save a modified buffer when switching to another buffer:

```
:set autowrite
```

For more information, read:

```
'autowrite'  
'autowriteall'  
'hidden'
```

faq-8.3

8.3. How do I replace the buffer in the current window with a blank buffer?

You can use the ``:enew`` command to load an empty buffer in place of the buffer in the current window.

For more information, read:

```
:enew
```

faq-8.4

8.4. Is there a keyboard shortcut to load a buffer by the buffer number?

You can use the **CTRL-^** command to load a buffer by specifying the buffer number. For example, to load buffer number 5, you have to use the 5 **CTRL-^** command.

For more information, read:

```
CTRL-^
```

faq-8.5

8.5. How do I open all the current buffers in separate windows?

You can use the `:ball` or `:sball` commands to open all the buffers in the buffer list:

```
:ball
```

If you want all buffers to be opened in new tabs, simply prefix the `:tab` command:

```
:tab :sball
```

For more information, read:

```
:ball
```

faq-8.6

8.6. How do I close (delete) a buffer without exiting Vim?

You can use any of `:bdelete`, `:bwipeout` or `:bunload` commands to delete a buffer without exiting Vim. For example:

```
:bdelete file1
```

For more information, read:

```
:bdelete  
:bwipeout  
:bunload
```

faq-8.7

8.7. When I use the command `:%bd` to delete all the buffers, not all the buffers are deleted. Why?

In the `:%bd` command, the "%" range will be replaced with the starting and ending line numbers in the current buffer. Instead of using "%" as the range, you should specify numbers for the range. For example, to delete all

the buffers, you can use the command ``:1,9999bd``.

For more information, read:

```
:bd
```

(This behaviour has been fixed with patch 7.4.530)

faq-8.8

8.8. How do I display the buffer number of the current buffer/file?

You can use `2<CTRL-G>` command to display the buffer number for the current file/buffer. **Note** the use of count before the **CTRL-G** command. If the count is greater than 1, then Vim will display the buffer number.

You can also use the following command to display the current buffer number:

```
:echo bufnr("%")
```

You can also include the `"%n"` field to the `'statusline'` option to display the current buffer number on the statusline.

For more information, read:

```
CTRL-G  
bufnr()  
:echo  
'statusline'
```

faq-8.9

8.9. How do I delete a buffer without closing the window in which the buffer is displayed?

You can use the following command to open the next buffer and delete the current buffer.

```
:bnext | bdelete #
```

For more information, read:

```
:bnext  
:bdelete  
:buffers
```

faq-8.10

8.10. How do I map the `<Tab>` key to cycle through and open all the buffers?

You can use the following two map commands, to map the `<Ctrl-Tab>` key to open the next buffer and the `<Ctrl-Shift-Tab>` key to open the previous buffer:

```
:nnoremap <C-Tab> :bnext<CR>  
:nnoremap <S-C-Tab> :bprevious<CR>
```

Note, this might not work in the terminal version of Vim.

For more information, read:

```
:bnext  
:bprevious
```

SECTION 9 - WINDOWS

faq-9

faq-9.1

9.1. What is the difference between a Vim window and a buffer?

A Vim buffer is a file loaded into memory for editing. The original file remains unchanged until you write the buffer to the file. A Vim window is a viewport onto a buffer. You can use multiple windows on one buffer or several windows on different buffers.

For more information, read:

```
usr_08.txt  
22.4  
windows-intro  
Q_wi
```

faq-9.2

9.2. How do I increase the width of a Vim window?

You can increase the width of a Vim window using one of the following commands:

```
:vert resize +N  
:vert resize -N  
:vert resize N
```

You can also use **CTRL-W <** or **CTRL-W >** or **CTRL-W |** commands.

For more information, read:

```
:vertical-resize  
CTRL-W_>  
CTRL-W_<  
window-resize
```

faq-9.3

9.3. How do I zoom into or out of a window?

You can zoom into a window (close all the windows except the current window) using the "**<CTRL-W>o**" command or the `:only`` ex command.

You can use the "**<CTRL-W>_**" command or the `:resize`` ex command to increase the current window height to the highest possible without closing other windows.

You can use the "`<CTRL-W>|`" command or the `:vertical resize`` ex command to increase the current window width to the highest possible without closing other windows.

You can use the "`<CTRL-W>=`" command to make the height and width of all the windows equal.

You can also set the following options to get better results with the above commands:

Method 1:

Set the `'winminheight'` option to 0:

```
:set winminheight=0
```

By default, this option is set to 1.

This option controls the minimum height of an inactive window (when it is not the current window). When the `'winminheight'` option is set to 0, only the status line will be displayed for inactive windows.

Method 2:

Set the `'noequalalways'` option and set the `'winheight'` option to a large value (like 99999):

```
:set noequalalways  
:set winheight=99999
```

Now, the active window will always open to its maximum size, while the other windows will stay present, but shrunken to just a status line.

With any of the above mentioned methods, you cannot restore the window layout after zooming into a window. If you want to restore the Vim window layout after zooming into a window, you can use the ZoomWin plugin. You can download this plugin from the Vim online website at:

https://www.vim.org/scripts/script.php?script_id=508

For more information, read:

```
CTRL-W_o  
window-resize  
'winminheight'  
'equalalways'  
'winheight'  
08.3
```

[faq-9.4](#)

9.4. How do I execute an ex command on all the open buffers or open windows or all the files in the argument list?

You can use the `:bufdo`` command to execute an ex command on all the open buffers. You can use the `:windo`` command to execute an ex command on all the open windows. You can use the `:argdo`` command to execute an ex

command on all the files specified in the argument list. And finally you can use the `:tabdo` command to execute an ex command in all open tab pages.

For more information, read:

```
:windo
:bufdo
:argdo
:tabdo
26.3
```

faq-10

SECTION 10 - MOTION

faq-10.1

10.1. How do I jump to the beginning (first line) or end (last line) of a file?

You can use "G" command to jump to the last line in the file and the "gg" command to jump to the first line in the file.

For more information, read:

```
G
gg
```

faq-10.2

10.2. In insert mode, when I press the `<Esc>` key to go to command mode, the cursor moves one character to the left (except when the cursor is on the first character of the line). Is it possible to change this behavior to keep the cursor at the same column?

No. It is not possible to change this behavior. The cursor is always positioned on a valid character (unless you have virtual-edit mode enabled). So, if you are appending text to the end of a line, when you return to command mode the cursor **must** drop back onto the last character you typed. For consistency sake, the cursor drops back everywhere, even if you are in the middle of a line.

You can use the **CTRL-O** or `CTRL-\ CTRL-O` command in insert mode to execute a single ex command and return back to insert mode without moving the cursor column.

For more information, read:

```
'virtualedit'
i_CTRL-O
i_CTRL-\_CTRL-O
```

faq-10.3

10.3. How do I configure Vim to maintain the horizontal cursor position when scrolling with the `<Page Up>`, `<Page Down>`, etc keys?

You can reset the **'startofline'** option to keep the cursor at the same horizontal location when scrolling text:

```
:set nostartofline
```

For more information, read:

```
'startofline'
```

faq-10.4

10.4. Some lines in a file are more than the screen width and they are all wrapped. When I use the j, k keys to move from one line to the next, the cursor is moved to the next line in the file instead of the next line on the screen. How do I move from one screen line to the next?

You can use the gj and gk commands to move from one screen line to the next/previous screen line. The j and k commands move the cursor from one file line to the next file line. You can also avoid the line wrapping by resetting the **'wrap'** option:

```
:set nowrap
```

For more information, read:

```
gj  
gk  
'wrap'
```

You can use the following mappings:

```
:map <Up> gk  
:imap <Up> <C-o>gk  
:map <Down> gj  
:imap <Down> <C-o>gj  
:noremap j gj  
:noremap k gk
```

faq-10.5

10.5. What is the definition of a sentence, paragraph and section in Vim?

A sentence is defined as ending at a ".", "!" or "?" followed by either the end of a line, or by a space (or two) or tab. Which characters and the number of spaces needed to constitute a sentence ending is determined by the **'joinspaces'** and **'coptions'** options.

A paragraph begins after each empty line, and also at each of a set of paragraph macros, specified by the pairs of characters in the **'paragraphs'** option.

A section begins after a form-feed (<C-L>) in the first column and at each of a set of section macros, specified by the pairs of characters in the **'sections'** option.

For more information, read:

sentence
'joinspaces'
'cptions'
paragraph
section
word

faq-10.6

10.6. How do I jump to beginning or end of a sentence, paragraph or a section?

You can use the following motion commands to jump to the beginning or end of a sentence or a paragraph or a section:

<u>motion</u>	<u>position</u>	<u>where</u>
(beginning	current sentence
)	end	current sentence
{	beginning	current paragraph
}	end	current paragraph
[]	end	previous section
[[beginning	current section
][end	current section
]]	beginning	next section

Each of these motions can be preceded by a number which will extend the jump forward (or backward).

For more information, read:

[object-motions](#)

faq-10.7

10.7. I have lines in a file that extends beyond the right extent of the screen. How do I move the Vim view to the right to see the text off the screen?

You can use one of the following commands to horizontally scroll the screen to the left or right:

<u>cmd</u>	<u>scroll to</u>
zl	scroll to the left
zh	scroll to the right
zL	scroll half a screenwidth to the left
zH	scroll half a screenwidth to the right
zs	scroll to position the cursor at the start of the screen
ze	scroll to position the cursor at the end of the screen

You can use the `g0` command to move the cursor to the first character of the screen line and the `g$` command to move the cursor to the last character of the screen line without scrolling the screen.

For more information, read:

[scroll-horizontal](#)

[faq-10.8](#)

10.8. How do I scroll two or more buffers simultaneously?

You can set the **'scrollbind'** option for each of the buffers to scroll them simultaneously.

For more information, read:

['scrollbind'](#)
[scroll-binding](#)
['scrollopt'](#)
['cursorbind'](#)

[faq-10.9](#)

10.9. When I use my arrow keys, Vim changes modes, inserts weird characters in my document but doesn't move the cursor properly. What's going on?

There are a couple of things that could be going on: either you are using Vim over a slow connection or Vim doesn't understand the key sequence that your keyboard is generating.

If you are working over a slow connection (such as a 2400 bps modem), you can try to set the **'timeout'** or **'ttimeout'** option. These options, combined with the **'timeoutlen'** and **'ttimeoutlen'** options, may fix the problem.

The preceding procedure will not work correctly if your terminal sends key codes that Vim does not understand. In this situation, your best option is to map your key sequence to a matching cursor movement command and save these mappings in a file. You can then `:source` the file whenever you work from that terminal.

For more information, read:

['timeout'](#)
['ttimeout'](#)
['timeoutlen'](#)
['ttimeoutlen'](#)
[:map](#)
[vt100-cursor-keys](#)

[faq-10.10](#)

10.10. How do I configure Vim to move the cursor to the end of the previous line, when the left arrow key is pressed and the cursor is currently at the beginning of a line?

You can add the "<" flag to the **'whichwrap'** option to configure Vim to move the cursor to the end of the previous line, when the left arrow key is pressed and the cursor is currently at the beginning of a line:

```
:set whichwrap+=<
```


Similarly, to move the cursor the beginning of the next line, when the right arrow key is pressed and the cursor is currently at the end of a line, add the ">" flag to the **'whichwrap'** option:

```
:set whichwrap+=>
```

The above will work only in normal and visual modes. To use this in insert and replace modes, add the "[" and "]" flags respectively.

For more information, read:

```
'whichwrap'  
05.7
```

[faq-10.11](#)

10.11. How do I configure Vim to stay only in insert mode (modeless editing)?

You can set the **'insertmode'** option to configure Vim to stay only in insert mode:

```
:set insertmode
```

By setting this option, you can use Vim as a modeless editor. If you press the <Esc> key, Vim will not go to the normal mode. To execute a single normal mode command, you can press **CTRL-O** followed by the normal mode command. To execute more than one normal command, you can use **CTRL-L** followed by the commands. To return to insert mode, press the <Esc> key. To disable this option, reset the **'insertmode'** option:

```
:set noinsertmode
```

You can also start vim using the "evim" command or you can use "vim -y" to use Vim as a modeless editor.

You can also start Vim in insert mode using the ``:startinsert`` ex command.

For more information, read:

```
'insertmode'  
:startinsert  
:stopinsert  
i_CTRL-O  
i_CTRL-L  
evim  
evim-keys
```

[faq-10.12](#)

10.12. How do I display some context lines when scrolling text?

You can set the **'scrolloff'** option to display a minimal number of screen lines (context) above and below the cursor.

```
:set scrolloff=10
```

For more information, read:

```
'scrolloff'  
'sidescrolloff'
```

faq-10.13

10.13. How do I go back to previous cursor locations?

You can go back to the cursor location before the latest jump using the `'` or ``` command. You can use the **CTRL-O** command to go back to older cursor positions and the **CTRL-I** command to go to the newer cursor positions in the jump list.

For more information, read:

```
03.10  
mark-motions  
jump-motions
```

=====

faq-11

SECTION 11 - SEARCHING TEXT

faq-11.1

11.1. After I searched for a text with a pattern, all the matched text stays highlighted. How do I turn off the highlighting temporarily/permanently?

The **'hlsearch'** option controls whether all the matches for the last searched pattern are highlighted or not. By default, this option is not enabled. If this option is set in a system-wide vimrc file, then you can turn off the search highlighting by using the following command:

```
:set nohlsearch
```

To temporarily turn off the search highlighting, use

```
:nohlsearch
```

You can also clear the search highlighting, by searching for a pattern that is not in the current file (for example, search for the pattern "asdf").

You can use this mapping, to clear the search highlighting when redrawing the window pressing **CTRL-L**

```
:nnoremap <silent> <C-L> <C-L>:nohls<CR>
```

For more information, read:

```
'hlsearch'  
:nohlsearch
```

faq-11.2

11.2. How do I enter a carriage return character in a search pattern?

You can either use "\r" or <CTRL-V><CTRL-M> to enter a carriage return character in a pattern. In Vim scripts, it is better to use "\r" for the carriage return character.

For more information, read:

[sub-replace-special](#)

faq-11.3

11.3. How do I search for the character ^M?

You can enter the ^M character in a search command by first pressing the **CTRL-V** key and then pressing the **CTRL-M** key.

```
/^V^M
```

You can also use the "\r" character. In Vim scripts, "\r" is preferred.

For more information, read:

```
c_CTRL-V
using_CTRL-V
/\r
```

faq-11.4

11.4. How can I search/replace characters that display as "~R", "~S", etc.?

You can use the "ga" command to display the ASCII value/code for the special character. For example, let us say the ASCII value is 142. Then you can use the following command to search for the special character:

```
/^V142
```

where, ^V is entered by pressing **CTRL-V**.

For more information, read:

```
ga
using_CTRL_V
24.8
```

faq-11.5

11.5. How do I highlight all the non-printable characters in a file?

You can use the following commands and search pattern to highlight all the non-printable characters in a file:

```
:set hlsearch
/\(\p\|\$\)\@!.
```

For more information, read:

```
/\p
/bar
/$
/\(
/\@!
'hlsearch'
```

faq-11.6

11.6. How do I search for whole words in a file?

You can search for whole words in a file using the \< and \> atoms. For example:

```
/\
```

The \< atom matches the beginning of the word and the \> atom matches the end of the word.

For more information, read:

```
/\<
/\>
```

faq-11.7

11.7. How do I search for the current word under the cursor?

You can press the * key to search forward for the current word under the cursor. To search backward, you can press the # key. **Note** that only whole keywords will be searched using these commands.

For more information, read:

```
star
#
gstar
g#
03.8
search-commands
```

faq-11.8

11.8. How do I search for a word without regard to the case (uppercase or lowercase)?

To always ignore case while searching for a pattern, set the **'ignorecase'** option:

```
:set ignorecase
```

To ignore case only when searching a particular pattern, use the special \c directive:

```
/\c<pattern>
```

For more information, read:

```
'ignorecase'  
/ignorecase  
\c
```

faq-11.9

11.9. How do I search for words that occur twice consecutively?

You can use one of the following search commands to locate words that occur twice consecutively:

```
\/(\<\w\+)\_s\+1\  
\/(\<\k\+)\_s\+1\  
\/(\<\w\+)\_s\+1\  
\/(\<\k\+)\_s\+1
```

The main difference is the use of "\w" and "\k", where the latter is based on the '**iskeyword**' option which may include accented and other language specific characters.

For more information, read:

```
\/1  
\/(  
\/)  
\/<  
\/>  
\/w  
\/k  
\/+  
\/_x  
'iskeyword'
```

faq-11.10

11.10. How do I count the number of times a particular word occurs in a buffer?

You can use the following set of commands to count the number of times a particular word occurs in a buffer:

```
:let cnt=0  
:g/\<your_word\>/let cnt=cnt+1  
:echo cnt
```

This only counts the number of lines where the word occurs. You can also use the following command:

```
:%s/\<word\>/&/gn
```

To count the number of alphabetic words in a file, you can use

```
:%s/\a\+/&/gn
```

To count the number of words made up of non-space characters, you can use

```
:%s/\S\+/\&/gn
```

For more information, read:

```
count-items  
word-count  
v_g_CTRL-G  
12.5  
:s_flags
```

[faq-11.11](#)

11.11. How do I place the cursor at the end of the matched word when searching for a pattern?

You can use the "e" offset to the search command to place the cursor at the end of the matched word. For example

```
/mypattern/e
```

For more information about search offsets, read:

```
search-offset  
/
```

[faq-11.12](#)

11.12. How do I search for an empty line?

You can search for an empty line using:

```
/^$
```

or

```
/^\\s*$
```

The latter also matches lines, that consist only of white space, while the former only matches true empty lines. For more information, read:

```
/^  
/$  
/\\s  
/star  
search-commands
```

[faq-11.13](#)

11.13. How do I search for a line containing only a single character?

You can search for a line containing only a single character using:

```
/^\\s*\\a\\s*$
```

For more information, read:

```
/^
```

```
/\a
/>\s
/star
/$(
```

faq-11.14

11.14. How do I search and replace a string in multiple files?

You can use the `:argdo`, `:bufdo`, `:windo` or `:tabdo` commands to execute an ex command on multiple files. For example:

```
:argdo %s/foo/bar/g|upd
```

For more information, read:

```
:argdo
:bufdo
:windo
:tabdo
```

faq-11.15

11.15. I am using the `:s` substitute command in a mapping. When a search for a pattern fails, the map terminates. I would like the map to continue processing the next command, even if the substitute command fails. How do I do this?

You can use the "e" flag to the substitute command to continue processing other commands in a map, when a pattern is not found.

For more information, read:

```
:s_flags
```

faq-11.16

11.16. How do I search for the n-th occurrence of a character in a line?

To search for the n-th occurrence of a character in a line, you can prefix the "f" command with a number. For example, to search for the 5th occurrence of the character @ in a line, you can use the command 5f@. This assumes the cursor is at the beginning of the line - and that this first character is not the one you are looking for.

For more information, read:

```
f
F
t
T
;
,
```

faq-11.17

11.17. How do I replace a tab (or any other character) with a hard return (newline) character?

You can replace a tab (or any other character) with a hard return (newline) character using the following command:

```
:s/\t/\r/
```

Note that in the above command, if you use `\n` instead of `\r`, then the tab characters will not be replaced by a new-line character.

For more information, read:

```
sub-replace-special  
NL-used-for-Nul  
CR-used-for-NL
```

faq-11.18

11.18. How do I search for a character by its ASCII value?

You can search for a character by its ASCII value by pressing **CTRL-V** followed by the decimal or hexadecimal or octal value of that character in the search `/` command. To determine the ASCII value of a character you can use the `:ascii` or the `"ga"` command.

For example, to search for the ASCII character with value 188 (¼), you can use one of the following search commands:

```
/<CTRL-V>188  
/<CTRL-V>o274  
/<CTRL-V>xBC  
/<CTRL-V>u00bc
```

You can also search for the character with the decimal/octal/hex number using a collation `[]` like this:

```
/[\d188]  
/[\o274]  
/[\xbc]  
/[\u00bc]
```

Alternatively, you can use the special atom `\%d` `\%o` `\%x` `\%u`:

```
/\%d188  
/\%o274  
/\%xbc  
/\%u00bc
```

Or you use digraphs to enter the character. For example enter:

```
/CTRL-K14
```

to search for the above character.

For more information, read:


```
i_CTRL-V_digit
:ascii
ga
/\]
/\%d
digraphs
```

faq-11.19

11.19. How do I search for long lines?

You can search for long lines or lines containing more than a specific number of characters using the Vim regular-expressions in the search command. For example, to search for all the lines containing more than 80 characters, you can use one of the following commands:

```
/^\.{80}.\+$
/^\.*\%>80c.*$
/^\.*\%>80v.*$
```

For more information, read:

```
/\{
/\%c
/\%v
```

faq-11.20

11.20. How do I display all the lines in the current buffer that contain a specified pattern?

You can use the following command to display all the lines in the current buffer that contain a specified pattern:

```
:g/<pattern>/p
```

For example, the following command will display all the lines in the current buffer that contain "vim":

```
:g/vim/p
```

Since `:p` is the default command to be executed for the ex command `:g`, you can also use:

```
:g/vim
```

If you also want the corresponding line numbers, then you can use the following command:

```
:g/<pattern>/#
```

For more information, read:

```
:global
:print
:number
```

faq-11.21

11.21. How do I search for a text string that spans multiple lines?

You can search for a text string that spans multiple lines using the `_x` regular expression atom. For example, to search for the text string "Hello World", you can use the following search command:

```
/Hello\_sWorld
```

This will match the word "Hello" followed by a newline character and then the word "World" at the beginning of the next line. This will also match the word "Hello" immediately followed by a space character and then the word "World". When searching for the "Hello World" string, to include the space characters at the end and beginning of the line, you can use the following search command:

```
/Hello\_s\+World
```

For more information, read:

```
27.8  
pattern-atoms  
/\_  
pattern-searches
```

faq-11.22

11.22. How do I search for a pattern within the specified range of lines in a buffer?

You can search for a pattern within a range of lines using the `\%>l` and `\%<l` regular expression atoms.

For example, to search for the word "white" between the lines 10 and 30 in a buffer, you can use the following command:

```
/white\%>9l\%<31l
```

For more information, read:

```
/\%l
```

faq-11.23

11.23. How do I clear the last searched pattern?

The last searched pattern is stored in the `"/` register. You can clear this register using the following command:

```
:let @/=""
```

To clear the last search pattern whenever a buffer is unloaded, you can use the following command:

```
:autocmd BufUnload * let @/ = ""
```

For more information, read:

```
@/  
:let-@  
autocmd-searchpat  
last-pattern
```

faq-11.24

11.24. Why does this pattern "a.\{-}p\@" not match?

"\{-}" doesn't just mean "as few as possible", it means "as few as possible to make the whole pattern succeed". If it didn't match the "p", the whole pattern would fail (because of the "p\@!") so it does match the "p". It is a longer match, but it is the shortest match that makes the whole pattern succeed.

If you wanted "as few as possible regardless" you would use "\@>", which basically divides a pattern up so that the pieces either side behave independently. If the pattern were "a.\{-}\@>p\@" then ".\{-}" would always match nothing because that's the smallest match that can succeed when there are not other restrictions. The whole pattern then would behave the same as "ap\@", i.e. it would match any "a" not followed by a "p").

This means, it matches as few as possible "a"s without trying to keep going until Vim finds the longest match. This means, it will still match "ap".

faq-11.25

11.25. How can I use "/" within a pattern, without escaping it?

When using / to search for a pattern, you need to escape all "/" within the pattern, because they would otherwise terminate the pattern. So you can't directly search for /usr/share/doc/ but need to search for \usr\share\doc\

The easiest solution around that, would be to use "?" to start a backward search and afterwards use /<CR> to use the last search-pattern in forward direction.

If you have a Vim, that has the eval-feature built in (which needs at least a normal built or higher), you can also directly paste the pattern into the search register:

```
:let @/ = '/usr/share/doc/'
```

Then use "n" to jump to the next occurrence.

See also the help at

```
@/  
/<CR>
```

11.26. How can I operate on a search match?

The "gn" command makes it easy to operate on regions of text that match the current search pattern. By default, it will search forward for the last used search pattern and visually select the match. If the cursor is already on the match, it will be visually selected. If you used the "gn" command after an operator (e.g. "c" to change text), it will be applied on the match.

If Visual mode is active before using gn, the visual selection will be extended until the end of the next match.

The "gN" commands works similar but searches backwards.

This allows to repeat simple operations on each match. For example, you might want to change each occurrence of apples by peaches. So you search using "/apple" then you can use "cgnpeach<Esc>" to replace the current match by peach. Now you can use the dot "." command to redo the replacement for the rest of the buffer.

See also the help at

```
gn
gN
```

SECTION 12 - CHANGING TEXT

12.1. How do I delete all the trailing white space characters (SPACE and TAB) at the end of all the lines in a file?

You can use the `:substitute` command on the entire file to search and remove all the trailing white space characters:

```
:%s/\s\+$//
```

For more information, read:

```
:%
:s
/\s
/\+
/$
```

12.2. How do I replace all the occurrences of multiple consecutive space characters to a single space?

You can use the following command to replace all the occurrences of multiple consecutive space characters to a single space:

```
:%s/ \{2,\}/ /g
```

Alternatively use:

```
:%s/ \+/ /g
```

For more information, read:

```
:%  
:s  
/\{  
:s_flags
```

faq-12.3

12.3. How do I reduce a range of empty lines into one line only?

You can use the following command to reduce a range of empty lines into one line only:

```
:v/./.,/./-1join
```

The explanation for this command is below:

<u>part</u>	<u>description</u>
<code>:v/./</code>	Execute the following command for all lines not containing a character (empty lines).
<code>.,</code>	Use the current line as the start of the range of lines.
<code>/./</code>	Use the line containing a character as the last line.
<code>-1</code>	Adjust the range of lines to end with the line before the last line.
<code>j</code>	Join the lines in the range.

Note that this will give an error message if the empty lines are at the end of the file. To correct this, you have to add a temporary line at the end of the file, execute the command and then remove the temporary line.

For more information, read:

```
:v  
:join  
cmdline-ranges  
collapse
```

faq-12.4

12.4. How do I delete all blank lines in a file? How do I remove all the lines containing only space characters?

To remove all blank lines, use the following command:

```
:g/^\$/d
```

To remove all lines with only whitespace (spaces or tabs) in them, use the

following command:

```
:g/^\s\+$/d
```

To remove all the lines with only whitespace, if anything, use the following command:

```
:g/^\s*$/d
```

[faq-12.5](#)

12.5. How do I copy/yank the current word?

You can use the "yiw" (yank inner word without whitespace) command or the "yaw" (yank a word with whitespace) command to copy/yank the current word.

For more information, read:

```
04.6  
04.8  
iw  
yank  
text-objects  
objects
```

[faq-12.6](#)

12.6. How do I yank text from one position to another position within a line, without yanking the entire line?

You can specify a motion command with the yank operator (y) to yank text from one position to another position within a line. For example, to yank from the current cursor position till the next letter x, use yfx or Fx or tx or Tx. To yank till the nth column, use n|. To yank till the next occurrence of a "word", use /word. To do a yank till the nth column on another line, first mark the position using the "ma" command, go to the start of the yank position, and then yank till the mark using y`a (note the direction of the quote)

For more information, read:

```
yank  
motion.txt  
04.6
```

[faq-12.7](#)

12.7. When I yank some text into a register, how do I append the text to the current contents of the register?

When you specify the register for some operation, if you use the upper-case for the register name, then the new text will be appended to the existing contents. For example, if you have some text in the register "a". If you want to append some new text to this, you have to use the "A" register name. If you use the lowercase register name, then the contents of the register will be overwritten with the new text.

For more information, read:

```
quote
quote_alpha
10.1
```

[faq-12.8](#)

12.8. How do I yank a complete sentence that spans over more than one line?

To yank a complete sentence that spans over more than one line you have to use the yank operator followed by a motion command. For example:

```
y)
```

From inside the sentence you can use "yi)" to yank the sentence.

For more information, read:

```
yank
{motion}
object-motions
04.6
```

[faq-12.9](#)

12.9. How do I yank all the lines containing a pattern into a buffer?

You can use the `:global` command to yank all the lines containing the pattern into a register and then paste the contents of the register into the buffer:

```
:let @a=''
:g/mypattern/y A
```

The first command, clears the contents of the register "a". The second command copies all the lines containing "mypattern" into the register "a". **Note** that the capital letter "A" is used to append the matched lines. Now you can paste the contents of register "a" to a buffer using "ap command.

If you only want to collect all matches, you can use a different approach. For that run the `:s` command with the flags "gn" so that it won't actually change the buffer ("n" flag) but select each match ("g" flag). Combining this with the "\=" part in the replacement part, you can copy each match to e.g. a list. Altogether this looks like this:

```
:let list=[]
:%s/pattern/\=add(list, submatch(0))/gn
```

Now all matches will be in the list and you can post process them as wanted.

For more information, read:

```
:g
```

```
:y
:let-register
quote_alpha
put
registers
:registers
sub-replace-\=
```

faq-12.10

12.10. How do I delete all the lines in a file that do not contain a pattern?

You can use ``:v`` command to delete all the lines that do not contain a pattern:

```
:v/pattern/d
```

or

```
:g!/pattern/d
```

For more information, read:

```
:v
:g
```

faq-12.11

12.11. How do I add a line before each line with "pattern" in it?

You can use the following command to add a line before each line with "pattern" in it:

```
:g/pattern/normal! Oi<line of text goes here>
```

Alternatively you can yank the line using the Y command and then insert the line using the following command:

```
:g/pattern/put!
```

For more information, read:

```
:g
:put
insert
0
```

faq-12.12

12.12. Is there a way to operate on a line if the previous line contains a particular pattern?

You can use the ``:global`` command to operate on a line, if the previous line contains a particular pattern:

```
:g/<pattern>/+{cmd}
```


For more information, read:

```
:g
:range
```

faq-12.13

12.13. How do I execute a command on all the lines containing a pattern?

You can use the `:global` (`:g`) command to execute a command on all the lines containing a pattern.

```
:g/my pattern/d
```

If you want to use a non-Ex command, then you can use the `:normal` command:

```
:g/my pattern/normal {command}
```

Unless you want the normal mode commands to be remapped, consider using a `:normal!` command instead (note the "!").

For more information, read:

```
:global
:v
:normal
```

faq-12.14

12.14. Can I copy the character above the cursor to the current cursor position?

In Insert mode, you can copy the character above the cursor to the current cursor position by typing **CTRL-Y**. The same can be done with the characters below the cursor by typing **CTRL-E**.

For more information, read:

```
i_CTRL-Y
i_CTRL-E
```

faq-12.15

12.15. How do I insert a blank line above/below the current line without entering insert mode?

You can use the `:put` ex command to insert blank lines. For example, try

```
:put ='
:put! ='
```

For more information, read:

```
:put
```

faq-12.16

12.16. How do I insert the name of the current file into the current buffer?

There are several ways to insert the name of the current file into the current buffer. In insert mode, you can use the `<C-R>%` or the `<C-R>=expand("%")` command. In normal mode, you can use the ``:put =@%`` command.

For more information, read:

```
i_CTRL-R
expand()
!!
```

faq-12.17

12.17. How do I insert the contents of a Vim register into the current buffer?

In insert mode, you can use the `<CTRL-R><register>` command to insert the contents of `<register>`. For example, use `<CTRL-R>a` to insert the contents of register "a" into the current buffer.

In normal mode, you can use the ``:put <register>`` command to insert the contents of `<register>`. For example, use the ``:put d`` command to insert the contents of register "d" into the current buffer.

For more information, read:

```
i_CTRL-R
i_CTRL-R_CTRL-R
i_CTRL-R_CTRL-O
i_CTRL-R_CTRL-P
:put
```

faq-12.18

12.18. How do I move the cursor past the end of line and insert some characters at some columns after the end of the line?

You can set the `'virtualedit'` option to move the cursor past the end-of-line and insert characters in a column after the end-of-line. To start the virtual mode, use

```
:set virtualedit=all
```

For more information, read:

```
'virtualedit'
```

faq-12.19

12.19. How to replace the word under the cursor (say: junk) with "foojunkbar" in Vim?

There are several ways to do this. If the word is the first such word on the line, use the following command:

```
:exe "s/".expand("<word>")."/foo&bar/"
```

To match specifically you could use a more complex substitution like this:

```
:exe 's/\<' . expand("<word>") . '\%>' . (col(".") - 1) . 'c\>/foo&bar/'
```

You can also use the command: `ciwfoo<C-R>"bar<Esc>`

For more information, read:

```
:substitute
expand()
col()
/\%c
```

[faq-12.20](#)

12.20. How do I replace a particular text in all the files in a directory?

You can use the `:argdo` command to execute the substitute command on all the files specified as arguments:

```
:args *
:argdo %s/<your_text>/<replacement_text>/ge | update
```

For more information, read:

```
:args_f
:argdo
:s_flags
```

[faq-12.21](#)

12.21. I have some numbers in a file. How do I increment or decrement the numbers in the file?

You can use the **CTRL-A** key to increment the number and the **CTRL-X** key to decrement the number. You can also specify the number to increment/decrement from the number by specifying a count to the key. This works for decimal, octal and hexadecimal numbers. You can change the base used by Vim for this operation by modifying the `'nrformats'` option.

For more information, read:

```
26.2
CTRL-A
CTRL-X
'nrformats'
```

[faq-12.22](#)

12.22. How do I reuse the last used search pattern in a `:substitute` command?

To reuse the last used search pattern in a `:substitute` command, don't specify a new search pattern:

```
:s/pattern/newtext/  
:s//sometext/
```

In the second `:s`` command, as a search pattern is not specified, the pattern specified in the first `:s`` command "pattern" will be used.

If you want to change the search pattern but repeat the substitution pattern you can use the special right hand side, you can use the tilde character:

```
:s/newpattern/~/
```

For more information, read:

```
:S  
:&  
:~  
&  
sub-replace-special
```

faq-12.23

12.23. How do I change the case of a string using the `:substitute`` command?

You can use special characters in the replacement string for a `:substitute`` command to change the case of the matched string. For example, to change the case of the string "MyString" to all uppercase, you can use the following command:

```
:%s/MyString/\U&/g
```

To change the case to lowercase, you can use the following command:

```
:%s/MyString/\L&/g
```

To change the case of the first character in all the words in the current line to uppercase, you can use the following command:

```
:s/\< \(.\) \( \k*\) \> /\u\1\L\2/g
```

For more information, read:

```
sub-replace-special  
:substitute  
\U  
\L  
\u
```

faq-12.24

12.24. How do I enter characters that are not present in the keyboard?

You can use digraphs to enter characters that are not present in the keyboard. You can use the `:digraphs`` command to display all the currently

defined digraphs. You can add a new digraph to the list using the `:digraphs` command.

For more information, read:

```
digraphs
'digraph'
24.9
```

faq-12.25

12.25. Is there a command to remove any or all digraphs?

No. The digraphs table is defined at compile time. You can only add new ones. Adding a command to remove digraphs is on the todo list.

faq-12.26

12.26. In insert mode, when I press the backspace key, it erases only the characters entered in this instance of insert mode. How do I erase previously entered characters in insert mode using the backspace key?

This is traditional vi behaviour. You can set the `'backspace'` option to erase previously entered characters in insert mode:

```
:set backspace=indent,eol,start
```

For more information, read:

```
'backspace'
i_backspacing
```

faq-12.27

12.27. I have a file which has lines longer than 72 characters terminated with "+" and wrapped to the next line. How can I quickly join the lines?

You can use the `:global` command to search and join the lines:

```
:g/+$ /j
```

This will, however, only join every second line. A couple of more complex examples which will join all consecutive lines with a "+" at the end are:

```
:g/+$/,/\(^|[^\+])$/j
:g/+$ /mark a | .,/\(^|[^\+])$/s/+$// | 'a,.j
```

For more information, read:

```
:g
:j
:mark
```

faq-12.28

12.28. How do I paste characterwise yanked text into separate lines?

You can use the `:put` command to paste characterwise yanked text into new lines:

```
:put =@"
```

For more information, read:

```
:put  
quote_="
```

faq-12.29

12.29. How do I change the case (uppercase, lowercase) of a word or a character or a block of text?

You can use the `"~` command to switch the case of a character.

You can change the case of the word under the cursor to uppercase using the `"gUiw` or `"viwU` command and to lowercase using the `"guiw` or `"viwu` command.

You can switch the case (upper case to lower case and vice versa) of the word under the cursor using the `"viw~` or `"g~iw` command.

You can use the `"gUgU` command to change the current line to uppercase and the `"gugu` command to change the current line to lowercase.

You can use the `"g~g~` command to switch the case of the current line. You can use the `"g~{motion}` or `"{Visual}~` commands to switch the case of a block of text.

If you set `'tildeop'` the `"~` command behaves like an operator and expects a motion command to act on. If you have

```
:set tildeop
```

and you want to change the case from the current cursor position to the end of line, simply use `"~$`.

For more information, read:

```
case  
'tildeop'
```

faq-12.30

12.30. How do I enter ASCII characters that are not present in the keyboard?

You can enter ASCII characters that are not present in the keyboard by pressing **CTRL-V** and then the ASCII character number. You can also use digraphs to enter special ASCII characters.

For more information, read:

i_CTRL-V_digit
digraphs
45.5

faq-12.31

12.31. How do I replace non-printable characters in a file?

To replace a non-printable character, you have to first determine the ASCII value for the character. You can use the `:ascii` ex command or the "ga" normal-mode command to display the ASCII value of the character under the cursor.

You can enter the non-printable character by entering **CTRL-V** followed by the decimal number 1-255 (with no leading zero), or by x and a hex number 00-FF, or by an octal number 0-0377 (with leading zero), or by u and a hex number 0-FFFF, or by U and a hex number 0-7FFFFFFF

Another alternative is to use the `:digraphs` ex command to display the digraphs for all characters, together with their value in decimal and alpha. You can enter a non-printable character by entering **CTRL-K** followed by two alphanumeric characters (a digraph).

For more information, read:

`:ascii`
`i_CTRL-V`
`i_CTRL-V_digit`
`:digraphs`

faq-12.32

12.32. How do I remove duplicate lines from a buffer?

You can use the following user-defined command to remove all the duplicate lines from a buffer:

```
:command -range=% Uniq <line1>,<line2>g/^%\<<line2>l\(.*\)\n\1$/d
```

Add the above command to your `.vimrc` file and invoke `:Uniq` to remove all the duplicate lines.

faq-12.33

12.33. How do I prefix all the lines in a file with the corresponding line numbers?

You can prefix the lines in a file with the corresponding line number in several ways. Some of them are listed below:

```
:%s/^/\=line('.'). ' '  
:%s/^/\=printf('%5d ', line('.'))/  
:%s/^/\=strpart(line('.').', 0, 5)  
:%s/^/\=strpart(' ', strlen(line('.'))).line('.'). ' '
```

The last two commands will pad the line numbers with space characters. The last command will right align the numbers and the command before that will

left align the numbers.

If you don't want to number consecutive lines but rather non-consecutive regions, you can also use this idiom:

```
:let i = 1
:g/TODO/s/^\=\=printf('%2d.',i)|let i+=1
```

This first initializes the variable `i` with 1. In the next line, a ``:g`` command is used to perform a substitute command only on lines, that match "TODO". After the substitute command has taken place, the variable `i` will be incremented by 1.

For more information, read:

```
sub-replace-special
line()
expr6
strpart()
printf()
:execute
:global
```

faq-12.34

12.34. How do I exchange (swap) two characters or words or lines?

You can exchange two characters with the "xp" command sequence. The "x" will delete the character under the cursor and "p" will paste the just deleted character after the character under the cursor. This will result in exchanging the two characters.

You can exchange two words with the "deep" command sequence (start with the cursor in the blank space before the first word).

You can exchange two lines with the "ddp" command sequence. The "dd" will delete the current line and "p" will paste the just deleted line after the current line. This will result in exchanging the two lines.

All of the above operations will change the " unnamed register.

You can use the ``:m +`` ex command to exchange two lines without changing the unnamed register.

For more information, read:

```
x
p
dd
d
e
linewise-register
quotequote
:move
```


12.35. How do I change the characters used as word delimiters?

Vim uses the characters specified by the **'iskeyword'** option as word delimiters. The default setting for this option is "@,48-57,_,192-255".

For example, to add ":" as a word delimiter, you can use

```
:set iskeyword+=:
```

To remove "_" as a word delimiter, you can use

```
:set iskeyword=-_
```

For more information, read:

```
'iskeyword'  
word
```

=====

SECTION 13 - COMPLETION IN INSERT MODE

13.1. How do I complete words or lines in insert mode?

In insert mode, you can complete words using the **CTRL-P** and **CTRL-N** keys. The **CTRL-N** command searches forward for the next matching keyword. The **CTRL-P** command searches backwards for the next matching keyword.

In insert mode, you can use the **CTRL-X CTRL-L** command sequence to complete lines that starts with the same characters as in the current line before the cursor. To get the next matching line, press the **CTRL-P** or **CTRL-N** keys. There are a lot of other keys/ways available to complete words in insert mode.

Vim supports completion of the following items:

```
CTRL-X CTRL-L   whole lines
CTRL-X CTRL-N   keywords in the current file
CTRL-X CTRL-K   words from a dictionary
CTRL-X CTRL-T   words from a thesaurus
CTRL-X CTRL-I   current and included files
CTRL-X CTRL-]   tags
CTRL-X CTRL-F   file names
CTRL-X CTRL-D   macro definitions (also in included files)
CTRL-X CTRL-V   Vim command line
CTRL-X CTRL-U   User defined completion
CTRL-X CTRL-O   Omni completion
```

User defined completions and omni completions are often set by filetype plugins.

For more information, read:

24.3
ins-completion

faq-13.2

13.2. How do I complete file names in insert mode?

In insert mode, you can use the **CTRL-X CTRL-F** command sequence to complete filenames that start with the same characters as in the current line before the cursor.

For more information, read:

compl-filename

faq-13.3

13.3. I am using **CTRL-P/CTRL-N** to complete words in insert mode. How do I complete words that occur after the just completed word?

You can use **CTRL-X CTRL-N** and **CTRL-X CTRL-P** keys to complete words that are present after the just completed word.

For more information, read:

i_CTRL-X_CTRL-P
i_CTRL-X_CTRL-N
ins-completion

=====
faq-14

SECTION 14 - TEXT FORMATTING

faq-14.1

14.1. How do I format a text paragraph so that a new line is inserted at the end of each wrapped line?

You can use the "gq" command to format a paragraph. This will format the text according to the current **'textwidth'** setting. An alternative would be to use the "gw" command that formats like "gq" but does not move the cursor.

Note that the gq operator can be used with a motion command to operate on a range of text. For example:

gqgq - Format the current line
gqap - Format current paragraph
gwap - Format current paragraph (and don't move cursor)
gq3j - Format the current and the next 3 lines

For more information, read:

gq
gw
formatting

```
usr_25.txt
motion.txt
```

faq-14.2

14.2. How do I format long lines in a file so that each line contains less than "n" characters?

You can set the **'textwidth'** option to control the number of characters that can be present in a line. For example, to set the maximum width of a line to 70 characters, you can use the following command:

```
set textwidth=70
```

Now to break the long lines in a file to the length defined by the **'textwidth'** option, you can use

```
:g/./normal gq
```

For more information, read:

```
'textwidth'
gq
```

faq-14.3

14.3. How do I join short lines to form a paragraph?

First, make sure the **'textwidth'** option is set to a high value:

```
:set textwidth=99999
```

Next, join the short lines to form a paragraph using the command:

```
1GgqG
```

The above command will operate on the entire file. To do the formatting on all paragraphs in a specific range, use:

```
:'a,'bg/\S/normal gq}
```

For more information, read:

```
gq
G
gqg
```

faq-14.4

14.4. How do I format bulleted and numbered lists?

You can configure Vim to format bulleted and numbered lists using the **'formatoptions'** option. For example, you can format the list of the following format:

```
- this is a test. this is a test. this is a test. this is a test.
this is a test.
```

into this format:

```
- this is a test. this is a test. this is a test. this is a test.
  this is a test.
```

You can use the "n" flag in the **'formatoptions'** to align the text.

```
:set fo+=n
```

With this option, when formatting text, Vim will recognize numbered lists. For this option to work, the **'autoindent'** option also must be set.

For more information, read:

```
'formatoptions'  
fo-table  
format-comments
```

faq-14.5

14.5. How do I indent lines in insert mode?

In insert mode, you can press the **CTRL-T** key to insert one shiftwidth of indent at the start of the current line. In insert mode, you can use the **CTRL-D** key to delete one shiftwidth of indent at the start of the current line. You can also use the **CTRL-O >>** and **CTRL-O <<** commands to indent the current line in insert mode.

For more information, read:

```
i_CTRL-T  
i_CTRL-D  
i_0_CTRL-D  
i_CTRL-O  
>>  
<<
```

faq-14.6

14.6. How do I format/indent an entire file?

You can format/indent an entire file using the `gg=G` command, where

```
gg - Goto the beginning of the file  
=  - apply indentation  
G  - till end of file
```

For more information, read:

```
gg  
=  
G  
'formatprg'  
C-indenting
```

faq-14.7

14.7. How do I increase or decrease the indentation of the current line?

You can use the ">>" and "<<" commands to increase or decrease the indentation of the current line.

For more information, read:

```
shift-left-right
>>
<<
'shiftwidth'
```

faq-14.8

14.8. How do I indent a block/group of lines?

You can visually select the group of lines and press the > or < key to indent/unindent the lines. You can also use the following ex-command to indent the lines

```
:10,20>
```

For more information, read:

```
shift-left-right
v_>
v_<
:<
:>
```

faq-14.9

14.9. When I indent lines using the > or < key, the standard 8-tabstops are used instead of the current 'tabstop' setting. Why?

The number of spaces used when lines are indented using the ">" operator is controlled by the 'shiftwidth' option. The 'tabstop' setting is only used, when the 'shiftwidth' option is zero.

```
:set shiftwidth=4
```

For more information, read:

```
'shiftwidth'
>>
'softtabstop'
```

faq-14.10

14.10. How do I turn off the automatic indentation of text?

By default, the automatic indentation of text is not turned on. Check the configuration files (.vimrc, .gvimrc) for settings related to indentation. Make sure the `:filetype indent on` command is not present. If it is present, remove it. Also, depending on your preference, you may also want to check the value of the 'autoindent', 'smartindent', 'cindent' and

'**indentexpr**' options and turn them off as needed.

For more information, read:

```
:filetype-indent-off
'autoindent'
'smartindent'
'cindent'
'indentexpr'
```

faq-14.11

14.11. How do I configure Vim to automatically set the '**textwidth**' option to a particular value when I edit mails?

You can use the "FileType" autocommand to set the '**textwidth**' option:

```
autocmd FileType mail set tw=<your_value>
```

For more information, read:

```
:autocmd
FileType
usr_43.txt
```

faq-14.12

14.12. Is there a way to make Vim auto-magically break lines?

Yes. Set the '**textwidth**' option to the preferred length for a line. Then Vim will auto-magically break the newly entered lines. For example:

```
:set textwidth=75
```

For more information, read:

```
'textwidth'
ins-textwidth
'formatoptions'
fo-table
formatting
```

faq-14.13

14.13. I am seeing a lot of ^M symbols in my file. I tried setting the '**fileformat**' option to "dos" and then "unix" and then "mac". None of these helped. How can I hide these symbols?

When a file is loaded in Vim, the format of the file is determined as below:

- If all the lines end with a new line (<NL>), then the fileformat is "unix".
- If all the lines end with a carriage return (<CR>) followed by a new line (<NL>), then the fileformat is "dos".
- If all the lines end with carriage return (<CR>), then the fileformat is "mac".

If the file has some lines ending with <CR> and some lines ending with <CR> followed by a <NL>, then the fileformat is set to "unix".

You can change the format of the current file, by saving it explicitly in dos format:

```
:w ++ff=dos
```

To display the format of the current file, use

```
:set fileformat?
```

The above behavior is also controlled by the **'fileformats'** option. You can try the following commands:

```
:set fileformats+=unix
:e <your_file>
:set fileformat=unix
:w
```

To remove the carriage return (<CR>) character at the end of all the lines in the current file, you can use the following command:

```
:%s/\r$//
```

To force Vim to use a particular file format, when editing a file, you can use the following command:

```
:e ++ff=dos filename
```

For more information, read:

```
'fileformats'
'fileformat'
file-formats
DOS-format-write
Unix-format-write
Mac-format-write
dos-file-formats
23.1
++ff
```

faq-14.14

14.14. When I paste some text into a Vim buffer from another application, the alignment (indentation) of the new text is messed up. How do I fix this?

When you paste text into a GUI Vim using the mouse, Vim is able to detect that you are pasting text. So all the indentation related settings (like autoindent, smartindent, cindent, etc.) are ignored and the text is pasted literally.

When pasting text into a Vim running in a terminal (like xterm) using

the mouse, Vim may not be able to detect that you are pasting text. This depends on several things: the capability of the terminal to pass the mouse events to Vim, Vim is compiled to handle mouse events and access the clipboard, the DISPLAY variable is set properly, the Vim **'mouse'** option is set correctly.

If Vim is able to detect that you are pasting text using the mouse, then the pasted text will be inserted literally.

If Vim is not able to detect that you are pasting using the mouse, then it will see the pasted text as though you literally typed the text. After the first line from the pasted text is inserted, when Vim encounters the newline character, because of the indentation settings, the next line will start indented. The spaces at the beginning of the second line in the pasted text will be inserted leading to additional indentation. This will be repeated for subsequent lines. So the pasted text will be inserted with stair case indentation.

You can fix this problem in a terminal Vim in several ways:

1. Build Vim with the `+mouse` and `+xterm_clipboard` compile-time options. The normal or big or huge build of Vim includes these options. Set the **'mouse'** option to either "a" or include "i". When pasting text using the mouse, don't press the Shift key. This will work only if Vim can access the X display. For more information, read the following Vim help topics:

```
+feature-list
'mouse'
<MiddleMouse>
x11-selection
xterm-clipboard
```

- 1.1 Some Linux distributions build their terminal vim packages without X support. This makes no sense and leaves many users with the impression that Vim in terminal mode doesn't support some operations such as properly pasting text with a mouse.

If your distribution includes `gvim`, which it almost certainly does these days, the solutions to this include the following.

- a) Start Vim as

```
gvim -v
```

- b) Put this alias in your shell's configuration file, e.g. `~/.bashrc`:

```
alias vim='gvim -v'
```

- c) Put the following command in a file named "vim" and put that file in your `~/bin` directory:

```
gvim -v "$@"
```


- d) Link the distribution's gvim to ~/bin/vim with the following command, which needs to be executed only once.

```
ln -s $(which gvim) ~/bin/vim
```

For c) and d), make sure that ~/bin precedes /usr/bin in your PATH.

2. Paste the text using the **CTRL-R CTRL-O** * command. This will paste the text literally without any automatic indentation. If you want to paste the text and then fix the indentation, then you can use **CTRL-R CTRL-P** *. These commands will work only if Vim can access the X display. For more information, read the following Vim help topics:

```
i_CTRL-R_CTRL-O  
i_CTRL-R_CTRL-P  
quotestar
```

3. Set the **'paste'** option before pasting the text. This option will disable the effect of all the indentation related settings. Make sure to turn off this option using `:set nopaste`` after pasting the text. Otherwise the Vim indentation feature will not work. Do not permanently set the **'paste'** option in your .vimrc file. If you are going to repeat these steps often, then you can set the **'pastetoggle'** option to a key. When you press the specified key, the **'paste'** option will be toggled. You can press the key once before pasting the text and the press the key once after pasting the text. **Note** that when the **'paste'** option is set, all the mappings and abbreviations are disabled. For more information, read the following Vim help topics:

```
'paste'  
'pastetoggle'
```

You can also refer to the following topics in the user manual:

```
04.7  
09.3
```

[faq-14.15](#)

- 14.15. When there is a very long wrapped line (wrap is "on") and a line doesn't fit entirely on the screen it is not displayed at all. There are blank lines beginning with "@" symbol instead of wrapped line. If I scroll the screen to fit the line the "@" symbols disappear and the line is displayed again. What Vim setting control this behavior?

You can set the **'display'** option to "lastline" to display as much as possible of the last line in a window instead of displaying the "@" symbols.

```
:set display=lastline
```

For more information, read:

'display'

faq-14.16

14.16. How do I convert all the tab characters in a file to space characters?

You can use the `:retab` command to update all the tab characters in the current file with the current setting of `'expandtab'` and `'tabstop'`. For example, to convert all the tabs to white spaces, use

```
:set expandtab
:retab
```

For more information, read:

```
:retab
'expandtab'
'tabstop'
25.3
```

faq-14.17

14.17. What Vim options can I use to edit text that will later go to a word processor?

You can set the following options to edit text that will later go into a word processor:

```
:set wrap
:set linebreak
:set textwidth=0
:set showbreak=>>>
```

You can use the "gk" and "gj" commands to move one screen line up and down. For more information, read:

```
'wrap'
'linebreak'
'textwidth'
'showbreak'
gk
gj
```

faq-14.18

14.18. How do I join lines without adding or removing any space characters?

By default, when you join lines using the "J" or `:join` command, Vim will replace the line break, leading white space and trailing white space with a single space character. If there are space characters at the end of a line or a line starts with the ")" character, then Vim will not add a space character.

To join lines without adding or removing any space characters, you can use the gJ or `:join!` commands.

For more information, read:

```
gJ
:join
J
10.5
'joinspaces'
'coptions'
'formatoptions'
```

faq-15

SECTION 15 - VISUAL MODE

faq-15.1

15.1. How do I do rectangular block copying?

You can do rectangular block copying in Vim using the blockwise visual mode. To start blockwise visual mode use the **CTRL-V** key. Move the cursor using any of the motion commands and then use the y operator to yank to visually selected text.

If **CTRL-V** does not work as expected, it may have been remapped to **CTRL-Q** by the mswin.vim script which is often sourced by a vimrc on Windows machines to mimic some common short cuts from other programs.

For more information, read:

```
04.4
blockwise-visual
visual-mode
Q_vi
```

faq-15.2

15.2. How do I delete or change a column of text in a file?

You can use the Vim block-wise visual mode to select the column of text and apply an operator (delete, change, copy, etc) on it.

For more information, read:

```
visual-block
visual-operators
```

faq-15.3

15.3. How do I apply an ex-command on a set of visually selected lines?

When you select a range of lines in visual mode, the < register is set to the start of the visual region and the > register is set to the end of the visual region. You can use these registers to specify the range for an ex command. After visually selecting the lines, press ":" to go to the command mode. Vim will automatically insert the visual range '<,>'. You can run any ex-command on the visual range.

For more information, read:

```
V_:  
'<  
'>
```

faq-15.4

15.4. How do I execute an ex command on a column of text selected in Visual block mode?

All the ex commands operate on whole lines only. If you try to execute an ex command on a column of text selected in visual block mode, Vim will operate on all the selected lines (instead of the selected columns). You can use the `vis.vim` or `NrrwRgn` plugin script from <https://www.vim.org> scripts archive to do this.

For more information, read:

```
cmdline-ranges  
10.3  
cmdline-lines
```

faq-15.5

15.5. How do I select the entire file in visual mode?

You can select the entire file in visual mode using `ggVG`.

```
gg - go to the beginning of the file.  
V - Start linewise visual mode  
G - goto the end of the file.
```

For more information, read:

```
gg  
linewise-visual  
G
```

faq-15.6

15.6. When I visually select a set of lines and press the `>` key to indent the selected lines, the visual mode ends. How can I reselect the region for further operation? (or) How do I re-select the last selected visual area again?

You can use the `"gv"` command to reselect the last selected visual area. You can also use the marks `'<` and `'>` to jump to the beginning or the end of the last selected visual area.

For more information, read:

```
gv  
'<  
'>
```

faq-15.7

15.7. How do I jump to the beginning/end of a visually selected region?

You can use the "o" command to jump to the beginning/end of a visually selected region.

For more information, read:

[v_o](#)

faq-15.8

15.8. When I select text with mouse and then press : to enter an ex command, the selected text is replaced with the : character. How do I execute an ex command on a text selected using the mouse similar to the text selected using the visual mode?

This will happen if you have configured Vim to use select mode instead of Visual mode by setting the '**selectmode**' option. Check the value of this option:

```
:set selectmode?
```

This mode is known as selectmode and is similar to the visual mode. This option is also automatically set when you use the "behave mswin" command. Select mode looks like visual mode, but it is similar to the selection mode in MS-Windows.

For more information, read:

```
Select-mode  
'selectmode'  
09.4  
:behave
```

faq-15.9

15.9. When I select a block of text using the mouse, Vim goes into selection mode instead of Visual mode. Why?

The '**selectmode**' option controls whether Select mode will be started when selecting a block of text using the mouse. To start Visual mode when selecting text using mouse, remove the "mouse" value from the '**selectmode**' option:

```
:set selectmode-=mouse
```

Note that by default, the '**selectmode**' option will be set to empty, so that always visual mode is used.

For more information, read:

```
'selectmode'  
Select-mode  
:behave
```

15.10. How do I visually select the last copy/pasted text?

You can use the '[' and ']' marks to visually select the last copy/pasted text. The '[' mark is set to the beginning of the last changed/yanked text and the ']' mark is set to the end of the last changed/yanked text. To visually select this block of text use the command '[v']

For more information, read:

```
'[
']
`a
v
```

SECTION 16 - COMMAND-LINE MODE

16.1. How do I use the name of the current file in the command mode or an ex command line?

In the command line, the "%" character represents the name of the current file. In some commands, you have to use `expand("%)` to get the filename:

```
:!perl %
```

Another example is to load the latex generated pdf file from the file you are currently editing:

```
!:xpdf %<.pdf
```

For more information, read:

```
:_%
cmdline-special
expand()
```

16.2. How do I edit the text in the Vim command-line effectively?

You can use the command-line window for editing Vim command-line text. To open the Vim command-line window use the "q:" command in normal mode. In command-line mode, use the **CTRL-F** key. In this window, the command line history will be displayed. You can use normal Vim keys/commands to edit any previous/new command line. To execute a command line, press the enter/return key.

In a similar vain, the search history can be edited with "q/" and "q?" commands.

For more information, read:

16.3. How do I switch from Vi mode to Ex mode?

You can use the Q command to switch from Vi mode to Ex mode. To switch from Ex mode back to the Vi mode, use the `:vi` command.

For more information, read:

```
Q
gQ
Ex-mode
:vi
```

16.4. How do I copy the output from an ex-command into a buffer?

To copy the output from an ex-command into a buffer, you have to first get the command output into a register. You can use the `:redir` command to get the output into a register. For example,

```
:redir @a
:g/HelloWord/p
:redir END
```

Now the register "a" will contain the output from the ex command `:g/HelloWord/p`. Now you can paste the contents of the register "a" into a buffer. You can also send or append the output of an ex-command into a file using the `:redir` command.

You can prefix the `:global` command with `:silent`, to avoid having the lines printed to the screen.

To redirect the output from an ex-command to a file, you can use the following set of commands:

```
:redir > myfile
:g/HelloWord/p
:redir END
```

For more information, read:

```
:redir
:silent
```

16.5. When I press the `<Tab>` key to complete the name of a file in the command mode, if there are more than one matching file names, then Vim completes the first matching file name and displays a list of all matching filenames. How do I configure Vim to only display the list of all the matching filenames and not complete the first one?

You can modify the `'wildmode'` option to configure the way Vim completes

filenames in the command mode. In this case, you can set the **'wildmode'** option to "list":

```
:set wildmode=list
```

For more information, read:

'wildmode'

faq-16.6

16.6. How do I copy text from a buffer to the command line and from the command line to a buffer?

To copy text from a buffer to the command line, after yanking the text from the buffer, use "<CTRL-R>0" in the command line to paste the text. You can also yank the text to a specific register and use **CTRL-R <register>** to paste the text to the command line. You can use **CTRL-R CTRL-W** to paste the word under the cursor in the command line.

To copy text from the command line into a buffer, you can paste the contents of the : register using the ":p command. The most recently executed command line is stored in the : register.

Another approach for copying and pasting text to and from the command line is to open the command line window using q: from normal mode or **CTRL-F** from the command-line mode. In the command line window you can use all the Vim commands to edit the command line.

For more information, read:

```
c_CTRL-R  
quote_  
cmdline-window
```

faq-16.7

16.7. How do I put a command onto the command history without executing it?

To put a command onto the command history without executing it, press the <Esc> key to cancel the command.

An alternative solution, is to use the histadd() function like this:

```
:call histadd(':', 'echo strftime("%c")')
```

For more information, read:

```
c_<Esc>  
histadd()
```

faq-16.8

16.8. How do I increase the height of the command-line?

You can increase the height of the command-line by changing the **'cmdheight'** option:


```
:set cmdheight=2
```

For more information, read:

```
'cmdheight'  
hit-enter  
05.7
```

faq-17

SECTION 17 - VIMINFO

faq-17.1

17.1. When I invoke Vim, I get error messages about illegal characters in the viminfo file. What should I do to get rid of these messages?

You can remove the \$HOME/.viminfo or the \$HOME/_viminfo file to get rid of these error messages.

For more information, read:

```
viminfo-errors  
viminfo-file-name  
viminfo  
21.3
```

faq-17.2

17.2. How do I disable the viminfo feature?

By default, the viminfo feature is disabled. If the viminfo feature is enabled by a system-wide vimrc file, then you can disable the viminfo feature by setting the '**viminfo**' option to an empty string in your local .vimrc file:

```
:set viminfo=""
```

For more information, read:

```
'viminfo'
```

faq-17.3

17.3. How do I save and use Vim marks/commands across Vim sessions?

You can save and restore Vim marks across Vim sessions using the viminfo file. To use the viminfo file, make sure the '**viminfo**' option is not empty. To save and restore Vim marks, the '**viminfo**' option should not contain the "f" flag or should have a value greater than zero for the "f" option.

You can also use the viminfo file to synchronize the commandline history across different sessions using `:wvimfo` and `:rviminfo` commands together with the FocusGained and FocusLost autocommands:

```
augroup viminfo
```

```
    au!  
    au FocusLost * wviminfo  
    au FocusGained * rviminfo  
augroup end
```

Note, this will only work reliably, when Vim can detect the FocusLost and FocusGained autocommands correctly. This means it should work with GVim but might depend on your terminal for konsole vim.

For more information, read:

```
21.3  
viminfo  
'viminfo'  
:wviminfo  
:rviminfo  
FocusLost  
FocusGained
```

faq-18

SECTION 18 - REMOTE EDITING

faq-18.1

18.1. How do I open a file with existing instance of gvim? What happened to the Vim 5.x OpenWithVim.exe and SendToVim.exe files?

Starting with Vim6, the OLE version of OpenWithVim.exe and SendToVim.exe Vim utilities are replaced by the new client-server feature. To open the file j.txt with an existing instance of Gvim (MyVim), use:

```
$ gvim --servername MyVim --remote-silent j.txt
```

To list the server names of all the currently running Vim instances, use

```
$ vim --serverlist
```

To get more information about client-server feature, read

```
client-server
```

faq-18.2

18.2. How do I send a command to a Vim server to write all buffers to disk?

You can use the Vim remote server functionality to do this:

```
$ gvim --servername myVIM --remote-send "<C-\><C-N>:wall<CR>"
```

For more information, read:

```
client-server  
CTRL-\_CTRL-N  
:wall
```

faq-18.3

18.3. Where can I get the documentation about the Vim remote server functionality?

You can get more information about the Vim remote server functionality by reading

[client-server](#)

=====

faq-19

SECTION 19 - OPTIONS

faq-19.1

19.1. How do I configure Vim in a simple way?

You can use the `:options` command to open the Vim option window:

```
:options
```

This window can be used for viewing and setting all the options.

For more information, read:

[:options](#)

faq-19.2

19.2. How do I toggle the value of an option?

You can prefix the option with "inv" to toggle the value of the option:

```
:set invignorecase  
:set invhlsearch
```

You can also suffix the option with "!" to toggle the value:

```
:set ignorecase!  
:set hlsearch!
```

For more information, read:

[set-option](#)

faq-19.3

19.3. How do I set an option that affects only the current buffer/window?

Some of the Vim options can have a local or global value. A local value applies only to a specific buffer or window. A global value applies to all the buffers or windows.

When a Vim option is modified using the `:set` command, both the global and local values for the option are changed. You can use the `:setlocal` command to modify only the local value for the option and the `:setglobal` command to modify only the global value.

You can use the `:setlocal` command to set an option that will affect only the current file/buffer:

```
:setlocal textwidth=70
```

Note that not all options can have a local value. You can use `:setlocal` command to set an option locally to a buffer/window only if the option is allowed to have a local value.

You can also use the following command to set an option locally:

```
:let &l:{option-name} = <value>
```

For more information, read:

```
:setlocal  
local-options
```

[faq-19.4](#)

19.4. How do I use space characters for a Vim option value?

To use space characters in a Vim option value, you have to escape the space character. For example:

```
:set tags=tags\ /usr/tags
```

For more information, read:

```
option-backslash
```

[faq-19.5](#)

19.5. Can I add (embed) Vim option settings to the contents of a file?

You can use modelines to add Vim option settings to the contents of a file. For example, in a C file, you can add the following line to the top or the bottom of the file:

```
/* vim:sw=4: */
```

This will set the `'shiftwidth'` option to 4, when editing that C file. For this to work, the `'modeline'` option should be set. By default, the `'modeline'` option is set. An alternative example is given in this document in the first line.

The `'modelines'` settings specifies the number of lines that will be checked for the Vim set commands.

For more information, read:

```
21.6  
modeline  
auto-setting  
'modeline'
```

`'modelines'`

faq-19.6

19.6. How do I display the line numbers of all the lines in a file?

You can set the `'number'` option to display the line numbers for all the lines.

```
:set number
```

For more information, read:

`'number'`

faq-19.7

19.7. How do I change the width of the line numbers displayed using the `'number'` option?

You can set the minimum number of columns to be used for line numbering by setting the `'numberwidth'` option:

```
:set numberwidth=3
```

This set's the width for the line number to 3 digits, which is enough, if your buffer contains less than 999 lines. However, if your current buffer contains more lines than 999, the `'numberwidth'` will be adjusted accordingly, so that the maximum line number will fit on the screen.

faq-19.8

19.8. How do I display (view) all the invisible characters like space, tabs and newlines in a file?

You can set the `'list'` option to see all the invisible characters in your file.

```
:set list
```

With this option set, you can view space characters, tabs, newlines, trailing space characters and wrapped lines.

To not display the invisible characters (which is the default), you have to reset the `'list'` option:

```
:set nolist  
(or)  
:set list!
```

The `':set list!'` command will toggle the current setting of the boolean `'list'` option.

You can modify the `'listchars'` option to configure how and which invisible characters are displayed. For example, with the following command all the trailing space characters will be displayed with a "." character.

```
:set listchars=trail:.
```

For more information, read:

```
'listchars'  
'list'
```

[faq-19.9](#)

19.9. How do I configure Vim to always display the current line and column number?

You can set the **'ruler'** option to display current column and line number in the status line:

```
:set ruler
```

For more information, read:

```
'ruler'
```

[faq-19.10](#)

19.10. How do I display the current Vim mode?

You can set the **'showmode'** option to display the current Vim mode. In Insert, Replace and Visual modes, Vim will display the current mode on the last line.

```
:set showmode
```

For more information, read:

```
'showmode'
```

[faq-19.11](#)

19.11. How do I configure Vim to show pending/partial commands on the status line?

You can set the **'showcmd'** option to display pending/partial commands in the status line:

```
:set showcmd
```

For more information, read:

```
'showcmd'
```

[faq-19.12](#)

19.12. How do I configure the Vim status line to display different settings/values?

You can set the **'statusline'** option to display different values/settings in the Vim status line.

For more information, read:

```
'statusline'  
'laststatus'  
'rulerformat'  
'ruler'
```

faq-19.13

19.13. How do I configure Vim to display status line always?

You can set the **'laststatus'** option to 2 to display the status line always.

```
:set laststatus=2
```

For more information, read:

```
'laststatus'
```

faq-19.14

19.14. How do I make a Vim setting persistent across different Vim invocations/instances/sessions?

To make a Vim option setting persistent across different Vim instances, add your setting to the `.vimrc` or `.gvimrc` file. You can also use the `:mkvimrc` command to generate a vimrc file for the current settings.

For more information, read:

```
save-settings  
vimrc  
gvimrc  
vimrc-intro  
:mkvimrc  
initialization
```

faq-19.15

19.15. Why do I hear a beep (why does my window flash) about 1 second after I hit the Escape key?

This is normal behavior. If your window flashes, then you've got the visual bell on. Otherwise, you should hear a beep.

Vim needs a timeout to tell the difference between a simple escape and, say, a cursor key sequence. When you press a key in normal mode (and even in insert mode) and that key is the beginning of a mapping, Vim waits a certain amount of time to see if the rest of the mapping sequence follows. If the mapping sequence is completed before a given timeout period, the mapping for that sequence of keys is applied. If you interrupt the mapping, the normal actions associated with the keys are executed.

For example, if you have a mapping defined as ``:imap vvv Vim is great!!`` and you type "vvv" quickly, the "Vim is great!!" will be inserted into your text. But if you type "vv v" then that is what will put into your text. This is also true if you type "vvv" too slowly where "too slowly" is longer than the value for the timeout option. Setting the timeout option to a

larger value can help alleviate problems that appear when using function keys over a slow line.

For more information, read:

`'timeout'`

[faq-19.16](#)

19.16. How do I make the "c" and "s" commands display a "\$" instead of deleting the characters I'm changing?

To make the "c" and "s" commands display a "\$" instead of deleting the characters, add the \$ flag to the **'coptions'** option:

```
:set coptions+=
```

For more information, read:

`'coptions'`

[faq-19.17](#)

19.17. How do I remove more than one flag using a single `:set` command from a Vim option?

You can remove more than one flag from a Vim option using a single `:set` command, by specifying the flags in exactly the same order as they appear in the option. For example, if you use the following command to remove the "t" and "n" flags from the **'formatoptions'** option:

```
:set formatoptions-=tn
```

The "t" and "n" flags will be removed from the **'formatoptions'** option, only if the **'formatoptions'** option contains these flags in this order: "tn". Otherwise, it will not remove the flags. To avoid this problem, you can remove the flags one by one:

```
:set formatoptions-=t formatoptions-=n
```

For more information, read:

```
:set-=
```

[faq-20](#)

SECTION 20 - MAPPING KEYS

[faq-20.1](#)

20.1. How do I know what a key is mapped to?

To see what a key is mapped to, use the following commands:

```
:map <key>
:map! <key>
```


You can also check the mappings in a particular mode using one of the `:cmap`, `:nmap`, `:vmap`, `:imap`, `:omap`, etc commands.

To find out, where the key has been mapped, prefix the `:verbose` command:

```
:verbose :map <key>
```

For more information, read:

```
map-listing  
map-overview
```

faq-20.2

20.2. How do I list all the user-defined key mappings?

You can list all the user-defined key mappings using:

```
:map
```

For more information, read:

```
map-listing
```

faq-20.3

20.3. How do I unmap a previously mapped key?

You can unmap a previously mapped key using the `:unmap` command:

```
:unmap <key>  
:unmap! <key>
```

For mode specific mappings, you can use one of the these commands:

```
:nunmap  
:vunmap  
:ounmap  
:iunmap  
:lunmap  
:cunmap
```

The following command will fail to unmap a buffer-local mapped key:

```
:unmap <key>
```

To unmap a buffer-local mapped key, you have to use the `<buffer>` keyword in the unmap command:

```
:unmap <buffer> <key>  
:unmap! <buffer> <key>
```

For more information, read:

```
:unmap  
map-modes
```

```
:map-local
mapleader
```

faq-20.4

20.4. I am not able to create a mapping for the `<xxx>` key. What is wrong?

- 1) First make sure, the key is passed correctly to Vim. To determine if this is the case, put Vim in Insert mode and then hit **CTRL-V** (or **CTRL-Q** if your **CTRL-V** is remapped to the paste operation (e.g. on Windows if you are using the `mwin.vim` script file) followed by your key.

If nothing appears in the buffer (and assuming that you have `'showcmd'` on, `^V` remains displayed near the bottom right of the Vim screen), then Vim doesn't get your key correctly and there is nothing to be done, other than selecting a different key for your mapping or using GVim, which should recognise the key correctly.

- 2) Possibly, Vim gets your key, but sees it as no different than something else. Say you want to map `<Ctrl-Right>`, then in Insert mode hit `CTRL-K` followed by `<Ctrl-Right>`. If Vim displays `<C-Right>` it has correctly seen the keystroke and you should be able to map it (by using `<C-Right>` as your `{lhs}`). If it displays `<Right>` it has seen the keystroke but as if you hadn't held `<Ctrl>` down: this means your terminal passes `<Ctrl-Right>` as if it were just `<Right>`. Anything else means the key has been misidentified.
- 3) If the key is seen, but not as itself and not as some recognizable key, then there is probably an error in the terminal library for the current terminal (termcap or terminfo database). In that case

```
:set term?
```

will tell you which termcap or terminfo Vim is using. You can try to tell vim, what termcode to use in that terminal, by adding the following to your vimrc:

```
if &term == <termname>
    set <C-Right>=<keycode>
endif
```

where `<termname>` above should be replaced by the value of `'term'` (with quotes around it) and `<keycode>` by what you get when hitting **CTRL-V** followed by `<Ctrl-Right>` in Insert mode (with nothing around it). `<C-Right>` should be left as-is (9 characters). Don't forget that in a `:set` command, white space is not allowed between the equal sign and the value, and any space, double quote, vertical bar or backslash present as part of the value must be backslash-escaped.

Now you should be able to see the keycode corresponding to the key and you can create a mapping for the key using the following command:

```
:map <C-Right> <your_command_to_be_mapped>
```

For more information, read:

```
map-keys-fails
:map-special-keys
key-codes
```

faq-20.5

20.5. Why does mapping the `<C-...>` key not work?

The only `<Ctrl>-<printable-key>` chords which Vim can reliably detect (because they are defined in the ASCII standard) are the following:

CTRL-@	0x00	NUL
CTRL-A to CTRL-Z	0x01 to 0x1A	
CTRL-a to CTRL-z	0x01 to 0x1A	
CTRL-[0x1B	ESC
CTRL-\	0x1C	
CTRL-]	0x1D	
CTRL-^	0x1E	
CTRL-_	0x1F	
CTRL-?	0x7F	DEL

Most of these, however, already have a function in Vim (and some are aliases of other keys: **CTRL-H** and `<BS>`, **CTRL-I** and `<Tab>`, **CTRL-M** and `<Enter>`, `CTRL-[` and `<Esc>`, `CTRL-?` and ``).

The "safest" keys to use in Vim for the `{lhs}` of a mapping are the F keys, with or without Shift: `<F2>` to `<F12>` and `<S-F1>` to `<S-F12>`. (Some OSes, including mine, intercept `<Ctrl-Fn>` and `<Alt-Fn>`, which never reach an application program such as vim or gvim).

You can try other combinations of `<Ctrl>` + any key, but they may either not work everywhere (e.g. the terminal might not pass that key to Vim, or they might have unintended side effects (e.g. mapping `<C-I>` means also to map `<Tab>`).

This is a known issue, that has been discussed and might be implemented in the future to enable Vim to distinguish between various keys even in console mode. (e.g.

https://groups.google.com/d/msg/vim_dev/2bp9UdfZ63M/sajb9KM0pNYJ)

faq-20.6

20.6. How do I map the numeric keypad keys?

First make sure that the numeric keypad keys are passed to Vim. Next, you can use the following command to map the numeric keypad keys:

```
:map <kSomething> <your_command>
```

where, `<kSomething>` can be `kHome`, `kEnd`, `kPageUp`, `kPageDown`, `kPlus`, `kMinus`, `kDivide`, `kMultiply`, `kEnter`, etc.

For more information, read:

20.7. How do I create a mapping that works only in visual mode?

You can create mappings that work only in specific modes (normal, command, insert, visual, etc). To create a mapping that works only in the visual mode, use the `:vmap` command:

```
:vmap <F3> <your mapping here>
```

This mapping will work in visual and select mode. If you want the map to work only in visual mode (excluding select mode), use:

```
:xmap <F3> <your mapping here>
```

and to have the mapping only work in select mode (but not visual mode), use:

```
:smap <F3> <your mapping here>
```

For more information, read:

```
:vmap  
:xmap  
:smap  
map-modes  
40.1
```

20.8. How do I create a mapping that works only in normal and operator pending mode (but not in visual mode)?

Using `:map` creates a mapping that works in normal, visual+select mode and operator pending mode. You can use `:nmap` to have the mapping only work in normal mode and `:vmap` to have the mapping only be defined for visual and select mode or use `:omap` to have the mapping only defined in operator pending mode.

But if you want to have a mapping defined, that works in both operator pending mode and normal mode, but not in visual and select mode, you need to first define the mapping using `:map` and afterwards delete the mapping for visual and select mode:

```
:map <f3> <your mapping here>  
:vunmap <f3>
```

20.9. In a Vim script, how do I know which keys to use for my mappings, so that the mapped key will not collide with an already used key?

Vim uses most of the keys in the keyboard. You can use the `<leader>` prefix

in maps to define keys which will not overlap with Vim keys. For example:

```
:map <leader>S <C-W>s
:map <leader>j <C-W>j
:map <leader>k <C-W>k
```

where by default `<leader>` gets substituted with a backslash (`\`), so the user would enter

```
\s
\j
\k
```

to invoke the above map commands. The user can change the `mapleader` variable to be whatever they wanted:

```
:let mapleader = ","
```

When writing a plugin or other script, more often than not, it is advisable to use `:noremap` instead of `:map` to avoid side effects from user defined mappings.

For more information, read:

```
<Leader>
<LocalLeader>
write-plugin
```

[faq-20.10](#)

20.10. How do I map the escape key?

You can map the Escape key to some other key using the `:map` command. For example, the following command maps the escape key to **CTRL-O**.

```
:map <C-O> <Esc>
```

[faq-20.11](#)

20.11. How do I map a key to perform nothing?

You can map a key to `<Nop>` to perform nothing when the key is pressed. For example, with the following mappings, the `<F7>` key will do nothing when pressed.

```
:map <F7> <Nop>
:map! <F7> <Nop>
```

For more information, read:

```
<Nop>
:map
:map!
map-modes
```

[faq-20.12](#)

20.12. I want to use the `<Tab>` key to indent a block of text and `<Shift-Tab>` key to unindent a block of text. How do I map the keys to do this? This behavior is similar to textpad, visual studio, etc.

Use the following mapping:

```
:inoremap <S-Tab> <C-O><lt><lt>
:nnoremap <Tab> >>
:nnoremap <S-Tab> <lt><lt>
:vnoremap <Tab> >
:vnoremap <S-Tab> <lt>
```

Note, that the `<S-Tab>` mapping will work only if Vim receives the correct key sequence. This is mostly the case with GUI Vim.

For more information, read:

```
:inoremap
:nnoremap
:vnoremap
<S-Tab>
i_CTRL-O
>>
<<
<lt>
```

[faq-20.13](#)

20.13. In my mappings the special characters like `<CR>` are not recognized. How can I configure Vim to recognize special characters?

Check the value of the `'coptions'` option:

```
:set coptions?
```

If this option contains the `"<"` flag, then special characters will not be recognized in mappings. Remove the `"<"` flag from `'coptions'` option:

```
:set cpo-=<
```

Also, check the value of the `'compatible'` option:

```
:set compatible?
```

The `'compatible'` option must be reset:

```
:set nocompatible
```

For more information, read:

```
'coptions'
'compatible'
```

[faq-20.14](#)

20.14. How do I use the `"|"` to separate multiple commands in a map?

You can escape the "|" character using backslash (\) to use "|" in a map.

```
:map _l :!ls \| more<CR>
```

You can also try the following command:

```
:map _l :!ls <bar> more<CR>
```

There are also other ways to do this.

For more information, read:

[map_bar](#)

[faq-20.15](#)

20.15. If I have a mapping/abbreviation whose ending is the beginning of another mapping/abbreviation, how do I keep the first from expanding into the second one?

Instead of using the `:map lhs rhs` command, use the `:noremap lhs rhs` command. For abbreviations, use "noreabbrev lhs rhs". The "nore" prefix prevents the mapping or abbreviation from being expanded again.

For more information, read:

```
:noremap  
:noreabbrev
```

[faq-20.16](#)

20.16. Why does it take a second or more for Vim to process a key, sometimes when I press a key?

Make sure you have not defined a mapping for this key using the following command:

```
:map <key>
```

If a mapping is defined for this key and the mapped key contains more than one character, then Vim will wait for the next character to be pressed to determine whether it is the mapped key or not. For example, if you have mapped "ab", then if you press "a", Vim will wait for the next key to be pressed. If the next key is "b", Vim will execute the mapped sequence. Otherwise, Vim will proceed with the normal processing of "a" followed by the next key. If the **'timeout'** option is set (which is the default), then Vim will timeout after waiting for the period specified with the **'timeoutlen'** option (default is 1 second).

For more information, read:

```
map-typing  
'timeoutlen'  
'timeoutlen'  
'timeout'
```

```
'timeout'  
vt100-cursor-keys  
slow-fast-terminal
```

faq-20.17

20.17. How do I map a key to run an external command using a visually selected text?

You can use the `:vmap` command to map a key in the visual mode. In the mapped command sequence, you have to first yank the text. The yanked text is available in the `''` register. Now, you can use the contents of this register to run the external command. For example, to run the external command "perldoc" on a visually selected text, you can use the following mapping:

```
:vmap <F7> y:!exec "!perldoc '" . @" . "'"<CR>
```

If you want the mapping to work in the visual mode, but not with the highlighted text, you can use the following command:

```
:vmap <F7> :<C-U>!perldoc <word><CR>
```

The above mapping will use the word under the cursor instead of the highlighted text. **Note** the use of the `<C-U>` before invoking the "perldoc" external command. The `<C-U>` is used to erase the range of text selected in the visual mode and displayed on the command line. If the visual range is not removed using `<C-U>`, then the output from the external command will replace the visually selected text.

For more information, read:

```
:vmap  
quote_quote  
:let-register  
c_CTRL-U  
:!cmd
```

faq-20.18

20.18. How do I map the **CTRL-I** key while still retaining the functionality of the `<Tab>` key?

The **CTRL-I** key and the `<Tab>` key produce the same keycode, so Vim cannot distinguish between the **CTRL-I** and the `<Tab>` key. When you map the **CTRL-I** key, the `<Tab>` key is also mapped (and vice versa). The same restriction applies for the `CTRL-[` key and the `<Esc>` key.

For more information, read:

[keycodes](#)

faq-20.19

20.19. How do I define a map to accept a count?

Use the `@=` command to use an expression. For example,


```
nnoremap = @='3l'
```

Now you can specify a count to the "=" command.

```
complex-repeat
```

faq-20.20

20.20. How can I make my normal mode mapping work from within Insert Mode?

Mappings in normal mode can be executed after **CTRL-O** from insert mode as well but if there are more commands included in the mapping `{rhs}`, only the first one will be executed in normal mode and the rest of `{rhs}` will be printed literally in insert mode. One of ways to workaround this problem is to make `{rhs}` be one command, via wrapping it to the function. For example:

```
function GetFontNameOfFirstChar()  
normal! 0  
echo getfontname()  
endfunction
```

```
:nmap <F9> :call GetFontNameOfFirstChar()<CR>
```

A more technical and detailed solution to this problem follows and can be found at https://groups.google.com/group/vim_dev/msg/75f1f2dfc00908bb

Not every normal mode-mapping is automatically suitable for execution via **CTRL-O** from within insert mode; you need to explicitly design your mappings for that purpose.

The **CTRL-O** command allows execution of one normal mode command from within insert mode, then returns to insert mode. If a normal mode mapping concatenates multiple normal mode commands, this breaks down in temporary normal mode and literally inserts the second part of the command into the buffer instead. To support execution of normal mode mappings from within insert mode, these strategies can be used:

- 1) Instead of concatenating multiple normal mode commands, use one `:normal` mapping:

```
:nnoremap <silent> zC :<C-U>normal! zCVzC<CR>
```

- 2) Concatenate multiple Ex commands via `<Bar>` on the rhs:

```
:nnoremap zC :<C-U>call MyMap1()<Bar>call MyMap2()<CR>
```

- 3) Shadow normal mode mappings by insert mode mappings that re-enter normal mode, then invoke the normal mode mapping:

```
:nnoremap <silent> <SID>MyMap2 :<C-U>call MyMap2()<CR>  
:inoremap <silent> <script> <SID>MyMap2 <C-\><C-O><SID>MyMap2  
:nnoremap <silent> <script> zC <SID>MyMap1<SID>MyMap2
```

- 4) Normal mode mappings that consist of multiple Ex command lines (and where Ex commands cannot be concatenated via `<Bar>`) replace ``:<C-U>`` with `<SID>NM`; the `<SID>NM` mapping enters normal mode for one ex command line:

```
:nnoemap <silent> <SID>NM :<C-U>
:inoremap <silent> <SID>NM <C-\><C-O>:
:nnoemap <silent> <script> zC <SID>MyMap1<SID>NMcall MyMap2()<CR>
```

- 5) If none of the above is possible, at least force normal mode for subsequent commands via `CTRL-\ CTRL-N` to avoid accidental insertion of the remainder of the mapping.

```
:nnoemap zC zC<C-\><C-N>VzCzz
```

For more information, read:

```
i_CTRL-O
map_bar
i_CTRL-\_CTRL-O
CTRL-\_CTRL-N
```

=====

SECTION 21 - ABBREVIATIONS

faq-21

faq-21.1

21.1. How do I auto correct misspelled words?

You can auto correct misspelled words using abbreviations. For example, the following abbreviation can be used to correct "teh" with "the":

```
:abbreviate teh the
```

Vim supports abbreviations in insert mode, replace mode and command-line mode.

For more information, read:

```
24.7
abbreviations
Q_ab
```

faq-21.2

21.2. How do I create multi-line abbreviations?

You can create multi-line abbreviations by embedding the "`<CR>`" key code in the text:

```
iabbrev #c -----<CR>-- Date:<CR>--<CR>-----
```

With the above abbreviation, when you type #c, it will be expanded to the following text:

-- Date:
--

For more information, read:

[abbreviations](#)

faq-21.3

21.3. When my abbreviations are expanded, an additional space character is added at the end of the expanded text. How do I avoid this character?

To avoid an additional space character at the end of the expanded text, you can expand the abbreviation by pressing the **CTRL-]** key. The abbreviation will be expanded without adding a space character at the end.

Another alternative is to use the following function and command:

```
function! Eatchar(pat)
    let c = nr2char(getchar())
    return (c =~ a:pat) ? ' ' : c
endfunction
command! -nargs=+ Iabbr execute "iabbr" <q-args> . "<C-R>=Eatchar('\s')<CR>"
```

Now, define your abbreviations using the new "Iabbr" command instead of the builtin `:iabbrev` command. With this command, after expanding the abbreviated text, the next typed space character will be discarded.

For more information, read:

[abbreviations](#)

faq-21.4

21.4. How do I insert the current date/time stamp into the file?

You can use the `strftime()` function to insert the current data/time stamp in a file. For example, you can use the following abbreviation:

```
iabbrev dts <C-R>=strftime("%y/%m/%d %H:%M")<CR>
```

With this abbreviation, when you type `dts` in insert mode, it will be expanded to the date/time stamp.

Some other forms of the above abbreviation are listed below:

```
iabbrev mdyl <C-R>=strftime("%a %d %b %Y")<CR>
iabbrev mdys <C-R>=strftime("%y%m%d")<CR>
iabbrev mdyc <C-R>=strftime("%c")<CR>
iabbrev hml <C-R>=strftime("%d/%m/%y %H:%M:%S")<CR>
iabbrev hms <C-R>=strftime("%H:%M:%S")<CR>
```

For more information, read:

```
strftime()
i_CTRL-R
```

faq-21.5

21.5. How do I prevent an abbreviation from expanding in insert mode?

You can prevent an abbreviation from expanding in insert mode by typing **CTRL-V** before the character after the abbreviated word.

For more information, read:

```
abbreviations
```

faq-22

SECTION 22 - RECORD AND PLAYBACK

faq-22.1

22.1. How do I repeat an editing operation (insertion, deletion, paste, etc)?

You can repeat the last editing operation using the "." command. This will repeat the last simple change like a insert, delete, change, paste, etc.

For more information, read:

```
04.3
single-repeat
Q_re
```

faq-22.2

22.2. How I record and repeat a set of key sequences?

You can use the "q" command in normal mode to record a set of key sequences and store it in a register. For example, in the normal mode you can press q followed by a register name `{0-9a-zA-Z}` to start the recording. To end/stop the recording press q again. You can playback/repeat the recorded key sequences by pressing @ followed by the register name. e.g. @a.

Another approach is to start Vim with the "-w" command-line argument.

```
$ vim -w <file_name>
```

Vim will record all the characters typed in the session in the specified file "file_name". You can use the recorded file with the "-s" command line argument to play it back:

```
$ vim -s <file_name>
```

For more information, read:

```
10.1
recording
```

```
-w
-s
```

faq-22.3

22.3. How do I edit/modify a recorded set of key sequences?

The recorded key sequences are stored in a register. You can paste the contents of the register into a Vim buffer, edit the pasted text and again yank the text into the register. You can also use the `:let` command to modify the register. For example:

```
:let @a = "iHello World\<Esc>"
```

For more information, read:

```
recording
10.1
:let-register
<>
'coptions'
```

faq-22.4

22.4. How do I write recorded key sequences to a file?

The recorded key sequences are stored in a register. You can paste the contents of the register into a Vim buffer. Now you can save the buffer into a file. You can also modify the pasted text and again yank into the register to modify the recorded key sequence. For example, if you record a set of key sequences using `qa q`. The recorded key sequences are stored in the register "a". You can paste the contents of register "a" using "ap.

For more information, read:

```
recording
10.1
```

faq-22.5

22.5. I am using register 0 to record my key sequences (i.e. `q0 q`). In the recorded key sequences, I am yanking some text. After the first replay of the recorded key sequence, I am no longer able to play it back.

Register 0 contains the text from the last yank operation. In your recorded key sequence, when the yank is performed, register 0 is overwritten with the yanked text. So your recording stored in register 0 is lost. You have to use some other register.

For more information, read:

```
registers
```

=====
faq-23

SECTION 23 – AUTOCOMMANDS

faq-23.1

23.1. How do I execute a command when I try to modify a read-only file?

You can use the FileChangedRO autocommand event to execute a command when a read-only file is modified. For example, you can use this event to checkout a read-only file:

```
:autocmd FileChangedRO * call MyCheckoutFunction()
```

For more information, read:

[FileChangedRO](#)

faq-23.2

23.2. How do I execute a command every time when entering a buffer?

You can use the BufEnter autocommand event to execute a command every time when entering a buffer. For example:

```
:autocmd BufEnter *.c set formatoptions=croqt
```

For more information, read:

[BufEnter](#)

faq-23.3

23.3. How do I execute a command every time when entering a window?

You can use the WinEnter autocommand event to execute a command every time when entering a window. For example:

```
:autocmd WinEnter *.c call MyFunction()
```

For more information, read:

[WinEnter](#)

faq-23.4

23.4. From an autocmd, how can I determine the name of the file or the buffer number for which the autocommand is executed?

You can use the special words `<file>` or `<abuf>` in an autocmd to get the name of the file or the buffer number for which the autocommand is executed.

For more information, read:

```
:<file>  
:<abuf>  
:<amatch>
```

faq-23.5

23.5. How do I automatically save all the changed buffers whenever Vim loses focus?

You can define an autocommand for the FocusLost event which will save all the modified buffers whenever Vim loses focus:

```
:autocmd FocusLost * wall
```

For more information, read:

```
FocusLost  
:wall
```

[faq-23.6](#)

23.6. How do I execute/run a function when Vim exits to do some cleanup?

You can use VimLeave autocmd event to execute a function just before Vim exits. For example,

```
:autocmd VimLeave * call MyCleanupFunction()
```

For more information, read:

```
VimLeave
```

[faq-24](#)

SECTION 24 - SYNTAX HIGHLIGHT

[faq-24.1](#)

24.1. How do I turn off/on syntax highlighting?

By default, the Vim syntax highlighting is turned off. To enable the syntax highlighting, you can use one of the following commands:

```
:syntax enable
```

or

```
:syntax on
```

To disable the syntax highlighting, you can use the following command:

```
:syntax off
```

For more information, read:

```
06.1  
06.4  
:syntax-enable  
:syntax-on  
:syn-clear
```

[faq-24.2](#)

24.2. How do I change the background and foreground colors used by Vim?

Vim uses the "Normal" highlight group for the background and foreground colors. To change the foreground/background colors, you have to modify the "Normal" highlight group. For example, to set the background color to blue and foreground color to white, you can use

```
:highlight Normal ctermbg=blue ctermfg=white guibg=blue guifg=white
```

If you are using the Motif or the Athena version of the GUI Vim, then you can modify the foreground and background resource names in the .Xdefaults files to change the colors:

```
Vim.foreground:    Black
Vim.background:    Wheat
```

You can also use the "-foreground" and "-background" command-line arguments to specify the foreground and background colors. These arguments are supported only in the Motif or Athena versions:

```
$ gvim -foreground Black -background Wheat
```

For more information, read:

```
:highlight
.Xdefaults
-gui
```

[faq-24.3](#)

24.3. How do I change the highlight colors to suit a dark/light background?

You can set the **'background'** option to either "dark" or "light" to change the highlight colors to suit a dark/light background:

```
:set background=dark
```

For more information, read:

```
'background'
06.2
```

[faq-24.4](#)

24.4. How do I change the color of the line numbers displayed when the ``:set number`` command is used?

The line numbers displayed use the LineNr highlighting group. To display the current colors used, use

```
:hi LineNr
```

To change the color modify the LineNr highlight group. For example:

```
:hi linenr guifg=red guibg=black
```


This will give red numbers on a black background in GVIM.

For more information, read:

```
:highlight
```

[faq-24.5](#)

24.5. How do I change the background color used for a Visually selected block?

You can modify the "Visual" highlight group to change the color used for a visually selected block:

```
:highlight Visual guibg=red
```

For more information, read:

```
:highlight  
hl-Visual
```

[faq-24.6](#)

24.6. How do I highlight the special characters (tabs, trailing spaces, end of line, etc) displayed by the '**list**' option?

You can modify the "NonText" and "SpecialKey" highlight groups to highlight the special characters displayed by the '**list**' option:

```
:highlight NonText guibg=red  
:highlight SpecialKey guibg=green
```

The "NonText" highlighting group is used for "eol", "extends" and "precedes" settings in the '**listchars**' option. The "SpecialKey" highlighting group is used for the "tab" and "trail" settings.

For more information, read:

```
'listchars'  
hl-NonText  
hl-SpecialKey
```

[faq-24.7](#)

24.7. How do I specify a colorscheme in my .vimrc/.gvimrc file, so that Vim uses the specified colorscheme every time?

You can specify the color scheme using the `:colorscheme` command in your .vimrc or .gvimrc file:

```
colorscheme evening
```

For more information, read:

```
:colorscheme
```

[faq-24.8](#)

24.8. Vim syntax highlighting is broken. When I am editing a file, some parts of the file is not syntax highlighted or syntax highlighted incorrectly.

Vim doesn't read the whole file to parse the text for syntax highlighting. It starts parsing wherever you are viewing the file. That saves a lot of time, but sometimes the colors are wrong. A simple fix is refreshing the screen using the **CTRL-L** key. Or scroll back a bit and then forward again. You can also use the command:

```
:syntax sync fromstart
```

Note that this might considerably slow down the screen refreshing.

For more information, read:

```
:syn-sync  
:syn-sync-first
```

[faq-24.9](#)

24.9. Is there a built-in function to syntax-highlight the corresponding matching bracket?

Yes. Vim includes the matchparen Plugin as standard plugin that is enabled by default. Whenever the cursor moves over an item defined with the **'matchpairs'** option, Vim will highlight the corresponding bracket using the MatchParen highlighting group.

However, if the corresponding parenthesis is not visible in the current window, the cursor won't jump to it.

The matchit plugin provides a similar function, that lets the cursor jump to related items (e.g. "if", "else", "endif" items) and skips matches in comments. This uses the % command to jump to corresponding items. Though both plugins provide similar functions they are unrelated and work differently.

For more information, read:

```
matchparen  
'matchpairs'  
matchit-install  
matchit-intro
```

[faq-24.10](#)

24.10. How do I turn off the C comment syntax highlighting?

You can use the following command to turn off C comment syntax highlighting:

```
:highlight clear comment
```

For more information, read:

ft-c-syntax

faq-24.11

24.11. How do I add my own syntax extensions to the standard syntax files supplied with Vim?

You should not modify the syntax files supplied with Vim to add your extensions. When you install the next version of Vim, you will lose your changes. Instead you should create a file under the `~/.vim/after/syntax` directory with the same name as the original syntax file and add your additions to this file.

For more information, read:

```
mysyntaxfile-add  
'runtimepath'
```

faq-24.12

24.12. How do I replace a standard syntax file that comes with the Vim distribution with my own syntax file?

You can replace a standard syntax file that comes with the Vim distribution by creating a file with the same name as the original syntax file and placing it in the vim runtime syntax (`~/.vim/syntax`) directory. For example, to replace the `c.vim` syntax file in a Unix system, place the new `c.vim` in the `~/.vim/syntax` directory. In a MS-Windows system, place the new syntax file in the `$HOME/vimfiles/syntax` or `$VIM/vimfiles/syntax` directory.

For more information, read:

```
mysyntaxfile-replace  
44.11  
mysyntaxfile
```

faq-24.13

24.13. How do I highlight all the characters after a particular column?

You can use the `:match` command to highlight all the characters after a particular column:

```
:match Todo '\%>75v.\+'
```

This will highlight all the characters after the 75th column.

You can also set the `'colorcolumn'` option to highlight a particular column:

```
:set colorcolumn=+2
```

which highlights 2 columns after the current `'textwidth'` setting (alternatively, you can use the exact column number).

For more information, read:

```
:match
/\%v
/\+
/\.
'colorcolumn'
```

faq-24.14

24.14. How do I convert a source file (.c, .h, etc) with the Vim syntax highlighting into a HTML file?

You can use the 2html.vim script to convert a source file into a HTML file with the Vim syntax highlighting. Use the following command:

```
:TOhtml
```

For more information, read:

```
convert-to-HTML
:TOhtml
```

faq-24.15

24.15. How do I list the definition of all the current highlight groups?

You can list the definition of all the current highlight groups using the `:highlight` (without any arguments) ex command.

For more information, read:

```
:highlight
```

faq-24.16

24.16. How can I embed one syntax highlighting language into another one?

It is possible to include one syntax highlighting into another one, however most of the currently deployed syntax highlighting scripts are not prepared to be included into another syntax script.

You can however create your own custom script to define your own regions, which will be highlighted with a different language.

See the wiki for a comprehensive solution:

http://vim.wikia.com/wiki/Different_syntax_highlighting_within_regions_of_a_file

For more information, read:

```
:syn-include
sh-awk
```

=====

SECTION 25 - VIM SCRIPT WRITING

faq-25

faq-25.1

25.1. How do I list the names of all the scripts sourced by Vim?

You can use the `:scriptnames` command to list the names of all the scripts sourced by Vim:

```
:scriptnames
```

For more information, read:

```
:scriptnames
```

[faq-25.2](#)

25.2. How do I debug Vim scripts?

Vim has built-in support for a primitive debugger to debug Vim plugins and scripts. Using this debugger you can set breakpoints and step through the plugin functions.

For more information, read:

```
debug-scripts  
-D
```

[faq-25.3](#)

25.3. How do I locate the script/plugin which sets a Vim option?

You can use the `:verbose` command to locate the plugin/script which last modified a Vim option. For example:

```
:verbose set textwidth?
```

For more information, read:

```
:set-verbose  
:verbose
```

[faq-25.4](#)

25.4. I am getting some error/informational messages from Vim (possibly when running a script), the messages are cleared immediately. How do I display the messages again?

You can use the `:messages` command to display the previous messages.

```
:messages
```

For more information, read:

```
:messages  
:echoerr  
:echomsg  
message-history
```

[faq-25.5](#)

25.5. How do I save and restore a plugin specific information across Vim

invocations?

Vim will save and restore global variables that start with an uppercase letter and don't contain a lower case letter. For this to work, the `'viminfo'` option must contain the `!"` flag. Vim will store the variables in the viminfo file.

For more information, read:

```
'viminfo'  
viminfo-file  
variables
```

[faq-25.6](#)

25.6. How do I start insert mode from a Vim function?

You can use the `:startinsert` command to start the insert mode from inside a Vim function.

For more information, read:

```
:startinsert
```

[faq-25.7](#)

25.7. How do I change the cursor position from within a Vim function?

You can use the `cursor()` function to position the cursor.

```
call cursor(lnum, col)
```

Alternatively, use the `setpos()` function:

```
call setpos('.', [bufnum, lnum, col, off])
```

which set's the cursor in the buffer `bufnum` to line `lnum`, column `col` and offset for `'virtualedit'`. You can use the `getpos()` function, to return a list with these values, that can then be fed back to the `setpos()` function.

If you want to save and restore the viewpoint on a window, use the `winsaveview()` and `winrestview()` function calls.

You can also use the following command to change the cursor position:

```
exe "normal! " . lnum . "G" . col . "|"
```

For more information, read:

```
cursor()  
bar  
getpos()  
setpos()  
winsaveview()  
winrestview()
```

faq-25.8

25.8. How do I check the value of an environment variable in the .vimrc file?

You can use prefix the environment variable name with the "\$" character to use it from a Vim script/function. You can refer to the value of an environment variable using the \$env_var syntax:

```
if $EDITOR == 'vi'
endif
```

For more information, read:

[expr-env](#)

faq-25.9

25.9. How do I check whether an environment variable is set or not from a Vim function?

You can use the exists() function to check for the existence of an environment variable.

```
if exists("$MY_ENV_VAR")
endif
```

For more information, read:

[exists\(\)](#)
[expr-env](#)

faq-25.10

25.10. How do I call/use the Vim built-in functions?

You can use the `:call` command to invoke a Vim built-in function:

```
:call cursor(10,20)
```

You can use the `:echo` command to echo the value returned by a function:

```
:echo char2nr('a')
```

You can use the `:let` command to assign the value returned by a function to a variable:

```
:let a = getline('.')
```

To store the return value from a function into a Vim register, you can use the following command:

```
:let @a = system('ls')
```

The above command will store the output of the "ls" command into the register "a".

For more information, read:

```
:call
:echo
:let
:let-register
user-functions
usr_41.txt
```

[faq-25.11](#)

25.11. I am using some normal mode commands in my Vim script. How do I avoid using the user-defined mappings for these normal mode commands and use the standard Vim functionality for these normal mode commands?

You can use the `:normal!` command in your script to invoke a normal-mode command. This will use the standard functionality of the normal mode command and will not use the user-defined mapping.

For more information, read:

```
:normal
```

[faq-25.12](#)

25.12. How do I get the current visually selected text into a Vim variable or register?

You can get the current visually selected text into a Vim variable by yanking the text into Vim register and then assigning the contents of the register into the variable:

```
:normal! gvy
:let myvar = @"
```

The above command copies the visually selected text into the variable "myvar".

You can also use the command:

```
:normal! gv"*y
```

In the above command, `gv` reselects the last visually selected text and the rest of the command copies the selected text into the `*` (clipboard) register. Alternatively, you can set the "a" flag in the `'guioptions'` option to automatically copy a visually selected text into the `*` register. To do this as part of a visual map, you can use a command similar to the one shown below:

```
:vmap <F3> "*y:call ...
```

For more information, read:

```
gv
```



```
:normal
:let-@
quotestar
clipboard
registers
```

faq-25.13

25.13. I have some text in a Vim variable "myvar". I would like to use this variable in a `:s` substitute command to replace a text "mytext". How do I do this?

You can use the `:execute` command to evaluate the variable:

```
:execute '%s/mytext/' . myvar . '/'
```

For more information, read:

```
:execute
```

You can also use "\=" in the substitute command to evaluate the variable:

```
:%s/mytext/\=myvar/
```

For more information, read:

```
sub-replace-special
```

faq-25.14

25.14. A Vim variable (bno) contains a buffer number. How do I use this variable to open the corresponding buffer?

The `:buffer` command will not accept a variable name. It accepts only a buffer number or buffer name. You have to use the `:execute` command to evaluate the variable into the corresponding value. For example:

```
:execute "buffer " . bno
```

For more information, read:

```
:execute
```

faq-25.15

25.15. How do I store the value of a Vim option into a Vim variable?

You can prefix the option name with the "&" character and assign the option value to a Vim variable using the `:let` command. For example, to store the value of the **'textwidth'** option into the Vim variable "old_tw", you can use the following command:

```
:let old_tw = &tw
```

To explicitly save buffer local options, use the prefix "l:"

```
:let old_tw = &l:tw
```

If you want to explicitly select the global option, use the "g:" prefix to the option name.

To do the opposite, to set the `'textwidth'` option with the value stored in the "old_tw" variable, you can use the following command:

```
:let &tw = old_tw
```

For more information, read:

```
expr-option  
:let-option
```

faq-25.16

25.16. I have copied and inserted some text into a buffer from a Vim function. How do I indent the inserted text from the Vim function?

You can use the following command to format the just inserted text:

```
:normal '[=]
```

For more information, read:

```
'[  
' ]  
=  
:normal
```

faq-25.17

25.17. How do I get the character under the cursor from a Vim script?

You can use the `getline()` function and use string index `[]` to get the character:

```
:echo getline(".")[col(".") - 1]
```

In the above command, `getline(".")` returns the text in the current line. The indexing of the string starts at zero, and you can get a single character in a string by its index with the "string[index]" notation. The `col(".")` returns the column of the cursor position; the adjustment is to get the right character of the string. However, this does NOT work with multibyte characters as this command only returns the byte index.

Alternatively, you can use the following sequence of commands to get the character under the cursor:

```
normal! vy  
let ch=@"
```

Note, that the above commands will change the '<' and '>' marks.

For more information, read:

```
getline()  
col()  
expr-[]
```

faq-25.18

25.18. How do I get the name of the current file without the extension?

You can get the name of the current file without the extension using:

```
:echo expand("%:r")
```

With some commands, you can use the file name modifiers directly:

```
:cd %:p:h  
:!gcc -o %:r.o %  
:!xpdf %<.pdf
```

For more information, read:

```
filename-modifiers  
expand()  
cmdline-special  
fnamemodify()
```

faq-25.19

25.19. How do I get the basename of the current file?

You can use the ":t" filename modifier to get the basename of the current file:

```
:echo expand("%:t")
```

For more information, read:

```
filename-modifiers
```

faq-25.20

25.20. How do I get the output from a Vim function into the current buffer?

You can insert the return value from a function using the following command in insert mode:

```
<C-R>=MyFunc()
```

Note, that this will only insert the return value of the function.

For more information, read:

```
i_CTRL-R  
i_CTRL-R_CTRL-R  
i_CTRL-R_CTRL-O  
expression
```

faq-25.21

25.21. How do I call external programs from a Vim function?

There are several ways to call external programs from a Vim function. You can use the builtin `system()` function to invoke external programs and get the result:

```
:let output = system("ls")
```

You can also use `!` ex-command to run an external command.

For more information, read:

```
system()  
:!  
10.9
```

[faq-25.22](#)

25.22. How do I get the return status of a program executed using the ``:!'`` command?

You can use the predefined Vim `v:shell_error` variable to get the return status of the last run shell command.

For more information, read:

```
v:shell_error
```

[faq-25.23](#)

25.23. How do I determine whether the current buffer is modified or not?

You can check the value of the `'modified'` option to determine whether the current buffer is modified:

```
:set modified?
```

From a Vim script, you can check the value of the `'modified'` option:

```
if &modified  
    echo "File is modified"  
endif
```

For more information, read:

```
'modified'
```

[faq-25.24](#)

25.24. I would like to use the carriage return character in a normal command from a Vim script. How do I specify the carriage return character?

You can use the ``:execute`` command to specify the special (control) character in a normal mode command:

```
:execute "normal \<CR>"
```

```
:execute "normal ixxx\<Esc>"
```

For more information, read:

```
:execute  
expr-quote
```

faq-25.25

25.25. How do I split long lines in a Vim script?

You can split long lines in a Vim script by inserting the backslash character ("\") at the start of the next line. For example,

```
set comments=sr:/*,mb:*,el:*/,  
    \://,  
    \b:##,  
    \:%,  
    \n:>,  
    \fb:-
```

For more information, read:

```
line-continuation
```

faq-25.26

25.26. When I try to "execute" my function using the `:execute Myfunc()` command, the cursor is moved to the top of the current buffer. Why?

The `:execute` command runs the ex command specified by the argument. In the case of the following command:

```
:execute Myfunc()
```

The call to `Myfunc()` will return 0. The `:execute` command will run the ex command `:0`, which moves the cursor to the top of the file. To call a Vim function, you should use the `:call` command instead of the `:execute` command:

```
:call Myfunc()
```

For more information, read:

```
:call  
:execute  
:echo  
user-functions  
41.5  
41.6  
cmdline-lines
```

faq-25.27

25.27. How do I source/execute the contents of a register?

If you have yanked a set of Vim commands into a Vim register (for example register "a"), then you can source the contents of the register using one of the following commands:

```
:@a  
or  
:exe @a
```

For more information, read:

[:@](#)

[faq-25.28](#)

25.28. After calling a Vim function or a mapping, when I press the "u" key to undo the last change, Vim undoes all the changes made by the mapping/function. Why?

When you call a function or a mapping, all the operations performed by the function/mapping are treated as one single operation. When you undo the last operation by pressing "u", all the changes made by the function/mapping are reversed.

For more information, read:

[undo-redo](#)
[:map-undo](#)

[faq-25.29](#)

25.29. How can I call a function defined with s: (script local function) from another script/plugin?

The s: prefix for a Vim function name is used to create a script local function. A script local function can be called only from within that script and cannot be called from other scripts. To define a function in a script/plugin, so that it can be called from other plugins/scripts, define the function without the s: prefix.

For more information, read:

[script-variable](#)
[script-local](#)
[:scriptnames](#)

[faq-25.30](#)

25.30. Is it possible to un-source a sourced script? In other words, reverse all the commands executed by sourcing a script.

No. It is not possible to reverse or undo all the commands executed by sourcing a script.

For more information, read:

[:source](#)

SECTION 26 - PLUGINS

faq-26.1

26.1. How do I set different options for different types of files?

You can create filetype plugins to set different options for different types of files. You should first enable filetype plugins using the command:

```
:filetype plugin on
```

A filetype plugin is a vim script that is loaded whenever Vim opens or creates a file of that type. For example, to ensure that the **'textwidth'** option is set to 80 when editing a C program (filetype "c"), create one of the following files:

```
~/vim/ftplugin/c.vim (Unix)
%HOME%\vimfiles\ftplugin\c.vim (Windows)
```

with the following text in it:

```
setlocal textwidth=80
```

You can also use autocommands to set specific options when editing specific type of files. For example, to set the **'textwidth'** option to 75 for only *.txt files, you can use the following autocmd:

```
autocmd BufRead *.txt setlocal textwidth=80
```

For more information, read:

```
filetype-plugin
add-filetype-plugin
:autocmd
40.3
```

faq-26.2

26.2. I have downloaded a Vim plugin or a syntax file or a indent file, or a color scheme or a filetype plugin from the web. Where should I copy these files so that Vim will find them?

You can place the Vim runtime files (plugins, syntax files, indent files, color schemes, filetype plugins, etc) under one of the directories specified in the **'runtimepath'** option. To determine the current value of the **'runtimepath'** option, use the following command:

```
:set runtimepath
```

For Unix systems, this is usually the "\$HOME/.vim" directory. For MS-Windows systems, this is usually the \$VIM\vimfiles or \$HOME\vimfiles directory. Depending on the type of the runtime file, you have to place it under a specific directory under the above runtime directory. The names of the directories are listed below:

name	description
colors/	color scheme files
compiler/	compiler files
doc/	documentation
ftplugin/	filetype plugins
indent/	indent scripts
keymap/	key mapping files
lang/	menu translations
plugin/	plugin scripts
syntax/	syntax files
tutor/	files for vimtutor

For more information, read:

```
your-runtime-dir
'runtimepath'
:runtime
```

faq-26.3

26.3. How do I extend an existing filetype plugin?

You can extend an existing filetype plugin by creating a file in the `after/` directory in any of the `'runtimepath'` directories.

- for small changes to be done after (and in addition to) what is already done by the ftplugin installed with Vim, use an `after-`directory, as follows (replacing `foobar` by the `'filetype'` of the concerned files):
 - For changes private to one user:
 - on Windows:


```
$HOME/vimfiles/after/ftplugin/foobar.vim
```
 - on Unix-like OSes:


```
$HOME/.vim/after/ftplugin/foobar.vim
```
 - For changes affecting all users on the system:


```
$VIM/vimfiles/after/ftplugin/foobar.vim
```
- when replacing the whole filetype-plugin by a different version, or when installing a new ftplugin for some filetype not yet supported by Vim out of the box: use the same paths without the `after/` in them. In that case you should place near the start of your plugin an `"if... finish... endif... let"` block like the one in the plugins distributed with Vim.

All the above paths are given in Vim terminology (which is similar to Unix terminology, but is understood even by Vim for Windows); they don't exist by default, so the first time you need them you will have to create them using `mkdir` (on any OS including DOS/Windows) or `md` (on DOS/Windows only). `$VIM` and, on DOS/Windows, `$HOME`, do not necessarily exist outside Vim. If `$HOME` has no value (or no valid value) inside Vim, you can use `$VIM` instead; but on any but possibly very old versions of Windows, `$HOMEDRIVE` and `$HOMEPATH` are defined by the system, and if

\$HOME is undefined at Vim startup, Vim will set it by expanding \$HOMEDRIVE\$HOMEPATH before sourcing your vimrc. To know which values Vim uses, you can type (in a running Vim):

```
:echo $VIM
:echo $HOME
```

If you placed the file in the after/ftplugin runtime directory, then Vim will first source the existing filetype plugin file and then will source the new file. If you placed the file in the \$VIMRTUNTIME/ftplugin runtime directory, then Vim will first source the new file and then will source the existing filetype plugin file.

For more information, read:

```
ftplugin-override
filetype-plugin
add-filetype-plugin
'runtimepath'
```

faq-26.4

26.4. How do I turn off loading the Vim plugins?

You can reset the **'loadplugins'** option to turn off loading the plugins:

```
:set noloadplugins
```

You can also specify the "--noplugin" command line argument to stop loading the plugins:

```
$ vim --noplugin
```

For more information, read:

```
'loadplugins'
--noplugin
load-plugins
```

faq-26.5

26.5. How do I turn on/off loading the filetype plugins?

By default, Vim will not load the filetype plugins. You can configure Vim to load filetype plugins using the command:

```
filetype plugin on
```

You can turn off loading the filetype plugins using:

```
filetype plugin off
```

For more information, read:

```
:filetype-plugin-on
```

```
:filetype-plugin-off
:filetype
```

faq-26.6

26.6. How do I override settings made in a file type plugin in the global ftplugin directory for all the file types?

You can use an autocommand triggered on the FileType event:

```
au FileType * set formatoptions=xyz
```

This should at least be after "filetype on" in your vimrc. Best is to put it in your "myfiletypefile" file, so that it's always last.

If you want to override a setting for a particular filetype, then create a file with the same name as the original filetype plugin in the ~/.vim/after/ftplugin directory. For example, to override a setting in the c.vim filetype plugin, create a c.vim file in the ~/.vim/after/ftplugin directory and add your preferences in this file.

For more information, read:

```
ftplugin-override
ftplugins
myfiletypefile
```

faq-26.7

26.7. How do I disable the Vim directory browser plugin?

To disable the directory browsing Vim plugin, add the following line to your .vimrc file:

```
let g:loaded_netrw = 1
```

For more information, read:

```
netrw
```

faq-26.8

26.8. How do I set the filetype option for files with names matching a particular pattern or depending on the file extension?

You can set the **'filetype'** option for files with names matching a particular pattern using an autocmd. For example, to set the **'filetype'** option to "c" for all files with extension ".x", you can use the following autocmd:

```
autocmd! BufRead,BufNewFile *.x      setfiletype c
```

A better alternative to the above approach is to create a filetype.vim file in the ~/.vim directory (or in one of the directories specified in the **'runtimepath'** option) and add the following lines:

```
" my filetype file
```

```
if exists("did_load_filetypes")
  finish
endif
augroup filetypedetect
  au! BufRead,BufNewFile *.x      setfiletype c
augroup END
```

For more information, read:

```
new-filetype
43.2
:setfiletype
```

SECTION 27 - EDITING PROGRAM FILES

faq-27

27.1. How do I enable automatic indentation for C/C++ files?

faq-27.1

You can enable file-type based indentation using:

```
:filetype indent on
```

If you want to only enable automatic C indentation, then use:

```
:set cindent
```

For more information, read:

```
'cindent'
C-indenting
filetype
```

faq-27.2

27.2. How do I configure the indentation used for C/C++ files?

You can configure the Vim C indentation by modifying the value of the **'cinoptions'**, **'cinkeys'** and **'cinwords'** options.

For more information, read:

```
'cindent'
'cinoptions'
'cinkeys'
'cinwords'
C-indenting
cinoptions-values
'smartindent'
```

faq-27.3

27.3. How do I turn off the automatic indentation feature?

By default, the automatic indentation is not turned on. You must have

configured Vim to do automatic indentation in either `.vimrc` or `.gvimrc` files. You can disable automatic indentation using either,

```
:filetype indent off
```

or

```
:set nocindent
```

Also, check the setting for the following options:

```
:set autoindent?  
:set smartindent?  
:set indentexpr?
```

For more information, read:

```
'cindent'  
:filetype-indent-off  
'autoindent'  
'smartindent'  
'indentexpr'
```

[faq-27.4](#)

27.4. How do I change the number of space characters used for the automatic indentation?

You can modify the `'shiftwidth'` option to change the number of space characters used for the automatic indentation:

```
:set shiftwidth=4
```

For more information, read:

```
'shiftwidth'
```

[faq-27.5](#)

27.5. I am editing a C program using Vim. How do I display the definition of a macro or a variable?

You can use the `"[d` command to display the definition of a macro, `"[i` command to display the definition of a variable, `"gd` to goto the local declaration of a variable and `"gD` to go to the global Declaration.

For more information, read:

```
[d  
[i  
gd  
gD  
include-search  
29.4  
29.5
```

faq-27.6

27.6. I am editing a C program using Vim. How do I jump to the beginning or end of a code block from within the block?

You can use "[{" command to jump to the beginning of the code block and "]}" to jump to the end of the code block from inside the block.

For more information, read:

```
[{  
}]  
various-motions
```

faq-27.7

27.7. When editing C++ files and when inserting new lines above or below a comment (//) line, Vim automatically inserts the C++ comment character (//) at the beginning of the line. How do I disable this?

This automatic insertion of the comment leader (//) when new lines are added is controlled by three flags in the **'formatoptions'** option: "c", "r" and "o". "c" enables auto-wrapping of comment lines when typing extends beyond the right margin. "r" enables the automatic insertion of the comment leader when <Enter> is pressed while editing a comment line. "o" enables the automatic insertion of the comment leader when a new line is opened above or below an existing comment line by typing O or o in Normal mode.

You can stop Vim from automatically inserting the comment leader when typing <Enter> within a comment or when opening a new line by removing the "r" and "o" flags from **'formatoptions'**.

```
:set formatoptions-=r  
:set formatoptions-=o
```

The default filetype plugin for C and C++ files (\$VIMRUNTIME/ftplugin/c.vim) adds the "r" and "o" flags to the **'formatoptions'** option. If you want to override this for C++ files, then you can add the above lines to the ~/.vim/after/ftplugin/cpp.vim file.

For more information, read:

```
formatoptions  
30.6  
format-comments  
filetype-plugins  
ftplugin-override
```

faq-27.8

27.8. How do I add the comment character "#" to a set of lines at the beginning of each line?

First, select the first character in all the lines using visual block mode (CTRL-V). Press "I" to start inserting characters at the beginning of the

line. Enter the comment character and then stop the insert mode by pressing `<Esc>`. Vim will automatically insert the entered characters at the beginning of all the selected lines.

For more information, read:

```
visual-block  
blockwise-operators  
v_b_I
```

faq-27.9

27.9. How do I edit a header file with the same name as the corresponding C source file?

You can use the following command to edit a header file with the same name as the corresponding C source file:

```
:e %:~:~:~.h
```

You can use the following command to edit the file in a new split window:

```
:sp %:~:~:~.h
```

In the above commands, the percent sign expands to the name of the current file. The ``:t`` modifier extracts the tail (last component) of the filename. The ``:r`` modifier extracts the root of the filename. The `.h` is appended to the resulting name to get the header filename.

Another approach is to use the following command:

```
:sfind %:~:~:~.h
```

This command will search for the header file in the directories specified in the `'path'` option.

For more information, read:

```
cmdline-special  
filename-modifiers  
:sfind  
'path'
```

faq-27.10

27.10. How do I automatically insert comment leaders while typing comments?

To automatically insert comment leaders while typing comments, add the `"r"` and `"o"` flags to the `'formatoptions'` option.

```
:set formatoptions+=ro
```

You may also want to add the `"c"` flag to auto-wrap comments using the `'textwidth'` option setting and the `"q"` flag to format comments with the `"gq"` command:

```
:set formatoptions=croq
```

For more information, read:

```
30.6
format-comments
'comments'
fo-table
```

faq-28

SECTION 28 - QUICKFIX

faq-28.1

28.1. How do I build programs from Vim?

You can use the `:make` command to build programs from Vim. The `:make` command runs the program specified by the `'makeprg'` option.

For more information, read:

```
30.1
:make_makeprg
'makeprg'
'makekeef'
:make
quickfix
```

faq-28.2

28.2. When I run the make command in Vim I get the errors listed as the compiler compiles the program. When it finishes this list disappears and I have to use the `:clist` command to see the error message again. Is there any other way to see these error messages?

You can use the `:copen` or `:cwindow` command to open the quickfix window that contains the compiler output. You can select different error lines from this window and jump to the corresponding line in the source code.

For more information, read:

```
:copen
:cwindow
quickfix
```

faq-28.3

28.3. How can I perform a command for each item in the quickfix/location list?

Starting from Vim 7.4.858 Vim provides the new commands `:cfdo`, `:cdo`, `:lfdo` and `:ldo`. They work by iterating over all items in the quickfix list and performing a command on each. The difference is, that the `:lfdo` and `:ldo` commands iterate over the location list entries, while the `:cfdo` and `:cdo` commands operate on the items in the quickfix list. Also, the `:cfdo` and `:lfdo` operate on all different files, while the `:cdo` and

`:ldo` commands operate on each item in the quickfix/location list.

For example you could vimgrep all C files in the current directory for a search string "Foobar":

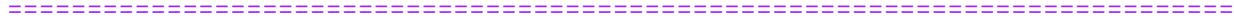
```
:vimgrep /Foobar/ *.c
```

and as this populates your quickfix list, you could simply replace all occurrences by using:

```
:cdo :%s/Foobar/Foobaz | upd
```

For more information, read:

```
:cfdo  
:cdo
```



SECTION 29 - FOLDING

faq-29

faq-29.1

29.1. How do I extend the Vim folding support?

You can use the **'foldexpr'** option to fold using a user specified function. For example, to fold subroutines of the following form into a single line:

```
sub foo {  
  my $barf;  
  $barf = 3;  
  return $barf;  
}
```

You can use the following commands:

```
set foldmethod=expr  
set foldexpr=MyFoldExpr(v:lnum)  
fun! MyFoldExpr(line)  
  let str = getline(a:line)  
  if str =~ '^sub\>'  
    return '1'  
  elseif str =~ '^}'  
    return '<1'  
  else  
    return foldlevel(a:line - 1)  
  endif  
endfun
```

For more information, read:

```
'foldexpr'  
fold-expr
```

faq-29.2

29.2. When I enable folding by setting the **'foldmethod'** option, all the folds are closed. How do I prevent this?

You can set the **'foldlevelstart'** option to a particular value to close only folds above the specified value.

```
:set foldlevelstart=99
```

For more information, read:

```
'foldlevelstart'  
'foldlevel'  
fold-foldlevel
```

[faq-29.3](#)

29.3. How do I control how many folds will be opened when I start editing a file?

You can modify the **'foldlevelstart'** option to control the number of folds that will be opened when you start editing a file. To start editing with all the folds closed:

```
:set foldlevelstart=0
```

To start editing with all the folds opened, you can use

```
:set foldlevelstart=999
```

For more information, read:

```
'foldlevelstart'
```

[faq-29.4](#)

29.4. How do I open and close folds using the mouse?

You can click on the + and - characters displayed at the leftmost column to open and close fold. For this to work, you have to set the **'foldcolumn'** to a value greater than zero:

```
:set foldcolumn=2
```

For more information, read:

```
'foldcolumn'
```

[faq-29.5](#)

29.5. How do I change the text displayed for a closed fold?

You can use the **'foldtext'** option to change the text displayed for a closed fold.

For more information, read:

```
'foldtext'
```

fold-foldtext
'fillchars'

faq-29.6

29.6. How do I store and restore manually created folds across different Vim invocations?

You can use the ``:mkview`` command to store manually created folds. Later, you can use the ``:loadview`` command to restore the folds. For this to work, the `'viewoptions'` must contain "folds".

For more information, read:

28.4
:mkview
:loadview
'viewoptions'
'viewdir'
:mksession
'sessionoptions'

faq-29.7

29.7. I have enabled syntax based folding. Why is Vim so slow?

Syntax based folding is currently rather slow in Vim and will possibly slow down Vim considerably. There is an issue in the todo list to fix this, but the todo list is rather long and it may take a while until this will be fixed.

You can find the issue in the todo list, if you read

[todo.txt](#)

followed by a search for "folding with `'foldmethod'`"

A workaround is to temporarily set the `foldmethod` to manual while in insert mode. This is described in the wiki at:

http://vim.wikia.com/wiki/Keep_folds_closed_while_inserting_text

=====

faq-30

SECTION 30 - VIM WITH EXTERNAL APPLICATIONS

faq-30.1

30.1. Can I run a shell inside a Vim window?

Since Version 8.1 Vim comes with a terminal window included. It allows to run a shell inside an ordinary Vim window (e.g. split) asynchronously and interact with the shell using the normal Vim commands.

When the focus is in the terminal window, typed keys will be sent to the job and is called terminal mode. You can click outside of the terminal window to move keyboard focus elsewhere, alternatively one can

use **CTRL-W** to navigate between different Vim windows. To feed **CTRL-W** into the terminal, one needs to use **CTRL-W** .

To map keys specifically for terminal mode, use the new `:tmap` command. After typing **CTRL-W** the terminal window will switch to Terminal-Normal mode (this can be used to move the cursor around, scroll the window, etc. Just like normal mode).

To interact between the terminal and Vim, Vim implements several interfaces using `term_sendkeys()`, `terminal-api` and the client-server mechanism.

For more information, read:

```
terminal
mapmode-t
Terminal-Normal
terminal-communication
```

faq-30.2

30.2. How do I pass the word under the cursor to an external command?

You can use the special keyword `<word>` to pass the word under the cursor to an external command. For example:

```
:!dict <word>
```

For more information, read:

```
:<word>
```

faq-30.3

30.3. How do I get the output of a shell command into a Vim buffer?

You can use the `:r !`` command to get the output of a shell command into a Vim buffer. For example, to insert the output of the "ls" shell command, you can use the following command:

```
:r !ls
```

To insert the output of the shell command above the first line use the following command:

```
:0r !ls
```

For more information, read:

```
:r!
```

faq-30.4

30.4. How do I pipe the contents of the current buffer to an external command and replace the contents of the buffer with the output from the command?

You can use the `:!` command to pipe the contents of the current buffer to an external command and replace the contents of the buffer with the output from the command. For example, to sort the contents of the current buffer, using the Unix `sort` command, you can use the following command:

```
:%!sort
```

To sort only lines 10-20, you can use the following command

```
:10,20!sort
```

Also, if you want to pipe a buffer to an external command but not put the results back in the buffer, you can use

```
:w !sort
```

The above command will pipe the entire buffer to the `sort` command. Note, that the space between the `"w"` and the `"!"` is critical. To pipe only a range of lines, you can use

```
:10,20w !sort
```

The above command will pipe the lines 10-20 to the `sort` command.

For more information, read:

```
:range!  
10.9  
:w_c
```

[faq-30.5](#)

30.5. How do I sort a section of my file?

You use the `:sort` command like this:

```
:5,100sort
```

Using the `:sort` command provides many options, you can sort numerical on the first found decimal number using:

```
:%sort n
```

Or you can specify to sort on the text, starting at virtual column 8:

```
:%sort /*\%8v/
```

Alternatively can pipe a section of the file to the Unix `"sort"` utility to sort the file. For example:

```
:5,100!sort
```

You can also use a visual block, and use the `"!sort"` command on the selected block.

See also:

```
:sort  
filter
```

[faq-30.6](#)

30.6. How do I use Vim as a pager?

You can use Vim as a pager using the `$VIMRUNTIME/macros/less.sh` shell script, supplied as part of the standard Vim distribution. This shell script uses the `$VIMRUNTIME/macros/less.vim` Vim script to provide less like key bindings.

For more information, read:

```
less
```

[faq-30.7](#)

30.7. How do I view Unix man pages from inside Vim?

You can view Unix man pages, inside Vim, using the `man.vim` plugin supplied as part of the standard Vim distribution. To use this plugin, add the following line to your startup vimrc file:

```
runtime ftplugin/man.vim
```

You can also press the `K` key to run the program specified by the `'keywordprg'` option with the keyword under the cursor. By default, `'keywordprg'` is set to run `man` on the keyword under the cursor.

For more information, read:

```
ft-man-plugin  
K  
'keywordprg'
```

[faq-30.8](#)

30.8. How do I change the diff command used by the Vim diff support?

By default, the Vim diff support uses the `'diff'` command. You can change this by changing the `'diffexpr'` option.

For more information, read:

```
diff-diffexpr  
'diffexpr'
```

[faq-30.9](#)

30.9. How do I use the Vim diff mode without folding?

You can use the following command-line to start Vim with two filenames and use the diff mode without folding:

```
$ vim -o file1 file2 "+windo set diff scrollbind scrollopt+=hor nowrap"
```

If you like vertically split windows, then replace "-o" with "-O".

For more information, read:

`vimdiff`

SECTION 31 - GUI VIM

faq-31

faq-31.1

31.1. How do I create buffer specific menus?

Adding support for buffer specific menus is in the Vim TODO list. In the mean time, you can try Michael Geddes's plugin, `buffermenu.vim`:

https://www.vim.org/scripts/script.php?script_id=246

faq-31.2

31.2. How do I change the font used by GUI Vim?

You can change the **'guifont'** option to change the font used by GUI Vim. To display the current value of this option, you can use

```
:set guifont?
```

You can add the displayed font name to the `.vimrc` file to use the font across Vim sessions. For example, add the following line to the `.vimrc` file to use Andale Mono font.

```
set guifont=Andale_Mono:h10:cANSI
```

For Win32, GTK and Photon version of Vim, you can use the following command to bringup a dialog which will help you in changing the `guifont`:

```
:set guifont=*
```

You can also use the `-font` Vim command line option to specify the font used for normal text.

For more information, read:

```
'guifont'  
'guifontset'  
'guifontwide'  
font-sizes  
-font  
-boldfont  
-italicfont  
-menufont  
-menufontset
```

faq-31.3

31.3. When starting GUI Vim, how do I specify the location of the GVIM

window?

You can use the "-geometry" command line argument to specify the location of the GUI Vim window. For example:

```
$ gvim -geometry 80x25+100+300
```

For more information, read:

```
31.4  
-geom
```

faq-31.4

31.4. How do I add a horizontal scrollbar in GVim?

You can enable the horizontal scrollbar by modifying the **'guioptions'** option:

```
:set guioptions+=b
```

For more information, read:

```
'guioptions'  
gui-horiz-scroll
```

faq-31.5

31.5. How do I make the scrollbar appear in the left side by default?

You can add the "l" flag to the **'guioptions'** option to make the scrollbar appear in the left side.

```
:set guioptions+=l  
:set guioptions-=r
```

For more information, read:

```
'guioptions'  
gui-scrollbar
```

faq-31.6

31.6. How do I remove the Vim menubar?

You can remove the Vim menubar by removing the "m" flag from the **'guioptions'** option:

```
:set guioptions-=m
```

For more information, read:

```
'guioptions'
```

faq-31.7

31.7. I am using GUI Vim. When I press the <Alt> key and a letter, the menu starting with that letter is selected. I don't want this behavior as

I want to map the `<Alt>-<key>` combination. How do I do this?

You can use the `'winaltkeys'` option to disable the use of the `<Alt>` key to select a menu item:

```
:set winaltkeys=no
```

For more information, read:

```
'winaltkeys'  
:simalt
```

faq-31.8

31.8. Is it possible to scroll the text by dragging the scrollbar so that the cursor stays in the original location?

The way Vim is designed, the cursor position has to be in a visible spot in normal, visual, select and insert mode. This cannot be changed without modifying Vim. When the scrollbar is used, the cursor will be moved so that it is always visible. Another approach to solving this problem is to use the Vim marks. You can mark the current cursor position using `ma`. Then scroll to a different part of the text and jump back to the old position using ``a`. You can also try the following suggestion from the Vim Online website:

<http://vim.wikia.com/wiki/VimTip320>

For more information, read:

```
mark-motions
```

faq-31.9

31.9. How do I get `gvim` to start browsing files in a particular directory when using the `':browse'` command?

You can set the `'browsedir'` option to the default directory to use for the `':browse'` command.

```
:set browsedir='<your_dir>'
```

For more information, read:

```
'browsedir'
```

faq-31.10

31.10. For some questions, like when a file is changed outside of Vim, Vim displays a GUI dialog box. How do I replace this GUI dialog box with a console dialog box?

You can set the `"c"` flag in the `'guioptions'` option to configure Vim to use console dialogs instead of GUI dialogs:

```
:set guioptions+=c
```


For more information, read:

['guioptions'](#)

[faq-31.11](#)

31.11. I am trying to use GUI Vim as the editor for my xxx application. When the xxx application launches GUI Vim to edit a file, the control immediately returns to the xxx application. How do I start GUI Vim, so that the control returns to the xxx application only after I quit Vim?

You have to start GUI Vim with the "-f" (foreground) command line option:

```
$ gvim -f
```

By default, GUI Vim will disconnect from the program that started Vim. With the '-f' option, GUI Vim will not disconnect from the program that started it.

For more information, read:

[gui-fork](#)
[-f](#)

[faq-31.12](#)

31.12. Why does the "Select Font" dialog doesn't show all the fonts installed in my system?

Vim supports only fixed width (mono-spaced) fonts. Proportional fonts are not supported. In the "Select Font" dialog, only fixed width fonts will be displayed.

For more information, read:

[font-sizes](#)
['guifont'](#)

[faq-31.13](#)

31.13. How do I use the mouse in Vim command-line mode?

You can set the "c" flag in the **'mouse'** option to use mouse in the Vim command-line mode:

```
:set mouse+=c
```

For more information, read:

[mouse-using](#)
[gui-mouse](#)
[09.2](#)

[faq-31.14](#)

31.14. When I use the middle mouse button to scroll text, it pastes the last copied text. How do I disable this behavior?

You can map the middle mouse button to `<Nop>` to disable the middle mouse button:

```
:map <MiddleMouse> <Nop>
:map! <MiddleMouse> <Nop>
```

For more information, read:

```
gui-mouse-mapping
<Nop>
```

faq-31.15

31.15. How do I change the location and size of a GUI Vim window?

You can use the `:winpos` command to change the Vim window position. To change the size of the window, you can modify the `'lines'` and `'columns'` options.

For example, the following commands will position the GUI Vim window at the X,Y co-ordinates 50,50 and set the number of lines to 50 and the number of columns to 80.

```
:winpos 50 50
:set lines=50
:set columns=80
```

The arguments to the `:winpos` command specify the pixel co-ordinates of the Vim window. The `'lines'` and `'columns'` options specify the number of lines and characters to use for the height and the width of the window respectively.

For more information, read:

```
31.4
:winpos
'lines'
'columns'
GUIEnter
```

faq-31.16

31.16. When splitting the Vim window vertically, Vim changes the position.

This is a known problem. When you are splitting the Vim window, Vim will try to draw a scrollbar. Since this changes the gui window, Vim tries to resize its main window to keep the same position and this will cause Vim to move its position. This happens on Windows with a multi-window setup or a window that was "snapped" to a certain position.

A workaournd to this problem is, to remove gui scrollbars, e.g.

```
:set guioptions-=L
```

SECTION 32 - VIM ON UNIX

faq-32.1

32.1. I am running Vim in a xterm. When I press the **CTRL-S** key, Vim freezes. What should I do now?

Many terminal emulators and real terminal drivers use the **CTRL-S** key to stop the data from arriving so that you can stop a fast scrolling display to look at it (also allowed older terminals to slow down the computer so that it did not get buffer overflows). You can start the output again by pressing the **CTRL-Q** key.

When you press the **CTRL-S** key, the terminal driver will stop sending the output data. As a result of this, it will look like Vim is hung. If you press the **CTRL-Q** key, then everything will be back to normal.

You can turn off the terminal driver flow control using the "stty" command:

```
$ stty -ixon -ixoff
```

or, you can change the keys used for the terminal flow control, using the following commands:

```
$ stty stop <char>
$ stty start <char>
```

faq-32.2

32.2. I am seeing weird screen update problems in Vim. What can I do to solve this screen/display update problems?

You have to use a proper terminal emulator like xterm with correct TERM settings (TERM=xterm) and a correct terminfo/termcap file. For more information, read:

```
'term'
```

faq-32.3

32.3. I am using the terminal/console version of Vim. In insertmode, When I press the backspace key, the character before the cursor is not erased. How do I configure Vim to do this?

You have to make sure that Vim gets the correct keycode for the backspace key. You can try using the command:

```
:fixdel
```

Make sure the TERM environment variable is set to the correct terminal name. You can try using the "stty" command:

```
$ stty erase ^H
```

where, you have to enter the ^H character by pressing the **CTRL-V** key and

then the **CTRL-H** key. Also check the value of your **'backspace'** setting.

For more information, read:

```
:fixdel
Linux-backspace
NetBSD-backspace
'backspace'
```

faq-32.4

32.4. I am using Vim in a xterm. When I quit Vim, the screen contents are restored back to the original contents. How do I disable this?

The xterm has a capability called "alternate screen". If this capability is present, vim switches to that alternate screen upon startup and back on exit, thus restoring the original screen contents. To disable this feature, add the following line to your .vimrc file:

```
:set t_ti= t_te=
```

For more information, read:

```
'restorescreen'
xterm-screens
```

faq-32.5

32.5. When I start Vim, it takes quite a few seconds to start. How do I minimize the startup time?

This may be related to Vim opening the X display for setting the xterm title and using the X clipboard. Make sure the DISPLAY variable is set to point to the correct host. Try using the command line:

```
$ vim -X
```

This will prevent Vim from opening the X display. With this command-line option, the X clipboard cannot be used and also Vim will not be able to change the xterm title.

You can also set the **'clipboard'** option to

```
:set clipboard=exclude:.*
```

This has the same effect as using the -X command-line argument.

For more information, read:

```
-X
'clipboard'
```

If the clipboard is not the cause of the slow startup, it might be a plugin that slows down Vim. In that case, you can use the --startuptime argument to debug this further. You can do:

```
$ vim --startuptime vim_startup.log
```

and the timing will be written to the file vim_startup.log. For even more advanced profiling, you can use the profiling feature, that is available in huge builds of Vim. To do so, call Vim like this:

```
$ vim --cmd 'profile start profile.log' \  
  --cmd 'profile func *' \  
  --cmd 'profile file *' \  
  -c 'profdel func *' \  
  -c 'profdel file *' \  
  -c 'qa!'
```

After running this, you will have a file profile.log in your current directory. To further analyse this, open the file profile.log and run:

```
" Open profile.log file in vim first  
:let timings=[]  
:g/^SCRIPT/call add(timings, [getline('.')[len('SCRIPT  '):], matchstr(getline(line('.')),  
:enew  
:call setline('.', ['count total (s)  self (s)  script']+map(copy(timings), 'printf("%
```

For more information, read:

```
--startuptime  
profiling
```

faq-32.6

32.6. How can I make the cursor in gvim in unix stop blinking?

You can modify the **'guicursor'** option, to stop the cursor from blinking. For example:

```
:set guicursor=a:blinkon0
```

For more information, read:

```
'guicursor'
```

faq-32.7

32.7. How do I change the menu font on GTK Vim?

You can modify the ~/.gtkrc file to change the menu font on GTK Vim. For example:

```
style "default"  
{ font ="smooth09" }  
class "*" style "default"
```

The last line changes the font of all widgets.

For more information, read:

```
gui-gtk
```

32.8. How do I prevent **CTRL-Z** from suspending Vim?

You can map **CTRL-Z** to prevent the suspending. Here are some suggestions:

- Make **CTRL-Z** do nothing:

```
:map <C-Z> <Nop>
```

- Make **CTRL-Z** start a shell:

```
:map <C-Z> :shell<CR>
```

- Make **CTRL-Z** give an error message:

```
:map <C-Z> :\"suspending disabled<CR>
```

For the last example, the double quote is necessary in order to keep the message on the status line.

32.9. When I kill the xterm running Vim, the Vim process continues to run and takes up a lot of CPU (99%) time. Why is this happening?

When Vim is built with support for Python interface, you will have this problem. This is a known problem with the python thread library and Vim. To solve this problem, use a Vim binary built without the Python interface.

For more information, read:

```
+python
python
```

32.10. How do I get the Vim syntax highlighting to work in a Unix terminal?

The easiest and simplest way to get Vim syntax highlighting is to use the GUI version of Vim (GVIM). To get syntax highlighting to work in the console/terminal version of Vim, you have to run a terminal emulator (like Xfree86 xterm or rxvt or dtterm) that supports color. **Note** that if a terminal emulator supports changing the background and foreground colors, that does not mean that it also supports ANSI escape sequences for changing the color. You can download the latest version of Xfree86 xterm from <https://invisible-island.net/xterm/xterm.html>

You can download the latest version of rxvt from <https://rxvt.org>. You have to install the terminfo/termcap file that supports colors for the terminal emulator. Also, set the TERM environment variable to the correct name of the term that supports colors.

You can use the colortest.vim script supplied with the Vim runtime package to test the color setup. To use this script, follow these steps:

```
:e $VIMRUNTIME/syntax/colortest.vim
```

```
:source %
```

For more information, read:

```
06.2  
terminal-colors  
termcap-colors  
startup-terminal  
xterm-color  
colortest.vim
```

faq-33

SECTION 33 - VIM ON MS-WINDOWS

faq-33.1

33.1. In MS-Windows, **CTRL-V** doesn't start the blockwise visual mode. What happened?

The mswin.vim script provides key mappings and options to make Vim behave like a MS-Windows application. One of the keys mapped is **CTRL-V** which is used for pasting text in MS-Windows applications. This will disable the use of **CTRL-V** to start the blockwise visual mode. The mswin.vim script maps **CTRL-Q** for starting the blockwise visual mode. So you can use **CTRL-Q** instead of **CTRL-V**.

For more information, read:

```
CTRL-V  
CTRL-V-alternative  
CTRL-Q  
10.5
```

faq-33.2

33.2. When I press the **CTRL-Y** key, it acts like the **CTRL-R** key. How do I configure Vim to treat **CTRL-Y** as **CTRL-Y**?

The mapping of the **CTRL-Y** key to the **CTRL-R** key is done by the mswin.vim script. The mswin.vim script maps **CTRL-Y** to make Vim behave like a standard MS-Windows application. This is explained in `:help CTRL-Y`. You can either comment out the line in mswin.vim that maps the **CTRL-Y** key or you can remove the line in your `.vimrc` file that sources the mswin.vim script.

faq-33.3

33.3. How do I start GUI Vim in a maximized window always?

You can use the `:simalt` command to maximize the Vim window. You can use the GUIEnter autocmd to maximize the Vim window on startup:

```
autocmd GUIEnter * simalt ~x
```

For more information, read:

```
:simalt
```

GUIEnter
gui-win32-maximized

faq-33.4

33.4. After doing some editing operations, Vim freezes. The cursor becomes an empty rectangle. I am not able enter any characters. What is happening?

Most probably, you used the mouse wheel to scroll the text in Vim. There is a known problem in using intellimouse mouse wheel with Vim. To avoid this problem, disable Universal scrolling support for Vim.

For more information, read:

[intellimouse-wheel-problems](#)

faq-33.5

33.5. I am using Windows XP, the display speed of maximized GVim is very slow. What can I do to speed the display updates?

This may be due to the fact that you have enabled 'Smooth edges of screen fonts' in the display properties. Try turning off font smoothing or try changing the smoothing method to "Standard".

faq-33.6

33.6. What are the recommended settings for using Vim with cygwin?

You may want to set the following shell related Vim settings:

```
:set shellcmdflag=-c
:set shellquote=
:set shellslash      " Use the forward slash for expansion.
:set shellxquote=\"
:set shell=d:\cygwin\bin\bash.exe " Use the bash shell
:set shellpipe=2>&1| tee
:set shellredir=>%s 2>&1
```

faq-33.7

33.7. I am trying to use GNU diff with Vim diff mode. When I run the diff from command line, it works. When I try to use the diff with Vim it doesn't work. What should I do now?

There is a problem with using GNU diff with Vim. You can try using the GNU diff.exe built by Ron Aaron from the following link:

<http://www.mossbayeng.com/~ron/vim/builds.html>

(This page no longer exists.)

faq-33.8

33.8. Is it possible to use Vim as an external editor for MS-Windows Outlook email client?

You can use the "cubiclevim" COM Add-In to use Vim as an external editor for MS-Windows Outlook email client. Visit the following URL for more

information:

<https://sourceforge.net/projects/cubiclevim/>

Note, that currently this works only with MS-Office 2000 and XP.

Also the plugin OutlookVim might be worth a look:

https://www.vim.org/scripts/script.php?script_id=3087

faq-33.9

33.9. I am using Vim to edit HTML files. How do I start internet explorer with the current file to preview the HTML file?

You can use the following command:

```
:!start c:\progra~1\intern~1\iexplore.exe file://%:~p<CR>
```

faq-33.10

33.10. I would like to use Vim with Microsoft Visual Studio. How do I do this?

You have to download and use the OLE version of Vim (for example: gvim61ole.zip). This file also contains instructions on how to use Vim with Visual Studio.

For more information, read:

[MSVisualStudio](#)

faq-33.11

33.11. Where do I place the _vimrc and _gvimrc files?

You can place the _vimrc and _gvimrc files under the directory pointed to by the VIM environment variable. If you are sharing this system with other users, then you can place the files in a directory and set the HOME environment variable to this directory.

For more information, read:

[\\$HOME-use
_vimrc](#)

faq-33.12

33.12. Every time I save a file, Vim warns about the file being changed outside of Vim. Why?

If you get the following warning message, every time you save a file:

```
WARNING: The file has been changed since reading it!!!  
Do you really want to write to it (y/n)?
```

then this problem could be related to a bug in MS-Windows on the day daylight saving time starts. Vim remembers the timestamp of the file after

it was written. Just before the next write the timestamp is obtained again to check if the file was changed outside of Vim. This works correctly, except on the day daylight saving time starts.

This problem will go away the next day after the day the daylight saving time starts.

For more information, read:

W11

=====

SECTION 34 - PRINTING

faq-34

faq-34.1

34.1. How do I print a file along with line numbers for all the lines?

You can set the **'printoptions'** option and use the `:hardcopy` command to print your file:

```
:set printoptions=number:y
:hardcopy
```

For more information, read:

```
'printoptions'
:hardcopy
```

faq-34.2

34.2. How do I print a file with the Vim syntax highlighting colors?

You can use the `:hardcopy` command to print a file with the Vim syntax highlighting colors. You can also convert your file to a HTML file using the `2html.vim` script and print the HTML file.

For more information, read:

```
syntax-printing
2html.vim
:hardcopy
printing
```

=====

SECTION 35 - BUILDING VIM FROM SOURCE

faq-35

faq-35.1

35.1. How do I build Vim from the sources on a Unix system?

For a Unix system, follow these steps to build Vim from the sources:

- Download the source and run-time files archive (vim-#.tar.bz2) from the <ftp://ftp.vim.org/pub/vim/unix> directory.

- Extract the archive using the bzip2 and tar utilities using the command:

```
$ bunzip2 -c <filename> | tar -xf -
```

- Alternatively, download the source from the git repository:

```
https://github.com/vim/vim/releases/
```

- Alternatively, download the source from the mercurial repository:

```
https://bitbucket.org/vim-mirror/vim/downloads/
```

- Run the "make" command to configure and build Vim with the default configuration.

- Run "make install" command to install Vim in the default directory.

To enable/disable various Vim features, before running the "make" command you can run the "configure" command with different flags to include/exclude the various Vim features. To list all the available options for the "configure" command, use:

```
$ configure --help
```

For more information, read:

```
install
```

faq-35.2

35.2. How do I install Vim in my home directory or a directory other than the default installation directory in Unix?

To install Vim in a directory other than the default installation directory, you have to specify the directory using the --prefix option while running the configure script.

```
$ ./configure --prefix=/users/xyz
```

You can enable/disable various Vim feature by supplying different arguments to the configure script. For more information about all these options, run:

```
$ ./configure --help
```

For more information, read:

```
install-home  
install
```

faq-35.3

35.3. How do I build Vim from the sources on a MS-Windows system?

For a MS-Windows system, Vim can be built using either the Visual C++ compiler or the Borland C++ compiler or the Ming GCC compiler or the cygwin gcc compiler. Follow these steps to build Vim from the sources for MS-Windows:

- Download the source (vim##src.zip), runtime (vim##rt.zip) and the extra (vim-##-extra.tar.gz) archives from the <ftp://ftp.vim.org/pub/vim/pc> directory.
- Extract the archives into a directory (for example, c:\vimsrc)

- Alternatively, download the source from the git repository:
<https://github.com/vim/vim/releases/>
- Alternatively, download the source from the mercurial repository:
<https://bitbucket.org/vim-mirror/vim/downloads/>
- Depending on the installed compiler, you can use the corresponding makefile to build the Vim sources. For Visual C++ use the Make_mvc.mak makefile, for borland C++ use the Make_bc5.mak makefile, for ming GCC use the Make_ming.mak makefile, for cygwin gcc use the Make_cyg.mak makefile.

Depending on whether you want to build the GUI version of Vim or the console version of Vim, you have to pass different arguments to the makefiles. After successfully building the sources, you can copy the vim.exe or gvim.exe file to the desired directory along with the files from the runtime archive.

For more information, read:

[install](#)

[faq-35.4](#)

35.4. The Vim help, syntax, indent files are missing from my Vim installation. How do I install these files?

The Vim help, syntax, indent and other runtime files are part of the Vim runtime package. You need to download and install the Vim runtime package. For example, for MS-Windows, the name of the Vim 6.1 runtime package is vim61rt.zip.

For more information, read:

[install](#)

[faq-35.5](#)

35.5. I have built Vim from the source and installed the Vim package using "make install". Do I need to keep the Vim source directory?

No. Once you have built and installed Vim in some directory other than the original source directory (for example, /usr/bin or /usr/local/bin), then you can remove the source directory.

[faq-35.6](#)

35.6. How do I determine the Vim features which are enabled at compile time?

You can use the ``:version`` command to determine the Vim features that are enabled at compile time. The features that are enabled will be prefixed with a "+". The features that are not enabled will be prefixed with a "-".

If you want to test for a feature in a script, you can use the `has()` function:

```
if has("menu")
    " Set up some menus
```

```
endif
```

For more information, read:

```
:version  
+feature-list  
has()
```

[faq-35.7](#)

35.7. Can I build Vim without the GUI support?

Yes. You can build Vim by optionally enabling/disabling many of the features including GUI.

For more information, read:

```
install
```

[faq-35.8](#)

35.8. When building Vim on a Unix system, I am getting "undefined reference to term_set_winsize" error. How do I resolve this error?

You will get this error when the build process is not able to locate the term lib, termcap or ncurses library. You have to install the ncurses-dev package to resolve this error.

[faq-35.9](#)

35.9. Vim configure keeps complaining about the lack of gtk-config while trying to use GTK 2.03. This is correct, since in GTK 2 they moved to using the generic pkg-config. I can get pkg-config to list the various includes and libs for gtk, but for some reason the configure script still isn't picking this up.

Use the following shell script named gtk-config:

```
#!/bin/sh  
pkg-config gtk+-2.0 $1 $2
```

[faq-35.10](#)

35.10. I did successfully download the sources and compiled Vim on Unix. But feature ... still does not work. What is wrong and how can I fix it?

You should first check, that you are actually running your self compiled Vim and not the system's provided version. So first check your \$PATH setting.

Depending on your compile options, some features might not be included in your build of Vim. You can use the `:version` command to determine the Vim features that are enabled at compile time. The features that are enabled will be prefixed with a "+". The features that are not enabled will be prefixed with a "-".

The easiest way to include all features is to build the huge version. To do

this, you have to specify the `--with-features` option while running the configure script:

```
$ ./configure --with-features=huge
```

Nevertheless, a feature could still be disabled at compile time, if the configure script can't find the required libraries for those features (e.g. for clipboard integration, your Vim needs to be linked against the X11 development libraries).

There are several ways to install the required libraries:

- 1) On a Debian based distribution, you can use the package manager "apt" to install all required dependencies. As superuser, run the command:

```
$ apt-get build-dep vim-gtk
```

This makes sure all required libraries needed to compile the vim-gtk package will be installed. (This requires, that your sources list contains deb-src entries. See your distribution manual on how to enable this, if the above command did not work.)

- 2) In openSUSE you can use the package manager "zypper" to install all required libraries. This requires, that there is a source version of the package installable from a configured repository (which by default is not the case). Use:

```
$ zypper search -t srcpackage vim
```

to find out, whether or not there exists a source version in the repository. If there is none, you'll need to add a source repository. For openSUSE 11.2 you could use, e.g.

```
$ zypper ar
```

```
http://download.opensuse.org/source/distribution/11.2/repo/oss/src-11.2
```

(one line)

Once you have a source version available in your repositories, use this command to install all needed requirements:

```
$ zypper source-install --build-deps-only vim
```

- 3) On a Fedora/RedHat based system, you can use

```
$ yum-builddep vim-enhanced
```

- 4) Run configure with your options and watch for missing libraries:

```
$ ./configure --with-features=huge 2>&1 |tee logfile
```

This will run configure and record the output into the file "logfile". You need to check the logfile for missing dependencies. Consider this output:

```
checking --disable-gtktest argument... gtk test enabled
checking for pkg-config... /usr/bin/pkg-config
checking for GTK - version >= 2.2.0... no
```

Here you can see, that the gtk libraries are missing and therefore no GTK gui version can't be build. So you need to install the GTK library in your system, with your package manager or by compiling it yourself. Then run the configure script again and check, that it finds the library.

In theory, those provided dependencies by your distribution might still lack some libraries, that are needed for features, that simply are not enabled in your distribution and therefore those commands in 1-3 won't install it. At the very least, this provides a jumping point and you need to track down the required missing packages using method 4 from above. But usually, this works good enough for most people and you won't have to bother with the fourth method.

For more information, read:

```
:version
+feature-list
```

SECTION 36 - VARIOUS

faq-36

36.1. How do I edit binary files with Vim?

faq-36.1

You can set the following options to edit binary files in Vim:

```
:set binary
:set display=uhex
```

You can also use the "-b" command-line option to edit a binary file:

```
$ vim -b <binary_file_name>
```

You can also use the xxd utility (part of the Vim distribution) to edit binary files.

For more information, read:

```
23.4
edit-binary
hex-editing
-b
'binary'
'endofline'
'display'
```

faq-36.2

36.2. How do I disable the visual error flash and the error beep?

You can disable both the visual error flash and the error beep using the following command:

```
:set visualbell t_vb=
```

For more information, read:

```
'visualbell'  
'errorbells'  
t_vb
```

[faq-36.3](#)

36.3. How do I display the ascii value of a character displayed in a buffer?

You can use the "ga" command to display the ascii value of a displayed character.

For more information, read:

```
ga  
:ascii
```

[faq-36.4](#)

36.4. Can I use zero as a count for a Vim command?

You cannot use zero as a count for a Vim command, as "0" is a command on its own, moving to the first column of the line.

For more information, read:

```
0  
count
```

[faq-36.5](#)

36.5. How do I disable the Vim welcome screen?

You can disable the Vim welcome screen, by adding the "I" flag to the '**shortmess**' option:

```
:set shortmess+=I
```

For more information, read:

```
:intro  
'shortmess'
```

[faq-36.6](#)

36.6. How do I avoid the "hit enter to continue" prompt?

Vim will prompt you with the "hit enter to continue" prompt, if there are some messages on the screen for you to read and the screen is about to be

redrawn. You can add the "T" flag to the **'shortmess'** option to truncate all messages. This will help in avoiding the hit-enter prompt:

```
:set shortmess+=T
```

You can also increase the command height by setting the **'cmdheight'** option:

```
:set cmdheight=2
```

For more information, read:

```
hit-enter  
avoid-hit-enter  
'shortmess'  
'cmdheight'
```

[faq-36.7](#)

36.7. How do I invoke Vim from command line to run a group of commands on a group of files?

There are several ways to invoke Vim from command line to run a group of commands on a group of files. You can use a set of "-c" command line options to specify a group of commands:

```
$ vim -c "<ex_command_1>" -c "<ex_command_2>" *.txt
```

Each of the ex-command specified with the "-c" command line option is executed one by one sequentially. You can also use a single "-c" command line option and the "|" character to separate the ex commands:

```
$ vim -c "<ex_command_1> | <ex_command_2>" *.txt
```

In the above command, if an ex command fails, then all the remaining ex commands will not be executed.

For example, to replace "ABC" with "DEF" in a file from the command-line, you can use the following command:

```
$ vim -c "%s/ABC/DEF/ge | update" myfile.txt
```

To replace "ABC" with "DEF" in multiple files from the command-line, you can use the following command:

```
$ vim -c "argdo %s/ABC/DEF/ge | update" *.txt
```

You can store the group of commands into a file and use the "-s" command line option to run the commands on a set of files. For example, if the group of commands are stored in the file mycmds.txt, then you can use the following command:

```
$ vim -s mycmds.txt *.pl
```

For more information, read:

```
-C  
-S
```

faq-36.8

36.8. How do I use a normal mode command from insert mode without leaving the insert mode?

You can use a normal command from insert mode, without leaving the insert mode, by first pressing the **CTRL-O** key and then follow that with a single normal mode command.

To execute more than one normal mode command, press the **CTRL-L** key, followed by any number of normal mode commands and then press `<Esc>` to get back to the insert mode. (This only works, when the `'insertmode'` option is set).

For more information, read:

```
i_CTRL-O  
i_CTRL-L
```

faq-36.9

36.9. How do I start Vim in insert mode?

You can start Vim in insert mode using the `:startinsert` ex command.

```
$ vim +startinsert myfile.txt
```

The above command will open the file "myfile.txt" and start insert mode with the cursor in front of the first character on the first line. To open the file and start appending after the last character on the last line, you can use the following command:

```
$ vim + +startinsert! myfile.txt
```

For more information, read:

```
:startinsert
```

faq-36.10

36.10. How do I use Copy and Paste with Vim?

You should first check the output of the `:version` command and make sure that `+xterm-clipboard` is present.

When running Vim in an xterm, you can either let Vim control the mouse or let xterm control the mouse. This is configured by the `'mouse'` option.

If the `'mouse'` option is not set (or set to the default value), then Vim will not control the mouse. You cannot move the Vim text cursor using the mouse. When you select some text using the mouse, xterm will copy it to the X11 cut buffer. When you press both the mouse buttons, xterm will paste the text from the cut buffer.

If the **'mouse'** option is set to "a" or some other value, then Vim controls the mouse. The mode (normal or insert or visual, etc) in which Vim controls the mouse is configured by the **'mouse'** option. You can move the Vim text cursor using the mouse. When you select some text, the **'clipboard'** option setting is used to determine whether to transfer the selected text to the clipboard or not. The default setting is to transfer the selected text to the clipboard. If you want to use the xterm selection mechanism in this mode, then you can press the **<Shift>** key. If you press **<Shift>** key when selecting text using the mouse, then Vim doesn't control the mouse and xterm controls the mouse.

In the GUI mode, Copy and Paste should just work, depending on the **'mouse'** setting. For more information, read:

```
'clipboard'  
x11-selection  
clipboard  
'go-a'  
'mouse'  
xterm-copy-paste  
09.3
```

faq-36.11

36.11. Why shouldn't I modify the files in the system runtime directory?

Just be careful about modifying files under \$VIMRUNTIME, which usually is /usr/share/vim/vimXX (Unix) or C:\Program Files\vim\vimXX (Windows) and XX being the version for which it applies, e.g. 73 for Vim 7.3.

One should generally avoid modifying those files because they may be replaced during an upgrade of your Vim installation and your changes will be lost. Also, if you upgrade to a new major or minor revision of Vim (e.g., from 7.3 to 7.4), the new version of Vim will use a different \$VIMRUNTIME directory and while your changes won't be lost, they will be ignored.

Consequently, take a look at

[filetypes](#)

for an explanation of several ways to modify Vim's response to different filetypes and where to put those modifications so that they will not be overwritten.

=====

faq-37

SECTION 37 - UNICODE

Author: Tony Mechelynck <antoine.mechelynck AT belgacom.net>

faq-37.1

37.1. Is it possible to create Unicode files using Vim?

Yes. It may be more or less complicated depending on the keyboard and fonts available to you, but it is always possible to encode any possible Unicode codepoint (and some illegal ones) into a file. To create a Unicode file using Vim, you should have compiled Vim with the "+multi_byte" compile-time option. You can get more information about Unicode from the following sites:

<http://www.unicode.org>
<https://www.cl.cam.ac.uk/~mgk25/unicode.html>

For more information, read:

```
multibyte  
usr_45.txt
```

faq-37.2

37.2. Which Vim settings are particularly important for editing Unicode files?

The most important are the various "encoding" options, i.e., **'encoding'**, **'fileencoding'**, **'fileencodings'** and **'termencoding'**. The boolean option **'bomb'** is also significant.

For more information, read:

```
'encoding'  
'fileencoding'  
'fileencodings'  
'termencoding'  
'bomb'
```

faq-37.3

37.3. What is the **'encoding'** option?

Basically, the **'encoding'** option defines how Vim will represent your data internally. However, all Unicode encodings are represented internally as utf-8 and converted (if necessary) when reading and writing.

For more information, read:

```
'encoding'
```

faq-37.4

37.4. How does Vim name the various Unicode encodings?

Utf-8 is called utf-8 or utf8; utf-16 is called ucs-2 or ucs2; utf-32 is called ucs-4 or ucs4. Also, you may specify endianness (except for utf-8 which does not vary for endianness) by appending le for little-endian or be for big-endian. If you create a file with an encoding of ucs-2 or ucs-4 without specifying endianness, Vim will use what is typical of your machine.

For more information, read:

encoding-names
encoding-values
encoding-table

faq-37.5

37.5. How does Vim specify the presence or absence of a byte-order mark?

When reading a file, if the **'fileencodings'** option includes "ucs-bom", Vim will check for a byte-order mark. When writing a file, if the **'bomb'** option is set, Vim will write a byte-order mark on files whose encoding warrants it.

For more information, read:

'fileencodings'
'bomb'

faq-37.6

37.6. What is the **'fileencoding'** option?

The **'fileencoding'** option defines the particular encoding which Vim will use to write a file. If empty, then the value of the **'encoding'** option is the default.

For more information, read:

'fileencoding'

faq-37.7

37.7. What is the **'fileencodings'** option?

The **'fileencodings'** option defines the heuristics used by Vim when opening an existing file. It is a comma separated list of encodings. A special name, "ucs-bom" is used to indicate that Vim should check for the presence of a byte-order mark; however, it will not be recognised if it comes after "utf-8". Normally, "ucs-bom" (if present) should be first in the list.

When Vim opens a file, it checks it against the encodings listed in **'fileencodings'**. The first one that matches is used. If there is no match, then Vim sets **'fileencoding'** to the null string, i.e., the value of **'encoding'** will be used.

For more information, read:

'fileencodings'
'encoding'

faq-37.8

37.8. What is the **'termencoding'** option?

The **'termencoding'** option defines how your keyboard encodes the data you type. If empty, Vim assumes that it has the same value as **'encoding'**. Usually it should be set to something that matches your locale.

For more information, read:

`'termencoding'`
`locale`

faq-37.9

37.9. What is the **'bomb'** option?

When reading a file with "ucs-bom" present in the **'fileencodings'** option, Vim will set the **'bomb'** option on or off depending on the presence or absence of a byte-order mark at the start of the file. When writing, Vim will write a byte-order mark if the **'bomb'** option is set. You may set or unset it manually do make Vim write, or not write, the b.o.m.

For more information, read:

`'bomb'`

faq-37.10

37.10. Where can I find an example of a typical use of all these options?

There is a "tip", with explains them in different words with an example, at

<http://vim.wikia.com/wiki/VimTip246>

faq-37.11

37.11. How can I insert Unicode characters into a file using Vim?

Several methods are available:

- Characters present on your keyboard can be typed in the usual way, even those which require a "dead-key" prefix, like (for instance) the circumflex on French keyboards.
- Characters for which a digraph is defined can be typed as two characters prefixed by **CTRL-K**.
- If you have set the **'digraph'** option, you can enter the characters for which a digraph is defined as `<char1><BS><char2>`.
- Any character can be entered by using a **CTRL-V** prefix (or **CTRL-Q** if **CTRL-V** is remapped to paste from the clipboard).

For more information, read:

`digraphs`
`'digraph'`
`i_CTRL-V_digit`

faq-37.12

37.12. How can I know which digraphs are defined and for which characters?

First set the **'encoding'** option properly (for instance, to utf-8), then use the ``:digraphs`` command to list the currently defined digraphs.

Alternatively, the help file contains the complete set of all digraphs. So you can easily search that list there.

For more information, read:

```
:digraphs  
'encoding'  
digraph-table
```



```

! !! # #{} $ $HOME $HOME-use $HOME-windows $MYGVIMRC $MYVIMRC $VIM $VIM-use
$VIMRUNTIME $VIM_POSIX $quote % %:. %:8 %:S %:e %:gs %:h %:p %:r %:s %:t %:~
& ' ' ' ( ' ) ' . '0 ' < ' > 'A '[ ' ] '^ 'a 'acd' 'ai' 'akm' 'al' 'aleph'
'allowrevins' 'altkeymap' 'ambiwidth' 'ambw' 'anti' 'antialias' 'ap' 'ar'
'arab' 'arabic' 'arabicshape' 'ari' 'arshape' 'as' 'asd' 'autochdir'
'autoindent' 'autoprint' 'autoread' 'autosave' 'autoselldir' 'autowrite'
'autowriteall' 'aw' 'awa' 'b:context_ignore_makefile' 'b:context_include'
'b:mp_metafun' 'background' 'backspace' 'backup' 'backupcopy' 'backupdir'
'backupext' 'backskip' 'balloondelay' 'ballooneval' 'balloonevalterm'
'balloonexpr' 'bdir' 'bdlay' 'beautify' 'belloff' 'beval' 'bevalterm' 'bex'
'bexpr' 'bf' 'bg' 'bh' 'bin' 'binary' 'biosk' 'bioskey' 'bk' 'bkc' 'bl' 'bo'
'bomb' 'breakat' 'breakindent' 'breakindentopt' 'bri' 'briopt' 'brk'
'browsedir' 'bs' 'bsdir' 'bsk' 'bt' 'bufhidden' 'buflisted' 'buftype'
'casemap' 'cb' 'cc' 'ccv' 'cd' 'cdh' 'cdhome' 'cdpath' 'cedit' 'cf' 'cfu'
'ch' 'character' 'charconvert' 'ci' 'cin' 'cindent' 'cink' 'cinkeys' 'cino'
'cinoptions' 'cinscopedecls' 'cinsd' 'cinw' 'cinwords' 'clipboard' 'cm'
'cmdheight' 'cmdwinheight' 'cmp' 'cms' 'co' 'cocu' 'cole' 'colorcolumn'
'columns' 'com' 'comments' 'commentstring' 'compatible' 'complete'
'completefunc' 'completeopt' 'completepopup' 'completeslash' 'concealcursor'
'conceallevel' 'confirm' 'consk' 'conskey' 'copyindent' 'cot' 'cp' 'cpo'
'cptions' 'cpp' 'cpt' 'crb' 'cryptmethod' 'cscopepathcomp' 'cscopeprg'
'cscopequickfix' 'cscoperelative' 'cscopetag' 'cscopetagorder'
'cscopeverbose' 'csl' 'cspc' 'csprg' 'csqf' 'csre' 'cst' 'csto' 'csverb'
'cuc' 'cul' 'culopt' 'cursorbind' 'cursorcolumn' 'cursorline' 'cursorlineopt'
'cwh' 'debug' 'deco' 'def' 'define' 'delcombine' 'dex' 'dg' 'dict'
'dictionary' 'diff' 'diffexpr' 'diffopt' 'digraph' 'dip' 'dir' 'directory'
'display' 'dy' 'ea' 'ead' 'eadirection' 'eb' 'ed' 'edcompatible' 'ef' 'efm'
'ei' 'ek' 'emo' 'emoji' 'enc' 'encoding' 'endoffile' 'endofline' 'eof' 'eol'
'ep' 'equalalways' 'equalprg' 'errorbells' 'errorfile' 'errorformat'
'esckey' 'et' 'eventignore' 'ex' 'expandtab' 'exrc' 'fcl' 'fcs' 'fdc' 'fde'
'fdi' 'fdl' 'fdls' 'fdm' 'fdn' 'fdo' 'fdt' 'fe' 'fen' 'fenc' 'fencs' 'fex'
'ff' 'ffs' 'fic' 'fileencoding' 'fileencodings' 'fileformat' 'fileformats'
'fileignorecase' 'filetype' 'fillchars' 'fixendofline' 'fixeol' 'fk' 'fkmap'
'fl' 'flash' 'flp' 'fml' 'fmr' 'fo' 'foldclose' 'foldcolumn' 'foldenable'
'foldexpr' 'foldignore' 'foldlevel' 'foldlevelstart' 'foldmarker'
'foldmethod' 'foldminlines' 'foldnestmax' 'foldopen' 'foldtext' 'formatexpr'
'formatlistpat' 'formatoptions' 'formatprg' 'fp' 'fs' 'fsync' 'ft'
'g:context_extra_options' 'g:context_ignore_makefile' 'g:context_include'
'g:mf_other_macros' 'g:mf_plain_macros' 'g:mf_plain_modes' 'g:mp_close_tag'
'g:mp_metafun' 'g:mp_mfplain_macros' 'g:mp_open_tag' 'g:mp_other_macros'
'g:mp_plain_macros' 'g:no_context_maps' 'g:no_mf_maps' 'g:no_mp_maps' 'gcr'
'gd' 'gdefault' 'gfm' 'gfn' 'gfs' 'gfw' 'ghr' 'gli' 'go' 'go-!' 'go-A' 'go-F'
'go-L' 'go-M' 'go-P' 'go-R' 'go-T' 'go-a' 'go-b' 'go-c' 'go-d' 'go-e' 'go-f'
'go-g' 'go-h' 'go-i' 'go-k' 'go-l' 'go-m' 'go-p' 'go-r' 'go-t' 'go-v' 'gp'
'gr' 'graphic' 'grepformat' 'grepprg' 'gtl' 'gtt' 'guicursor' 'guifont'
'guifontset' 'guifontwide' 'guiheadroom' 'guiligatures' 'guioptions' 'guipty'
'guitablabel' 'guitabtooltip' 'hardtabs' 'helpfile' 'helpheight' 'helplang'
'hf' 'hh' 'hi' 'hid' 'hidden' 'highlight' 'history' 'hk' 'hkmap' 'hkmap'
'hkp' 'hl' 'hlg' 'hls' 'hlsearch' 'ht' 'ic' 'icon' 'iconstring' 'ignorecase'
'im' 'imactivatefunc' 'imactivatekey' 'imaf' 'imak' 'imc' 'imcmdline' 'imd'
'imdisable' 'imi' 'iminsert' 'ims' 'imsearch' 'imsf' 'imst' 'imstatusfunc'
'imstyle' 'inc' 'include' 'includeexpr' 'incsearch' 'inde' 'indentexpr'
'indentkeys' 'indk' 'inex' 'inf' 'infercase' 'insertmode' 'is' 'isf'
'isfname' 'isi' 'isident' 'isk' 'iskeyword' 'isp' 'isprint' 'joinspaces'

```

'jop' 'js' 'jumpoptions' 'key' 'keymap' 'keymodel' 'keyprotocol' 'keywordprg'
'km' 'kmp' 'kp' 'kpc' 'langmap' 'langmenu' 'langnoremap' 'langremap'
'laststatus' 'lazyredraw' 'lbr' 'lcs' 'linebreak' 'lines' 'linespace' 'lisp'
'lispoptions' 'lispwords' 'list' 'listchars' 'lm' 'lmap' 'lnr' 'loadplugins'
'lop' 'lpl' 'lrm' 'ls' 'lsp' 'luadll' 'lw' 'lz' 'ma' 'macatsui' 'magic'
'makeef' 'makeencoding' 'makeprg' 'mat' 'matchpairs' 'matchtime' 'maxcombine'
'maxfuncdepth' 'maxmapdepth' 'maxmem' 'maxmempattern' 'maxmemtot' 'mco' 'mef'
'menc' 'menuitems' 'mesg' 'mfd' 'mh' 'mis' 'mkspellmem' 'ml' 'mle' 'mls' 'mm'
'mmd' 'mmp' 'mmt' 'mod' 'modeline' 'modelineexpr' 'modelines' 'modifiable'
'modified' 'more' 'mouse' 'mousef' 'mousefocus' 'mousehide' 'mousem'
'mousemev' 'mousemodel' 'mousemoveevent' 'mouses' 'mouset' 'mousetime'
'mp' 'mps' 'msm' 'mzq' 'mzquantum' 'mzschemedll' 'mzschemegcdll'
'nf' 'noacd' 'noai' 'noakm' 'noallowrevins' 'noaltkeymap' 'noanti'
'noantialias' 'noar' 'noarab' 'noarabic' 'noarabicshape' 'noari' 'noarshape'
'noas' 'noasd' 'noautochdir' 'noautoindent' 'noautoread' 'noautosave'
'noautosHELLdir' 'noautowrite' 'noautowriteall' 'noaw' 'noawa' 'nobackup'
'noballooneval' 'noballoonevalterm' 'nobeval' 'nobevalterm' 'nobin'
'nobinary' 'nobiosk' 'nobioskey' 'nobk' 'nobl' 'nobomb' 'nobreakindent'
'nobri' 'nobuflisted' 'nocdh' 'nocdhome' 'nocf' 'noci' 'nocin' 'nocindent'
'nocompatible' 'noconfirm' 'noconsk' 'noconskey' 'nocopyindent' 'nocp'
'nocrb' 'nocscoperelative' 'nocscopetag' 'nocscopeverbose' 'nocsre' 'nocst'
'nocsvrb' 'nocuc' 'nocul' 'nocursorbind' 'nocursorcolumn' 'nocursorline'
'nodeco' 'nodelcombine' 'nodg' 'nodiff' 'nodigraph' 'noea' 'noeb' 'noed'
'noedcompatible' 'noek' 'noemo' 'noemoji' 'noendoffile' 'noendoffline' 'noeof'
'noeol' 'noequalalways' 'noerrorbells' 'noesckey' 'noet' 'noex'
'noexpandtab' 'noexrc' 'nofen' 'nofic' 'nofileignorecase' 'nofixendoffline'
'nofixeol' 'nofk' 'nofkmap' 'nofoldenable' 'nofs' 'nofsync' 'nogd'
'nogdefault' 'noguipty' 'nohid' 'nohidden' 'nohk' 'nohkmap' 'nohkmap'
'nohkp' 'nohls' 'nohlsearch' 'noic' 'noicon' 'noignorecase' 'noim' 'noimc'
'noimcmdline' 'noimd' 'noimdisable' 'noincsearch' 'noinf' 'noinfercase'
'noinsertmode' 'nois' 'nojoinspaces' 'nojs' 'nolangnoremap' 'nolangremap'
'nolazyredraw' 'nolbr' 'nolinebreak' 'nolisp' 'nolist' 'nolnr'
'noloadplugins' 'nolpl' 'nolrm' 'nolz' 'noma' 'nomacatsui' 'nomagic' 'nomh'
'noml' 'nomle' 'nomod' 'nomodeline' 'nomodelineexpr' 'nomodifiable'
'nomodified' 'nomore' 'nomousef' 'nomousefocus' 'nomousehide' 'nomousemev'
'nomousemoveevent' 'nonu' 'nonumber' 'noodev' 'noopendev' 'nopaste' 'nopi'
'nopreserveindent' 'nopreviewwindow' 'noprompt' 'nopvw' 'noreadonly'
'norelativenumber' 'noremap' 'norestorescreen' 'norevins' 'nori'
'norigleft' 'norl' 'nornu' 'noro' 'nors' 'noru' 'noruler' 'nosb' 'nosc'
'noscb' 'noscf' 'noscrollbind' 'noscrollfocus' 'noscs' 'nosecure' 'nosft'
'noshellslash' 'noshelltemp' 'noshiftround' 'noshortname' 'noshowcmd'
'noshowfulltag' 'noshowmatch' 'noshowmode' 'nosi' 'nosm' 'nosmartcase'
'nosmartindent' 'nosmarttab' 'nosmd' 'nosmoothscroll' 'nosms' 'nosn' 'nosol'
'nospell' 'nosplitbelow' 'nosplitright' 'nospr' 'nosr' 'nossl' 'nosta'
'nostartoffline' 'nostmp' 'noswapfile' 'noswf' 'nota' 'notagbsearch'
'notagrelative' 'notagstack' 'notbi' 'notbidi' 'notbs' 'notermbidi'
'notermguicolors' 'noterse' 'notextauto' 'notextmode' 'notf' 'notgc' 'notgst'
'notildeop' 'notimeout' 'notitle' 'noto' 'notop' 'notr' 'nottimeout'
'nottybuiltin' 'nottyfast' 'notx' 'noudf' 'noundofile' 'novb' 'novice'
'novisualbell' 'nowa' 'nowarn' 'nowb' 'noweirdinvert' 'nowfh' 'nowfw' 'nowic'
'nowildignorecase' 'nowildmenu' 'nowinfixheight' 'nowinfixwidth' 'nowiv'
'nowmnu' 'nowrap' 'nowrapscan' 'nowrite' 'nowriteany' 'nowritebackup' 'nows'
'noxtermcodes' 'nrformats' 'nu' 'number' 'numberwidth' 'nuw' 'odev' 'oft'
'ofu' 'omnifunc' 'op' 'open' 'opendev' 'operatorfunc' 'opfunc' 'optimize'

'option' 'osfiletype' 'pa' 'packpath' 'para' 'paragraphs' 'paste'
'pastetoggle' 'patchexpr' 'patchmode' 'path' 'pdev' 'penc' 'perldll' 'pex'
'pexpr' 'pfn' 'ph' 'pheader' 'pi' 'pm' 'pmbcs' 'pmbfn' 'popt' 'pp'
'preserveindent' 'previewheight' 'previewpopup' 'previewwindow' 'printdevice'
'printencoding' 'printexpr' 'printfont' 'printhead' 'printmbcharset'
'printmbfont' 'printoptions' 'prompt' 'pt' 'pumheight' 'pumwidth' 'pvh' 'pvp'
'pvw' 'pw' 'pythondll' 'pythonhome' 'pythonthreedll' 'pythonthreehome' 'pyx'
'pyxversion' 'qe' 'qftf' 'quickfixtextfunc' 'quote' 'quoteescape' 'rdt' 're'
'readonly' 'redraw' 'redrawtime' 'regexpengine' 'relativenumber' 'remap'
'renderoptions' 'report' 'restorescreen' 'revins' 'ri' 'rightleft'
'rightleftcmd' 'rl' 'rlc' 'rnu' 'ro' 'rop' 'rs' 'rtp' 'ru' 'rubydll' 'ruf'
'ruler' 'rulerformat' 'runtimepath' 'sb' 'sbo' 'sbr' 'sc' 'scb' 'scf' 'scl'
'scr' 'scroll' 'scrollbind' 'scrollfocus' 'scrolljump' 'scrolloff'
'scrollopt' 'scs' 'sect' 'sections' 'secure' 'sel' 'selection' 'selectmode'
'sessionoptions' 'sft' 'sh' 'shcf' 'shell' 'shellcmdflag' 'shellpipe'
'shellquote' 'shellredir' 'shellslash' 'shelltemp' 'shelltype' 'shellxescape'
'shellxquote' 'shiftround' 'shiftwidth' 'shm' 'shortmess' 'shortname'
'showbreak' 'showcmd' 'showcmdloc' 'showfulltag' 'showmatch' 'showmode'
'showtabline' 'shq' 'si' 'sidescroll' 'sidescrolloff' 'signcolumn' 'siso'
'sj' 'slm' 'sloc' 'slow' 'slowopen' 'sm' 'smartcase' 'smartindent' 'smarttab'
'smc' 'smd' 'smoothscroll' 'sms' 'sn' 'so' 'softtabstop' 'sol' 'sourceany'
'sp' 'spc' 'spell' 'spellcapcheck' 'spellfile' 'spelllang' 'spelloptions'
'spellsuggest' 'spf' 'spk' 'spl' 'splitbelow' 'splitkeep' 'splitright' 'spo'
'spr' 'sps' 'sr' 'srr' 'ss' 'ssl' 'ssop' 'st' 'sta' 'stal' 'startofline'
'statusline' 'stl' 'stmp' 'sts' 'su' 'sua' 'suffixes' 'suffixesadd' 'sw'
'swapfile' 'swapsync' 'swb' 'swf' 'switchbuf' 'sws' 'sxe' 'sxq' 'syn'
'synmaxcol' 'syntax' 't_#2' 't_#4' 't_%1' 't_%i' 't_&8' 't_8b' 't_8f' 't_8u'
't_@7' 't_AB' 't_AF' 't_AL' 't_AU' 't_BD' 't_BE' 't_CF' 't_CS' 't_CV' 't_Ce'
't_Co' 't-Cs' 't_DL' 't_Ds' 't_EC' 't_EI' 't_F1' 't_F2' 't_F3' 't_F4' 't_F5'
't_F6' 't_F7' 't_F8' 't_F9' 't_GP' 't_IE' 't_IS' 't_K1' 't_K3' 't_K4' 't_K5'
't_K6' 't_K7' 't_K8' 't_K9' 't_KA' 't_KB' 't_KC' 't_KD' 't_KE' 't_KF' 't_KG'
't_KH' 't_KI' 't_KJ' 't_KK' 't_KL' 't_PE' 't_PS' 't_RB' 't_RC' 't_RF' 't_RI'
't_RK' 't_RS' 't_RT' 't_RV' 't_Ri' 't_SC' 't_SH' 't_SI' 't_SR' 't_ST' 't_Sb'
't_Sf' 't_Si' 't_TE' 't_TI' 't_Te' 't_Ts' 't_Us' 't_VS' 't_WP' 't_WS' 't_XM'
't_ZH' 't_ZR' 't_al' 't_bc' 't_cd' 't_ce' 't_cl' 't_cm' 't_cs' 't_da' 't_db'
't_dl' 't_ds' 't_fd' 't_fe' 't_fs' 't_k1' 't_k2' 't_k3' 't_k4' 't_k5' 't_k6'
't_k7' 't_k8' 't_k9' 't_k;' 't_kB' 't_kD' 't_kI' 't_kN' 't_kP' 't_kb' 't_kd'
't_ke' 't_kh' 't_kl' 't_kr' 't_ks' 't_ku' 't_le' 't_mb' 't_md' 't_me' 't_mr'
't_ms' 't_nd' 't_op' 't_se' 't_so' 't_sr' 't_star7' 't_te' 't_ti' 't_ts'
't_u7' 't_ue' 't_us' 't_ut' 't_vb' 't_ve' 't_vi' 't_vs' 't_xn' 't_xs' 'ta'
'tabline' 'tabpagemax' 'tabstop' 'tag' 'tagbsearch' 'tagcase' 'tagfunc'
'taglength' 'tagrelative' 'tags' 'tagstack' 'tal' 'tb' 'tbi' 'tbidi' 'tbis'
'tbs' 'tc' 'tcdll' 'tenc' 'term' 'termbidi' 'termencoding' 'termguicolors'
'termwinkey' 'termwincscroll' 'termwinsize' 'termwintype' 'terse' 'textauto'
'textmode' 'textwidth' 'tf' 'tfu' 'tgc' 'tgst' 'thesaurus' 'thesaurusfunc'
'tildeop' 'timeout' 'timeoutlen' 'title' 'titlelen' 'titleold' 'titlestring'
'tl' 'tm' 'to' 'toolbar' 'toolbariconsize' 'top' 'tpm' 'tr' 'ts' 'tsl' 'tsr'
'tsrfu' 'ttimeout' 'ttimeoutlen' 'ttm' 'tty' 'ttybuiltin' 'ttyfast' 'ttym'
'ttymouse' 'ttyscroll' 'ttytype' 'tw' 'twk' 'twsl' 'twsl' 'twsl' 'twsl' 'twsl' 'twsl'
'udf' 'udir' 'ul' 'undodir' 'undofile' 'undolevels' 'undoreload'
'updatecount' 'updatetime' 'ur' 'ut' 'varsofttabstop' 'vartabstop' 'vb' 'vbs'
'vdir' 've' 'verbose' 'verbosefile' 'vfile' 'vi' 'viewdir' 'viewoptions'
'vif' 'viminfo' 'viminfofile' 'virtualedit' 'visualbell' 'vop' 'vsts' 'vts'
'w1200' 'w300' 'w9600' 'wa' 'wak' 'warn' 'wb' 'wc' 'wcm' 'wcr' 'wd'

```

'weirdinvert' 'wfh' 'wfw' 'wh' 'whichwrap' 'wi' 'wic' 'wig' 'wildchar'
'wildcharm' 'wildignore' 'wildignorecase' 'wildmenu' 'wildmode' 'wildoptions'
'wim' 'winaltkeys' 'wincolor' 'window' 'winfixbuf' 'winfixheight'
'winfixwidth' 'winheight' 'winminheight' 'winminwidth' 'winptydll' 'winwidth'
'wiv' 'wiv' 'wm' 'wmh' 'wmnu' 'wmw' 'wop' 'wrap' 'wrapmargin' 'wrapscan'
'write' 'writeany' 'writebackup' 'writedelay' 'ws' 'ww' 'xtermcodes' '{ } (
) + ++bad ++bin ++builtin_terms ++edit ++enc ++ff ++nobin ++opt +ARP
+GUI_Athena +GUI_GTK +GUI_Motif +GUI_Photon +GUI_neXtaw +X11 +acl +arabic
+autochdir +autocmd +autoservername +balloon_eval +balloon_eval_term +browse
+byte_offset +channel +cindent +clientserver +clipboard +clipboard_working
+cmd +cmdline_compl +cmdline_hist +cmdline_info +cmdwin +comments +conceal
+cryptv +cscope +cursorbind +cursorshape +debug +dialog_con +dialog_con_gui
+dialog_gui +diff +digraphs +directx +dnd +emacs_tags +eval +ex_extra
+extra_search +farsi +feature-list +file_in_path +find_in_path +float
+folding +footer +fork +gettext +hangul_input +iconv +iconv/dyn
+insert_expand +ipv6 +job +jumplist +keymap +lambda +langmap +libcall
+linebreak +lispindent +listcmds +localmap +lua +lua/dyn +menu +mksession
+modify_fname +mouse +mouse_dec +mouse_gpm +mouse_gpm/dyn +mouse_jsbterm
+mouse_netterm +mouse_ptyterm +mouse_sgr +mouse_sysmouse +mouse_urxvt
+mouse_xterm +mouseshape +multi_byte +multi_byte_ime +multi_lang +mzscheme
+mzscheme/dyn +netbeans_intg +num64 +ole +packages +path_extra +perl
+perl/dyn +persistent_undo +popupwin +postscript +printer +profile +python
+python/dyn +python3 +python3/dyn +python3/dyn-stable +quickfix +reltime
+rightleft +ruby +ruby/dyn +scrollbind +signs +smartindent +sodium +sound
+spell +startuptime +statusline +sun_workshop +syntax +system()
+tag_any_white +tag_binary +tag_old_static +tcl +tcl/dyn +termguicolors
+terminal +terminfo +termresponse +textobjects +textprop +tgetent +timers
+title +toolbar +unix +user_commands +vartabs +vertsplits +vim9script +viminfo
+virtualedit +visual +visualextra +vreplace +vtp +wildignore +wildmenu
+windows +writebackup +xattr +xfontset +xim +xpm +xpm_w32 +xsmp
+xsmp_interact +xterm_clipboard +xterm_save , - -+ -+ / -+c -+reverse -+rv --
--- --clean --cmd --echo-wid --gui-dialog-file --help --literal --log
--nofork --nopugin --not-a-term --remote --remote-expr --remote-send
--remote-silent --remote-tab --remote-tab-silent --remote-tab-wait
--remote-tab-wait-silent --remote-wait --remote-wait-silent --role
--serverlist --servername --socketid --startuptime --ttyfail --version
--windowid -> -? -A -C -D -E -F -H -L -M -N -O -P -R -S -T -U -V -W -X -Z -b
-background -bg -boldfont -borderwidth -bw -c -d -dev -display -e -f -fg
-file -fn -font -foreground -g -geom -geometry -geometry-example -gui -h -i
-iconic -italicfont -l -m -menufont -menufontset -menuheight -mf -mh -n -nb
-o -p -q -qf -r -register -reverse -rv -s -s-ex -scrollbarwidth -silent -sw
-t -tag -u -ul -unregister -v -vim -w -w_nr -x -xrm -y . . . .Xdefaults .aff
.dic .exerc .gvimrc .netrwbook .netrwhist .vimrc / /$ / . // //; /<CR> /[[.
/[[= /[\n] /[] \ \ $ \ \%# \ \%#= \ \%$ \ \% 'm \ \% ( \ \% ( \ ) \ \% < 'm \ \% < c \ \% < l
\ \% < v \ \% > 'm \ \% > c \ \% > l \ \% > v \ \% C \ \% U \ \% V \ \% [ \ \% ^ \ \% c \ \% d \ \% l \ \% o
\ \% u \ \% v \ \% x \ \% & \ \% ( \ \% ( \ ) \ \% ) \ \% + \ \% . \ \% 1 \ \% 2 \ \% 3 \ \% 9 \ \% < \ \% = \ \% > \ \% ? \ \% @ !
\ \% @ < ! \ \% @ < = \ \% @ = \ \% @ > \ \% A \ \% C \ \% D \ \% F \ \% H \ \% I \ \% K \ \% L \ \% M \ \% O \ \% P \ \% S \ \% U \ \% V
\ \% W \ \% X \ \% Z \ \% [ \ \% \ \ \% ] \ \% ^ \ \% _ \ \% _ $ \ \% _ . \ \% _ A \ \% _ D \ \% _ F \ \% _ H \ \% _ I \ \% _ K \ \% _ L
\ \% _ O \ \% _ P \ \% _ S \ \% _ U \ \% _ W \ \% _ X \ \% _ [ \ \% _ ^ \ \% _ a \ \% _ d \ \% _ f \ \% _ h \ \% _ i \ \% _ k \ \% _ l
\ \% _ o \ \% _ p \ \% _ s \ \% _ u \ \% _ w \ \% _ x \ \% a \ \% b \ \% bar \ \% c \ \% d \ \% e \ \% f \ \% h \ \% i \ \% k \ \% l
\ \% m \ \% n \ \% o \ \% p \ \% r \ \% s \ \% star \ \% t \ \% u \ \% v \ \% w \ \% x \ \% z ( \ \% z ( \ ) \ \% z 1 \ \% z 2 \ \% z 3
\ \% z 4 \ \% z 5 \ \% z 6 \ \% z 7 \ \% z 8 \ \% z 9 \ \% ze \ \% zs \ \% { \ \% { - \ \% ~ \ \% ^ \ \% _ CTRL-G \ \% _ CTRL-L
\ \% _ CTRL-T /atom /bar /branch /character-classes /collection /concat /dyn
/ignorecase /magic /multi /ordinary-atom /pattern /piece /star /zero-width /~

```

0 01.1 01.2 01.3 01.4 02.1 02.2 02.3 02.4 02.5 02.6 02.7 02.8 03.1 03.10 03.2
03.3 03.4 03.5 03.6 03.7 03.8 03.9 04.1 04.10 04.2 04.3 04.4 04.5 04.6 04.7
04.8 04.9 05.1 05.2 05.3 05.4 05.5 05.6 05.7 05.8 05.9 06.1 06.2 06.3 06.4
06.5 06.6 07.1 07.2 07.3 07.4 07.5 07.6 07.7 08.1 08.2 08.3 08.4 08.5 08.6
08.7 08.8 08.9 09.1 09.2 09.3 09.4 0o 0x 10.1 10.2 10.3 10.4 10.5 10.6 10.7
10.8 10.9 11.1 11.2 11.3 11.4 12.1 12.2 12.3 12.4 12.5 12.6 12.7 12.8 1gd 1gd
20.1 20.2 20.3 20.4 20.5 21.1 21.2 21.3 21.4 21.5 21.6 22.1 22.2 22.3 22.4
23.1 23.2 23.3 23.4 23.5 24.1 24.10 24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9
25.1 25.2 25.3 25.4 25.5 26.1 26.2 26.3 26.4 27.1 27.2 27.3 27.4 27.5 27.6
27.7 27.8 27.9 28.1 28.10 28.2 28.3 28.4 28.5 28.6 28.7 28.8 28.9 29.1 29.2
29.3 29.4 29.5 2html.vim 30.1 30.2 30.3 30.4 30.5 30.6 31.1 31.2 31.3 31.4
31.5 32.1 32.2 32.3 32.4 40.1 40.2 40.3 41.1 41.10 41.11 41.12 41.2 41.3 41.4
41.5 41.6 41.7 41.8 41.9 42 42.1 42.2 42.3 42.4 43.1 43.2 44.1 44.10 44.11
44.12 44.2 44.3 44.4 44.5 44.6 44.7 44.8 44.9 45.1 45.2 45.3 45.4 45.5 50.1
50.2 50.3 51.1 51.2 51.3 51.4 52.1 52.2 52.3 52.4 52.5 8g8 90.1 90.2 90.3
90.4 90.5 : :! :!! :!cmd :!start :# :#! :\$:% :& :&& :' :++ :, :-- :. :/
:0file :2match :3match :. :. :8 :S :e :gs :h :p :r :s :t :~ :; :<
:<abuf> :<afile> :<amatch> :<cWORD> :<cexpr> :<cfile> :<client> :<cword>
:<script> :<sfile> :<sflnum> :<slnum> :<stack> := :> :? :@ :@: :@@ :AdaLines
:AdaRainbow :AdaSpaces :AdaTagDir :AdaTagFile :AdaTypes :Arguments :Asm
:Break :Cargo :Cbench :Cbuild :Cclean :Cdoc :Cfilter :Cinit :Cinstall :Clear
:CompilerSet :ConTeXt :ConTeXtJobsStatus :ConTeXtLog :ConTeXtStopJobs
:Continue :Cpublish :Crun :Cruntarget :Csearch :Ctest :Cupdate :DiffOrig
:DoMatchParen :Down :Evaluate :Explore :Finish :FixBeginfigs :Frame :GLVS
:Gdb :GetLatestVimScripts_dat :GnatFind :GnatPretty :GnatTags :Hexplore :LP
:LPE :LPF :Lexplore :Lfilter :LogiPat :Man :MkVimball :N :NetrwClean
:Nexplore :Next :NoMatchParen :Nr :Nread :Ns :Nsource :Ntree :Nw :Nwrite
:Over :P :Pexplore :Print :Program :Rexplore :RmVimball :Run :RustEmitAsm
:RustEmitIr :RustExpand :RustFmt :RustFmtRange :RustInfo :RustInfoClipboard
:RustInfoToFile :RustPlay :RustRun :RustTest :Sexplore :Source :Step :Stop
:TOhtml :TarDiff :Tbreak :Termdebug :TermdebugCommand :Texplore :Until :Up
:UseVimball :Var :Vexplore :VimballList :Vimuntar :Winbar :X :XMLent :XMLns
:[range] :\bar :_! :_# :_## :_#0 :_#< :_#n :_% :_%% :_%%% :_%%0 :_%%< :_%%n
:_% :_%< :a :ab :abbreviate :abbreviate-<buffer> :abbreviate-local
:abbreviate-verbose :abc :abclear :abo :aboveleft :abstract :add :al :all
:am :amenu :an :anoremenu :append :ar :arg :arga :argadd :argd :argded
:argdedupe :argdelete :argdo :arge :argedit :argglobal :arglocal :args
:args_f :args_f! :argu :argument :as :ascii :au :aug :augroup :augroup-delete
:aun :aunmenu :autocmd :autocmd-block :autocmd-verbose :b :bN :bNext :ba :bad
:badd :ball :balt :bar :bd :bdel :bdelete :be :behave :bel :belowright :bf
:bfirst :bl :blast :bm :bmodified :bn :bnext :bo :botright :bp :bprevious :br
:bre :brea :break :breaka :breakadd :breakd :breakdel :breakl :breaklist
:brewind :bro :browse :browse-set :bu :buf :bufdo :buffer :buffer-! :buffers
:bun :bunload :bw :bwipe :bwipeout :c :cN :cNext :cNf :cNfile :ca :cab
:cabbrev :cab :cabclear :cabo :cabove :cad :cadd :caddbuffer :cadde
:caddexpr :caddf :caddfile :caf :cafter :cal :call :cat :catch :cb :cbe
:cbefore :cbel :cbelow :cbo :cbottom :cbuffer :cc :ccl :cclose :cd :cd- :cdo
:ce :center :cex :cexpr :cf :cfdo :cfi :cfile :cfir :cfirst :cg :cgetb
:cgetbuffer :cgete :cgetexpr :cgetfile :ch :change :changes :chd :chdir :che
:chec :check :checkpath :checkt :checktime :chi :chistory :cl :cla :class
:clast :cle :clearjumps :clist :clo :close :cm :cmap :cmap_l :cmapc
:cmapclear :cme :cmenu :cn :cne :cnew :cnewer :cnext :cnf :cnfile :cno :cnor
:cnorea :cnoreaabbrev :cnoremap :cnoreme :cnoremenu :co :col :colder :colo
:colorscheme :com :comc :comclear :command :command-addr :command-bang

:command-bar :command-buffer :command-complete :command-completion
:command-completion-custom :command-completion-customlist :command-count
:command-keepscrip :command-modifiers :command-nargs :command-range
:command-register :command-repl :command-verbose :comment :comp :compiler
:con :conf :confirm :cons :const :continue :cope :copen :copy :count_quit :cp
:cpf :cpfile :cprev :cprevious :cq :cquit :cr :crewind :cs :cscope :cstag :cu
:cun :cuna :cunabbrev :cunmap :cunme :cunmenu :cw :cwindow :d :de :debug
:debug-name :debugg :debuggreedy :def :defc :defcompile :defer :del :delc
:delcommand :delcr :delete :delf :delfunction :delm :delmarks :di :dif :diff
:diffg :diffget :diffo :diffoff :diffp :diffpatch :diffpu :diffput :diffs
:diffsplit :diffst :diffthis :diffupdate :dig :digraphs :dis :disa
:disassemble :display :dj :djump :dl :dli :dlist :do :doau :doaut :doautoa
:doautoall :doautocmd :dp :dr :drop :ds :dsearch :dsp :dsplit :e :e# :ea
:earlier :ec :echo :echo-redraw :echoc :echoconsole :echoe :echoerr :echoh
:echohl :echom :echomsg :echon :echow :echowin :echowindow :edit :edit!
:edit!_f :edit_# :edit_f :el :else :elsei :elseif :em :emenu :en :end
:endclass :enddef :endenum :endf :endfo :endfor :endfunction :endif
:endinterface :endt :endtry :endw :endwhile :ene :ene! :enew :enew! :enum
:eval :ex :exe :exe-comment :execute :exi :exit :exp :export :exu :exusage :f
:fi :file :file_f :filename :files :filet :filetype :filetype-indent-off
:filetype-indent-on :filetype-off :filetype-overview :filetype-plugin-off
:filetype-plugin-on :filt :filter :fin :fina :final :finally :find :fini
:finish :fir :first :fix :fixdel :fo :fold :foldc :foldclose :foldd :folddo
:folddoc :folddoclosed :folddoopen :foldo :foldopen :for :fu :func-abort
:func-closure :func-dict :func-range :function :function-verbose :g :global
:go :goto :gr :grep :grepa :grepadd :gu :gui :gv :gvim :h :ha :hardcopy :help
:helpc :helpclose :helpf :helpfind :helpg :helpgrep :helpt :helptags :hi
:hi-clear :hi-default :hi-link :hi-normal :hi-normal-cterm :hide :highlight
:highlight-default :highlight-link :highlight-normal :highlight-verbose :his
:history :history-indexing :hor :horizontal :i :ia :iabbrev :iabc :iabclear
:if :ij :ijump :il :ilist :im :imap :imap_l :imapc :imapclear :ime :imenu
:imp :import :import-as :import-cycle :in :index :ino :inor :inorea
:inoreabbrev :inoremap :inoreme :inoremenu :insert :interface :intro :is
:isearch :isp :isplit :iu :iuna :iunabbrev :iunmap :iunme :iunmenu :j :join
:ju :jumps :k :kee :keep :keepa :keepalt :keepj :keepjumps :keepmarks :keep
:keeppatterns :l :lN :lNext :lNf :lNfile :la :lab :labove :lad :laddb
:laddbuffer :laddexpr :laddf :laddfile :laf :lafter :lan :lang :language
:last :lat :later :lb :lbe :lbefore :lbel :lbelow :lbo :lbottom :lbuffer :lc
:lcd :lcd- :lch :lchdir :lcl :lclose :lcs :lcscope :ldo :le :left :lefta
:leftabove :leg :legacy :let :let%= :let+= :let-\$:let-& :let-= :let-@
:let-environment :let-heredoc :let-option :let-register :let-unpack :let..=
:let.= :let/= :let=<< :letstar= :lex :lexpr :lf :lfdo :lfi :lfile :lfir
:lfirst :lg :lge :lgetb :lgetbuffer :lgete :lgetexpr :lgetfile :lgr :lgrep
:lgrepa :lgrepadd :lh :lhelpgrep :lhi :lhistory :list :ll :lla :llast :lli
:llist :lm :lma :lmak :lmake :lmap :lmap_l :lmapc :lmapclear :ln :lne :lnew
:lnewer :lnext :lnf :lnfile :lnoremap :lo :loadk :loadkeymap :loadview :loc
:lock :lockmarks :lockv :lockvar :lol :lolder :lop :lopen :lp :lpf :lpfile
:lprev :lprevious :lr :lrewind :ls :lt :ltag :lu :lua :luado :luafile :lunmap
:lv :lvimgrep :lvimgrepa :lvimgrepadd :lw :lwindow :m :ma :mak :make
:make_makeprg :map :map! :map-<buffer> :map-<expr> :map-<nowait>
:map-<script> :map-<silent> :map-<special> :map-<unique> :map-alt-keys
:map-arguments :map-buffer :map-cmd :map-cmd-key :map-commands
:map-expression :map-local :map-meta-keys :map-modes :map-nowait
:map-operator :map-script :map-silent :map-special :map-special-chars

:map-special-keys :map-super-keys :map-undo :map-unique :map-verbose :map_l
:map_l! :mapc :mapc! :mapclear :mapclear! :mark :marks :mat :match :me :menu
:menu-<script> :menu-<silent> :menu-<special> :menu-disable :menu-enable
:menu-script :menu-silent :menu-special :menut :menutrans :menutranslate :mes
:messages :mk :mkexrc :mks :mksession :mksp :mkspell :mkv :mkvi :mkvie
:mkview :mkvimrc :mo :mod :mode :move :mz :mzf :mzfile :mzscheme :n :nbclose
:nbkey :nbstart :ne :new :next :next_f :nm :nmap :nmap_l :nmapc :nmapclear
:nme :nmenu :nn :nnoremap :nnoreme :nnoremenu :no :no! :noa :noautocmd :noh
:nohlsearch :nor :nore :norea :noreabbrev :noreme :noremap :noremap! :noreme
:noremenu :norm :normal :normal-range :nos :noswapfile :nu :number :nun
:nunmap :nunme :nunmenu :o :ol :oldfiles :om :omap :omap_l :omapc :omapclear
:ome :omenu :on :only :ono :onoremap :onoreme :onoremenu :op :open :opt
:options :ou :ounmap :ounme :ounmenu :ownsyntax :p :pa :packadd :packl
:packloadall :pc :pclose :pe :ped :pedit :perl :perld :perlido :po :pop :popu
:popup :pp :ppop :pr :pre :preserve :prev :previous :print :pro :prof :profd
:profdel :profile :promptfind :promptr :promptrepl :ps :psearch :ptN :ptNext
:pta :ptag :ptf :ptfirst :ptj :ptjump :ptl :ptlast :ptn :ptnext :ptp
:ptprevious :ptr :ptrewind :pts :ptselect :pu :public :put :pw :pwd
:pwd-verbose :py :py3 :py3do :py3file :pydo :pyf :pyfile :python :python3
:pythonx :pyx :pyxdo :pyxfile :q :qa :qall :quit :quita :quitall :quote :r
:r! :range :range! :range-closed-fold :range-offset :range-pattern :re :read
:read! :rec :recover :recover-crypt :red :redi :redir :redo :redr :redraw
:redraws :redrawstatus :redrawt :redrawtabline :reg :registers :res :resize
:ret :retab :retab! :retu :return :rew :rewind :ri :right :rightb :rightbelow
:ru :rub :ruby :rubyd :rubydo :rubyf :rubyfile :rundo :runtime :rv :rviminfo
:s :s% :sI :sIc :sIe :sIg :sIl :sIn :sIp :sIr :sN :sNext :s\= :s_# :s_I :s_c
:s_e :s_flags :s_g :s_i :s_l :s_n :s_p :s_r :sa :sal :sall :san :sandbox
:sargument :sav :saveas :sb :sbN :sbNext :sba :sball :sbf :sbfir :sbl
:sblast :sbm :sbmodified :sbn :sbnnext :sbp :sbprevious :sbr :sbrewind
:sbuffer :sc :scI :sce :scg :sci :scl :scp :scr :script :scripte
:scriptencoding :scriptnames :scriptv :scriptversion :scs :scscope :se
:search-args :set :set+= :set-! :set-& :set-&vi :set-&vim :set-= :set-args
:set-browse :set-default :set-inv :set-termcap :set-verbose :set= :set^=
:set_env :setf :setfiletype :setg :setglobal :setl :setlocal :sf :sfi :sfind
:sfir :sfirst :sg :sgI :sgc :sge :sgi :sgl :sgn :sgp :sgr :sh :shell :si :sic
:sie :sig :sign :sign-define :sign-fname :sign-jump :sign-list :sign-place
:sign-place-list :sign-undefine :sign-unplace :sil :silent :silent! :sim
:simalt :sin :sip :sir :sl :sl! :sla :slast :sleep :sleep! :sm :smagic :smap
:smap_l :smapc :smapclear :sme :smenu :smile :sn :snext :sno :snomagic :snor
:snore :snoremap :snoreme :snoremenu :so :sor :sort :source :source!
:source-range :source_crnl :sp :spe :spell :spelled :spelledump :spellgood :spelli
:spellinfo :spellr :spellra :spellrare :spellrepall :spellu :spellundo
:spellw :spellwrong :split :split_f :spr :sprevious :sr :srI :src :sre
:srewind :srg :sri :srl :srn :srp :st :sta :stag :star :star-compatible
:star-visual-range :start :startgreplace :startinsert :startreplace :static
:stj :stjump :stop :stopi :stopinsert :sts :stselect :su :substitute
:substitute-repeat :sun :sunhide :sunm :sunmap :sunme :sunmenu :sus :suspend
:sv :sview :sw :swapname :sy :syn :syn-arguments :syn-case :syn-cchar
:syn-clear :syn-cluster :syn-conceal :syn-conceal-implicit :syn-concealends
:syn-contained :syn-containedin :syn-contains :syn-context
:syn-default-override :syn-define :syn-display :syn-enable :syn-end
:syn-excludenl :syn-ext-match :syn-extend :syn-file-remarks :syn-files
:syn-fold :syn-foldlevel :syn-include :syn-iskeyword :syn-keepend
:syn-keyword :syn-lc :syn-leading :syn-list :syn-manual :syn-match

:syn-matchgroup :syn-multi-line :syn-nextgroup :syn-off :syn-on :syn-oneline
:syn-pattern :syn-pattern-offset :syn-priority :syn-qstart :syn-region
:syn-reset :syn-skip :syn-skipempty :syn-skipnl :syn-skipwhite :syn-spell
:syn-start :syn-sync :syn-sync-ccomment :syn-sync-first :syn-sync-fourth
:syn-sync-linebreaks :syn-sync-maxlines :syn-sync-minlines :syn-sync-second
:syn-sync-third :syn-transparent :sync :syncbind :syntax :syntax-enable
:syntax-off :syntax-on :syntax-reset :syntime :t :tN :tNext :ta :tab :tabN
:tabNext :tabc :tabclose :tabd :tabdo :tabe :tabedit :tabf :tabfind :tabfir
:tabfirst :tabl :tablast :tabm :tabmove :tabn :tabnew :tabnext :tabo :tabonly
:tabp :tabprevious :tabr :tabrewind :tabs :tag :tags :tc :tcd :tcd- :tch
:tchdir :tcl :tclD :tcldo :tclf :tclfile :te :tearoff :ter :terminal :tf
:tfirst :th :this :throw :tj :tjump :tl :tlast :tlm :tlmenu :tln :tlnoremenu
:tlu :tlunmenu :tm :tma :tmap :tmap_l :tmapc :tmapclear :tmenu :tn :tnext
:tno :tnoremap :topleft :tp :tprevious :tr :trewind :try :ts :tselect :tu
:tunma :tunmap :tunmenu :type :u :un :una :unabbreviate :undo :undoj
:undojoin :undol :undolist :unh :unhide :unl :unlet :unlet-\$
:unlet-environment :unlo :unlockvar :unm :unm! :unmap :unmap! :unme :unmenu
:unmenu-all :uns :unsilent :up :update :v :var :ve :ver :verb :verbose
:verbose-cmd :version :vert :vertical :vertical-resize :vglobal :vi :vie
:view :vim :vim9 :vim9cmd :vim9s :vim9script :vimgrep :vimgrepa :vimgrepadd
:visual :visual_example :viu :viusage :vm :vmap :vmap_l :vmapc :vmapclear
:vme :vmenu :vn :vne :vnew :vnoremap :vnoreme :vnoremenu :vs :vsplit :vu
:vunmap :vunme :vunmenu :w :w! :wN :wNext :w_a :w_c :w_f :wa :wall :wh :while
:wi :win :winc :wincmd :windo :winp :winpos :winsize :wn :wnext :wp
:wprevious :wq :wqa :wqall :write :write_a :write_c :write_f :wundo :wv
:wvminfo :x :xa :xall :xit :xm :xmap :xmap_l :xmapc :xmapclear :xme :xmenu
:xn :xnoremap :xnoreme :xnoremenu :xr :xrestore :xu :xunmap :xunme :xunmenu
:y :yank :z :z! :z# :~ ; < <2-LeftMouse> <3-LeftMouse> <4-LeftMouse> << <>
<A- <A-LeftMouse> <A-RightMouse> <BS> <Bar> <Bslash> <C- <C-Del> <C-End>
<C-Home> <C-Insert> <C-Left> <C-LeftMouse> <C-PageDown> <C-PageUp> <C-Right>
<C-RightMouse> <C-ScrollWheelDown> <C-ScrollWheelLeft> <C-ScrollWheelRight>
<C-ScrollWheelUp> <C-Tab> <CR> <CSI> <Char-> <Char> <Cmd> <CursorHold> <D-
<D-c> <D-v> <D-x> <Down> <Drop> <EOL> <End> <Enter> <Esc> <F10> <F11>
<F12> <F13> <F14> <F15> <F16> <F17> <F18> <F19> <F1> <F2> <F3> <F4> <F5> <F6>
<F7> <F8> <F9> <Help> <Home> <Ignore> <Insert> <Leader> <Left> <LeftDrag>
<LeftMouse> <LeftRelease> <LocalLeader> <M- <MiddleDrag> <MiddleMouse>
<MiddleRelease> <Mouse> <MouseDown> <MouseMove> <MouseUp> <NL> <Nop> <NuL>
<PageDown> <PageUp> <Plug> <Return> <Right> <RightDrag> <RightMouse>
<RightRelease> <S- <S-Del> <S-Down> <S-End> <S-F10> <S-F11> <S-F12> <S-F1>
<S-F2> <S-F3> <S-F4> <S-F5> <S-F6> <S-F7> <S-F8> <S-F9> <S-Home> <S-Insert>
<S-Left> <S-LeftMouse> <S-Right> <S-RightMouse> <S-ScrollWheelDown>
<S-ScrollWheelLeft> <S-ScrollWheelRight> <S-ScrollWheelUp> <S-Tab> <S-Up>
<S-xF1> <S-xF2> <S-xF3> <S-xF4> <SID> <SNR> <ScriptCmd> <ScrollWheelDown>
<ScrollWheelLeft> <ScrollWheelRight> <ScrollWheelUp> <Space> <Tab> <Undo>
<Up> <abuf> <afile> <amatch> <args> <bang> <buffer=N> <buffer=abuf> <cWORD>
<cexpr> <cfile> <character> <client> <count> <cword> <f-args> <k0> <k1> <k2>
<k3> <k4> <k5> <k6> <k7> <k8> <k9> <kDivide> <kEnd> <kEnter> <kHome> <kMinus>
<kMultiply> <kPageDown> <kPageUp> <kPlus> <kPoint> <line1> <line2> <lt>
<mods> <nomodeline> <q-args> <q-mods> <range> <reg> <register> <script>
<sfile> <sflnum> <slnum> <stack> <xCSI> <xDown> <xEnd> <xEnd>-xterm <xF1>
<xF1>-xterm <xF2> <xF2>-xterm <xF3> <xF3>-xterm <xF4> <xF4>-xterm <xHome>
<xHome>-xterm <xLeft> <xRight> <xUp> = == > >> >backtrace >bt >cont >down
>finish >frame >interrupt >next >quit >step >up >where ? ?<CR> ?? @ @/ @: @=
@@ @r A ACL ANSI-C ATTENTION Abbreviations Aleph Amiga Arabic Atari B BeBox

BeOS Blob Blobs Boolean Bram Bram-Moolenaar BufAdd BufCreate BufDelete
BufEnter BufFilePost BufFilePre BufHidden BufLeave BufNew BufNewFile BufRead
BufReadCmd BufReadPost BufReadPre BufUnload BufWinEnter BufWinLeave
BufWipeout BufWrite BufWriteCmd BufWritePost BufWritePre C C-editing
C-indenting C89 C99 COMSPEC CR-used-for-NL **CTRL-6 CTRL-<PageDown>**
CTRL-<PageUp> CTRL-A CTRL-B CTRL-C CTRL-D CTRL-E CTRL-F CTRL-G CTRL-H CTRL-I
CTRL-J CTRL-L CTRL-M CTRL-N CTRL-O CTRL-P CTRL-Q CTRL-R CTRL-T CTRL-U
CTRL-U-changed **CTRL-V CTRL-V**-alternative **CTRL-W CTRL-W_+ CTRL-W_- CTRL-W_:**
CTRL-W_< CTRL-W_<BS> CTRL-W_<CR> CTRL-W_<Down> CTRL-W_<Enter> CTRL-W_<Left>
CTRL-W_<Right> CTRL-W_<Up> CTRL-W_= CTRL-W_> CTRL-W_CTRL-B CTRL-W_CTRL-C
CTRL-W_CTRL-D CTRL-W_CTRL-F CTRL-W_CTRL-H CTRL-W_CTRL-I CTRL-W_CTRL-J
CTRL-W_CTRL-K CTRL-W_CTRL-L CTRL-W_CTRL-N CTRL-W_CTRL-O CTRL-W_CTRL-P
CTRL-W_CTRL-Q CTRL-W_CTRL-R CTRL-W_CTRL-S CTRL-W_CTRL-T CTRL-W_CTRL-V
CTRL-W_CTRL-W CTRL-W_CTRL-X CTRL-W_CTRL-Z CTRL-W_CTRL-] CTRL-W_CTRL-^
CTRL-W_CTRL-_ **CTRL-W_F CTRL-W_H CTRL-W_J CTRL-W_K CTRL-W_L CTRL-W_P CTRL-W_R**
CTRL-W_S CTRL-W_T CTRL-W_W CTRL-W_] CTRL-W_^ CTRL-W_� CTRL-W_b CTRL-W_bar
CTRL-W_c CTRL-W_d CTRL-W_f CTRL-W_g<Tab> CTRL-W_gF CTRL-W_gT CTRL-W_g]
CTRL-W_g_CTRL-] CTRL-W_gf CTRL-W_gt CTRL-W_g} **CTRL-W_h CTRL-W_i CTRL-W_j**
CTRL-W_k CTRL-W_l CTRL-W_n CTRL-W_o CTRL-W_p CTRL-W_q CTRL-W_r CTRL-W_s
CTRL-W_t CTRL-W_v CTRL-W_w CTRL-W_x CTRL-W_z CTRL-W_} **CTRL-X CTRL-Y CTRL-Z**
CTRL-_CTRL-G CTRL-_CTRL-N CTRL-] CTRL-^ CTRL-{char} Channel Channels
Chinese Class Cmd-event CmdUndefined Cmdline Cmdline-mode CmdlineChanged
CmdlineEnter CmdlineLeave CmdwinEnter CmdwinLeave ColorScheme ColorSchemePre
Command-line Command-line-mode CompleteChanged CompleteDone CompleteDonePre
ConPTY Contents Cscope CursorHold CursorHold-example CursorHoldI CursorIM
CursorMoved CursorMovedI D DOS DOS-format DOS-format-write Dict Dictionaries
Dictionary Dictionary-function DiffUpdated Digraphs DirChanged DirChangedPre
E E10 E100 E1001 E1002 E1003 E1004 E1005 E1006 E1007 E1008 E1009 E101 E1010
E1011 E1012 E1013 E1014 E1015 E1016 E1017 E1018 E1019 E102 E1020 E1021 E1022
E1023 E1024 E1025 E1026 E1027 E1028 E1029 E103 E1030 E1031 E1032 E1033 E1034
E1035 E1036 E1037 E1038 E1039 E104 E1040 E1041 E1042 E1043 E1044 E1047 E1048
E1049 E105 E1050 E1051 E1052 E1053 E1054 E1055 E1056 E1057 E1058 E1059 E106
E1060 E1061 E1062 E1063 E1064 E1065 E1066 E1067 E1068 E1069 E107 E1071 E1072
E1073 E1074 E1075 E1077 E1078 E1079 E108 E1080 E1081 E1082 E1083 E1084 E1085
E1087 E1088 E1089 E109 E1090 E1091 E1092 E1093 E1094 E1095 E1096 E1097 E1098
E1099 E11 E110 E1100 E1101 E1102 E1103 E1104 E1105 E1106 E1107 E1109 E111
E1110 E1111 E1112 E1113 E1114 E1115 E1116 E1117 E1118 E1119 E112 E1120 E1121
E1122 E1123 E1124 E1125 E1126 E1127 E1128 E1129 E113 E1130 E1131 E1132 E1133
E1134 E1135 E1136 E1138 E1139 E114 E1140 E1141 E1142 E1143 E1144 E1145 E1146
E1147 E1148 E1149 E115 E1150 E1151 E1152 E1153 E1154 E1155 E1156 E1157 E1158
E1159 E116 E1160 E1161 E1162 E1163 E1164 E1165 E1166 E1167 E1168 E1169 E117
E1170 E1171 E1172 E1173 E1174 E1175 E1176 E1177 E1178 E1179 E118 E1180 E1181
E1182 E1183 E1184 E1185 E1186 E1187 E1188 E1189 E119 E1190 E1191 E1192 E1193
E1194 E1195 E1196 E1197 E1198 E1199 E12 E120 E1200 E1201 E1202 E1203 E1204
E1205 E1206 E1207 E1208 E1209 E121 E1210 E1211 E1212 E1213 E1214 E1215 E1216
E1217 E1218 E1219 E122 E1220 E1221 E1222 E1223 E1224 E1225 E1226 E1227 E1228
E1229 E123 E1230 E1231 E1232 E1233 E1234 E1236 E1237 E1238 E1239 E124 E1240
E1241 E1242 E1243 E1244 E1245 E1246 E1247 E1248 E1249 E125 E1250 E1251 E1252
E1254 E1255 E1256 E1257 E1258 E1259 E126 E1260 E1261 E1262 E1263 E1264 E1265
E1266 E1267 E1268 E1269 E127 E1270 E1271 E1272 E1273 E1274 E1275 E1276 E1277
E1278 E1279 E128 E1280 E1281 E1282 E1283 E1284 E1285 E1286 E1287 E1288 E1289
E129 E1290 E1291 E1292 E1293 E1294 E1295 E1296 E1297 E1298 E1299 E13 E1300
E1301 E1302 E1303 E1304 E1305 E1306 E1307 E1308 E1309 E131 E1310 E1311 E1312
E1313 E1314 E1315 E1316 E1317 E1318 E1319 E132 E1320 E1321 E1322 E1323 E1324

E1325 E1326 E1327 E1328 E133 E1330 E1331 E1332 E1333 E1335 E1336 E1337 E1338
E1339 E134 E1340 E1341 E1342 E1343 E1344 E1345 E1346 E1347 E1348 E1349 E135
E1350 E1351 E1352 E1353 E1354 E1355 E1356 E1357 E1358 E1359 E136 E1360 E1361
E1362 E1363 E1364 E1365 E1366 E1367 E1368 E1369 E137 E1370 E1371 E1372 E1373
E1374 E1375 E1376 E1377 E1378 E1379 E138 E1380 E1381 E1382 E1383 E1384 E1385
E1386 E1387 E1388 E1389 E139 E1390 E1391 E1392 E1393 E1394 E1395 E1396 E1397
E1398 E1399 E140 E1400 E1401 E1402 E1403 E1405 E1406 E1407 E1408 E1409 E141
E1410 E1411 E1412 E1413 E142 E143 E144 E145 E146 E147 E148 E149 E15 E150
E1500 E1501 E1502 E1503 E1504 E1505 E1506 E1507 E1508 E1509 E151 E1510 E1511
E1512 E1513 E152 E153 E154 E155 E156 E157 E158 E159 E16 E160 E161 E162 E163
E164 E165 E166 E167 E168 E169 E17 E170 E171 E172 E173 E174 E175 E176 E177
E178 E179 E18 E180 E181 E182 E183 E184 E185 E186 E187 E188 E189 E19 E190 E191
E192 E193 E194 E195 E196 E197 E199 E20 E200 E201 E202 E203 E204 E205 E206
E207 E208 E209 E21 E210 E211 E212 E213 E214 E215 E216 E217 E218 E219 E22 E220
E221 E222 E223 E224 E225 E226 E227 E228 E229 E23 E230 E231 E232 E233 E234
E235 E236 E237 E238 E239 E24 E240 E241 E242 E243 E244 E245 E246 E247 E248
E249 E25 E250 E251 E252 E253 E254 E255 E257 E258 E259 E26 E260 E261 E262 E263
E264 E265 E266 E267 E268 E269 E27 E270 E271 E272 E273 E274 E275 E276 E277
E279 E28 E280 E282 E283 E284 E285 E286 E287 E288 E289 E29 E290 E292 E293 E294
E295 E296 E297 E298 E299 E30 E300 E301 E302 E303 E304 E305 E306 E307 E308
E309 E31 E310 E311 E312 E313 E314 E315 E316 E317 E318 E319 E32 E320 E321 E322
E323 E324 E325 E326 E327 E328 E329 E33 E330 E331 E332 E333 E334 E335 E336
E337 E338 E339 E34 E340 E341 E342 E343 E344 E345 E346 E347 E348 E349 E35 E350
E351 E352 E353 E354 E355 E356 E357 E358 E359 E36 E360 E362 E363 E364 E365
E366 E367 E368 E369 E37 E370 E371 E372 E373 E374 E375 E376 E377 E378 E379 E38
E380 E381 E382 E383 E384 E385 E386 E387 E388 E389 E39 E390 E391 E392 E393
E394 E395 E397 E398 E399 E40 E400 E401 E402 E403 E404 E405 E406 E407 E408
E409 E41 E410 E411 E412 E413 E414 E415 E416 E417 E418 E419 E42 E420 E421 E422
E423 E424 E425 E426 E427 E428 E429 E43 E430 E431 E432 E433 E434 E435 E436
E437 E438 E439 E44 E440 E441 E442 E443 E444 E445 E446 E447 E448 E449 E45 E450
E451 E452 E453 E454 E455 E456 E457 E458 E459 E46 E460 E461 E462 E463 E464
E465 E466 E467 E468 E469 E47 E470 E471 E472 E473 E474 E475 E476 E477 E478
E479 E48 E480 E481 E482 E483 E484 E485 E486 E487 E488 E489 E49 E490 E491 E492
E493 E494 E495 E496 E497 E498 E499 E50 E500 E501 E502 E503 E504 E505 E506
E507 E508 E509 E51 E510 E511 E512 E513 E514 E515 E516 E517 E518 E519 E52 E520
E521 E522 E523 E524 E525 E526 E527 E528 E529 E53 E530 E531 E532 E533 E534
E535 E536 E537 E538 E539 E54 E540 E542 E543 E544 E545 E546 E547 E548 E549 E55
E550 E551 E552 E553 E554 E555 E556 E557 E558 E559 E560 E561 E562 E563 E564
E565 E566 E567 E568 E570 E571 E572 E573 E574 E575 E576 E577 E579 E580 E581
E582 E583 E584 E585 E586 E587 E588 E589 E59 E590 E591 E592 E593 E594 E595
E596 E597 E598 E599 E60 E600 E601 E602 E603 E604 E605 E606 E607 E608 E609 E61
E610 E611 E612 E613 E614 E616 E617 E618 E619 E62 E620 E621 E622 E623 E624
E625 E626 E627 E628 E629 E63 E630 E631 E632 E633 E634 E635 E636 E637 E638
E639 E64 E640 E641 E642 E643 E644 E645 E646 E647 E648 E65 E650 E651 E652 E654
E655 E656 E657 E658 E659 E66 E660 E661 E662 E663 E664 E665 E666 E667 E668
E669 E67 E670 E671 E672 E673 E674 E675 E676 E677 E678 E679 E68 E680 E681 E682
E683 E684 E685 E686 E687 E688 E689 E69 E690 E691 E692 E693 E694 E695 E696
E697 E698 E699 E70 E700 E701 E702 E703 E704 E705 E706 E707 E708 E709 E71 E710
E711 E712 E713 E714 E715 E716 E717 E718 E719 E72 E720 E721 E722 E723 E724
E725 E726 E727 E728 E729 E73 E730 E731 E732 E733 E734 E735 E736 E737 E738
E739 E74 E740 E741 E742 E743 E744 E745 E746 E747 E748 E749 E75 E750 E751 E752
E753 E754 E755 E756 E757 E758 E759 E76 E760 E761 E762 E763 E764 E765 E766
E767 E768 E769 E77 E770 E771 E772 E773 E774 E775 E776 E777 E778 E779 E78 E780
E781 E782 E783 E784 E785 E786 E787 E788 E789 E79 E790 E791 E792 E793 E794

E795 E796 E797 E798 E799 E80 E800 E801 E802 E803 E804 E805 E806 E807 E808
E809 E81 E810 E811 E812 E813 E814 E815 E816 E817 E818 E819 E82 E820 E821 E822
E823 E824 E825 E826 E827 E828 E829 E83 E830 E831 E832 E833 E834 E835 E836
E837 E838 E84 E840 E841 E842 E843 E844 E845 E846 E847 E848 E849 E85 E850 E851
E852 E853 E854 E855 E856 E857 E858 E859 E86 E860 E861 E862 E863 E864 E865
E866 E867 E868 E869 E87 E870 E871 E872 E873 E874 E875 E876 E877 E878 E879 E88
E880 E881 E882 E883 E884 E885 E886 E887 E888 E889 E89 E890 E891 E892 E893
E894 E895 E896 E897 E898 E899 E90 E900 E901 E902 E903 E904 E905 E906 E907
E908 E909 E91 E910 E911 E912 E913 E914 E915 E916 E917 E918 E919 E92 E920 E921
E923 E924 E925 E926 E927 E928 E929 E93 E930 E931 E932 E933 E934 E935 E936
E937 E938 E939 E94 E940 E941 E942 E943 E944 E945 E946 E947 E948 E949 E95 E950
E951 E952 E953 E954 E955 E956 E957 E958 E959 E96 E960 E961 E962 E963 E964
E965 E966 E967 E968 E969 E97 E970 E971 E972 E973 E974 E975 E976 E977 E978
E979 E98 E980 E981 E982 E983 E984 E985 E986 E987 E988 E989 E99 E990 E991 E992
E993 E994 E995 E996 E997 E998 E999 EX EXINIT Elvis EncodingChanged Eterm Ex
Ex-mode ExitPre Exuberant_ctags F FALSE FAQ Farsi FileAppendCmd
FileAppendPost FileAppendPre FileChangedRO FileChangedShell
FileChangedShellPost FileEncoding FileExplorer FileReadCmd FileReadPost
FileReadPre FileType FileWriteCmd FileWritePost FileWritePre FilterReadPost
FilterReadPre FilterWritePost FilterWritePre Float FocusGained FocusLost
Folding FuncUndefined Funcref G GNOME GTK GTK+ GTK3 GUI GUI-X11 GUIEnter
GUIFailed GetLatestVimScripts GetLatestVimScripts-copyright
GetLatestVimScripts_dat Gnome H Haiku I ICCF IM-server IME Insert Insert-mode
InsertChange InsertCharPre InsertEnter InsertLeave InsertLeavePre Integer
Interface J Japanese Job Jobs K KDE KVM Kibaale Korean L Linux-backspace
List Lists LogiPat() LogiPat-flags Lua M MDI MS-DOS MS-Windows MSDOS
MSVisualStudio MVS Mac Mac-format Mac-format-write Macintosh Mark MenuPopup
MiNT ModeChanged Moolenaar MorphOS Motif Myspell MzScheme N N% N: N NFA
NL-used-for-Nul Neovim NetBSD-backspace NetBeans NetUserPass()
NoDefaultCurrentDirectoryInExePath None Normal Normal-mode Number Nvi O OS/2
OS2 OS390 OS390-Motif OS390-PuTTY OS390-bugs OS390-has-ebcdic
OS390-limitations OS390-open-source Object OffTheSpot OnTheSpot
Operator-pending Operator-pending-mode OptionSet OverTheSpot P PATHEXT PEP8
PHP_BracesAtCodeLevel PHP_IndentFunctionCallParameters
PHP_IndentFunctionDeclarationParameters PHP_autoformatcomment
PHP_default_indenting PHP_noArrowMatching PHP_outdentSLComments
PHP_outdentphpescape PHP_removeCRwhenUnix PHP_vintage_case_default_indent
Partial Pattern Perl Posix Python Q Q-command-changed QNX Q_ab Q_ac Q_ai Q_bu
Q_ce Q_ch Q_cm Q_co Q_ct Q_de Q_di Q_ed Q_et Q_ex Q_fl Q_fo Q_gu Q_in Q_km
Q_lr Q_ma Q_op Q_pa Q_qf Q_ra Q_re Q_sc Q_si Q_ss Q_st Q_sy Q_ta Q_tm Q_to
Q_ud Q_ur Q_vc Q_vi Q_vm Q_wi Q_wq QuickFixCmdPost QuickFixCmdPost-example
QuickFixCmdPre Quickfix QuitPre R RISC-OS RISCOS RemoteReply Replace
Replace-mode Root Ruby Russian S SHELL SQLGetType SQLSetType SafeState
SafeStateAgain Select Select-mode Select-mode-mapping Session
SessionLoad-variable SessionLoadPost ShellCmdPost ShellFilterPost SigUSR1
SourceCmd SourcePost SourcePre Special SpellFileMissing StdinReadPost
StdinReadPre String Sven-Guckes SwapExists Syntax T TCL TERM TOhtml-encoding
TOhtml-encoding-detect TOhtml-performance TOhtml-progress-color
TOhtml-uncopyable-text TOhtml-wrap-text TRUE TSQL TTpro-telnet Tab TabClosed
TabEnter TabLeave TabNew Tcl TermChanged TermResponse TermResponseAll
TermdebugStartPost TermdebugStartPre TermdebugStopPost TermdebugStopPre
Terminal-Job Terminal-Normal Terminal-mode TerminalOpen TerminalWinOpen
TextChanged TextChangedI TextChangedP TextChangedT TextYankPost Transact-SQL
U UTF-8 UTF8-xterm Uganda Unicode Unix Unix-format Unix-format-write User

UserGettingBored V VIMINIT VMS Vi View Vim9 Vim9-abstract-class Vim9-class
Vim9-class-member Vim9-class-overview Vim9-enum Vim9-script Vim9-simple-class
Vim9-type Vim9-using-interface VimEnter VimLeave VimLeavePre VimResized
VimResume VimSuspend Vimball-copyright Virtual-Replace-mode Visual
Visual-mode W W10 W11 W12 W13 W14 W15 W16 W17 W18 W19 W20 W21 W22 WORD WSL
WWW Win32 WinBar WinClosed WinEnter WinLeave WinNew WinNewPre WinResized
WinResized-event WinScrolled WinScrolled-event X X11 X11-icon
X11_mouse_shapes X1Drag X1Mouse X1Release X2Drag X2Mouse X2Release XIM XLFD Y
Y2K ZQ ZZ [[# [' [([++opt] [+cmd] [..] [/ [:alnum:] [:alpha:] [:backspace:]
[:blank:] [:cntrl:] [:digit:] [:escape:] [:fname:] [:graph:] [:ident:]
[:keyword:] [:lower:] [:print:] [:punct:] [:return:] [:space:] [:tab:]
[:upper:] [:xdigit:] [<MiddleMouse> [=] [D [I [M [P [S [[[] [_CTRL-D
[_CTRL-I [` [c [count] [d [f [i [m [p [pattern] [quotex] [range] [s [star [z
[{ \0]]#]'])] /]<MiddleMouse>]D]I]M]P]S] []]]_CTRL-D]_CTRL-I] `
]c]d]f]i]m]p]s]star]z]} ^ _ _exerc _gvimrc _vimrc ` `(`) ` -expansion
' . `0 `< `= `> `A `[`] `^ `` `a `quote `{ `}` a a' a(a) a4 a:0 a:000 a:1
a:firstline a:lastline a:var a< a> aB aW a[a] a` ab abandon abbreviations
abel.vim abs() abstract-method acos() active-buffer ada#Create_Tags()
ada#Jump_Tag() ada#Listtags() ada#Switch_Syntax_Option() ada#Word()
ada-compiler ada-ctags ada-extra-plugins ada-reference ada.vim add()
add-filetype-plugin add-global-plugin add-local-help add-option-flags
add-package add-plugin added-5.1 added-5.2 added-5.3 added-5.4 added-5.5
added-5.6 added-5.7 added-5.8 added-6.1 added-6.2 added-6.3 added-6.4
added-7.1 added-7.2 added-7.3 added-7.4 added-8.1 added-8.2 added-9.1
added-9.2 added-BeOS added-Mac added-VMS added-cmdline-args added-options
added-regexp added-various added-win32-GUI aff-dic-format after-directory
aleph alt alt-input alternate-file amiga-window and() anonymous-function
ant.vim ap apache.vim append() appendbuflines() aquote arabic.txt arabicfonts
arabickeymap arg-functions argc() argidx() arglist arglist-position
arglist-quit arglistid() argument-list argv() argv-variable arrow_modifiers
as asin() asm.vim asm68k asmh8300.vim assert-functions-details assert-return
assert_beeeps() assert_equal() assert_equalfile() assert_exception()
assert_fails() assert_false() assert_inrange() assert_match() assert_nobeep()
assert_notequal() assert_notmatch() assert_report() assert_true() at atan()
atan2() attr-list author auto-format auto-setting auto-shortname autocmd-<>
autocmd-buffer-local autocmd-buflocal autocmd-changes autocmd-define
autocmd-disable autocmd-events autocmd-events-abc autocmd-execute
autocmd-expand autocmd-functions autocmd-groups autocmd-intro autocmd-list
autocmd-nested autocmd-once autocmd-osfiletypes autocmd-patterns
autocmd-remove autocmd-searchpat autocmd-use autocmd.txt autocmd_add()
autocmd_delete() autocmd_get() autocmds-kept autocommand autocommand-events
autocommand-pattern autocommands autoformat autoloader autoloader-functions
avoid-hit-enter aw a{ a} b b: b:changedtick b:changelog_name
b:clojure_syntax_keywords b:clojure_syntax_without_core_keywords
b:current_syntax-variable b:netrw_lastfile b:rust_cargo_avoid_whole_workspace
b:rust_cargo_check_all_features b:rust_cargo_check_all_targets
b:rust_cargo_check_benches b:rust_cargo_check_examples
b:rust_cargo_check_tests b:rustfmt_autosave b:tex_stylish b:var b:yaml_schema
baan-folding baan-syntax baan.vim backslash backspace backspace-delete
backtick-expansion backup backup-changed backup-extension backup-table
balloon-eval balloon_gettext() balloon_show() balloon_split() bar bars
base_font_name_list basic.vim beep beos better-python-interface
beval_bufnr-variable beval_col-variable beval_lnum-variable
beval_text-variable beval_winid-variable beval_winnr-variable binary-number

bitwise-function bitwise-shift blob blob-concatenation blob-functions
blob-identity blob-index blob-literal blob-modification blob2list()
blockwise-examples blockwise-operators blockwise-put blockwise-register
blockwise-visual blowfish blowfish2 bold bom-bytes book bookmark books
boolean break-finally browse() browsedir() browsefilter bufadd() bufexists()
buffer-functions buffer-hidden buffer-list buffer-variable buffer-write
buffer_exists() buffer_name() buffer_number() buffers buffers-menu
buflisted() bufload() bufloaded() bufname() bufnr() bufwinid() bufwinnr()
bug-fixes-5 bug-fixes-6 bug-fixes-7 bug-fixes-8 bug-fixes-9 bug-fixes-9.1
bug-fixes-9.2 bug-reports bugreport.vim bugs builtin-function-details
builtin-function-list builtin-functions builtin-object-methods builtin-terms
builtin-tools builtin.txt builtin_terms byte-count byte2line() byteidx()
byteidxcomp() bzip2 c c.vim cW c_# c_## c_#< c_#n c_% c_%% c_%%## c_%%< c_%%n
c_<BS> c_<C-Left> c_<C-R> c_<C-R>_<C-A> c_<C-R>_<C-F> c_<C-R>_<C-L>
c_<C-R>_<C-O> c_<C-R>_<C-P> c_<C-R>_<C-R> c_<C-R>_<C-W> c_<C-Right> c_<CR>
c_ c_<Down> c_<End> c_<Esc> c_<Home> c_<Insert> c_<Left> c_<LeftMouse>
c_<MiddleMouse> c_<NL> c_<PageDown> c_<PageUp> c_<Right> c_<S-Down>
c_<S-Left> c_<S-Right> c_<S-Tab> c_<S-Up> c_<Tab> c_<Up> c_BS c_CR c_CTRL-A
c_CTRL-B c_CTRL-C c_CTRL-D c_CTRL-E c_CTRL-F c_CTRL-G c_CTRL-H c_CTRL-I
c_CTRL-J c_CTRL-K c_CTRL-L c_CTRL-M c_CTRL-N c_CTRL-P c_CTRL-Q c_CTRL-R
c_CTRL-R_= c_CTRL-R_CTRL-A c_CTRL-R_CTRL-F c_CTRL-R_CTRL-L c_CTRL-R_CTRL-O
c_CTRL-R_CTRL-P c_CTRL-R_CTRL-R c_CTRL-R_CTRL-W c_CTRL-SHIFT-Q c_CTRL-SHIFT-V
c_CTRL-T c_CTRL-U c_CTRL-V c_CTRL-W c_CTRL-Y c_CTRL-[c_CTRL-_CTRL-G
c_CTRL-_CTRL-N c_CTRL-_e c_CTRL-] c_CTRL-^ c_CTRL-_ c_Del c_Down c_End
c_Esc c_Home c_Insert c_Left c_Right c_Up c_ansi_constants c_ansi_typedefs
c_comment_strings c_curly_error c_digraph c_function_pointers c_functions
c_gnu c_no_ansi c_no_bracket_error c_no_bsd c_no_c11 c_no_c99 c_no_cformat
c_no_curly_error c_no_if0 c_no_tab_space_error c_no_trail_space_error
c_no_utf c_space_errors c_syntax_for_h c_wildchar call() carriage-return case
catch-all catch-errors catch-interrupt catch-order catch-text cc ceil()
cfilter-plugin ch.vim ch_canread() ch_close() ch_close_in() ch_evalexp()
ch_evalraw() ch_getbufnr() ch_getjob() ch_info() ch_log() ch_logfile()
ch_open() ch_read() ch_readblob() ch_readraw() ch_sendexpr() ch_sendraw()
ch_setoptoptions() ch_status() change-list-jumps change-name change-tabs
change.txt changed-5.1 changed-5.2 changed-5.3 changed-5.4 changed-5.5
changed-5.6 changed-5.7 changed-5.8 changed-6.1 changed-6.2 changed-6.3
changed-6.4 changed-7.1 changed-7.2 changed-7.3 changed-7.4 changed-8.1
changed-8.2 changed-9.1 changed-9.2 changelist changelog.vim changenr()
changetick changing channel channel-address channel-callback channel-close
channel-close-in channel-commands channel-demo channel-drop channel-functions
channel-functions-details channel-mode channel-more channel-noblock
channel-onetime-callback channel-open channel-open-options channel-raw
channel-timeout channel-use channel.txt char-variable char2nr() characterwise
characterwise-register characterwise-visual charclass() charcol()
charconvert_from-variable charconvert_to-variable charidx() charity charset
charset-conversion chdir() chill.vim chmod cindent() cinkeys-format cino-#
cino-(cino-) cino+ cino-/ cino-: cino-= cino-> cino-C cino-E cino-J cino-L
cino-M cino-N cino-P cino-U cino-W cino-^ cino-b cino-c cino-e cino-f cino-g
cino-h cino-i cino-j cino-k cino-l cino-m cino-n cino-p cino-star cino-t
cino-u cino-w cino-{ cino-} cinoptions-values class class-compile
class-method clear-undo clearmatches() client-server client-server-name
clientserver clipboard clipboard-autoselect clipboard-autoselectml
clipboard-autoselectplus clipboard-exclude clipboard-html clipboard-unnamed
clipboard-unnamedplus closure-indent close_cb closure cmdarg-variable

cmdbang-variable cmdline-arguments cmdline-changed cmdline-completion
cmdline-editing cmdline-history cmdline-lines cmdline-ranges cmdline-special
cmdline-too-long cmdline-window cmdline.txt cmdwin cmdwin-char cobol.vim
codeset coding-style col() coldfusion.vim collapse collate-variable
color-schemes color-xterm coloring colorscheme-override colortest.vim
command-attributes command-block command-line-functions command-line-window
command-mode compatible-default compile-changes-5 compile-changes-6
compile-changes-7 compile-changes-8 compile-changes-9 compile-changes-9.2
compiler-compaqada compiler-decada compiler-dotnet compiler-gcc compiler-gnat
compiler-hpada compiler-manx compiler-perl compiler-pyunit compiler-select
compiler-tex compiler-vaxada compl-current compl-define compl-dictionary
compl-filename compl-function compl-generic compl-keyword compl-omni
compl-omni-filetypes compl-spelling compl-stop compl-tag compl-thesaurus
compl-thesaurusfunc compl-vim compl-whole-line complete() complete-functions
complete-item-kind complete-items complete-popup complete-popuphidden
complete-script-local-functions complete-set-option complete_CTRL-E
complete_CTRL-Y complete_add() complete_check() complete_info()
complete_info_mode completed_item-variable completion-functions
complex-change complex-repeat compress conceal confirm() connection-refused
console-menus constructor context.vim control conversion-server
convert-to-HTML convert-to-XHTML convert-to-XML
convert_legacy_function_to_vim9 copy() copy-diffs copy-move copying copyright
cos() cosh() count count() count-bytes count-items count-variable
count1-variable cp-default cpo cpo-! cpo-# cpo-\$ cpo-% cpo-& cpo-+ cpo--
cpo-. cpo-/ cpo-; cpo-< cpo-> cpo-A cpo-B cpo-C cpo-D cpo-E cpo-F cpo-H cpo-I
cpo-J cpo-K cpo-L cpo-M cpo-O cpo-P cpo-R cpo-S cpo-W cpo-X cpo-Z cpo-\ cpo-a
cpo-b cpo-bar cpo-c cpo-d cpo-e cpo-f cpo-g cpo-i cpo-j cpo-k cpo-l cpo-m
cpo-n cpo-o cpo-p cpo-q cpo-r cpo-s cpo-star cpo-t cpo-u cpo-v cpo-w cpo-x
cpo-y cpo-{ cpp.vim crash-recovery creating-menus credits crontab cs-find
cs7-problem cscope cscope-commands cscope-find cscope-howtouse cscope-info
cscope-intro cscope-limitations cscope-options cscope-suggestions
cscope-win32 cscope_connection() cscopepathcomp cscopeprg cscopequickfix
cscoperelative cscopetag cscopetagorder cscopeverbose csh.vim cspc csprg csqf
csre cst cst0 csverb ctags ctags-gone cterm-colors ctrl ctype-variable
curly-braces-function-names curly-braces-names curpos-visual
current-directory current-file current_compiler cursor() cursor-blinking
cursor-down cursor-functions cursor-left cursor-motions cursor-position
cursor-right cursor-up cursor_down cursor_left cursor_right cursor_up cw
cweb.vim cynlib.vim d daB daW dab dap dart.vim das date-functions dav davs
daw dd debug-gcc debug-highlight debug-leaks debug-minidump debug-mode
debug-scripts debug-signs debug-vim debug-vs2005 debug-win32 debug-windbg
debug.txt debugbreak() debugger-compilation debugger-features
debugger-support debugger.txt dec-mouse decada_members deepcopy()
default-constructor defaults.vim defaults.vim-explained define-function
definition-search definitions delete() delete-insert delete-menus
deletebuflines() deleting demoserver.py design-assumptions design-compatible
design-decisions design-documented design-flexible design-goals
design-improved design-maintain design-multi-platform design-not
design-speed-size desktop.vim develop-spell develop-spell-suggestions
develop.txt development dgn dh diB diW dialog dialogs-added dib dict
dict-functions dict-identity dict-modification did_filetype() diff diff()
diff-diffexpr diff-func-examples diff-mode diff-options diff-original-file
diff-patchexpr diff-slow diff.txt diff.vim diff_filler() diff_hlID()
diff_translations digraph digraph-arg digraph-encoding digraph-table

digraph-table-mbyte digraph.txt digraph_get() digraph_getlist() digraph_set()
digraph_setlist() digraphs digraphs-changed digraphs-default digraphs-define
digraphs-use dip dircolors.vim dis disable-menus discard dist#vim dist#vim9
distribute-script distributed-plugins distribution diw dl do doc-file-list
docbk.vim docbksgml.vim docbkxml.vim docbook documentation-6 donate dos
dos-:cd dos-CTRL-Break dos-backslash dos-colors dos-file-formats
dos-locations dos-powershell dos-pwsh dos-shell dos-standard-mappings
dos-temp-files dosbatch.vim double-click download doxygen-syntax doxygen.vim
dp drag-n-drop drag-n-drop-win32 drag-status-line drop_file dtd.vim dtd2vim
dying-variable e easy echoraw() echospace-variable edit-a-file edit-binary
edit-dialogs edit-files edit-intro edit-no-break edit-paragraph-join
edit-window editing.txt editorconfig-install efm-%> efm-entries efm-ignore
eiffel.vim elixir.vim emacs-keys emacs-tags emacs_tags empty() encoding-names
encoding-table encoding-values encryption end end-of-file
enlightened-terminal environ() eol-and-eof erlang.vim err_buf err_cb err_mode
err_modifiable err_msg err_name err_teapot() err_timeout errmsg-variable
error-file-format error-messages errorformat errorformat-Jikes
errorformat-LaTeX errorformat-Perl errorformat-ant errorformat-changed
errorformat-jade errorformat-javac errorformat-multi-line
errorformat-separate-filename errorformats errors errors-variable escape
escape() escape-bar euphoria3.vim euphoria4.vim euro euro-digraph eval eval()
eval-examples eval-sandbox eval.txt event-variable eventhandler() eview evim
evim-keys evim.vim ex ex-cmd-index ex-edit-index ex-flags ex: except-autocmd
except-autocmd-Cmd except-autocmd-Post except-autocmd-Pre except-autocmd-ill
except-compatible except-examine except-from-finally except-hier-param
except-several-errors except-single-line except-syntax-err
except-syntax-error exception-handling exception-variable exclusive
exclusive-linewise executable() execute() execute-menus exepath() exim
exists() exists_compiled() exiting exiting-variable exp() expand() expand-env
expand-environment-var expandcmd() expr expr-! expr-!= expr-!=# expr-!=?
expr-!~ expr-!~# expr-!~? expr-\$\$' expr-\$quote expr-% expr-&& expr-' expr-+
expr-- expr-. expr-.. expr-/ expr-< expr-<# expr-<< expr-<= expr-<=# expr-<=?
expr-<? expr-== expr-==# expr-==? expr-==~ expr-==~# expr-==~? expr-> expr->#
expr->= expr->=# expr->=? expr->> expr->? expr-[:] expr-[] expr-barbar
expr-entry expr-env expr-env-expand expr-function expr-is expr-is# expr-is?
expr-isnot expr-isnot# expr-isnot? expr-lambda expr-nesting expr-number
expr-option expr-option-function expr-quote expr-register expr-star
expr-string expr-unary-+ expr-unary-- expr-variable expr1 expr10 expr11 expr2
expr3 expr4 expr5 expr6 expr7 expr8 expr9 expression expression-commands
expression-syntax excrc extend() extendnew() extends extension-removal
extensions-improvements f f-args-example false false-variable falsy
falsy-operator faq farsi farsi.txt fasm.vim fast-functions
fcs_choice-variable fcs_reason-variable feature-list feedkeys() fetch
file-browser-5.2 file-formats file-functions file-pattern file-read
file-searching file-type file-types file_readable() fileencoding-changed
filename-backslash filename-modifiers filereadable() filetype filetype-detect
filetype-ignore filetype-override filetype-plugin filetype-plugins
filetype.txt filetypepedetect-changed filetypes filewritable() filler-lines
filter filter() find-manpage find-replace finddir() findfile() fixed-5.1
fixed-5.2 fixed-5.3 fixed-5.4 fixed-5.5 fixed-5.6 fixed-5.7 fixed-5.8
fixed-6.1 fixed-6.2 fixed-6.3 fixed-6.4 fixed-7.1 fixed-7.2 fixed-7.3
fixed-7.4 flatten() flattennew() flexwiki.vim float-e float-functions
float-pi float2nr() floating-point-format floating-point-precision floor()
fmod() fname-variable fname_diff-variable fname_in-variable

fname_new-variable fname_out-variable fnameescape() framemodify() fo-/ fo-1
fo-2 fo-B fo-M fo-] fo-a fo-b fo-c fo-j fo-l fo-m fo-n fo-o fo-p fo-q fo-r
fo-t fo-table fo-v fo-w fold-behavior fold-colors fold-commands
fold-create-marker fold-delete-marker fold-diff fold-expr fold-foldcolumn
fold-foldlevel fold-foldtext fold-indent fold-manual fold-marker fold-methods
fold-options fold-syntax fold.txt foldclosed() foldclosedend()
folddashes-variable foldend-variable folding folding-functions foldlevel()
foldlevel-variable folds foldstart-variable foldtext() foldtextresult()
font-sizes fontset forced-motion foreach() foreground() fork form.vim
format-bullet-list format-comments format-formatexpr formatOtherKeys
formatting forth.vim fortran.vim freebasic.vim freebasic_no_comment_fold
freebasic_operators freebasic_space_errors freebasic_type_suffixes friendship
frombook ft-abel-syntax ft-ada-commands ft-ada-constants ft-ada-functions
ft-ada-indent ft-ada-omni ft-ada-options ft-ada-plugin ft-ada-syntax
ft-ada-variables ft-ant-syntax ft-apache-syntax ft-asciidoc-plugin
ft-asm-syntax ft-asm68k-syntax ft-asmh8300-syntax ft-asperl-syntax
ft-aspvbs-syntax ft-awk-plugin ft-bash-syntax ft-basic-syntax ft-c-omni
ft-c-syntax ft-ch-syntax ft-changelog-plugin ft-changelog-syntax
ft-chill-syntax ft-clojure-indent ft-clojure-syntax ft-cobol-syntax
ft-coldfusion-syntax ft-context ft-context-commands ft-context-intro
ft-context-mappings ft-context-settings ft-cpp-syntax ft-csh-syntax
ft-css-omni ft-cweb-syntax ft-cynlib-syntax ft-dart-syntax ft-dash-syntax
ft-desktop-syntax ft-dircolors-syntax ft-docbk-syntax ft-docbksgml-syntax
ft-docbkxml-syntax ft-dosbatch-syntax ft-dtd-syntax ft-eiffel-syntax
ft-elixir-syntax ft-erlang-syntax ft-euphoria-syntax ft-flexwiki-syntax
ft-form-syntax ft-forth-syntax ft-fortran-indent ft-fortran-plugin
ft-fortran-syntax ft-freebasic-plugin ft-freebasic-syntax ft-fvwm-syntax
ft-gitcommit-plugin ft-gprof-plugin ft-groff-syntax ft-gsp-syntax
ft-haskell-syntax ft-html-indent ft-html-omni ft-html-syntax ft-htlos-syntax
ft-ia64-syntax ft-inform-syntax ft-java-syntax ft-javascript-omni
ft-json-plugin ft-json-syntax ft-ksh-syntax ft-lace-syntax ft-lex-syntax
ft-lifelines-syntax ft-lisp-syntax ft-lite-syntax ft-lpc-syntax ft-lua-syntax
ft-mail-plugin ft-mail.vim ft-make-syntax ft-man-plugin ft-maple-syntax
ft-markdown-plugin ft-markdown-syntax ft-masm-syntax ft-mathematica-syntax
ft-matlab-indent ft-metafont ft-metafont-intro ft-metafont-mappings
ft-metafont-settings ft-metapost ft-metapost-commands ft-metapost-intro
ft-metapost-mappings ft-metapost-settings ft-mma-syntax ft-modula2-syntax
ft-moo-syntax ft-mysql-syntax ft-nlql-syntax ft-nasm-syntax ft-ncf-syntax
ft-nroff-syntax ft-ocaml-syntax ft-papp-syntax ft-pascal-syntax ft-pdf-plugin
ft-perl-syntax ft-php-indent ft-php-omni ft-php-syntax ft-php3-syntax
ft-phtml-syntax ft-plaintex-syntax ft-posix-syntax ft-postscr-syntax
ft-ppwiz-syntax ft-printcap-syntax ft-progress-syntax ft-ptcap-syntax
ft-python-indent ft-python-plugin ft-python-syntax ft-qf-plugin
ft-quake-syntax ft-r-indent ft-r-syntax ft-readline-syntax ft-rego-syntax
ft-rexx-syntax ft-rmd-plugin ft-rmd-syntax ft-rnoweb-plugin ft-rrst-plugin
ft-rrst-syntax ft-rst-plugin ft-rst-syntax ft-ruby-omni ft-ruby-syntax
ft-rust ft-scheme-syntax ft-sdl-syntax ft-sed-syntax ft-sgml-syntax
ft-sh-indent ft-sh-syntax ft-spec-plugin ft-spup-syntax ft-sql ft-sql-omni
ft-sql-syntax ft-sqlanywhere-syntax ft-sqlinformix-syntax ft-squirrel-syntax
ft-syntax-omni ft-tcsh-syntax ft-termcap-syntax ft-tex-plugin ft-tex-syntax
ft-tf-syntax ft-vb-syntax ft-verilog-indent ft-vhdl-indent ft-vim-indent
ft-vim-plugin ft-vim-syntax ft-xf86conf-syntax ft-xhtml-omni ft-xml-omni
ft-xml-syntax ft-xpm-syntax ft-yaml-indent ft-yaml-syntax ft-zimbu-plugin
ft-zsh-syntax ft_ada.txt ft_context.txt ft_mp.txt ft_ps1.txt ft_raku.txt

ft_rust.txt ft_sql.txt ftdetect ftp ftplugin ftplugin-docs ftplugin-name
ftplugin-override ftplugin-special ftplugins fullcommand() funcref()
function() function-argument function-key function-list
function-range-example function-search-undo function_key functions
fuzzy-matching fvwm.vim fvwm2rc fvwmrc g g# g\$ g& g' g'a g+ g, g- g0 g8 g:
g:NetrwTopLvlMenu g:Netrw_UserMaps g:Netrw_corehandler g:Netrw_funcref
g:actual_curbuf g:actual_curwin g:ada#Comment g:ada#Ctags_Kinds
g:ada#DotWordRegex g:ada#Keywords g:ada#WordRegex g:ada_abbrev
g:ada_all_tab_usage g:ada_begin_preproc g:ada_default_compiler
g:ada_extended_completion g:ada_extended_tagging g:ada_folding
g:ada_gnat_extensions g:ada_line_errors g:ada_no_tab_space_error
g:ada_no_trail_space_error g:ada_omni_with_keywords g:ada_rainbow_color
g:ada_space_errors g:ada_standard_types g:ada_with_gnat_project_files
g:ada_withuse_ordinary g:cargo_makeprg_params g:cargo_shell_command_runner
g:clojure_align_multiline_strings g:clojure_align_subforms
g:clojure_discard_macro g:clojure_fold g:clojure_fuzzy_indent
g:clojure_fuzzy_indent_blacklist g:clojure_fuzzy_indent_patterns
g:clojure_maxlines g:clojure_special_indent_words g:clojure_syntax_keywords
g:colors_name g:decada g:decada.Error_Format g:decada.Make()
g:decada.Make_Command g:decada.Unit_Name() g:filetype_csh g:filetype_r
g:ftplugin_rust_source_path g:gnat g:gnat.Error_Format g:gnat.Find()
g:gnat.Find_Program g:gnat.Make() g:gnat.Make_Command g:gnat.Pretty()
g:gnat.Pretty_Program g:gnat.Project_File g:gnat.Set_Project_File()
g:gnat.Tags() g:gnat.Tags_Command g:gzip_exec g:html_charset_override
g:html_diff_one_file g:html_dynamic_folds g:html_encoding_override
g:html_end_line g:html_expand_tabs g:html_font g:html_hover_unfold
g:html_id_expr g:html_ignore_conceal g:html_ignore_folding g:html_line_ids
g:html_no_doc g:html_no_foldcolumn g:html_no_invalid g:html_no_links
g:html_no_modeline g:html_no_pre g:html_no_progress g:html_number_lines
g:html_pre_wrap g:html_prevent_copy g:html_start_line g:html_use_css
g:html_use_encoding g:html_use_input_for_pc g:html_use_xhtml
g:html_whole_filler g:netrw_altfile g:netrw_alto g:netrw_altv g:netrw_banner
g:netrw_bannerbackslash g:netrw_browse_split g:netrw_browsex_support_remote
g:netrw_browsex_viewer g:netrw_bufsettings g:netrw_chgperm g:netrw_chgwin
g:netrw_clipboard g:netrw_compress g:netrw_ctags g:netrw_cursor
g:netrw_cygwin g:netrw_dav_cmd g:netrw_decompress g:netrw_dirhistmax
g:netrw_dynamic_maxfilenamelen g:netrw_errorlvl g:netrw_fastbrowse
g:netrw_fetch_cmd g:netrw_ffkeep g:netrw_file_cmd g:netrw_fname_escape
g:netrw_ftp g:netrw_ftp_browse_reject g:netrw_ftp_cmd g:netrw_ftp_list_cmd
g:netrw_ftp_options g:netrw_ftp_sizelist_cmd g:netrw_ftp_timelist_cmd
g:netrw_ftpextracmd g:netrw_ftpmode g:netrw_glob_escape g:netrw_gx
g:netrw_hide g:netrw_home g:netrw_http_cmd g:netrw_http_put_cmd
g:netrw_http_xcmd g:netrw_ignorenetcrc g:netrw_keepdir g:netrw_keepj
g:netrw_list_cmd g:netrw_list_cmd_options g:netrw_list_hide g:netrw_liststyle
g:netrw_localcopycmd g:netrw_localcopycmdopt g:netrw_localcopydircmd
g:netrw_localcopydircmdopt g:netrw_localmkdir g:netrw_localmkdiropt
g:netrw_localmovecmd g:netrw_localmovecmdopt g:netrw_maxfilenamelen
g:netrw_menu g:netrw_mkdir_cmd g:netrw_mousemaps g:netrw_nobeval g:netrw_nogx
g:netrw_preview g:netrw_rcp_cmd g:netrw_remote_mkdir g:netrw_rename_cmd
g:netrw_retmap g:netrw_rm_cmd g:netrw_rmdir_cmd g:netrw_rmf_cmd
g:netrw_rsync_cmd g:netrw_rsync_sep g:netrw_scp_cmd g:netrw_scpport
g:netrw_sepchr g:netrw_servername g:netrw_sftp_cmd g:netrw_silent
g:netrw_sizestyle g:netrw_sort_by g:netrw_sort_direction g:netrw_sort_options
g:netrw_sort_sequence g:netrw_special_syntax g:netrw_ssh_browse_reject

g:netrw_ssh_cmd g:netrw_sshport g:netrw_suppress_gx_mesg g:netrw_timefmt
g:netrw_tmpfile_escape g:netrw_uid g:netrw_use_errorwindow g:netrw_use_noswf
g:netrw_use_nt_rcp g:netrw_usetab g:netrw_win95ftp g:netrw_winsize
g:netrw_wiw g:netrw_xstrlen g:no_mail_maps g:no_plugin_maps g:plugin_exec
g:rust_bang_comment_leader g:rust_cargo_avoid_whole_workspace
g:rust_cargo_check_all_features g:rust_cargo_check_all_targets
g:rust_cargo_check_benches g:rust_cargo_check_examples
g:rust_cargo_check_tests g:rust_clip_command g:rust_conceal
g:rust_conceal_mod_path g:rust_conceal_pub g:rust_fold
g:rust_keep_autopairs_default g:rust_playpen_url g:rust_recommended_style
g:rust_shortener_url g:rust_use_custom_ctags_defs g:rustc_makeprg_no_percent
g:rustc_path g:rustfmt_autosave g:rustfmt_autosave_if_config_present
g:rustfmt_command g:rustfmt_emit_files g:rustfmt_fail_silently
g:rustfmt_options g:statusline_winid g:syntax_on g:tar_browseoptions
g:tar_cmd g:tar_copycmd g:tar_extractcmd g:tar_nomax g:tar_readoptions
g:tar_secure g:tar_writeoptions g:termdebug_config g:termdebugger
g:terminal_ansi_colors g:tex_comment_nospell g:tex_conceal g:tex_fast
g:tex_flavor g:tex_fold_enabled g:tex_isk g:tex_matchcheck g:tex_no_error
g:tex_nospell g:tex_stylish g:tex_subscripts g:tex_superscripts
g:tex_verbspell g:var g:vim_indent g:vim_indent_cont g:vimball_home
g:vimball_mkdir g:vimsyn_embed g:vimsyn_folding g:vimsyn_maxlines
g:vimsyn_minlines g:vimsyn_noerror g:yaml_schema g:zipPlugin_ext g:zip_exec
g:zip_extractcmd g:zip_nomax g:zip_shq g:zip_unzipcmd g:zip_zipcmd g;
g<Down> g<End> g<Home> g<LeftMouse> g<RightMouse> g<Tab> g<Up> g<kEnd> g? g??
g?g? g@ gD gE gF gH gI gJ gM gN gP gQ gR gT gU gUU gUgU gV g] g^ g_ g_CTRL-A
g_CTRL-G g_CTRL-H g_CTRL-] g` g`a ga garbagecollect() gd gdb gdb-version ge
gender-neutral get() get-ms-debuggers getbufinfo() getbufline()
getbufoneline() getbufvar() getcellwidths() getchangelist() getchar()
getcharmod() getcharpos() getcharsearch() getcharstr() getcmdcomptype()
getcmdline() getcmdpos() getcmdscreenpos() getcmdtype() getcmdwintype()
getcompletion() getcurpos() getcursorcharpos() getcwd() getenv()
getfontname() getfperm() getfsize() getftime() getftype() getimstatus()
getjumplist() getlatestvimscripts-install getline() getloclist()
getmarklist() getmatches() getmousepos() getmousethshape() getpid() getpos()
getqflist() getqflist-examples getreg() getreginfo() getregion()
getregion-notes getregtype() getscrip getscrip-autoinstall getscrip-data
getscrip-history getscrip-plugins getscrip-start getscripinfo()
gettabinfo() gettabvar() gettabwinvar() gettagstack() gettext() getwininfo()
getwinpos() getwinposx() getwinposy() getwinvar() gex gf gg gh gi gj gk
glob() glob2regpat() global-ime global-local global-variable
global_markfilelist globpath() glvs glvs-alg glvs-algorithm glvs-autoinstall
glvs-contents glvs-copyright glvs-data glvs-dist-install glvs-hist
glvs-install glvs-options glvs-plugins glvs-usage gm gn
gnat#Insert_Tags_Header() gnat#New() gnat-xref gnat_members gnome-session go
gp gpm-mouse gq gqap gqgq gqq gr graphic-option-gone greek grep groff.vim
gross-national-happiness group-name gs gsp.vim gstar gt gtk-css
gtk-tooltip-colors gtk3-slow gu gugu gui gui-IME gui-clipboard gui-colors
gui-extras gui-font gui-fontwide gui-footer gui-fork gui-functions gui-gnome
gui-gnome-session gui-gtk gui-gtk-socketid gui-horiz-scroll gui-init gui-kde
gui-mouse gui-mouse-focus gui-mouse-mapping gui-mouse-modeless gui-mouse-move
gui-mouse-select gui-mouse-status gui-mouse-various gui-pty gui-pty-erase
gui-resources gui-scrollbar gui-selections gui-shell gui-shell-win32
gui-start gui-toolbar gui-vert-scroll gui-w32 gui-w32-cmdargs gui-w32-dialogs
gui-w32-printing gui-w32-start gui-w32-various gui-w32-windowid gui-w32s

gui-wayland gui-win32-maximized gui-x11 gui-x11-athena gui-x11-compiling
gui-x11-gtk gui-x11-kde gui-x11-misc gui-x11-motif gui-x11-neXtaw
gui-x11-printing gui-x11-start gui-x11-various gui.txt gui_running
gui_w32.txt gui_x11.txt guifontwide_gtk guifontwide_win_mbyte guu gv gview
gvim gvimdiff gvimrc gw gwgw gww gzip gzip-autocmd gzip-example gzip-helpfile
g~ g~g~ g~~ h haiku-bugs haiku-colors haiku-compiling haiku-dragndrop
haiku-fonts haiku-general haiku-gui haiku-launch haiku-meta haiku-mouse
haiku-support-credits haiku-toolbar-images haiku-user-settings-dir
haiku-vimdir hangul hangulin.txt has() has-patch has-python has-pythonx
has_key() haskell.vim haslocaldir() hasmapto() hebrew hebrew.txt help
help-buffer-options help-context help-curwin help-summary help-tags
help-translated help-writing help-xterm-window help.txt helphelp helphelp.txt
hex-editing hex-number hidden-buffer hidden-changed hidden-menus
hidden-options hidden-quit highlight-args highlight-changed highlight-clear
highlight-cterm highlight-ctermbg highlight-ctermfg highlight-ctermfont
highlight-ctermul highlight-default highlight-font highlight-groups
highlight-gui highlight-guibg highlight-guifg highlight-guisp highlight-start
highlight-stop highlight-term highlightID() highlight_exists()
highlighting-functions hist-names histadd() histdel() histget() histnr()
history history-functions hit-enter hit-enter-prompt hit-return hitest.vim
hijkl hl-ColorColumn hl-Conceal hl-CurSearch hl-Cursor hl-CursorColumn
hl-CursorIM hl-CursorLine hl-CursorLineFold hl-CursorLineNr hl-CursorLineSign
hl-DiffAdd hl-DiffChange hl-DiffDelete hl-DiffText hl-Directory
hl-EndOfBuffer hl-ErrorMsg hl-FoldColumn hl-Folded hl-Ignore hl-IncSearch
hl-LineNr hl-LineNrAbove hl-LineNrBelow hl-MatchParen hl-Menu
hl-MessageWindow hl-ModeMsg hl-MoreMsg hl-NonText hl-Normal hl-Pmenu
hl-PmenuExtra hl-PmenuExtraSel hl-PmenuKind hl-PmenuKindSel hl-PmenuSbar
hl-PmenuSel hl-PmenuThumb hl-PopupNotification hl-Question hl-QuickFixLine
hl-Scrollbar hl-Search hl-SignColumn hl-SpecialKey hl-SpellBad hl-SpellCap
hl-SpellLocal hl-SpellRare hl-StatusLine hl-StatusLineNC hl-StatusLineTerm
hl-StatusLineTermNC hl-TOhtmlProgress hl-TabLine hl-TabLineFill hl-TabLineSel
hl-Terminal hl-Title hl-ToolbarButton hl-ToolbarLine hl-Tooltip hl-User1
hl-User1..9 hl-User9 hl-VertSplit hl-Visual hl-VisualNOS hl-WarningMsg
hl-WildMenu hl-debugBreakpoint hl-debugPC hl-lCursor hlID() hlexists()
hlget() hlsearch-variable hlset() holy-grail home home-replace hostname()
how-do-i how-to howdoi howto howto.txt hpterm hpterm-color html-flavor
html-folding html-indent html-indenting html.vim htmls.vim http i i' i(i)
i< i> iB iBus iW i[i] i_0_CTRL-D i_<BS> i_<C-End> i_<C-Home> i_<C-Left>
i_<C-PageDown> i_<C-PageUp> i_<C-Right> i_<CR> i_ i_<Down> i_<End>
i_<Esc> i_<F1> i_<Help> i_<Home> i_<Insert> i_<Left> i_<LeftMouse> i_<NL>
i_<PageDown> i_<PageUp> i_<Right> i_<S-Down> i_<S-Left> i_<S-Right>
i_<S-ScrollWheelDown> i_<S-ScrollWheelLeft> i_<S-ScrollWheelRight>
i_<S-ScrollWheelUp> i_<S-Up> i_<ScrollWheelDown> i_<ScrollWheelLeft>
i_<ScrollWheelRight> i_<ScrollWheelUp> i_<Tab> i_<Up> i_BS i_CTRL-<PageDown>
i_CTRL-<PageUp> i_CTRL-@ i_CTRL-A i_CTRL-B-gone i_CTRL-C i_CTRL-D i_CTRL-E
i_CTRL-F i_CTRL-G_<Down> i_CTRL-G_<Up> i_CTRL-G_CTRL-J i_CTRL-G_CTRL-K
i_CTRL-G_U i_CTRL-G_j i_CTRL-G_k i_CTRL-G_u i_CTRL-H i_CTRL-I i_CTRL-J
i_CTRL-K i_CTRL-L i_CTRL-M i_CTRL-N i_CTRL-O i_CTRL-P i_CTRL-Q i_CTRL-R
i_CTRL-R_ i_CTRL-R_= i_CTRL-R_CTRL-O i_CTRL-R_CTRL-P i_CTRL-R_CTRL-R
i_CTRL-SHIFT-Q i_CTRL-SHIFT-V i_CTRL-T i_CTRL-U i_CTRL-V i_CTRL-V_digit
i_CTRL-W i_CTRL-X i_CTRL-X_CTRL-D i_CTRL-X_CTRL-E i_CTRL-X_CTRL-F
i_CTRL-X_CTRL-I i_CTRL-X_CTRL-K i_CTRL-X_CTRL-L i_CTRL-X_CTRL-N
i_CTRL-X_CTRL-O i_CTRL-X_CTRL-P i_CTRL-X_CTRL-S i_CTRL-X_CTRL-T
i_CTRL-X_CTRL-U i_CTRL-X_CTRL-V i_CTRL-X_CTRL-Y i_CTRL-X_CTRL-Z

i_CTRL-X_CTRL-] i_CTRL-X_index i_CTRL-X_s i_CTRL-Y i_CTRL-Z i_CTRL-[
i_CTRL-_CTRL-G i_CTRL-_CTRL-N i_CTRL-_CTRL-O i_CTRL-] i_CTRL-^ i_CTRL-_
i_DEL i_Tab i_^_CTRL-D i_backspacing i_digraph i_esc i` ia64.vim ib iccf
iccf-donations icon-changed iconise iconize iconv() iconv-dynamic
ident-search idl-syntax idl.vim if_cscop.txt if_lua.txt if_mzsch.txt
if_ole.txt if_perl.txt if_pyth.txt if_ruby.txt if_sniff.txt if_tcl.txt
ignore-errors ignore-timestamp implements import-autoload import-legacy
import-map improved-autocmds-5.4 improved-quickfix improved-sessions
improved-viminfo improvements-5 improvements-6 improvements-7 improvements-8
improvements-9 improvements-9.2 in_bot in_buf in_io-buffer in_mode in_name
in_top inactive-buffer include-search inclusion inclusive incomp-small-6
incompatible-5 incompatible-6 incompatible-7 incompatible-8 incompatible-9
incompatible-9.2 indent() indent-expression indent.txt indentkeys-format
index index() index.txt indexof() info-message inform.vim informix
initialization inline-function input() inputdialog() inputlist()
inputrestore() inputsave() inputsecret() ins-completion ins-completion-menu
ins-expandtab ins-reverse ins-smarttab ins-softtabstop ins-special-keys
ins-special-special ins-textwidth insert insert() insert-index insert.txt
insert_expand inserting inserting-ex inserting-file insertmode-variable
install install-home install-registry instanceof() intel-itanium
intellimouse-wheel-problems interactive-functions interfaces-5.2
internal-error internal-variables internal-wordlist internet
interpolated-string interrupt() intro intro.txt inverse invert() ip iquote is
isabsolutepath() isdirectory() isinf() islocked() isnan() it italic items()
iw i{ i} j java-cinoptions java-indenting java.vim javascript-cinoptions
javascript-indenting job job-callback job-channel-overview job-close_cb
job-control job-drop job-err_cb job-err_io job-exit_cb job-functions
job-functions-details job-in_io job-noblock job-options job-out_cb job-out_io
job-start job-start-if-needed job-start-nochannel job-stoponexit job-term
job-timeout job_getchannel() job_info() job_setoptions() job_start()
job_status() job_stop() join() js_decode() js_encode() jsbterm-mouse json.vim
json_decode() json_encode() jtags jump-motions jumplist jumplist-stack
jumpto-diffs k kcc kde key-codes key-codes-changed key-mapping key-notation
key-variable keycodes keymap-accents keymap-file-format keymap-hebrew
keypad-0 keypad-9 keypad-comma keypad-divide keypad-end keypad-enter
keypad-home keypad-minus keypad-multiply keypad-page-down keypad-page-up
keypad-plus keypad-point keys() keytrans() kitty-keyboard-protocol
kitty-terminal known-bugs l l: l:var lCursor lace.vim lambda lang-variable
language-mapping language-server-protocol last-pattern last-position-jump
last_buffer_nr() latex-syntax lc_time-variable lcs-conceal lcs-eol
lcs-extends lcs-lead lcs-leadmultispace lcs-multispace lcs-nbsp lcs-precedes
lcs-space lcs-tab lcs-trail left-right-motions legacy-import len() less
letter lex.vim lhaskell.vim libcall() libcallnr() license lid lifelines.vim
limits line() line-continuation line-continuation-comment line2byte()
linefeed linewise linewise-register linewise-visual lisp.vim lispindent()
list list-concatenation list-functions list-identity list-index
list-modification list-repeat list2blob() list2str() listener_add()
listener_flush() listener_remove() lite.vim literal-Dict literal-string
lnum-variable load-plugins load-vim-script local-additions local-function
local-noglobal local-options local-variable local-variables
local_markfilelist locale locale-name localtime() location-list
location-list-file-window location-list-window log() log10() logiPat
logiPat-arg logiPat-caveat logiPat-cmd logiPat-contents logiPat-copyright
logiPat-examples logiPat-history logiPat-input logiPat-man logiPat-manual

logiPat-operators logiPat-pattern long-lines love lowercase lpc.vim lua
lua-blob lua-buffer lua-commands lua-dict lua-dynamic lua-eval lua-funcref
lua-list lua-luaeval lua-vim lua-vim-variables lua-window lua.vim luaeval() m
m' m< m> m[m] m` mac mac-bug mac-compile mac-darwin-feature mac-faq
mac-filename mac-lack mac-standard-mappings mac-vimfile macintosh macro
mail-list mail.vim maillist maillist-archive make.vim man.vim manpager.vim
manual-copyright map() map-<SID> map-CTRL-C map-ambiguous map-backslash
map-backtick map-bar map-comments map-empty-rhs map-error map-examples
map-keys-fails map-listing map-modes map-multibyte map-overview
map-precedence map-return map-self-destroy map-space_in_lhs map-space_in_rhs
map-table map-trailing-white map-typing map-which-keys map.txt map_CTRL-C
map_backslash map_bar map_empty_rhs map_return map_space_in_lhs
map_space_in_rhs maparg() mapcheck() maple.vim mapleader maplist()
maplocalleader mapmode-c mapmode-i mapmode-ic mapmode-l mapmode-n mapmode-nvo
mapmode-o mapmode-s mapmode-t mapmode-v mapmode-x mapnew() mapping
mapping-dict mapping-functions mapset() mark mark-functions mark-motions
markfilelist masm.vim match() match-highlight match-parens match-pattern
matchadd() matchaddpos() matcharg() matchbuflines() matchdelete() matchend()
matchfuzzy() matchfuzzypos() matchit-install matchlist() matchparen
matchstr() matchstrlist() matchstrpos() matlab-indent matlab-indenting max()
maxcol-variable mbyte-IME mbyte-XIM mbyte-combining mbyte-composing
mbyte-conversion mbyte-encoding mbyte-first mbyte-fonts-MSwin mbyte-fonts-X11
mbyte-func mbyte-keymap mbyte-locale mbyte-options mbyte-terminal mbyte-utf8
mbyte.txt menu-changes-5.4 menu-examples menu-priority menu-separator
menu-shortcut menu-text menu-tips menu.vim menu_info() menus merge
message-history message.txt messages meta method mf.vim min() missing-options
mkdir() mlang.txt mma.vim mode() mode-Ex mode-cmdline mode-ins-repl
mode-replace mode-switching modeless-and-clipboard modeless-selection
modeline modeline-local modeline-version modifyOtherKeys modula2.vim
modula2_iso_allow_lowline modula2_iso_disallow_octals
modula2_iso_disallow_synonyms modula2_pim_allow_lowline
modula2_pim_disallow_octals modula2_pim_disallow_synonyms
modula2_r10_allow_lowline moo.vim more-compatible more-prompt more-variables
motif-intellimouse motion-count-multiplied motion.txt mouse-mode-table
mouse-overview mouse-reporting mouse-scrolling-off mouse-swap-buttons
mouse-using mouse_col-variable mouse_lnum-variable mouse_win-variable
mouse_winid-variable movement mp.vim ms-dos msdos msql.vim mswin.vim
multi-byte multi-lang multi-repeat multibyte multibyte-ime multibyte-input
multilang multilang-menus multilang-messages multilang-scripts
multiple-constructors myfiletypefile myscripfile mysql mysyntaxfile
mysyntaxfile-add mysyntaxfile-replace mzeval() mzscheme mzscheme-buffer
mzscheme-commands mzscheme-dynamic mzscheme-examples mzscheme-funcref
mzscheme-mzeval mzscheme-sandbox mzscheme-setup mzscheme-threads mzscheme-vim
mzscheme-vimext mzscheme-window n n1ql.vim nasm.vim navigation nb-commands
nb-events nb-functions nb-messages nb-protocol_errors nb-special nb-terms
ncf.vim netbeans netbeans-commands netbeans-configure netbeans-debugging
netbeans-download netbeans-integration netbeans-intro netbeans-keybindings
netbeans-messages netbeans-parameters netbeans-preparation netbeans-problems
netbeans-protocol netbeans-run netbeans-setup netbeans-support netbeans-xpm
netbeans.txt netreadfixup netrw netrw-% netrw-- netrw-:Explore
netrw-:Hexlore netrw-:Lexlore netrw-:MF netrw-:MT netrw-:NetrwC
netrw-:NetrwMB netrw-:Rexplore netrw-:Sexlore netrw-:Texlore
netrw-:Vexplore netrw-C netrw-D netrw-I netrw-O netrw-P netrw-P18 netrw-P19
netrw-P20 netrw-P21 netrw-P22 netrw-R netrw-S netrw-Tb netrw-Th netrw-U

netrw-X netrw-a netrw-activate netrw-bookmark netrw-bookmarks netrw-browse
netrw-browse-cmds netrw-browse-maps netrw-browser netrw-browser-options
netrw-browser-settings netrw-browser-var netrw-browsing netrw-c-tab netrw-cB
netrw-cadaver netrw-call netrw-cb netrw-cd netrw-chgup netrw-clean
netrw-contents netrw-copyright netrw-cr netrw-createfile netrw-credits
netrw-ctrl-h netrw-ctrl-l netrw-ctrl-r netrw-ctrl_l netrw-curdir netrw-d
netrw-debug netrw-del netrw-delete netrw-dir netrw-dirlist netrw-downdir
netrw-edithide netrw-editwindow netrw-enter netrw-ex netrw-explore
netrw-explore-cmds netrw-expose netrw-externapp netrw-file netrw-filigree
netrw-fixup netrw-ftp netrw-ftype netrw-gb netrw-gd netrw-getftype netrw-gf
netrw-gh netrw-gitignore netrw-gn netrw-gp netrw-grep netrw-gx netrw-handler
netrw-help netrw-hexplore netrw-hide netrw-hiding netrw-history netrw-horiz
netrw-i netrw-incompatible netrw-internal-variables netrw-intro-browse
netrw-leftmouse netrw-lexplore netrw-list netrw-listbookmark netrw-listhack
netrw-login netrw-mA netrw-mB netrw-mF netrw-mT netrw-mX netrw-ma netrw-mb
netrw-mc netrw-md netrw-me netrw-mf netrw-mg netrw-mh netrw-middlemouse
netrw-ml_get netrw-mm netrw-modify netrw-mouse netrw-move netrw-mp netrw-mr
netrw-ms netrw-mt netrw-mu netrw-mv netrw-mx netrw-mz netrw-netrc
netrw-newfile netrw-nexplore netrw-noload netrw-nread netrw-ntree
netrw-nwrite netrw-o netrw-obtain netrw-options netrw-p netrw-p1 netrw-p10
netrw-p11 netrw-p12 netrw-p13 netrw-p14 netrw-p15 netrw-p16 netrw-p17
netrw-p2 netrw-p3 netrw-p4 netrw-p5 netrw-p6 netrw-p7 netrw-p8 netrw-p9
netrw-passwd netrw-password netrw-path netrw-pexplore netrw-preview
netrw-problems netrw-protocol netrw-prvwin netrw-pscp netrw-psftp netrw-putty
netrw-qF netrw-qL netrw-qb netrw-qf netrw-quickcom netrw-quickcoms
netrw-quickhelp netrw-quickmap netrw-quickmaps netrw-r netrw-read netrw-ref
netrw-refresh netrw-rename netrw-reverse netrw-rexplore netrw-rightmouse
netrw-s netrw-s-cr netrw-settings netrw-settings-window netrw-sexplore
netrw-sort netrw-sort-sequence netrw-sortsequence netrw-source netrw-ssh-hack
netrw-star netrw-starpatt netrw-starstar netrw-starstarpatt netrw-start netrw-t
netrw-texplore netrw-todo netrw-trailingslash netrw-transparent netrw-u
netrw-updir netrw-urls netrw-usermaps netrw-userpass netrw-v netrw-var
netrw-variables netrw-vexplore netrw-windows-netrc netrw-windows-s
netrw-write netrw-x netrw-xfer netrw.vim netrw_filehandler netterm-mouse
network new() new-5 new-6 new-7 new-8 new-9 new-GTK-GUI new-MzScheme
new-Select-mode new-View new-argument-list new-buftype new-cmdwin
new-color-schemes new-colorschemes-9 new-commands new-commands-5.4
new-conceal new-debug-itf new-debug-mode new-debug-support
new-define-operator new-diff-mode new-encryption new-evim new-ex-commands-5.2
new-file-browser new-file-writing new-filetype new-filetype-5.4
new-filetype-plugins new-filetype-scripts new-folding new-functions-5.2
new-global-values new-highlighting new-indent-flex new-items-6 new-items-7
new-items-8 new-items-9 new-line-continuation new-location-list new-lua
new-manpage-trans new-map-expression new-map-select new-more-encryption
new-more-highlighting new-more-unicode new-multi-byte new-multi-lang
new-multibyte new-netrw-explore new-network-files new-omni-completion
new-onemore new-operator-mod new-options-5.2 new-options-5.4 new-other-8.2
new-other-9.1 new-other-9.2 new-perl-python new-persistent-undo new-plugins
new-popup-compl new-popup-window new-posix new-print-multibyte new-printing
new-python3 new-regexp-engine new-runtime-dir new-script new-script-5.4
new-scroll-back new-search-path new-searchpatt new-session-files new-spell
new-style-testing new-tab-pages new-terminal-window new-text-properties
new-undo-branches new-unlisted-buffers new-user-defined new-user-manual
new-utf-8 new-vertsplitt new-vim-script new-vim-script-8 new-vim-script-9

new-vim-server new-vimgrep new-vimscript-8.2 new-virtedit news nextnonblank()
no-eval-feature no-type-checking no_buffers_menu no_mail_maps no_plugin_maps
nocombine non-greedy non-zero-arg none-function_argument none-variable
normal-index not-compatible not-edited notation notepad nr2char() nroff.vim
null null-anomalies null-compare null-details null-variable null-variables
null_blob null_channel null_class null_dict null_function null_job null_list
null_object null_partial null_string number_relativenumber numbered-function
numbermax-variable numbermin-variable numbersize-variable o o_CTRL-V o_V
o_object-select o_v object object-const-variable object-empty()
object-final-variable object-len() object-motions object-select
object-string() objects obtaining-extended ocaml.vim octal octal-nrformats
octal-number old-style-testing oldfiles-variable ole-activation ole-eval
ole-gethwnd ole-interface ole-methods ole-normal ole-registration
ole-sendkeys ole-setforeground omap-info omni-sql-completion online-help
opening-window operator operator-doubled operator-pending-index
operator-variable option-backslash option-list option-summary
option-value-function option-window option_restore() option_save()
optional-function_argument options options-changed options-in-terminal
options.txt optwin or() oracle os2 os390 os_390.txt os_amiga.txt os_beos.txt
os_dos.txt os_haiku.txt os_mac.txt os_mint.txt os_msdos.txt os_os2.txt
os_qnx.txt os_risc.txt os_unix.txt os_vms.txt os_win32.txt other-features
out_buf out_cb out_io-buffer out_mode out_modifiable out_msg out_name
out_timeout p pack-add package-create packages packload-two-steps page-down
page-up page_down page_up pager papp.vim paragraph partial pascal.vim
patches-8 patches-8.1 patches-8.2 patches-9 patches-9.1 patches-9.2
patches-after-8.2 patches-after-9.0 patches-after-9.1 pathshorten() pattern
pattern-atoms pattern-delimiter pattern-multi-byte pattern-multi-items
pattern-multibyte pattern-overview pattern-searches pattern.txt
patterns-composing pdev-option peace penc-option perl perl-Append perl-Blob
perl-Buffer perl-Buffers perl-Count perl-Delete perl-DoCommand perl-Eval
perl-Get perl-GetCursor perl-Msg perl-Name perl-Number perl-Set
perl-SetHeight perl-SetOption perl-Windows perl-compiling perl-dynamic
perl-editing perl-overview perl-patterns perl-using perl.vim perlevel()
persistent-undo pexpr-option pfn-option pheader-option photon-fonts
photon-gui php-comment php-indent php-indenting php.vim php3.vim phtml.vim
pi_getscript.txt pi_gzip.txt pi_logipat.txt pi_netrw.txt pi_paren.txt
pi_spec.txt pi_tar.txt pi_vimball.txt pi_zip.txt pkzip plaintex.vim plsqli
plugin plugin-details plugin-filetype plugin-special plugin_exec
plugin_name.txt pmbcs-option pmbfn-option popt-option popup popup-buffer
popup-callback popup-close popup-examples popup-filter popup-filter-errors
popup-filter-mode popup-function-details popup-functions popup-intro
popup-mapping popup-mask popup-menu popup-menu-added popup-position
popup-props popup-scrollbar popup-terminal popup-textprop-pos popup-usage
popup-window popup-window-functions popup.txt popup_atcursor() popup_beval()
popup_beval_example popup_clear() popup_close() popup_create()
popup_create_arguments popup_dialog() popup_dialog-example
popup_filter_menu() popup_filter_yesno() popup_findecho() popup_findinfo()
popup_findpreview() popup_getoptions() popup_getpos() popup_hide()
popup_list() popup_locate() popup_menu() popup_menu-shortcut-example
popup_move() popup_notification() popup_setoptions() popup_settext()
popup_show() popupmenu-completion popupmenu-keys popupwin ports-5.2 ports-6
posix posix-compliance posix-screen-size postgresql postscr.vim
postscript-cjk-printing postscript-print-encoding postscript-print-trouble
postscript-print-util postscript-printing pow() ppwiz.vim press-enter

press-return prevcount-variable preview-popup preview-window prevnonblank()
print-intro print-options print.txt printf() printf-\$ printf-% printf-B
printf-E printf-G printf-S printf-X printf-b printf-c printf-d printf-e
printf-f printf-g printf-o printf-s printf-x printing printing-formfeed
profile profiling profiling-variable progname-variable progpah-variable
progress.vim prompt-buffer prompt_getprompt() prompt_setcallback()
prompt_setinterrupt() prompt_setprompt() promptbuffer-functions pronounce
prop_add() prop_add_list() prop_clear() prop_find() prop_list() prop_remove()
prop_type_add() prop_type_change() prop_type_delete() prop_type_get()
prop_type_list() protected-method protected-variable ps1-about ps1-compiler
ps1-folding ps1-keyword ps1-syntax psql ptcap.vim pterm-mouse public-variable
pum_getpos() pumvisible() put put-Visual-mode py3eval() pyeval() python
python-.locked python-2-and-3 python-Dictionary python-Function python-List
python-VIM_SPECIAL_PATH python-_get_paths python-bindeval
python-bindeval-objects python-buffer python-buffers python-building
python-chdir python-command python-commands python-current python-dynamic
python-environment python-error python-eval python-examples python-fchdir
python-find_module python-foreach_rtp python-input python-options
python-output python-path_hook python-pyeval python-range python-special-path
python-stable python-stable-abi python-strwidth python-tabpage
python-tabpages python-vars python-vim python-vvars python-window
python-windows python.vim python2-directory python3 python3-directory
python3-stable-abi python3-version-variable python_x
python_x-special-comments pythonx pythonx-directory pyxeval() q
q-args-example q/ q: q? qf.vim qnx qnx-compiling qnx-general qnx-terminal
quake.vim quickfix quickfix-6 quickfix-ID quickfix-buffer
quickfix-changedtick quickfix-context quickfix-directory-stack
quickfix-error-lists quickfix-functions quickfix-gcc quickfix-index
quickfix-manx quickfix-parse quickfix-perl quickfix-size quickfix-title
quickfix-valid quickfix-window quickfix-window-ID quickfix-window-function
quickfix.txt quickref quickref.txt quote quote# quote% quote+ quote- quote.
quote/ quote0 quote1 quote2 quote3 quote4 quote9 quote: quote= quote_ quote_
quote_% quote_- quote_. quote_/ quote_: quote_= quote_alpha quote_number
quote_quote quote_~ quotea quotecommandquote quoteplus quotequote quotes
quotes.txt quotestar quote~ r r.vim raku-unicode rand() random range()
raw-terminal-mode rcp read-in-close-cb read-messages read-only-share
read-only-variable read-stdin readblob() readdir() readdirex() readfile()
readline.vim recording recover.txt recovery recursive_mapping redo
redo-register reduce() ref reference reference_toc reg_executing()
reg_recording() regexp regexp-changes-5.4 register register-faq
register-functions register-variable registers rego.vim regular-expression
reload reltime() reltimefloat() reltimestr() remote.txt remote_expr()
remote_foreground() remote_peek() remote_read() remote_send()
remote_startserver() remove() remove-filetype remove-option-flags rename()
rename-files repeat() repeat.txt repeating replacing replacing-ex
reselect-Visual resolve() restore-cursor restore-position restricted-mode
retab-example rethrow reverse() rexx.vim rgb.txt rgview rgvim right-justify
rileft rileft.txt riscos rmd.vim rot13 round() rrst.vim rst.vim rsync ruby
ruby-blob ruby-buffer ruby-command ruby-commands ruby-dynamic ruby-evaluate
ruby-globals ruby-message ruby-rubyeval ruby-set_option ruby-vim ruby-window
ruby.vim ruby_fold ruby_foldable_groups ruby_minlines ruby_no_expensive
ruby_operators ruby_space_errors ruby_spellcheck_strings rubyeval() russian
russian-intro russian-issues russian-keymap russian-l18n russian.txt rust
rust-auto-pairs rust-commands rust-intro rust-mappings rust-settings

rust-syntastic rview rvim rxvt s s/\& s/\@ s/\1 s/\2 s/\3 s/\9 s/\<CR> s/\=
s/\E s/\L s/\U s/\\ s/>\b s/>\e s/>\l s/>\n s/>\r s/>\t s/>\u s/>\~ s:netrw_passwd
s:var s<CR> sandbox sandbox-option save-file save-settings scheme.vim scp
scratch-buffer screenattr() screenchar() screenchars() screencol()
screenpos() screenrow() screenstring() script script-here script-local
script-variable scriptnames-dictionary scriptout-changed scriptversion
scriptversion-1 scriptversion-2 scriptversion-3 scriptversion-4
scroll-binding scroll-cursor scroll-down scroll-horizontal scroll-insert
scroll-mouse-wheel scroll-region scroll-smooth scroll-up scroll.txt
scrollbind-quickadj scrollbind-relative scrolling scrollstart-variable
sdl.vim search() search()-sub-match search-commands search-offset
search-pattern search-range search-replace searchcount() searchdecl()
searchforward-variable searchpair() searchpairpos() searchpos() section
sed.vim self send-money send-to-menu sendto sentence server-functions
server2client() serverlist() servername-variable session-file set-option
set-spc-auto setbufline() setbufvar() setcellwidths() setcharpos()
setcharesearch() setcmdline() setcmdpos() setcursorcharpos() setenv()
setfperm() setline() setloclist() setmatches() setpos() setqflist()
setqflist-action setqflist-examples setqflist-what setreg() settabvar()
settabwinvar() settagstack() setting-guifont setting-guitablable
setting-tabline setuid setwinvar() sftp sgml.vim sgr-mouse sh-awk sh-embed
sh.vim sha256() shell-window shell_error-variable shellescape() shift
shift-left-right shiftwidth() shm-A shm-C shm-F shm-I shm-O shm-S shm-T shm-W
shm-a shm-c shm-f shm-i shm-l shm-m shm-n shm-o shm-q shm-r shm-s shm-t shm-w
shm-x short-name-changed showing-menus sign-column sign-commands
sign-functions sign-functions-details sign-group sign-identifier sign-intro
sign-priority sign-support sign.txt sign_define() sign_getdefined()
sign_getplaced() sign_jump() sign_place() sign_placelist() sign_undefine()
sign_unplace() sign_unplacelist() signs simple-change simplify()
simulated-command sin() single-repeat sinh() sizeofint-variable
sizeoflong-variable sizeofpointer-variable skeleton skip_defaults_vim slice
slice() slow-fast-terminal slow-start slow-terminal socket-interface sort()
sorting sound-functions sound_clear() sound_playevent() sound_playfile()
sound_stop() soundfold() source-vim9-script space spec-customizing
spec-how-to-use-it spec-setting-a-map spec_chglog_format spec_chglog_prepend
spec_chglog_release_info special-buffers specifies speed-up spell
spell-ACCENT spell-AUTHOR spell-BAD spell-BREAK spell-CHECKCOMPOUNDCASE
spell-CHECKCOMPOUNDDUP spell-CHECKCOMPOUNDPATTERN spell-CHECKCOMPOUNDREP
spell-CHECKCOMPOUNDTRIPLE spell-CHECKSHARPS spell-CIRCUMFIX spell-COMMON
spell-COMPLEXPREFIXES spell-COMPOUND spell-COMPOUNDBEGIN spell-COMPOUNDEND
spell-COMPOUNDFIRST spell-COMPOUNDFLAG spell-COMPOUNDFORBIDFLAG
spell-COMPOUNDLAST spell-COMPOUNDMIDDLE spell-COMPOUNDMIN
spell-COMPOUNDPERMITFLAG spell-COMPOUNDRULE spell-COMPOUNDRULE
spell-COMPOUNDRULES spell-COMPOUNDSYLLABLE spell-COMPOUNDSYLMAX
spell-COMPOUNDWORDMAX spell-COPYRIGHT spell-EMAIL spell-FLAG spell-FOL
spell-FORBIDDENWORD spell-HOME spell-IGNOREEXTRA spell-KEEPCASE spell-KEY
spell-LANG spell-LEMMA_PRESENT spell-LOW spell-MAP spell-MAXNGRAMSUGS
spell-NAME spell-NEEDAFFIX spell-NEEDCOMPOUND spell-NOBREAK
spell-NOCOMPOUNDSUGS spell-NOSPLITSUGS spell-NOSUGFILE spell-NOSUGGEST
spell-ONLYINCOMPOUND spell-PFX spell-PFXPOSTPONE spell-PSEUDOROOT spell-RARE
spell-REP spell-SAL spell-SET spell-SFX spell-SLASH spell-SOFOFROM
spell-SOFOTO spell-SUGSWITHDOTS spell-SYLLABLE spell-SYLLABLENUM
spell-SpellFileMissing spell-TRY spell-UPP spell-VERSION spell-WORDCHARS
spell-aff-format spell-affix-chars spell-affix-comment spell-affix-flags

spell-affix-mbyte spell-affix-not-supported spell-affix-vim spell-cjk
spell-compound spell-dic-format spell-double-scoring spell-file-format
spell-functions spell-german spell-load spell-midword spell-mkspell
spell-quickstart spell-remarks spell-russian spell-sug-file spell-syntax
spell-wordlist-format spell-yiddish spell.txt spellbadword()
spellfile-cleanup spellfile.vim spellsuggest() split() splitfind splitview
sponsor sponsor-faq sponsor.txt spoon spup.vim sql-adding-dialects
sql-completion sql-completion-columns sql-completion-customization
sql-completion-dynamic sql-completion-filetypes sql-completion-maps
sql-completion-procedures sql-completion-static sql-completion-tables
sql-completion-tutorial sql-completion-views sql-dialects sql-macros
sql-matchit sql-navigation sql-object-motions sql-predefined-objects
sql-type-default sql-types sql.vim sqlanywhere sqlanywhere.vim sqlgettype
sqlinformix.vim sqlj sqlserver sqlsettype sqrt() squirrel.vim srand() sscanf
standard-plugin standard-plugin-list standout star starstar starstar-wildcard
start-of-file start-vimdiff starting starting-amiga starting.txt startup
startup-options startup-terminal state() static-tag status-line
statusmsg-variable stl-%{ str2float() str2list() str2nr() strcasestr()
strcharlen() strcharpart() strchars() strchr() strcspn() strdisplaywidth()
strftime() strgetchar() stridx() strikethrough string string()
string-functions string-match string-offset-encoding strlen() strpart()
strpbrk() strptime() strrchr() stridx() strspn() strstr() strtrans()
strutf16len() strwidth() style-changes style-compiler style-examples
style-functions style-names style-spaces style-various sub-menu-priority
sub-replace-\= sub-replace-expression sub-replace-special sublist submatch()
subscribe-maillist subscript substitute() substitute-CR substring suffixes
suspend swap-exists-choices swap-file swapchoice-variable
swapcommand-variable swapfile-changed swapfilelist() swapinfo() swapname()
swapname-variable sybase syn-sync-grouphere syn-sync-groupthere
syn-sync-linecont synID() synIDattr() synIDtrans() syncbind syncolor
synconcealed() synload-1 synload-2 synload-3 synload-4 synload-5 synload-6
synstack() syntax syntax-functions syntax-highlighting syntax-latex
syntax-loading syntax-printing syntax-tex syntax.txt syntax_cmd sys-file-list
sysmouse system() system-functions system-vimrc systemlist() s~ t t: t:var
t_#2 t_#4 t_%1 t_%i t_&8 t_8b t_8f t_8u t_@7 t_AB t_AF t_AL t_AU t_BD t_BE
t_CF t_CS t_CTRL-W_ t_CTRL-W_: t_CTRL-W_CTRL-C t_CTRL-W_CTRL-W t_CTRL-W_N
t_CTRL-W_gT t_CTRL-W_gt t_CTRL-W_quote t_CTRL-_CTRL-N t_CV t_Ce t_Co t-Cs
t_DL t_Ds t_EC t_EI t_F1 t_F2 t_F3 t_F4 t_F5 t_F6 t_F7 t_F8 t_F9 t_GP t_IE
t_IS t_K1 t_K3 t_K4 t_K5 t_K6 t_K7 t_K8 t_K9 t_KA t_KB t_KC t_KD t_KE t_KF
t_KG t_KH t_KI t_KJ t_KK t_KL t_PE t_PS t_RB t_RC t_RF t_RI t_RK t_RS t_RT
t_RV t_Ri t_SC t_SH t_SI t_SR t_ST t_Sb t_Sf t_Si t_TE t_TI t_Te t_Ts t_Us
t_VS t_WP t_WS t_XM t_ZH t_ZR t_al t_bc t_blob-variable t_bool-variable t_cd
t_cdl t_ce t_channel-variable t_ci t_cil t_cl t_class-variable t_cm t_cri
t_cs t_csc t_cv t_cvv t_da t_db t_dict-variable t_dl t_ds t_ed t_el t_f1
t_f10 t_f2 t_f3 t_f4 t_f5 t_f6 t_f7 t_f8 t_f9 t_fd t_fe t_float-variable t_fs
t_func-variable t_help t_il t_job-variable t_k1 t_k2 t_k3 t_k4 t_k5 t_k6 t_k7
t_k8 t_k9 t_k; t_kB t_kD t_kI t_kN t_kP t_kb t_kd t_ke t_kh t_kl t_kr t_ks
t_ku t_le t_list-variable t_mb t_md t_me t_mr t_ms t_nd t_none-variable
t_number-variable t_object-variable t_op t_se t_sf1 t_sf10 t_sf2 t_sf3 t_sf4
t_sf5 t_sf6 t_sf7 t_sf8 t_sf9 t_skd t_skl t_skr t_sku t_so t_sr t_star7
t_string-variable t_tb t_te t_ti t_tp t_ts t_ts_old t_typealias-variable t_u7
t_ue t_undo t_us t_ut t_vb t_ve t_vi t_vs t_xn t_xs tab tab-page
tab-page-commands tab-page-intro tab-page-other tabline-menu tabnew-autocmd
tabpage tabpage-variable tabpage.txt tabpagebuflist() tabpagenr()

tabpagewinnr() tag tag-! tag-binary-search tag-blocks tag-commands
tag-details tag-function tag-functions tag-highlight tag-matchlist
tag-old-static tag-overloaded tag-preview tag-priority tag-regexp tag-search
tag-security tag-skip-file tag-stack tagfiles() taglist() tags
tags-and-searches tags-file-changed tags-file-format tags-option tagsrch.txt
tagstack tagstack-examples tan() tanh() tar tar-contents tar-copyright
tar-history tar-manual tar-options tar-usage tcl tcl-beep tcl-buffer
tcl-buffer-append tcl-buffer-cmds tcl-buffer-command tcl-buffer-count
tcl-buffer-delcmd tcl-buffer-delete tcl-buffer-expr tcl-buffer-get
tcl-buffer-insert tcl-buffer-last tcl-buffer-mark tcl-buffer-option
tcl-buffer-set tcl-buffer-windows tcl-bugs tcl-command tcl-commands
tcl-dynamic tcl-ex-commands tcl-examples tcl-expr tcl-linenumbers tcl-misc
tcl-option tcl-output tcl-var-current tcl-var-lbase tcl-var-line tcl-var-lnum
tcl-var-range tcl-variables tcl-window tcl-window-buffer tcl-window-cmds
tcl-window-command tcl-window-cursor tcl-window-delcmd tcl-window-expr
tcl-window-height tcl-window-option tcsh-style tcsh.vim tear-off-menus
telnet-CTRL-] temp-file-name tempfile template tempname() term++close
term++open term-dependent-settings term-list term.txt term_dumpdiff()
term_dumpload() term_dumpwrite() term_getaltscreen() term_getansicolors()
term_getattr() term_getcursor() term_getjob() term_getline()
term_getscrolled() term_getsize() term_getstatus() term_gettitle()
term_gettty() term_list() term_scrape() term_sendkeys() term_setansicolors()
term_setapi() term_setkill() term_setrestore() term_setsize() term_start()
term_wait() termcap termcap-changed termcap-colors termcap-cursor-color
termcap-cursor-shape termcap-options termcap-title termdebug-commands
termdebug-communication termdebug-customizing termdebug-events
termdebug-example termdebug-frames termdebug-mappings termdebug-prompt
termdebug-starting termdebug-stepping termdebug-variables
termdebug_disasm_window termdebug_map_K termdebug_map_minus
termdebug_map_plus termdebug_popup termdebug_shortcuts termdebug_signs
termdebug_use_prompt termdebug_variables_window termdebug_wide
termdebug_winbar terminal terminal-api terminal-autoshellid terminal-bufname
terminal-client-server terminal-close terminal-colors terminal-communication
terminal-cursor-style terminal-debug terminal-debugger terminal-diff
terminal-diffscreeendump terminal-dumptest terminal-function-details
terminal-functions terminal-info terminal-job-index terminal-key-codes
terminal-mouse terminal-ms-windows terminal-options terminal-output-codes
terminal-resizing terminal-screendump terminal-session terminal-size-color
terminal-special-keys terminal-testing terminal-to-job terminal-typing
terminal-unix terminal-use terminal-window terminal.txt terminalprops()
terminfo termresponse-variable ternary test-functions test-functions-details
test_alloc_fail() test_autochdir() test_feedininput() test_garbagecollect_now()
test_garbagecollect_soon() test_getvalue() test_gui_event()
test_ignore_error() test_mswin_event() test_null_blob() test_null_channel()
test_null_dict() test_null_function() test_null_job() test_null_list()
test_null_partial() test_null_string() test_option_not_set() test_override()
test_refcount() test_setmouse() test_settime() test_srand_seed()
test_unknown() test_void() testing testing-support testing-variable
testing.txt tex-cchar tex-cole tex-conceal tex-error tex-folding
tex-matchcheck tex-math tex-morecommands tex-nospell tex-package tex-runon
tex-slow tex-stopzone tex-style tex-supersub tex-sync tex-verb tex.vim
text-functions text-objects text-objects-changed text-prop-changes
text-prop-functions text-prop-intro text-properties text-property-functions
textlock textprop textprop.txt tf.vim this_session-variable throw-catch

throw-expression throw-from-catch throw-variables throwpoint-variable
time-functions timer timer-functions timer_info() timer_pause() timer_start()
timer_stop() timer_stopall() timers timestamp timestamps tips tips.txt tmux
tmux-integration todo todo.txt toggle toggle-revins tolower() toolbar-icon
tooltips toupper() tr() trim() trojan-horse true true-variable trunc() truthy
try-conditionals try-echoerr try-finally try-nested try-nesting tutor
two-engines type() type-casting type-checking type-inference type-mistakes
typealias typename() u uganda uganda.txt undercurl underdashed underdotted
underdouble underline underline-codes undo undo-blocks undo-branches
undo-break undo-close-block undo-commands undo-persistence undo-redo
undo-remarks undo-tree undo-two-ways undo.txt undo_ftplugin undo_indent
undofile() undotree() unicode uniq() unix unlisted-buffer up-down-motions
uppercase urxvt-mouse use-visual-cmds useful-mappings usenet
user-cmd-ambiguous user-commands user-functions user-manual userfunc.txt
using-<Plug> using-menus using-scripts using-xxd using_CTRL-V usr_01.txt
usr_02.txt usr_03.txt usr_04.txt usr_05.txt usr_06.txt usr_07.txt usr_08.txt
usr_09.txt usr_10.txt usr_11.txt usr_12.txt usr_20.txt usr_21.txt usr_22.txt
usr_23.txt usr_24.txt usr_25.txt usr_26.txt usr_27.txt usr_28.txt usr_29.txt
usr_30.txt usr_31.txt usr_32.txt usr_40.txt usr_41.txt usr_42.txt usr_43.txt
usr_44.txt usr_45.txt usr_50.txt usr_51.txt usr_52.txt usr_90.txt usr_toc.txt
utf-8 utf-8-char-arg utf-8-in-xwindows utf-8-typing utf16idx() utf8 v v:
v:argv v:beval_bufnr v:beval_col v:beval_lnum v:beval_text v:beval_winid
v:beval_winnr v:char v:charconvert_from v:charconvert_to v:cmdarg v:cmdbang
v:collate v:colornames v:completed_item v:count v:count1 v:ctype v:dying
v:echospace v:errmsg v:errors v:event v:exception v:exiting v:false
v:fcs_choice v:fcs_reason v:fname v:fname_diff v:fname_in v:fname_new
v:fname_out v:folddashes v:foldend v:foldlevel v:foldstart v:hllsearch
v:insertmode v:key v:lang v:lc_time v:lnum v:maxcol v:mouse_col v:mouse_lnum
v:mouse_win v:mouse_winid v:none v:null v:numbermax v:numbermin v:numbersize
v:oldfiles v:operator v:option_command v:option_new v:option_old
v:option_oldglobal v:option_oldlocal v:option_type v:prevcount v:profiling
v:progname v:progpah v:python3_version v:register v:scrollstart
v:searchforward v:servername v:shell_error v:sizeofint v:sizeoflong
v:sizeofpointer v:statusmsg v:swapchoice v:swapcommand v:swapname v:t_TYPE
v:t_blob v:t_bool v:t_channel v:t_class v:t_dict v:t_float v:t_func v:t_job
v:t_list v:t_none v:t_number v:t_object v:t_string v:t_typealias
v:termblinkresp v:termrbgresp v:termresponse v:termrfgresp v:termstylerep
v:termu7resp v:testing v:this_session v:throwpoint v:true v:val v:var
v:version v:versionlong v:vim_did_enter v:warningmsg v>windowid v_! v_\$ v_
v_< v_<BS> v_ v_<Esc> v_= v_> v_C v_CTRL-A v_CTRL-C v_CTRL-G v_CTRL-H
v_CTRL-O v_CTRL-R v_CTRL-V v_CTRL-X v_CTRL-Z v_CTRL-_CTRL-G v_CTRL-_CTRL-N
v_CTRL-] v_D v_J v_K v_O v_P v_R v_S v_U v_V v_X v_Y v_a v_a' v_a(v_a) v_a<
v_a> v_aB v_aW v_a[v_a] v_a` v_ab v_ap v_aquote v_as v_at v_aw v_a{ v_a}
v_b< v_b<_example v_b> v_b>_example v_b_A v_b_A_example v_b_C v_b_D v_b_I
v_b_I_example v_b_c v_b_r v_b_r_example v_c v_d v_g? v_gF v_gJ v_gN v_gV v_g]
v_g_CTRL-A v_g_CTRL-G v_g_CTRL-X v_g_CTRL-] v_gf v_gn v_gq v_gv v_gw v_i v_i'
v_i(v_i) v_i< v_i> v_iB v_iW v_i[v_i] v_i` v_ib v_ip v_iquote v_is v_it
v_iw v_i{ v_i} v_o v_object-select v_p v_r v_s v_u v_v v_x v_y v_zy v_~ vab
val-variable valgrind values() var-functions variable-categories
variable-scope variable-types variables various various-cmds
various-functions various-motions various.txt vb.vim vba verbose version-5.1
version-5.2 version-5.3 version-5.4 version-5.5 version-5.6 version-5.7
version-5.8 version-6.1 version-6.2 version-6.3 version-6.4 version-7.0
version-7.1 version-7.2 version-7.3 version-7.4 version-8.0 version-8.1

version-8.2 version-9.0 version-9.1 version-9.2 version-variable version4.txt
version5.txt version6.txt version7.0 version7.1 version7.2 version7.3
version7.4 version7.txt version8.0 version8.1 version8.2 version8.txt
version9.0 version9.1 version9.2 version9.txt versionlong-variable vi
vi-differences vi-features vi: vi_diff.txt vib view view-diffs view-file
views-sessions vim-7.4 vim-8 vim-8.1 vim-8.2 vim-9 vim-9.0 vim-9.1 vim-9.2
vim-additions vim-announce vim-arguments vim-changelog vim-default-editor
vim-dev vim-mac vim-modes vim-modes-intro vim-raku vim-script-intro
vim-script-library vim-security vim-use vim-variable vim.b vim.g vim.t vim.v
vim.vim vim.w vim7 vim8 vim9 vim9-access-modes vim9-autoload vim9-boolean
vim9-class vim9-classes vim9-const vim9-curly vim9-debug vim9-declaration
vim9-declarations vim9-differences vim9-export vim9-false-true vim9-final
vim9-func-declaration vim9-function-defined-later vim9-gotchas
vim9-ignored-argument vim9-import vim9-lambda vim9-lambda-arguments
vim9-line-continuation vim9-literal-dict vim9-mix vim9-namespace
vim9-no-dict-function vim9-no-shorten vim9-rationale vim9-reload
vim9-s-namespace vim9-scopes vim9-string-index vim9-types vim9-unpack-ignore
vim9-user-command vim9-variable-arguments vim9.txt vim9class.txt vim9script
vim: vim_announce vim_dev vim_did_enter-variable vim_mac vim_starting vim_use
vimball vimball-contents vimball-extract vimball-history vimball-intro
vimball-manual vimball-windows vimdev vimdiff vimfiles viminfo viminfo-!
viminfo-% viminfo-' viminfo-/ viminfo-: viminfo-< viminfo-@ viminfo-c
viminfo-encoding viminfo-errors viminfo-f viminfo-file viminfo-file-marks
viminfo-file-name viminfo-h viminfo-n viminfo-quote viminfo-r viminfo-read
viminfo-read-write viminfo-s viminfo-timestamp viminfo-write vimrc
vimrc-filetype vimrc-intro vimrc-option-example vimrc_example.vim
vimscript-version vimscript-versions vimtutor virtcol() virtcol2col()
virtual-text visual-block visual-change visual-examples visual-index
visual-mode visual-operators visual-repeat visual-search visual-start
visual-use visual.txt visualmode() vms vms-authors vms-changes vms-compiling
vms-deploy vms-download vms-gui vms-notes vms-problems vms-started vms-usage
vote-for-features votes-counted votes-for-changes vreplace-mode
vt100-cursor-keys vt100-function-keys w w32-clientserver
w32-experimental-keycode-trans-strategy w32-xpm-support w: w:current_syntax
w:quickfix_title w:var waittime warningmsg-variable wdl-syntax wdl.vim
white-space whitespace wildcard wildcards wildmenu() win-scrolled-resized
win16 win32 win32-!start win32-PATH win32-backslashes win32-cmdargs
win32-colors win32-compiling win32-curdir win32-faq win32-gettext win32-gui
win32-hidden-menus win32-installer win32-mouse win32-open-with-menu
win32-popup-menu win32-problems win32-quotes win32-restore win32-startup
win32-term win32-vimrun win32-win3.1 win32-win95 win32s win_execute()
win_findbuf() win_getid() win_gettype() win_gotoid() win_id2tabwin()
win_id2win() win_move_separator() win_move_statusline() win_screenpos()
win_splitmove() winbufnr() wincol() window window-ID window-contents
window-exit window-functions window-move-cursor window-moving window-resize
window-size window-size-functions window-tag window-toolbar window-variable
windowid windowid-variable windows windows-3.1 windows-asynchronously
windows-icon windows-intro windows-starting windows.txt windows95 windows98
windowsme windowsversion() winheight() winid winlayout() winline() winnr()
winrestcmd() winrestview() winsaveview() winwidth() word word-count
word-motions wordcount() workbench workshop workshop-support workshop.txt
wrap-off write-compiler-plugin write-device write-fail write-filetype-plugin
write-library-script write-local-help write-permissions write-plugin
write-plugin-quickload write-quit write-readonly writefile() writing www x

x-input-method x-resources x11-clientserver x11-cut-buffer x11-selection
xattr xf86conf.vim xfontset xfree-xterm xim xim-input-style xterm
xml-folding xml-omni-datafile xml.vim xor() xpm.vim xterm-8-bit xterm-8bit
xterm-blink xterm-blinking-cursor xterm-bracketed-paste xterm-clipboard
xterm-codes xterm-color xterm-command-server xterm-copy-paste
xterm-cursor-keys xterm-end-home-keys xterm-focus-event xterm-function-keys
xterm-kitty xterm-modifier-keys xterm-mouse xterm-mouse-wheel xterm-resize
xterm-save-screen xterm-screens xterm-scroll-region xterm-shifted-keys
xterm-terminfo-entries xterm-true-color y yaml.vim yank ye-option-gone
year-2000 your-runtime-dir yy z z+ z- z. z/OS z<CR> z<Left> z<Right> z= zA zC
zD zE zF zG zH zL zM zN zN<CR> zO zOS zOS-Bugs zOS-Motif zOS-PuTTY
zOS-has-ebcdic zOS-limitations zOS-open-source zP zR zW zX z^ za zb zc zd ze
zf zg zh zi zip zip-contents zip-copyright zip-extension zip-history
zip-manual zip-usage zip-x zj zk zl zm zn zo zp zr zs zsh.vim zt zuG zuW zug
zuw zv zw zx zy zz { {Visual} {address} {arglist} {aupat} {bufname}
{char1-char2} {event} {file} {group-name} {lhs} {motion} {move-around}
{offset} {register} {rhs} {server} {subject} {{{ {} } }}} ~

The text of this document is taken from the Vim help pages and the Vim FAQ.

The files are converted to pdf using $\text{Xe}_{\text{L}}\text{A}_{\text{T}}\text{E}_{\text{X}}$ with the hyperref package. The conversion script is written by Nathan Grigg, based on the HTML conversion script written by [Carlo Teubner](#).

For more information, see <http://nathangrigg.net/vimhelp>.